DESIGN OF A FOUR ROTOR HOVERING VEHICLE

A Thesis

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Master of Science

by

Eryk Brian Nice

May 2004

ABSTRACT

Potential applications of autonomous vehicles range from unmanned surveillance to search and rescue applications dangerous to human beings. Vehicles specifically designed for hover flight have their own possible applications, including the formation of high gain airborne phased antenna arrays. With this specific application in mind, the Cornell Autonomous Flying Vehicle (AFV) team sought to produce a four rotor hovering vehicle capable of eventual untethered acrobatic autonomous flights.

The mechanical design of the AFV included both the selection of a battery-motor-gearing-prop combination for efficient thrust production and the design of a lightweight yet sufficiently stiff vehicle structure. The components chosen were selected from the variety of brushless motors, battery technologies and cell configurations, and fixed pitch propellers suited to use in a four rotor hovering vehicle. The vehicle structure settled upon achieved a high degree of stiffness with minimal weight through the use of thin walled aluminum compression members supported by stranded steel cable.

In addition to an efficient mechanical design, the vehicle also required onboard control and inertial navigation. In order to evaluate a variety of potential vehicle sensor, actuator, estimation, and control scenarios, a fully configurable nonlinear simulation of vehicle and sensor dynamics was also constructed. For the current iteration of the vehicle, a square root implementation of a Sigma Point Filter was used for estimation while a simple Linear Quadratic Regulator based on the nonlinear vehicle dynamics linearized about hover provided vehicle control. Sensory feedback on the current

vehicle included an onboard inertial measurement unit and a human observer, to be eventually replaced by GPS or an indoor equivalent.

While a hardware failure prevented the completion of a full range of tests, the team was able to complete a hands-free hover test that demonstrated the capabilities of the vehicle. Supplemented with various other final hardware tests, the vehicle demonstrated stable hover flight, potential vehicle endurance in the range of 10-15 minutes, and possible vertical acceleration of 0.8g beyond hover thrust. The final vehicle represented a significant achievement in terms of overall design and vehicle capability while future improvements will demonstrate more advanced nonlinear control algorithms and acrobatic flight maneuvers.

BIOGRAPHICAL SKETCH

Eryk was born 27 March 1981 in Tucson, AZ. His family moved to the east coast shortly thereafter and eventually settled in Frederick, MD. While growing up, Eryk cultivated interests that eventually ranged from lacrosse and mock trial to the assistant management of a small nishikigoi retail establishment. Following his graduation from St. John's Literary Institute at Prospect Hall in 1998, Eryk attended Cornell University's College of Engineering. While working towards a Bachelor of Science in Mechanical Engineering, Eryk continued to explore his interest in small business through involvement in the management of a small student-run stage lighting and sound company. Upon obtaining his undergraduate degree, and at the encouragement of friends, family, and faculty, he elected to remain at Cornell to pursue a Master of Science degree in design and control of mechanical systems. While future plans are uncertain, Eryk hopes to eventually own or manage his own engineering firm.

A man's brain is stored powder; it cannot be touched itself off; the fire must come from the outside. -Mark Twain

To all those who have supported, encouraged, challenged, and inspired me

ACKNOWLEDGEMENTS

I would first and foremost like to thank my advisor, Professor Raffaello D'Andrea. His financial support, intellectual contributions, and personal encouragements are what made my work possible. I would also like to thank Sean Breheny, my primary project partner. His electrical design work, late hour company, and occasional mechanical design suggestions were all major factors in the success of the project. The remaining two members of the design team, Ali Squalli and Jinwoo Lee, also provided invaluable support in the electronic engineering aspects of the vehicle design.

In addition to the above mentioned, I owe a lot to the Cornell College of Engineering faculty. The variety of their expertise would have been worthless without their willingness to offer support and intellectual guidance both in classes and on their own time. I would particularly like to thank Professor Mark Campbell for his help on vehicle state estimation.

Finally, I would like to thank those who have watched and helped me grow. Though not reflected in the technical content of this document, the support, encouragement, and love provided by friends and family are perhaps the greatest reason for my success.

This research was funded by Air Force Grant F49620-02-0388.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1:
INTRODUCTION

With the advent of new technologies ranging from global positioning systems to faster, smaller, and lighter computer processors, there has been a surge in development of unmanned vehicles. The benefits of unmanned vehicles include the removal of humans from harm's way and a degree of maneuverability and flexibility in deployment that has historically been unachievable when accommodations for a human pilot were necessary. Unmanned and autonomous vehicles are currently in development for use in air, over land, and in the water by both private and government agencies.

The Autonomous Flying Vehicle (AFV) project at Cornell University has been an ongoing attempt to produce a reliable autonomous hovering vehicle. The advantages of a hovering vehicle over a fixed wing flying vehicle include the minimal space required for takeoff and landing of the vehicle, maneuverability in obstacle-heavy environments, and the ability to maintain a static position and orientation if so desired. One of the more prominent demonstrations of autonomous hovering vehicle potential applications is the annual Aerial Robotics Competition hosted by the Association for Unmanned Vehicle Systems, International [7]. This competition draws research and project teams from around the world to compete in predefined autonomous missions. However, the competition is dominated primarily by converted hobbyist remote control (RC) helicopters well suited to the competition's focus on autonomous navigation and artificial intelligence. While the AFV shares some capabilities and potential applications with entrants in this competition, the AFV project specifically

has oriented its design efforts towards short range reconnaissance and multiple vehicle formation flight. The formation flight application provides both a foundation for another concept, encompassed in the airborne Phase Antenna Array (PAA) project, as well as a demonstration of both single vehicle control and distributed multi-vehicle control algorithms [2]. The requirements of these specific applications, discussed further in the next chapter, include a level of precision, control, maneuverability, and ease of interface that was not readily provided by solutions based on modified available RC vehicles.

The legacy version of the flying vehicle was based on an uncommon, though not unique, four rotor hovering vehicle design. The design was inspired by the purchase of a remote controlled toy, the Roswell Flyer produced by Area 51 Technologies, that uses the concept of speed control of four props, two rotating in each direction, to enable human controlled vehicle hover. The toy was purchased by Professor Raffaello D'Andrea, the advisor to the AFV project. Though the origin of this conceptual design is unknown, there have been a number of research projects based on the idea. The Hoverbot project at the University of Michigan attempted to construct a four rotor hovering vehicle in 1993 by essentially tying together the tails of four RC helicopters. The project was quickly abandoned due to hardware difficulties, the most notable of which was the need to hand craft the pusher rotors necessary for the four rotor design [12]. The PipeDream project team at the Queensland University of Technology has designed and built a four rotor hovering vehicle based on model gas powered engines. Their current version of the vehicle unfortunately suffers from inadequate thrust and possible control issues. They are currently working on an improved design [11]. There are a number of additional projects that have also attempted to produce a four rotor flying vehicle without success, including the X4-flyer in Australia and the

Gizmocopter in California [6], [10]. The most common problems noted seem to revolve around inadequate thrust production and inability to produce a control system capable of achieving stable hover, though most projects make note of intent to remedy this in future versions.

A group in France claimed success in their attempts to control and track a four rotor hovering vehicle. While they employed tethered communication and flight times were limited, they were able to produce hands off hover flights that followed a simple trajectory. The group used a modified version of a commercially available RC vehicle, the Draganflyer IV, in order to focus on the stabilization and tracking issues inherent in the problem without concern for the mechanical design [4]. The Draganflyer IV actually appears to be a fourth generation version of the Roswell Flyer originally purchased by the Cornell AFV team [8]. Another project, the Stanford Mesicopter project, endeavors to produce a miniature version of a four rotor vehicle approximately the size of a quarter. Though they share the same design concept and control scheme, the scale of their project addresses very different design issues than those of previously mentioned projects in aerodynamics, control, and fabrication [9].

The difficulty inherent in producing a total hovering vehicle system capable of sustained, stable, untethered flight is evident from the problems encountered by the assorted teams mentioned. In fact, many of the difficulties encountered by other teams are mirrored in past phases of the Cornell AFV project. While past phases of the project made headway in development of simple hover control systems and electronic design, they were bogged down by implementation details and mechanical shortcomings. At the start of the current phase of the project, the prior team had produced a version of the vehicle which demonstrated certain conceptual

achievements, but was still incapable of stable hover flight due to a lack of adequate thrust. In addition, the legacy vehicle relied on both power and communication tethers and external sensing and processing [5], 0. The goals of the current project phase included migration to a fully self-contained vehicle with onboard power and navigation systems and wireless communication. Despite the burden of the additional power and INS payload, the vehicle was also to be capable of reasonably long hover flights. Additionally, a large degree of maneuverability was desired for potential future demonstration of acrobatic flight maneuvers and their accompanying nonlinear control algorithms. Meeting the above requirements would aid in the high degree of precise control necessary for the PAA application discussed.

Because of the ambitious nature of project goals, the development of the next generation of the AFV involved a complete redesign of the vehicle from the ground up. The new vehicle would share little in common with previous versions beyond the four-rotor hovering vehicle concept. Development of the new version of the AFV can be easily divided into five major stages:

- vehicle conceptual design
- analysis and component-level design and selection
- fabrication, assembly, hardware testing, and re-design
- simulation development and verification
- control and estimation development for future implementation

Though these five stages occasionally overlapped and sometimes interfered with one another, they can be discussed independently.

CHAPTER 2:
VEHICLE CONCEPTUAL DESIGN

The conceptual design phase included primarily the determination of the general layout and design of the next-generation AFV. The first step in this phase was the identification of design goals. After some debate, the team decided upon the following fundamental vehicle requirements:

- Ability to hover – required for desired airborne phased antennae array (PAA) application
- Maneuverability in all directions about hover – equally important in PAA application for tight multi-vehicle formations
- Endurance of no less than ten minutes – ten minutes was judged a practical minimum to allow for sufficient useful flight time between takeoff and landing
- Sufficient control effort beyond hover to ensure a controllable vehicle – previous versions of the AFV could not produce more than 5% residual thrust beyond hover and saturation prevented hover stability
- Onboard power supply and processing – realistic applications would not allow tethers

In addition to these primary requirements, the following qualities were identified as desirable if achievable without detriment to the primary requirements:

- Electric power supply – preferable for ease and safety of use and quiet, indoor operation
- High residual thrust to hover thrust ratio –an acrobatic vehicle was desirable for its ability to demonstrate controllability in difficult to perform maneuvers
- Minimal cost and complexity

APPENDIX A: BRAINSTORMING NOTES contains rough notes on the initial brainstorming stage of the new vehicle design process. A variety of vehicle configurations, propulsion methods, and general ideas were explored. Many of the items on the list were either implemented or going to be implemented until the problem they addressed was resolved by other means. For example, the use of several constant speed thrust generation props in addition to smaller maneuver props was heavily considered until the arrival of new battery technologies allowed for a maneuverable vehicle with only four thrust/maneuver combination props. Though the main thrust producing props could still extend the endurance and maneuverability of the vehicle, the cost savings of utilizing a simpler four prop design was significant. As an example of a brainstorming topic that was realized in the final version of the vehicle, the wire-tensioned structure proved to be a beneficial idea that saved significant structure weight while producing a vehicle body stiffness well beyond that achieved by previous generation structure designs [5].

Ultimately we decided to stick close to previous designs, utilizing four electric motors driven by an as-yet unselected battery technology. These four motors would drive four fixed-pitch propellers. These props would provide the thrust necessary to counter gravity while also providing sufficient residual thrust for control of roll and pitch (and subsequently forward and lateral velocity), yaw, and vertical velocity. The nature of the vehicle control was simple, yet clever. Of the four props, two would turn in the clockwise direction while two would turn in the counterclockwise direction. The prop type would match this rotation direction so that both are producing their most efficient thrust while rotating in the expected direction. The similarly-rotating props would be located opposite one another. Figure 2-1: Prop Rotation Direction provides a layout of the four props and their rotation direction.

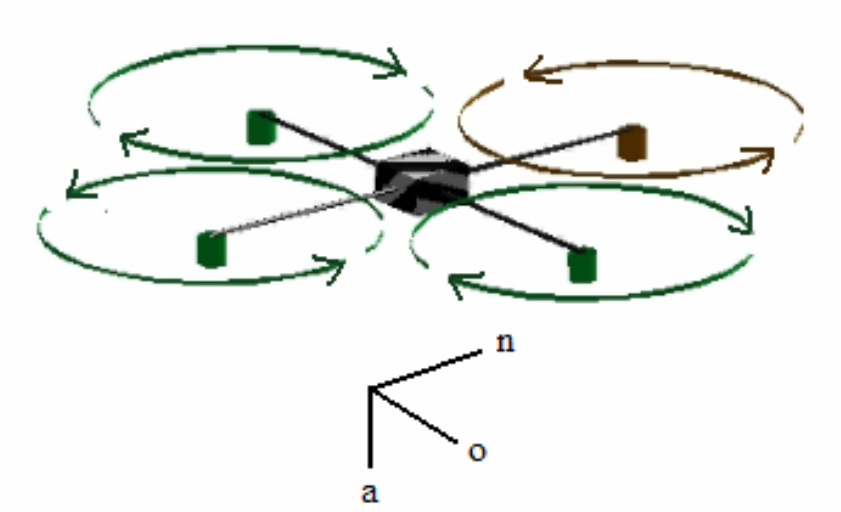


Figure 2-1: Prop Rotation Direction

At hover, all four props would be spinning at the same speed, producing zero net torque about any body axis and zero net force on the vehicle once gravity was taken into account. In order to roll or pitch the vehicle, one prop would speed up while its opposite partner in rotation direction would slow down. The result was a roll or pitch moment caused by the difference in thrust produced between the two props. However, since both props changing speed, one increasing while the other decreases, share a rotation direction, the reduction in drag on one prop is countered by the increase in drag on the other prop, resulting in no net torque about the yaw axis of the vehicle. Similarly, since one prop has sped up while the other slowed down, the net thrust has not changed maintaining zero net force vertically. When the vehicle needs to yaw, a pair of similarly-rotating props are sped up while the pair of props rotating in the opposite direction are slowed down. Since similarly rotating props are located across from one another, speeding up or slowing down both produces no roll or pitch body moment. Since two have sped up while two slow down, the net thrust also remains

constant, producing no change in vertical acceleration. However, since the two props spinning faster share the same rotation direction, the prop drag produces a nonzero net yaw torque. The last vehicle degree of freedom controlled, vertical acceleration, is the simplest of the four and is controlled merely by speeding up or slowing down all four props equally. Table 2-1: Prop Control Scheme depicts a summary of the vehicle control scheme.

Table 2-1: Prop Control Scheme

	Δ Prop 1	Δ Prop 2	Δ Prop 3	Δ Prop 4
ΔRoll+	+	0	-	0
ΔPitch+	0	-	0	+
ΔYaw+	+	-	+	-
ΔA- (up)	+	+	+	+

Note that the four prop layout is a minimal and efficient design. Unlike a helicopter's inefficient use of a tail rotor purely for cancellation of main rotor yaw torque, all power available to the AFV is utilized in thrust production or overcoming its associated propeller drag forces. Though the helicopter arguably reclaims some of this lost power through the efficiency of the large diameter main rotor, the four prop design also lends itself to a simple control scheme. As noted above the vehicle has direct control over four of its degrees of freedom (the remaining two, X and Y position, being coupled to Roll and Pitch because of the component of thrust acting along these axes when the vehicle is banked) through the simple speed control of the four motors driving the four props. The simple motor speed control employed eliminates the mechanical complexity of helicopter rotor blade pitch control linkages. In addition, the use of fixed pitch propellers provides some further gain in efficiency due to the asymmetric prop blade design. Helicopter blades, on the other hand, have predominantly symmetric cross sections due to some details of variable pitch control.

The structure settled upon would consist of a series of struts extending from the vehicle center to each motor/prop module. Four stiffening wires would be affixed to the end of each strut. These wires would travel to the end of a vertical strut extending above and below the vehicle center and to each of the strut ends adjacent to the current strut. The wires could provide significant stiffening of the struts without adding significant weight due to the high Young's modulus of steel. The diameter of the wire, the height of the vertical center struts (and thus the angle of the wires affixed to the strut ends), and the thickness of the struts themselves could all be varied as design parameters.

The details of specific component selection and design can be found in the following chapter. Information about components specifically related to the EE side of the design effort (eg, the Inertial Measurement Unit) can be found in the 2003 electronics documentation [1].

CHAPTER 3:
ANALYSIS AND COMPONENT-LEVEL DESIGN AND SELECTION

Once a general vehicle conceptual design was settled upon, the team needed to make specific choices regarding component selection and design. The mechanical aspects of vehicle design could be divided into the design of the battery/motor/gearing/prop combination (thrust-producing module) and the design of the overall structure. The design scale was driven by a preliminary electronics weight estimate. The estimate of 1.8kg heavily drove the remainder of design as this value coupled with structure weight determined the effective "payload" that the four thrust-producing modules would have to lift in addition to their own weight. The thrust modules needed to be able to each lift their own weight, one quarter of the expected electronics weight, and one quarter of the structure weight while supplying a residual amount of thrust sufficient for hover stability and maneuverability. Based on work with previous versions of the vehicle, it was decided that the residual thrust should fall in the range of 0.15 – 0.3 g excess thrust beyond vehicle weight. If higher values were obtainable, these were obviously preferable.

Much of the design effort fell into the development of a proper combination of batteries, motor, gearing reduction, and propeller to produce an effective thrust producing unit. Though the four components of the thrust unit were strongly coupled, variability in choices about gear ratio, number of cells to use in a battery pack, and prop diameter and pitch enabled a fair amount of latitude in treating these four categories somewhat independently. Minor tweaks could then be made to bring them all together as an efficient system. With this freedom, we worked to select what was

considered the best option available in each of the four categories. The specific analyses necessary to finalize the design could then be performed.

Motors

There were several options available in motor selection. Not only were there numerous brands to select from, but motors seemed to fall into three general categories. These categories included commercial brushed motors, commercial brushless motors, and hobby supplier brushless motors. Hobby supplier brushed motors were also available, but in limited sizes. The principal concern in motor selection was power output versus motor weight, as any weight added would require power expenditure to keep it aloft, with a secondary desire for reliable and long-term performance. Additionally, motors with an onboard encoder for brushed motors or Hall Effect sensors for brushless motors were ideal for ease of local motor speed control and brushless commutation. Finally, the motor performance level needed to fall within the desired range of motor performance. Neither a tiny nor an oversized motor could satisfy the requirements regardless of how efficient they might be.

Upon examination of motor specifications, it quickly became evident that brushless motors were able to provide much higher power to weight ratios than their brushed companions. This benefit seemed to be at the expense of easily available onboard sensing and simplicity of driving circuitry. While brushed motors need only a simple DC voltage applied to their terminals, brushless motor driving circuitry can be very complicated due to the complexities inherent in driving their internal torque-producing coils properly. The dramatic improvement in power to weight ratio of brushless motors as compared to brushed motors (the brushless producing as much as double the

power for some brands compared) was judged sufficient to work around the difficulties surrounding brushless motor commutation and sensing.

Having settled on brushless motors, it was still necessary to decide between hobby and professional-grade brushless motors. The hobby motors, built specifically for flight applications in some cases, seemed to outmatch the professional motors in power to weight ratio. Some of this was certainly due to the lightweight, less robust construction of the hobby motors, though there was also some slight ambiguity in exactly how to interpret the rather liberal hobby motor power ratings. While professional grade motors were rated conservatively for high duty cycle operation for indefinite periods of time, the hobby brushless motor specs were almost certainly intended for brief periods of high power output with a large degree of convective cooling. Separation of liberal power ratings from true design advantages achieved through design specifically for flight (such as the use of lighter weight metals in motor cans) proved difficult. However, when some of the best performing professional brushless motors were awarded a 50% power bonus in anticipation of potentially overdriving them, they still only just matched the specs provided by hobby motor manufacturers.

In addition to the power to weight ratio differences, the hobby brushless motors seemed to have fewer options available for high-resolution onboard sensors as compared to the professional motors. This lack of resolution was likely due to the same characteristic that aided in higher power ratings. The hobby motors use a few large diameter wire motor coils rather than the much higher number of windings found in commercial motors. This difference was easily observable in the significant cogging torque present in the hobby motors. Ultimately, once again, it was decided

that the benefits of the hobby brushless motors were significant and the primary disadvantage, the low resolution onboard sensors, could be worked around with the use of an external encoder geared to the motor drive shaft or the propeller shaft.

Initially the Astro 020 motor was selected. It had what was considered to be sufficient power ratings for minimal weight and the supplier was willing to provide us with custom versions (actually discontinued models) with Hall Effect sensors. The Astro motors also came with compact lightweight motor control boards, making them an attractive choice. After testing, however, it was decided that the motor speed control supplied by the Astro controllers was not of sufficient resolution and consistency to suit our needs. We chose instead to design custom motor control circuitry. This control circuitry allowed the motor to accept RPM commands and perform local feedback control on the motor/prop combination using the external encoder as a feedback sensor. The Hall Effect sensor was used primarily for ease of driving the motor coils.

Extensive work with the Astro 020 motors produced repeated motor failures. Examination of one failed motor revealed that, partly due to a somewhat questionable rotor design, the permanent magnets attached to the motor rotors were coming loose and jamming the rotors. We continued to encounter failures even after supplementary cooling fins were added to the motors and limits were placed on commanded motor torque. When the supplier repeatedly failed to deliver replacement orders in a timely fashion, we decided that a new motor supplier needed to be found. MaxCim Motors advertised a motor that looked promising. Discussions with the owner of the company revealed that the MaxCim motor possessed a higher resolution Hall Effect sensor, a significantly more robust design, significantly higher power ratings, and only slightly

higher weight than the Astro 020. The weight increase, the only perceived disadvantage, proved especially insignificant compared to the anticipated total vehicle weight. The owner also promised, and delivered, the MaxCim motors with a short turnaround time. The new motors proved extremely reliable and are currently the motors used onboard the AFV. Extensive use of the new motors produced no difficulties or failures. Specific motor characteristics can be found in APPENDIX B: COMPONENT CHARACTERISTICS.

Props

The initial search for propellers for the vehicle was confined to propellers commercially available in both pusher and tractor configurations (two of each were necessary for the vehicle control method employed). While custom props had been discussed, the cost would be large and the team lacked individuals with any knowledge of propeller design. Instead we looked into finding the best available props for efficiency in hover from the available list of props. This entailed both research into the performance of props and the purchase of an assortment of available propellers for testing. General web research and experimentation both quickly revealed that there were certain prop characteristics best suited for our application. Since hover performance was critical, the best props in forward flight applications were not optimal for use on the AFV. General web research (hobbyist forums, etc) revealed that the most efficient prop, as defined as static thrust over input power, was a large diameter, minimal bladed low pitch prop. An upper limit on prop diameter was imposed by both the weight of the prop itself and the gearing necessary to make a reasonably sized motor turn a prop of that size. A lower limit on the number of blades was imposed via simple balance concerns – two is a practical minimum, though there was mention of the use of counterbalanced single bladed propellers in endurance

competitions. A boundary on the pitch of the prop was imposed by the nature of the inefficiency of higher pitched props. In higher pitched props designed for forward flight applications, the pitch is so large that at zero forward speed the blade is significantly stalled, yielding very inefficient thrust production. As the prop moves forward at an increasing rate, the effective pitch angle of the prop in the oncoming flow is reduced until, at one point, flow once again becomes attached and the prop performs close to its optimum. Onboard the AFV, the prop will be operating primarily in zero forward speed conditions as the vehicle will predominantly be operating in hover. The best prop performance can therefore be achieved by selecting a prop that will produce fully attached flow at zero forward velocity. The critical range appeared to be a 10 - 14 degree attack angle at 0.75 chord length to ensure fully attached flow under zero free stream velocity conditions. Higher angles will produce stalled blades while lower angles will suffer from higher drag to thrust ratios than this ideal range.

The optimum choice at this point was clearly a low pitch, large diameter, two-bladed prop. Investigation revealed a general consensus among the hobbyist community that APC propellers excelled in the efficiency, weight, and stiffness categories important to propeller performance. Designs based on their props available in both pusher and tractor configurations yielded a workable vehicle solution with sufficient residual thrust for control, though it would have required the addition of a few main thrust producing props. This configuration was necessary due to the inefficiency associated with the fact that the props were above the optimum 10 – 14 degree angle of attack condition. Additional searching revealed an 18x6 (diameter x pitch, inches) “3D fun fly” propeller offered by APC. Though this prop was only available in tractor configuration, inquires revealed that APC was willing to provide a custom-made propeller for a reasonable fee. The fact that the pusher version would merely be a

mirror image of the existing prop removed the burden of custom prop design from our shoulders. The use of these new props coupled with the LiPoly battery technology that appeared midway through the project provided a tremendous boost to anticipated vehicle endurance and maneuverability and enabled us to scale back to a four-prop vehicle. The cost savings from only purchasing four motors, controllers, and battery packs rather than eight almost paid for the price of the custom propeller, and certainly would were multiple vehicles to be produced in the future. The 18x6 was settled upon for use in the final vehicle.

Note: Attempts to form a vehicle design around the props revealed that there was no simple way to perform a proper propeller analysis. So many parameters depended on specific details of prop design that analyses eventually relied upon a few freeware prop analysis programs, namely ThrustHP and PropSelector, and data from the manufacturer to make initial selections. Due to approximations and inaccuracies in these programs, though, they could not be relied upon for detailed design work. Later design, such as gear ratio and battery configuration selection, was done instead with the information obtained experimentally from the props ordered. Because the custom prop ordered was simply a mirror image of an available off-the-shelf design, we were able to conduct testing and identification of prop thrust and drag coefficients before the expense of custom prop production was invested. This identification proved valuable as even the data provided by the manufacturer of the props did not match with the values obtained in testing. It was only with the experimental data from testing of the actual prop that we were able to confidently move forward with vehicle design. Values obtained from testing can be found in APPENDIX B: COMPONENT CHARACTERISTICS.

Gearing

Due to the use of a large diameter prop that requires a fair amount of torque at a relatively low speed with a brushless motor, which tends to operate at high speeds and low torques, it was obvious that a relatively high gear reduction would be necessary. Unfortunately, the selection of off-the-shelf gearing packages was limited primarily to 3.5:1 and lower reductions. The decision was therefore made to build a custom gearbox with as close to the ideal reduction as was possible. Analysis revealed that the ideal gear ratio for the size of prop considered was significantly higher than a 7:1 reduction. However, after a reduction of 6.5:1 or so, there was diminished return for increased gearing. Given these results and available pulley sizes, the decision was made to go with a 6.7:1 reduction. This reduction was settled upon due to the additional restriction that the gearing reduction should be kept to a single stage in order to both maximize gearing efficiency and avoid the weight and expense of adding additional stages.

Unfortunately, a general rule of thumb regarding gearing is that no stage should provide greater than a 6:1 reduction in order to maintain a proper gear mesh. One proposed solution was the use of pulleys and belts rather than spur gears. Initially the option was suggested in order to allow for possible changing of gear ratios (by careful center to center distance, pulley size, and belt length selection) without making changes to the pulley box hardware. However, upon testing a version with a pulley belt reduction, we found that the pulley's appeared to operate with higher efficiency and much less noise than the high-speed spur gear equivalent. Testing further revealed that if the belt was kept sufficiently short with reasonable tension, the system could support high frequency control effort changes without chatter issues associated with stretching of the belt encountered for lower tension arrangements. In addition, it

was possible to trade off some center-to-center pulley distance and belt length for a better mesh between the belt and the smaller of the two pulleys. This trick allowed for a 100:15 tooth ratio, or 6.7:1 reduction. This brought the reduction very close to the best practical reduction ratio.

Note: the specific pulleys selected both have set screw hubs rather than the available Fairloc hubs. Fairloc hub pulleys were initially purchased, but due to the press fit join between the hub and the pulley there were several instances of pulley failure as the press fit came apart. Once the hub had vibrated loose the pulley itself could spin freely preventing any torque transmission. The set screw pulleys resolved this problem as the set screw passes through both the pulley material and the hub, acting essentially as a pin to prevent relative motion of the two parts. Please see APPENDIX B: COMPONENT CHARACTERISTICS for supplier information and details on the specific pulleys and belts used.

Batteries

The first step in battery selection was consideration of various available battery technologies. NiMH battery cells appeared to be the best in power density (power to weight ratio) while still being able to handle current drain at the rates anticipate for the motors (~25 amps). In particular, the best cell seemed to be the newer NiMH technologies from Panasonic. The HHR300SCP cell could handle a 20 amp drain rate for the targeted endurance, 5 – 10 minutes. The team purchased several packs and conducted extensive testing. This testing revealed large variability in performance of individual cells, reflected in abrupt but short drops in voltage near the end of the drain of the battery pack. While some cells could provide their current for nearly the entire rated capacity, other cells quit much earlier. Researching battery technologies did

reveal one means of increasing cell performance. The retailer who sold the NiMH cells primarily to RC hobbyists used a technique called cell "zapping" which entails discharging a large bank of high voltage capacitors through each cell. What little information available on this process suggested that the high voltage pulse spot-welds the internal connections of the batteries, thus reducing their internal resistance. Testing confirmed a significant (10%) improvement in voltage at a given drain rate as compared to unzapped cells. Unfortunately the lack of cell performance consistency still existed.

As this testing was going on, a few battery manufacturers were just beginning to market a new battery technology with impressive power to weight ratios. Some of the latest Lithium Polymer cells were able to handle large current drain rates (on the order of 7 – 10 A per cell versus the minimal .1A or so drain rates of previous LiPoly cells), but were typically three times the energy density of the best NiMH cells available. As batteries were the principle factor determining the weight of the vehicle, both directly through their own weight and indirectly through the motors and structure required to lift this weight, the savings accorded by moving to the LiPoly cells enabled previously unexpected performance. The LiPoly batteries not only enabled maneuverability on the order of 0.9 g excess above hover thrust, but also stretched the potential endurance to 15 – 25 minutes. In addition to these weight benefits, the cells themselves were much more homogenous in performance, providing consistent and reliable performance from cell to cell as compared to the NiMH cells studied. This consistency also allowed for the placement of cells in parallel to maximize battery pack performance and flexibility. The only disadvantages perceived in use of the LiPoly cells were limited early availability, which was remedied through contact with a distributor capable of supplying our relatively large demand, and cost. For

comparable total power provided, the LiPoly cells cost roughly 60% more than the NiMH technology cells. However, this cost was judged well worth the value of a lighter power source (and correspondingly scaled down vehicle) and more reliable, repeatable performance. The specific layout of the battery pack (number of cells in series/parallel) was left as a final design parameter to be selected as part of the integration of props, gearing, motors, and batteries into a single thrust producing module. Please see APPENDIX B: COMPONENT CHARACTERISTICS for discharge plots, supplier information, model number, and further details on the battery cells used.

Thrust Module

As mentioned previously, the best options available in propellers, motors, gearing, and batteries were selected. However, there was a good deal of matching done in this process. The gearing served to match the motor torque-speed curve as well as was possible to the prop drag-rpm curve. Insufficient gearing would cause the system to waste power as the motor became torque-limited below its max efficiency point, and an excess in gearing could limit the maximum speed of the prop, and thus the maximum achievable thrust for a selected prop. Similarly, once the motor, gearing, and prop was selected, the battery cells, available nominally in 3.6V 1200mAh units, had to be assembled in parallel and series to create the proper voltage/current source to match the rest of the thrust system. In some sense, gearing and current handling capability of the batteries were coupled. A large number of batteries in parallel would allow large current to flow, which would in turn allow large torque to be produced in the motor. This large torque could be passed through less gearing to turn a prop. However, keeping the weight of the batteries constant, more cells in parallel means that the total voltage of the pack would be lower, limiting the maximum speed of the

motor. However, since less gearing is used in this scenario, the maximum speed of the prop may well come out to be roughly the same as in the higher voltage, higher geared case.

This situation only becomes more complicated with the addition of PWM for motor voltage control, its associated effects, and battery cell internal resistance. In order to get a good rough idea of the desired operating point, however, basic analyses can be performed by choosing a current draw and voltage. The gearing ratio is then selected to force the motor to operate at that point for a given desired prop speed. The batteries can then be selected to provide this current at the stated voltage. The equations governing this relationship follow.

For an applied voltage, V, and desired prop RPM, α/G, where G is the gear ratio, the torque produced by the motor is:

$$\tau_m = \frac{k_i}{R}\left(V - \alpha k_v\right) \tag{3-1}$$

where R, k_i and k_v are parameters defining the motor performance with units Ohm, Nm/Amp, and Volts/RPM, respectively. In order for the motor to remain at a given speed, the torque produced by the motor applied to the prop, $G*\tau_m$ must cancel the nominal drag on the prop, D.

$$\dot{\alpha} = \frac{G*\tau_m - D}{J_t} = 0 \tag{3-2}$$

$$G*\tau_m = D = k_d(\alpha / G)^2 \tag{3-3}$$

where k_d is the coefficient of drag of the prop and J_t is the adjusted mass moment of inertia of the prop and motor rotor. The above relationships can be used to get a good idea of maximum battery/motor/gearing/prop thrust performance by inserting in the maximum voltage and current draw of the battery pack. An estimate of endurance can be obtained by calculating the hover point of the system from the relationship "thrust = $k_t(\alpha/G)^2$," setting thrust equal to the weight of one quarter of the vehicle and solving for alpha. This alpha can be used to compute a motor current draw. When this current draw is compared against the capacity of the battery pack, a rough approximate of endurance can be obtained.

It should be noted, however, that this lower current draw is theoretically obtained by applying a lower voltage to the system. PWM, the method used to obtain this effective lower voltage, has its own effects on battery performance. A more accurate analysis was developed by Sean Breheny on the EE side of the project. His analysis was used for the final battery pack configuration and gearing selections reflected in the current AFV. Information about his analysis can be found in the 2003 electronics documentation [1]. The above simplified method was suitable for all but final value tuning, though, and was used to initially select the smaller range of prop, motor, battery combinations reflected in the previous sections' discussions. A simple spreadsheet was assembled to compare maximum thrust and an endurance estimate across configurations. The weight of the vehicle was calculated simply as the sum of some constant mass (EE components, structure, etc) and some mass that was scaled with the number of battery cells and motor and prop sizes. This spreadsheet, *motor analysis.xls*, can be found on the AFVMechECD in the Analysis&Simulation folder.

The final battery configuration settled upon was an array of 2 cells in parallel by 7 cells in series per motor. This configuration yielded roughly 15 minutes endurance with a maximum vertical total thrust of 0.79 g above hover. An additional approximately 8 minutes of endurance and 0.15 g vertical thrust can be obtained by substitution of the 2x7 cell array with a 4x8 cell array. The maneuverability of the vehicle does not increase substantially because though the residual thrust increases drastically with the addition of more batteries, so too does the weight of the vehicle. The disadvantages to moving to the larger packs are the substantially higher battery cost (more than double) and the increase in prop hover RPM. The latter would necessitate a stiffer structure to ensure that the range of prop operating frequencies does not overlap the natural frequency of vehicle structure flexible modes.

In addition to the design details associated with the core thrust producing components, an encoder was selected to provide the high resolution sensing of prop speed necessary for local feedback control of the prop. The encoder selected was a fairly standard 1024 CPR optical encoder provided by US Digital. For details on this encoder, please see APPENDIX B: COMPONENT CHARACTERISTICS.

Structure

The structure of the vehicle needed to satisfy multiple requirements. Most generally, it needed to hold the various parts of the vehicle together while remaining as lightweight as possible. Additionally, the structure needed to have a modal natural frequency sufficiently large to avoid resonance with vibrations caused by the rotation of the propellers. The most effective solution to the design requirements seemed to be a wire-stiffened structure. A structure consisting primarily of members in pure tension and compression could provide the most efficient use of material for structure stiffness

and strength. Thin-walled aluminum tubing was decided upon for the radial compression members since it could provide the minimal strength required of the compression members while maintaining the stiffness required to prevent buckling. Stainless steel was used for the tensioning wire for its superior stiffness to weight ratio. Because the wire is only loaded in tension, the cross section can be shaped almost arbitrarily, allowing for the use of compact and flexible stranded wire.

An additional benefit of the wire-stiffened structure design, beyond its efficient conversion of weight to stiffness, is the ability to change the stiffness of the vehicle easily. By substitution of the wire with a similar wire of larger or smaller diameter, the stiffness and weight of the vehicle can be changed should it be decided that the current size is insufficiently stiff or overly and unnecessarily heavy for a given operation range of the vehicle propellers.

In order to perform an analysis to determine the appropriate wire and compression member sizes, a combination of ANSYS finite element modeling and a MATLAB m file was used. The MATLAB file *fourpropsplotted.m* performs a simplified analysis of the structure by examining the displacement of the end of a compression member co-located with the motor/prop combination. The compression member is assumed to be held fixed in rotation and displacement at the end that meets the center of the vehicle. Similarly, the wires connected to the end of the compression member where the motor/prop combination is located are assumed to be held fixed at their other ends. This is not an entirely valid assumption as two of the four wires run to adjacent motor/prop assemblies at the end of adjacent compression members. However, for the purposes of simplification, it was assumed and the more complex potential modes were left to ANSYS analysis.

Having constructed the problem in this manner, the code then effectively displaces the motor/prop combination in each of its principal directions, namely radially (along the axis extending from the vehicle center through the motor/prop combination), tangentially, and vertically, and determines a spring constant as a combination of stiffness contributed by the wires and the compression member. This spring constant is combined with the mass lumped at the end of the compression member consisting of the motor/prop assembly to produce an estimate of the natural resonant frequency of the arm in the direction examined. The same method is applied to rotational displacement about each of these three directions. The output, then, is a list of six computed frequencies, all of which must be reasonably higher than the highest frequency of normal prop rotation. This would ensure that there was no adverse interaction between prop rotation and structure vibration.

The expected hover prop rotation rate was approximately 66Hz given the prop coefficient of thrust k_t, the final vehicle weight of 6.2 kg, and the relationship between prop RPM and thrust production. The absolute highest prop rotation rate was found to be 90Hz given the limitations of the battery packs. It was therefore decided that the minimum resonant mode of the vehicle must have a frequency greater than 100Hz. This may seem somewhat close to the upper range 90Hz value, but the vehicle would rarely be performing at this peak level and even then for only very brief spurts of time. In addition, the least-stiff mode of the vehicle turned out to be the torsional mode about the radial direction, which is the least likely mode to be excited from imbalances in the prop. In order to help stiffen the structure against this mode, the compression member ends with “wings” were added. These extensions result in larger restorative torque being generated by the circumferential wire in response to rotation of the motor/prop combination about the axis of the compression member.

In order to verify the validity of the MATLAB file analysis, an ANSYS finite element model (FEM), Structure.db, was constructed. Though the final vehicle design was not constructed explicitly in an FEM, cases compared between the ANSYS FEM and MATLAB suggested that the MATLAB code was in agreement on modal shapes and in fact slightly conservative in its computations of modal frequency as compared to the more accurate ANSYS model, lending validity to use of the much more flexible MATLAB code to do the iterative design work and final wire/compression member size determinations. The files for both methods of structure analysis can be found in APPENDIX E: ELECTRONIC CONTENT.

In the end, a combination of material availability and MATLAB results determined the member dimensions. 1/16” 19-strand SS wire and 3/8” OD 0.028” wall thickness aluminum tubing was selected. Though these two selections work well for the vehicle, future versions may consider more strands of a smaller diameter (to maintain roughly the same cross-sectional area and stiffness) for the wire to aid in routing and handling of the wire. Also, given the superior performance of the final thrust modules, weight became less of a constraint on vehicle performance. Considering the relatively small percentage of total vehicle weight that structure comprises, thicker walled aluminum tubing could be considered. Though sufficient for the task, the thin-walled tubing is somewhat sensitive to buckling if loaded incorrectly. Please see Table B-4: Parts and Components Information in APPENDIX B: COMPONENT CHARACTERISTICS for supplier information and part numbers for the structure components.

CHAPTER 4:

FABRICATION, ASSEMBLY, HARDWARE TESTING, AND RE-DESIGN

Once all components were selected and all major fabricated parts were designed, what remained was the fabrication, assembly, hardware testing, and design iteration of the various vehicle subsystems. Except for the specific comments made below, fabrication and assembly is left to the skill and experience of the individual.

Pro/E Model

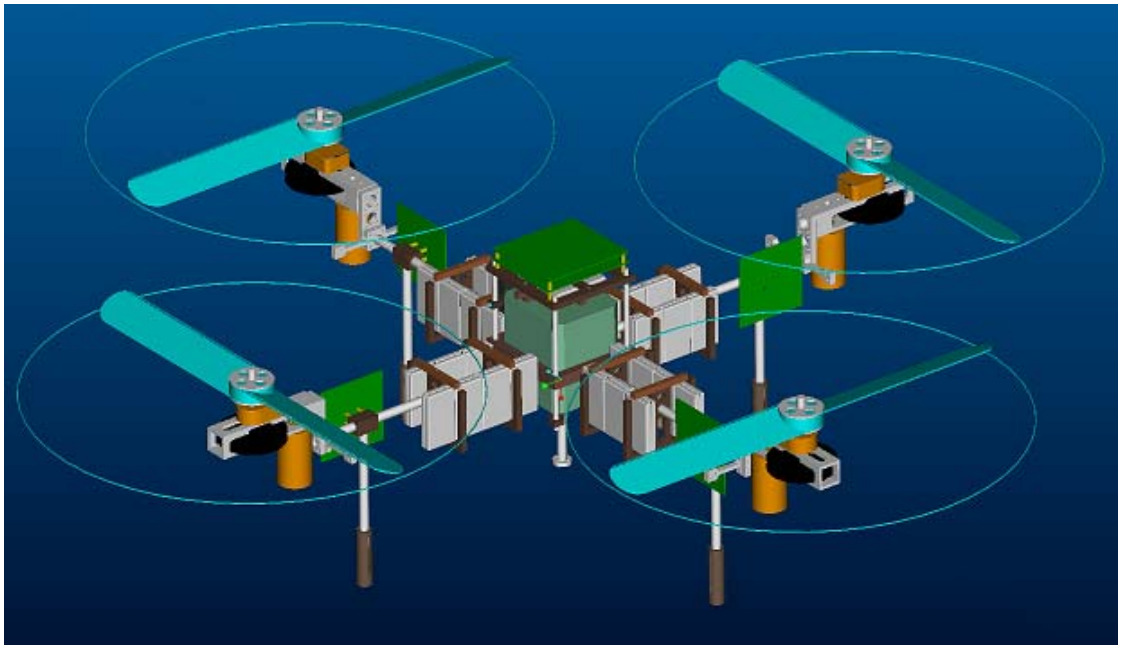


Figure 4-1: Pro/E Model of Assembled Vehicle

In order to aid in fabrication and redesign of the vehicle, it was first modeled in its entirety in Pro/Engineer. The Pro/E model can be found in the folder labeled ProE on the AFVMechECD. In order to access the model, simply specify this directory as the working directory. *AFVParts.xls*, also located on this cd at Documentation\2003-

2004\Designof4RotHoverVehicle\Part Data, contains a complete list of all final version parts present on the vehicle in the worksheet labeled *ProEparts*. The Excel file also contains a full list of all non-fabricated mechanical parts (eg, motors) along with supplier information in the worksheet named *Supplier&Stock info*. The ProEparts table is reproduced in Table F-7: Pro/E Files Information in Appendix F: Pro/E FILE INFORMATION AND MACHINING SPEC SHEETS. The Supplier&Stock info table is reproduced in part in Table B-4: Parts and Components Information in APPENDIX B: COMPONENT CHARACTERISTICS.

A few conventions were employed for simplicity in understanding and navigating the Pro/E model. All part file names begin with *prt_*. Similar convention applies to assemblies (*asm_*) and drawings (*drw_*). Drawings will be named to match their part with the exception of the file type prefix. Printouts of the drawings can be used for easy and accurate machining of replacement parts, should this become necessary. In all cases, part names should be reasonably intuitive, but when in doubt a part name can be easily obtained by clicking on it in its parent assembly.

All units are English, and are consistent with the Pro/E unit convention. Material density has been assigned to all parts to properly represent the mass of the finished part. For simple machined parts, this density is simply the density of the material they are machined from. For parts like the EE boards and motors, the density was obtained by dividing the final measured weight by the model volume. The unmodeled mass of the wire and turnbuckles are absorbed into a slightly higher density associated with the vehicle struts. The use of correct part densities allows the use of the Pro/E provided mass moment of inertia matrix for controller design.

In the fully assembled model, asm_bodycent.asm, all plastic parts constructed from Nylon 6/6 appear brown while all parts constructed from aluminum (6061 T6 or better alloy, except for small diameter threaded rod tubes) appear silver.

All screws used on the vehicle, excepting set screws and the IMU mounting screws but including the board mounting standoffs, are English 4-40 of varying lengths. These screws require a hole diameter of 0.089" for holes to be tapped, and 0.11" for through holes. Screw head types are specified in the Pro/E model, but should be apparent from application: pan heads where a wide or flat head is desired, deep socket heads where greater torque is desired and clearance allows.

The only remaining fastener type used are 5/64" rolled steel spring pins of varying length. These holes remain empty in the Pro/E model, but their location and function is obvious upon inspection of the model.

Assembly Comments

The majority of the vehicle assembly process is intuitive given the Pro/E model. There were, however, a few initial assembly tips that helped in the fabrication of a more robust vehicle.

- Tight tolerances are necessary in the fabrication of the pulley-box or the prop shafts. Any play either due to gaps between the shaft and the bearings, or between the bearings and the pulley-box will result in chatter and vibration when the prop is rotating. It is recommended that fine-grit sandpaper be used to do the final thousandth of an inch of material removal on the prop shafts to ensure a tight, almost press fit. The use of a sufficiently sharp bit with ample cutting fluid while machining the pulley box should be enough to ensure a tight

fit of the bearing into the pulley box. If absolutely necessary, a small bit of glue can be used to seat the bearing permanently in the box, though care must be taken that no glue makes its way into the bearing itself.

- Spring pin press fits should not be removed once assembled. Rather than permitting disassembly and reassembly of components by use of a loose spring pin press, the spring pins should be tightly pressed to ensure permanent assembly. Spring pins were used for their weight savings, not for their potential ability to be disassembled. In addition, parts joined by pins should be match drilled wherever possible.
- Care should be taken in the order in which components are pinned. All pins should be inserted via a press, and the order should be chosen such that the most difficult to assemble joints are accomplished first.

In addition to these one-time assembly details, there are a few procedures that should be kept in mind should any non-destructive assembly or disassembly become necessary. A detailed list of instructions is included in APPENDIX D: ASSEMBLY/DISASSEMBLY INSTRUCTIONS.

Some iteration was necessary to arrive at the final vehicle design. These iterations, including items such as design of shock-absorbing landing feet, implementation of a disassembly joint in the landing legs, and re-design and re-fabrication of the pulley-boxes to solve torsional flexibility issues, are all reflected in the final versions of the Pro/E model and the final vehicle itself.

There is still room for potential improvement of the AFV beyond those critical re-design steps already taken. While not necessary, the following improvements would

be desirable in either future versions of the vehicle or, given time, modifications of specific parts of the current vehicle:

- Lose weight where possible – the extension piece connecting the pulley box to the compression member end in particular is over-designed.
- Stiffer upper plate – the upper plate is currently constructed of plastic to preserve weight. While it is sufficient, it deforms noticeably when fully loaded. The addition of either a stiffening metal plate or the redesign of the plate would be beneficial as the deformation of this plate affects the EE board mounting.
- Changes to structure wire/tubing sizes – as mentioned in the structure section above, smaller diameter stranded wire or thicker walled aluminum tubing may be beneficial.
- Stiffer center strut mount – the current mount relies heavily on the strength provided by the steel IMU case. The mount can be re-machined from aluminum or stiffened by the addition of an aluminum insert in the event that the current IMU is no longer used.

Once the individual vehicle components were verified, it was necessary to assemble the entire vehicle for a whole-vehicle hardware test. In order to work with the assembled vehicle hardware, a landing platform that functioned also as a tethered power supply and vehicle constraint was constructed. In addition to this landing platform, a prop testing rig was also constructed for identification of prop parameters necessary in eventual control design and safe testing of an individual thrust module without concern for securing the entire vehicle.

Prop Testing Rig

Though analysis can answer many questions, ultimately testing confirmed the validity of thrust module analyses. A special testing mount was constructed for identification of prop parameters and testing of motor/pulley-box/prop combinations and local control. The prop testing rig can be used to perform thrust measurements by weighting or counter-weighting it appropriately. It can be converted to work similarly as a drag testing station by simply remounting the arm of the rig in the appropriate pivot hole. It can also be used as a secure and safe test bed for motor controller development and propeller parameter determination. It consists primarily of a mounting plate, which can be clamped to a convenient surface, a 1024 CPR digital encoder for arm angle information, and a boom arm that can, with the appropriate adapter installed, mount a full motor/pulley box/prop assembly. A vice applied at the pivot of the arm can lock the arm in place when the angular degree of freedom is not required of the rig. Figure 4-2: Prop Testing Rig depicts the testing rig fully assembled for thrust testing. Note the hole and channel cut for remounting the boom arm for torque/drag tests.

Figure 4-2: Prop Testing Rig

Landing Platform

In addition to the fabrication of the vehicle itself, it was decided that a special landing platform for the vehicle should be constructed. This platform had the initial purpose of providing a primarily open elevated surface for the vehicle to take off and land on in order to avoid the complications caused by propeller airflow interactions with the ground (ground effect) in takeoff and landing. The functionality of the platform was

expanded to include a leveling capability for the platform in order to help initialize the IMU and subtract the proper gravity vector. In addition, the platform was mounted on the top of a mobile cart which could transport the large lead acid battery supply employed in tethered power flight of the vehicle. The large variable resistor array used to help the lead acid battery source simulate the resistance of the onboard battery packs is also mounted conveniently on the cart. The top is removable for transportation and storage of the cart.

The final function of the cart was to constrain the vehicle during early hardware and controls tests. High tensile strength braided fishing line was used to tie the vehicle down to the platform. Depending on the specific test being conducted, various lengths of constraint tether could be played out. Even before any controller was developed for the vehicle, vehicle "hover" tests were performed by ramping up all four propellers to just above hover speed. This confirmed the absence of problems with excitation of vehicle flexible modes at any prop speed tested. In addition, tests of this nature helped test proper interaction of various system components, such as the communication between the main EE boards and the individual motor control boards.

Later testing of hover controllers utilized the constraint of the platform and fishing line to prevent the vehicle from flipping or allowing contact between a prop and any nearby objects, including the platform itself. This was a benefit particularly when controllers that turned out to be unstable were tested, though excessive constraint prevented proper knowledge of the effectiveness of an apparently stable controller because of the nonlinear interaction between the constraints and the vehicle. Figure 4-3: Landing Platform depicts the landing platform with all accessories mounted.

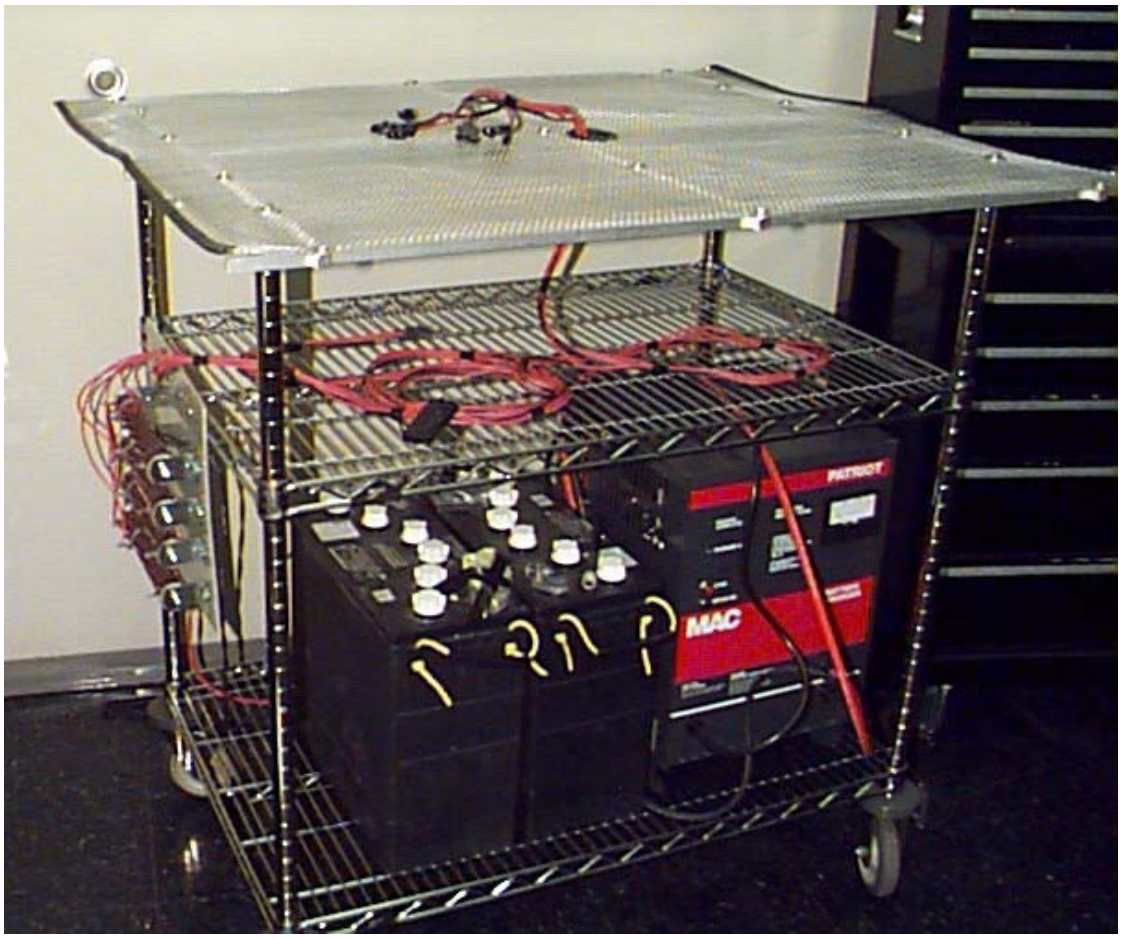


Figure 4-3: Landing Platform

Vehicle Testing

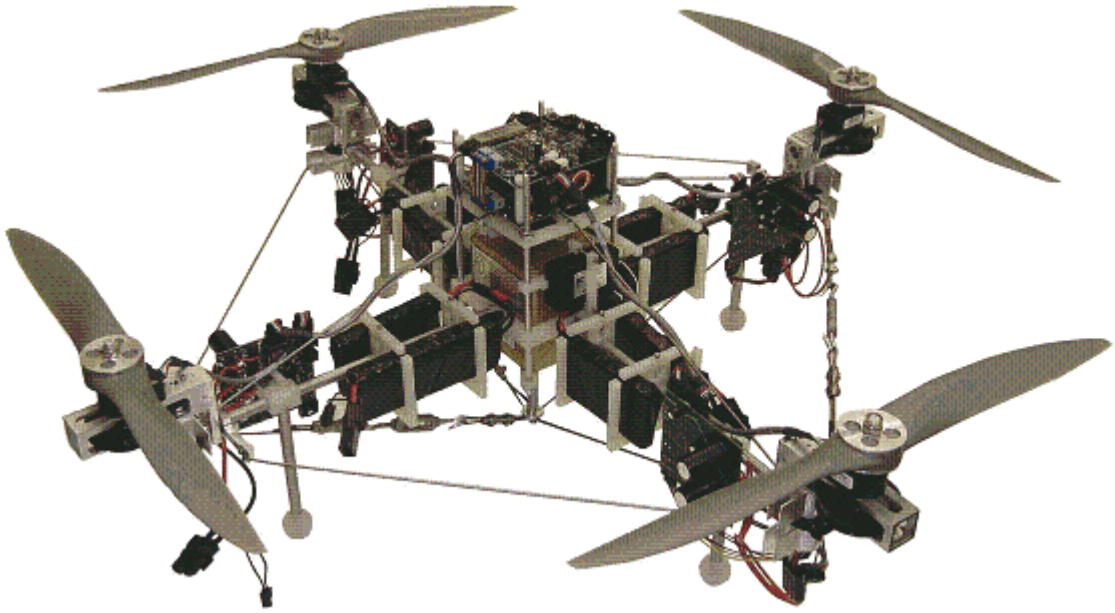


Figure 4-4: Fully Assembled AFV

Figure 4-4: Fully Assembled AFV depicts the fully assembled vehicle. Once the various vehicle subsystems were verified, the vehicle only needed a simple control and inertial navigation system (INS) before we could commence hover tests. The simple INS was developed by Sean Breheny. It consisted primarily of low pass and high pass filtering of the IMU data in an attempt to remove noise and constant bias from the measurements. The measurements were then rotated into the global coordinate system and integrated in order to keep track of global state. For further details on the INS used, please consult either the 2003 electronics documentation or *currfilterest.m* in APPENDIX E: ELECTRONIC CONTENT [1]. This initial INS provided somewhat inaccurate state information subject to drift. However, when combined with a human contribution in the form of velocity commands sent via a standard remote control transmitter/receiver, the INS performed suitably well for simple hover tests.

The controller utilized was a simple gain matrix. This gain matrix multiplied the current best state estimate in order to calculate four commanded thrust values for the four propellers. These thrust values were passed through a nonlinear transformation to obtain four propeller RPM commands which were then fed to the four local prop control loops. The gain matrix was formed as a simple combination of gains based on linearized decoupled dynamics. For example, a positive x velocity error would multiply a single gain. The resulting values would then be added and subtracted to the two appropriate props to cause the vehicle to bank back towards the negative x direction. The same was done for the other velocities, the Euler angles, and the Euler angular rates. The commands for each prop from each gain multiplication would then all be simply added together to form the commanded thrust for that prop. These gains were predominantly tuned by hand with guidance from propeller commanded thrust saturation values and expected disturbance magnitudes. The bulk of the tuning was done in an early version of the vehicle simulation, to be discussed in the following chapter.

Once the INS and control algorithms were in place and debugged, we were able to perform our first controlled flight tests. The vehicle was kept on a fairly short tether during these initial tests in order to prevent damage to either the vehicle or its surroundings. The vehicle did have enough play, however, to provide us with feedback on typical prop RPM excursions from hover RPM in response to natural disturbances given the controller currently loaded. In addition, we were able to take actual in-flight IMU data in order to improve upon the accuracy of the simulation of sensor noise used in tuning the controller. After only a few iterations of control gain matrices, the vehicle demonstrated very stable hover. Noise in commanded prop speeds was minimal, suggesting that the gains were not excessively large. Despite this,

though, the vehicle responded quite strongly to attempts to disturb the vehicle. While the vehicle, as a product of the design, is unable to directly resist disturbances in the plane of the propellers, the vehicle was quick to bank in opposition of forces applied in this plane. In addition, it was extremely difficult to disturb the vehicle in any of its angular degrees of freedom. This was primarily due to the large gains assigned to these degrees of freedom because of their importance both to physical vehicle stability and to the stability of the decoupled linearized controller. Finally, while the short constraint tethers prevented a truly unhindered view of controller performance and vehicle stability, there were extended periods of time where the velocity error was sufficiently small to allow the vehicle to hover in place, constraints slack. During these periods the vehicle remained extremely still without any human intervention either directly via forces applied to the vehicle or via the wireless RC link. A video of one of the hover tests performed, *ActualHoverTest.avi,* is available in the 2003 Documentation folder on the AFVMechECD. Though this video was not of the latest, most stable flights, it does show a large degree of stability in hover.

The final vehicle weighed approximately 6.22 kg. During the simple hover tests performed we were able to verify parameters such as the hover prop RPM and typical control deviations from this value. As it turns out, we actually underestimated the coefficient of thrust of the prop slightly. The result was a vehicle that hovered at slightly lower power consumption than anticipated. Given this information, the actual vehicle likely would outperform the predicted maneuverability and endurance. These specs were not tested, though, as all initial hover tests were performed with the power tether for simplicity. Verification of vehicle endurance and other predicted performance specs were left to later, less constrained flight tests.

Unfortunately before we were able to perform less constrained flight tests, the AFV suffered a crash and the IMU was damaged. As the IMU was the most expensive component, our only option was to send it off for repair. The repair bill quoted was much higher than expected, and at this time the team decided to consider lower cost alternatives to the high end IMU used. The extensive delivery time required for a new unit unfortunately meant that the actual integration of a new IMU would extend beyond the scope of the current project phase. However, armed with the data collected from flight tests and in anticipation of the new IMU, extensive work was done on the development of a more accurate simulation, a more complex filtering scheme, and more straightforward control. This work is detailed in the following two chapters.

CHAPTER 5:
SIMULATION DEVELOPMENT AND VERIFICATION

In preparation for vehicle hover tests and in order to aid in development and tuning of the vehicle control system, we opted to work on the development of a simulation of the full vehicle nonlinear dynamics. Early versions of the simulation contained the nonlinear dynamics model with simple white process noise driving the system and white additive sensor noise corrupting the true inertial measurements. This model was sufficient to tune the simple control system used on board the vehicle. Unfortunately, because of the simplistic nature of the noise simulation, the controller was only truly representative of the real system at the beginning of flight tests when accumulated state estimation errors were still small. Both in order to more accurately model the true vehicle and sensor dynamics and in anticipation of future more complicated controllers and vehicle maneuvers, the simulation was expanded significantly. With an accurate representation of both vehicle and sensor dynamics and the freedom to test a wide range of control algorithms, the simulation would become a valuable tool in future project development.

As mentioned previously, in order to aid in controller and estimator design, a full nonlinear dynamics model of the AFV was developed. This model was combined with an assortment of other model components and integrated into a Simulink model, *ThreeDAFVsimworkingvelocity.mdl*. The model in its current form performs global state feedback control. The Simulink model contains force and torque disturbances, sensor bias and noise, a linearized hover feedback controller, a nonlinear state estimator, and plotting windows for various state and performance comparison variables. The model is capable of producing a text file with a linear gain matrix for

use onboard the actual vehicle. It is also capable of producing an AVI video visualization of the AFV given the state simulation time history.

If the model is built and tuned carefully in order to closely match the true system, the model can be used as an effective tool to design, tune, and test controllers and state estimators. Discussion of control and estimation logic and tuning can be found in CHAPTER 6: CONTROL AND ESTIMATION DEVELOPMENT FOR FUTURE IMPLEMENTATION. The major components of the model can be identified as follows:

Simulation Parameter File

The file *ThreeDAFVmodelconsts.m* contains an extensive list of parameters both entered and computed. This file must be run before every simulink model run. In addition to defining process and sensor noise levels the file is also responsible for providing vehicle parameters such as prop pitch, motor and prop constants, and geometry information. This information was obtained primarily through a combination of hardware testing and analyses performed in the solid Pro/E model of the vehicle. The degree of precision in machining coupled with the convenience of the tools available in Pro/E contributed in many ways to the accuracy of the final simulation. The file also defines estimator noise matrices and initializes the state estimate, its covariance matrices, and variables necessary for the running of the simpler filtering model used on the AFV. In addition to all this, the file is also responsible for computation of an LQR gain matrix based on the linearized state dynamics computed from the simplified state dynamics found in *ThreeDAFVstatedervThrust.m*. Most changes to the simulation are accomplished through changes to the parameters contained in this file.

Full Nonlinear AFV Dynamics

The full nonlinear AFV dynamics are derived in Appendix C: DERIVATION OF AFV DYNAMICS using Euler angles and global position variables. Note that the use of Euler angles for state result in a singularity at theta = 90 degrees. The derived dynamic model is implemented, among other places, within *ThreeDAFVstatedervNew.m* which is called every simulation time step by the Simulink model. In addition to providing the true state derivatives that Simulink uses to track the true vehicle state, this file also produces the local angular rates and accelerations that a perfect set of sensors would measure. This true measurement is fed to the next block within the model, the IMU dynamics.

IMU Bias/Noise Corruption

In order to accurately replicate typical sensor measurements on board the vehicle, the signal corruption block of the model was developed. This block corrupts the true vehicle acceleration and rotation rates with possible sensor rotation and displacement from the vehicle center of mass and a bias offset and white noise. The measurements are rotated in *IMUgeometry.m* according to the parameters in the constants file. Accelerations contributed by centrifugal force from vehicle rotation combined with accelerometer offset from the center of mass are added to the accelerometer measurements. Sensor bias is randomly generated at the start of a run according to turn-on sensor bias specs. The bias is then subject to random walk driven by white noise and bounded by feedback. The measurement models used in simulation and some discussion of the tuning of parameters such as the white noise power driving the bias random walk are included in APPENDIX C: DERIVATION OF AFV DYNAMICS. The corrupted measurements are then fed at the same rate as the true IMU produces measurements to the filtering block, which attempts to estimate bias

and cull accurate sensor readings from the noisy signals. It is important that the model is tuned carefully to match the bias magnitude and drift expected from the actual sensors. As in all aspects of the model, the more closely the sensor inaccuracies can match the true system, the better the results achieved when controllers and estimators are tuned on the simulated system.

State Estimation

In the true system sensor noise and bias values will not be known. The state estimator, run in *estimatestate.m*, is therefore blind to the true vehicle state tracked by the simulation. Instead the state estimation block attempts to, at the proper simulated estimation frequency, produce a best estimate of current vehicle state given input from both the IMU and a human operator, as discussed in the next chapter. In order to do this it also estimates all six sensor biases associated with the IMU accelerometers and rate gyros in addition to the state. The core of the estimator is a square root implementation of a Sigma Point filter. This filter is realized in *srspf.m*, generously provided by Professor Mark Campbell of the Cornell MAE department, and makes use of the nonlinear dynamics file *ThreeDAFVstatedervNoVolts.m* and its parent file *ffunmine.m*. The control effort inputs are the four prop RPM values and their rate of change, provided in the true system by the encoders located on the four prop shafts. The nonlinear measurement equations used by the filter include *IMUgeometry_f.m* and its parent file *hfunmine.m*. In the simulink model, the simpler filtering implemented on board the vehicle is run in parallel to the SR SPF in the file *currfilterest.m*. The Simulink model presents several plots for performance comparison between the two filtering methods. The effectiveness of the more advanced filtering method is discussed in more detail in the following chapter.

Hover Controller

Once the estimator has provided its best guess at current vehicle state, the state is multiplied by the linear gain matrix created in *ThreeDAFVmodelconsts.m*. The results of this multiplication are four prop commanded thrust values. These four thrust values are added to thrust values commanded by the human operator and are then passed through a nonlinear transformation, *thrusttorads.m*, to compute desired prop RPM values. These four RPM values are passed through PID blocks that simulate the four local motor controllers onboard the vehicle. The time constants of the simulated response to commanded changes in prop speed match closely with the values recorded in actual testing due to matching of the variable R in the motor dynamics to the best calculated value based on the same prop controller tests. The output of the simulated PID gain blocks are voltages which are then fed back into the full nonlinear dynamics model discussed above. The loop continues for the duration of the simulation.

Post-Simulation Processing

Once a Simulink simulation has been run with satisfactory results, the data loaded into the workspace can be processed to generate either a controller text file to be loaded onto the AFV or an AVI movie of the simulated AFV. Running *animate_afv.m* will produce the AVI movie given the simulation data in the Matlab workspace. Edits to the boundaries of the virtual camera for the simulation can be made in the same file. The AVI movie occasionally provides a visualization of AFV behavior that can convey much more information than the two dimensional plots of various individual vehicle states. The original versions of the animation files were written by Sean Breheny with help from Professor Raffaello D'Andrea. The files were modified for use with the current vehicle and simulation information.

Any additional model information should be either self-explanatory or covered in more depth within the actual code. All code discussed in this section can be found in APPENDIX E: ELECTRONIC CONTENT.

CHAPTER 6:
CONTROL AND ESTIMATION DEVELOPMENT FOR FUTURE IMPLEMENTATION ONBOARD THE VEHICLE

Once the hardware testing, vehicle assembly, and basic hover flight tests were complete, the work that remained to be done was the construction of more advanced INS and control algorithms for eventual use onboard the AFV. The initial attempt was rather simplistic in nature as, at the time, the team was primarily interested in simply achieving stable hover. The DSP utilized onboard, though powerful, had its processing limits. As such, the initial inertial navigation system consisted primarily of high-pass and low-pass filtering of the inertial sensor data in an attempt to remove sensor bias and noise from the measurements. The control scheme implemented was the simple decoupled controller discussed in previous chapters. While this system worked well, and even enabled some good hover tests, the system was still prone to inaccuracies and required a fair amount of hand tuning. Some of the inaccuracies stemmed from the fact that selection of filter corner frequencies resulted in either failure to fully filter out noise or bias or the filtering out of true vehicle motion measurements.

Due to an assortment of project developments, including the availability of certain new technologies and the drift inherent in the current system (though a human could trim the vehicle to keep it from drifting, this trim value would itself constantly increase due to the lack of estimation of the bias parameters themselves), some changes to the vehicle are anticipated including the use of a lower cost IMU and the integration of a more powerful onboard processor. It was also decided that both the effectiveness of a more complex filtering scheme and the lower limit for a less

accurate IMU would be evaluated in the existing simulation before making the desired changes to the system. Though the actual implementation of new filtering scheme and the lower cost IMU extends beyond the scope of this document, the changes were examined in detail in the simulation constructed.

Estimation

The primary function of the vehicle state estimator is to provide an up to date best guess at the current vehicle state given potentially noisy or biased measurements of some function of the state and the current control input. For the current iteration of the vehicle, the measurements are provided primarily by a combination of a six degree of freedom inertial measurement unit providing three axis angular rates and accelerations and a human observer, to be discussed in further detail below. The state to be estimated most convenient to the desired functionality of the vehicle and the anticipated control scheme is a combination of vehicle global position and velocity and Euler angles and their rates. In order to make use of the measurement equations derived in APPENDIX C: DERIVATION OF AFV DYNAMICS, the biases associated with the six IMU sensors are also estimated. The estimation loop is summarized in Figure 6-1 and is discussed in more detail below.

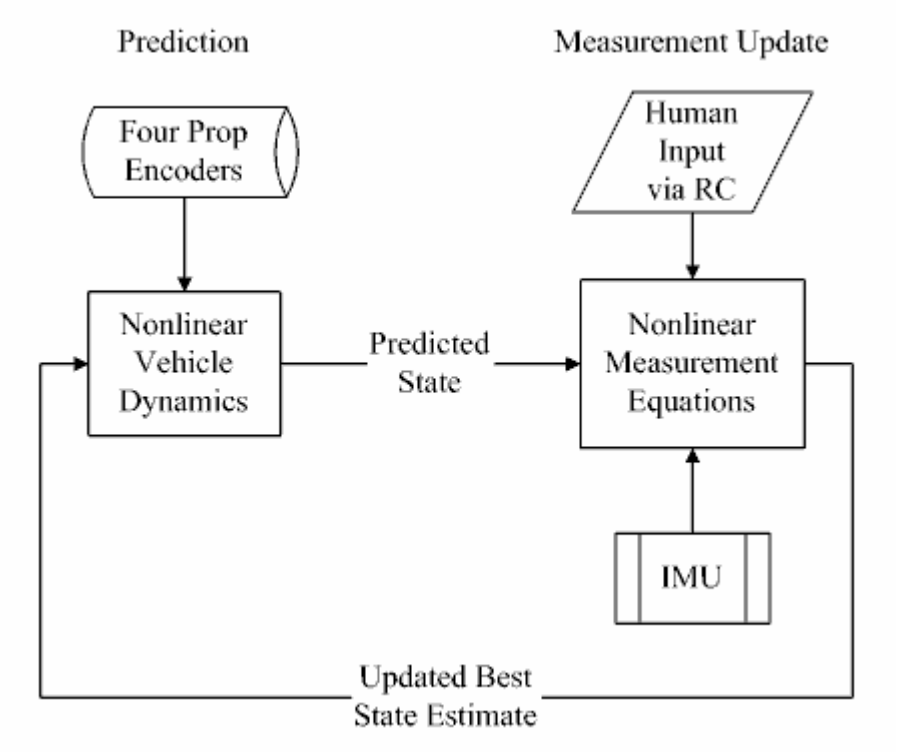


Figure 6-1: Estimation Loop

The vehicle dynamics are extremely nonlinear. This is largely due to the desire to control and track the vehicle in global coordinates while control effort (prop thrust and drag) is applied in local coordinates. In addition, inertial measurements and the bias parameters to be estimated are in the local frame. A nonlinear state estimator was therefore necessary. Initially an Extended Kalman Filter was designed to handle the estimation of both the state and the six IMU sensor bias parameters. This filter was cumbersome to implement, though, with extremely large and complex Jacobian terms because of the highly nonlinear dynamics. Evaluating these terms consumed a large amount of processor power and transcription of these matrices from derivation to implementation presented many opportunities for error. The next filter considered was a square root implementation of a Sigma Point Filter (SRSPF). The processing

time for this filter was at least comparable if not better than the EKF. In addition, it was much simpler to implement due to the direct use of the original dynamics equations rather than their complex Jacobian. Opportunity for error was thus significantly reduced. The filter also seemed to converge much more quickly than the EKF, and was much more robust to a range of tuned noise values. The measurement update step was also simplified dramatically with the use of the SRSPF. Because the accelerometers cannot all be located exactly at the vehicle center of mass, they measure a centrifugal force term associated with the angular rates as well. This lumps further nonlinearities upon the nonlinear dynamics (since the measurement is inertial, not of state directly) and further complicated the Jacobians used in the EKF.

The initial configuration of the AFV was to use the IMU as all available measurements. The calibration step, unfortunately, uses accelerometer information to initialize vehicle angle, preventing a proper calibration of IMU accelerometer bias errors. Though rate gyros also have bias issues, they are dramatically less significant both because of the identification of bias in the calibration step and because of the observability of angle through the vehicle dynamics. The accelerometer bias therefore would continually tell the vehicle that it was accelerating, resulting in continually increasing velocity error. In addition, velocity and position are not observable through an inertial sensor. Accumulated error therefore cannot be eliminated. The high-pass filtering utilized helped, but did not eliminate this problem. In early hover tests a human operator would send velocity commands to the AFV in order to cancel out the drift that would develop. By sending velocity commands roughly equal to the current velocity error, the AFV could be made to hover in place. This dependence on human operation was both undesirable and could even cause problems as the velocity command required grew ever larger. The first step to help the problem was the

implementation of the more complex filtering scheme, the SRSPF. The estimation of the accelerometer bias parameters alone decreased velocity error significantly. Figure 6-2: Accelerometer Bias Estimate Errors (m/s2) vs Time (s) and Figure 6-3: |Original Filtering| - |SR SPF| Velocity Estimate Errors (m/s) vs. Time (s) show both the success in estimation of the accelerometer bias parameters as well as the superior performance of the SRSPF as compared to the original filtering scheme. The test case for these figures, as all future figures, is for an initial roll angle of pi/15, or 12.5 degrees. An animation of this test case, *DocsTestCase.avi*, can be found in the 2003 Documentation folder on the AFVMechECD. Note that the parameter estimate error remains small in time despite the drifting true bias. Despite this improvement, however, the aforementioned problem with accumulated error was still an issue. Before an accurate estimate of the bias parameters could be established, some error would already have accumulated. In addition, the parameter estimation is not perfect due to the presence of a certain level of process noise.

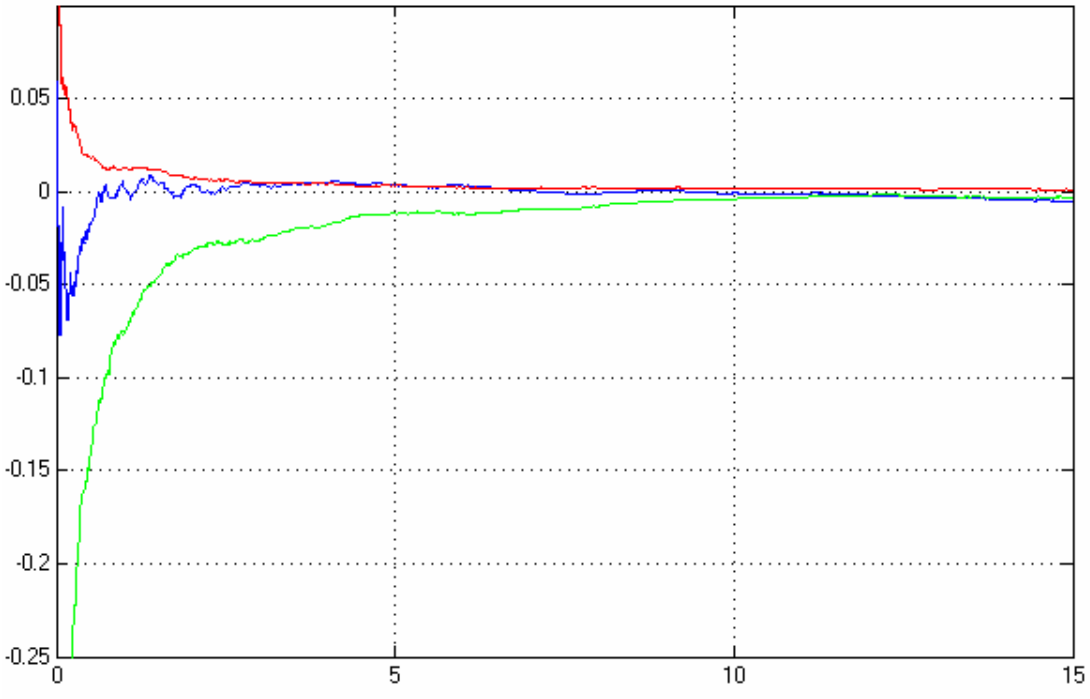


Figure 6-2: Accelerometer Bias Estimate Errors (m/s2) vs Time (s)

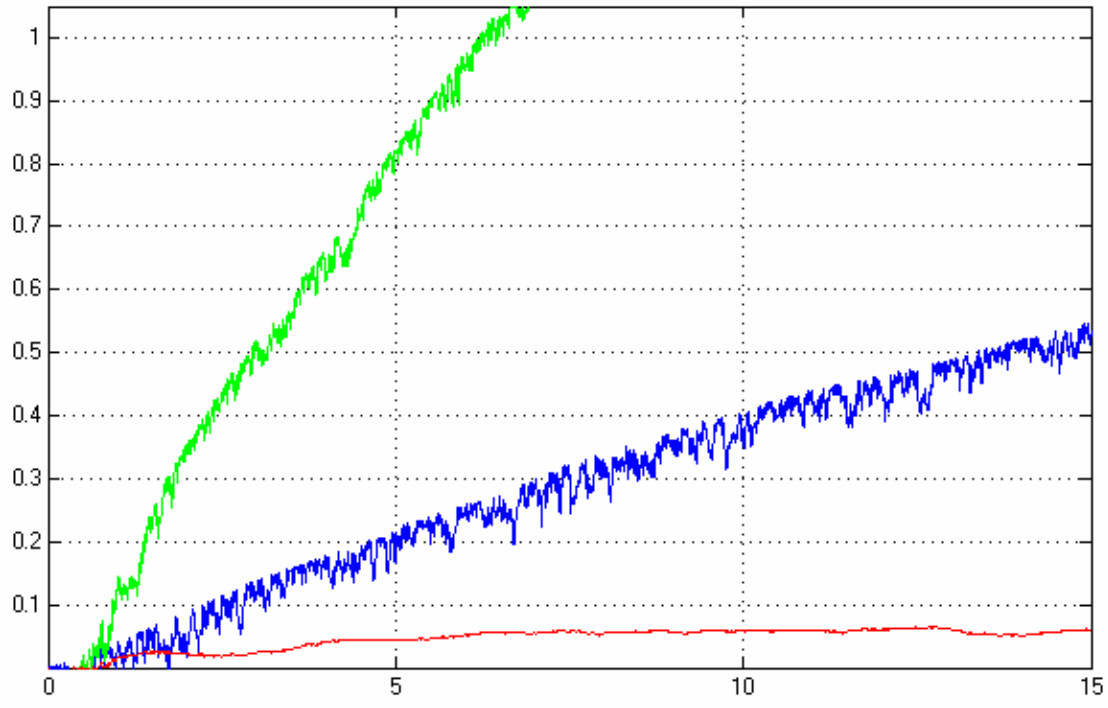


Figure 6-3: |Original Filtering| - |SR SPF| Velocity Estimate Errors (m/s) vs. Time (s)

The use of a human observer was still necessary at this point. However, whereas before the necessary human input for error cancellation was continuously increasing, the vehicle could now be made to hover in place with a constant trim input. While this was an improvement, ideally the vehicle would remain stationary with little to no human input. Though thrown together auto-trimming methods could likely produce this desired behavior, it was proposed that we treat the human input as both a control input and a measurement. As a simple case, if the human input is 1m/s in one direction, this could be treated as a measurement of 1 m/s in the opposite direction. In fact, a human observer attempting to drive vehicle state error to zero can be effectively modeled as a PD controller (on position) with relatively noisy/drifty PD gains. This treatment of the human input makes global velocity and position observable. This human state correction will ideally require less and less human input as time passes. Once the accelerometer biases are estimated fairly accurately and the accumulated error has been cancelled out by the human input, the vehicle should hover well without any additional human input. If it does drift due to small inaccuracies, a small human input is all that is needed. Ideally, the noise parameters would be tuned to allow for use of accelerometer information for fast dynamics while the human input helps zero out errors. While the variation in human gains could cause a problem, there are three possible solutions. The first is to estimate the PD gains themselves. This would increase the estimated parameters significantly, and is not desirable. The second option is to associate a large amount of sensor noise with the human "measurement" based on constant gains. The third option, which turned out to work the best in simulation, was to simply assume the human measurements have little noise and allow temporary errors in velocity and position estimates. This requires slightly more human input to keep the vehicle still than if the parameters were estimated, but the vehicle can still be driven to a decent hover without excessive effort or additional filter

complexity. Figure 6-4: SR SPF Velocity Estimate Errors (m/s) vs. Time (s) shows the velocity error as a function of time making use of a noisy human measurement. Note that while there is still some noise in the hover, this is primarily due to the vehicle's lack of ability to produce forces to directly counter process noise in the x and y directions. Instead, in order to react to forces in the horizontal plane, the vehicle must roll or pitch, and this reaction takes time.

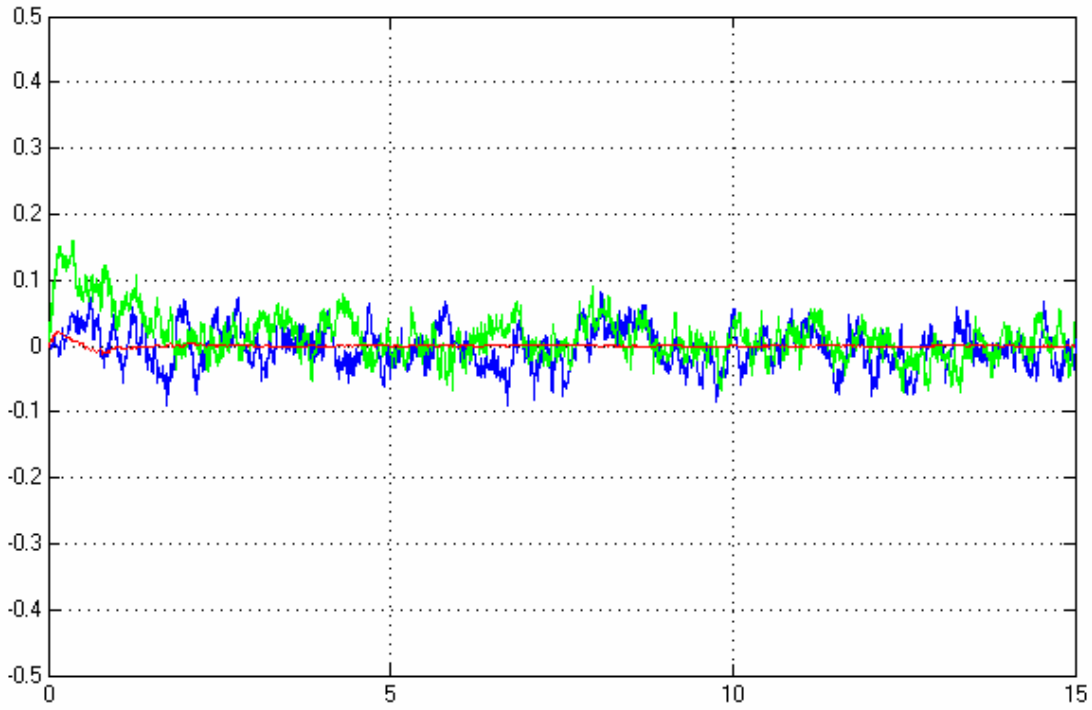


Figure 6-4: SR SPF Velocity Estimate Errors (m/s) vs. Time (s)

It should be noted that ultimately the human will be removed from the estimation and control loop completely. The use of human input in the loop for a so-called autonomous vehicle was only considered with the expectation that future versions of the vehicle would replace the human input with GPS or some indoor simulated GPS package. The addition of an absolute position measurement of this nature coupled

with the estimation of onboard inertial sensor bias parameters would allow for extremely accurate and drift-free navigation even if the absolute position information was available only at a relatively slow rate.

In addition to the real-time estimation of vehicle state and sensor bias parameters, batch processing of flight data can potentially be used for estimation of vehicle physical parameters such as the true positions of accelerometers, IMU rotation from the vehicle principal axes, and prop thrust and drag coefficients. No estimation of this nature has yet been performed due to the lack of the unconstrained flight data necessary for filtering of this type.

Estimator Tuning

The estimator noise matrices were created largely using the actual noise levels produced in the simulation. While this may seem like a cheat, the noise levels used in simulation have been tuned to best represent worst case noise levels in the true system. Even with close to the "true" noise levels available to the filter, though, some further tuning was necessary. Specifically, as touched upon earlier, there was some need to set the ratio of accelerometer noise to human measurement noise to produce the proper balance of accelerometer measurements and human input used for position and velocity estimation. Since the double integration of the accelerometers can drive the position estimate off extremely quickly, the human measurement was given less noise than would be expected from the variation in human PD gains. The noise matrices can be found in *ThreeDAFVmodelconsts.m* in APPENDIX E: ELECTRONIC CONTENT. Experimentation with the system suggests that the current balance of noise levels produces an estimate that trusts the human measurement roughly 75% while trusting the accelerometers roughly 25%. In addition to this deviation from "true" noise levels,

the estimates produced originally were still fairly noisy. A noisy estimate, unfortunately, results in noisy control since the control used is a simple gain matrix. It was therefore necessary to adjust the noise levels to produce a smoother estimate. Most of the noise observed was in the position and velocity state estimates. The solution, then, was to either reduce the process noise parameters or to increase the measurement noise parameters. Excessive decrease in the process noise parameters would prevent the estimate from changing to follow changes in true state due to process noise. Excessive increase in measurement noise values would result in a similar discounting of the accelerometer measurements. The best compromise was to change both parameters by the same amount, a factor of ten. The final velocity state estimates were still noisy, but remained both accurate and smooth enough to prevent excessive control jitter. The later portion of Figure 6-5: Motor 2 Voltage (V) vs. Time (s) shows the typical noise on the control effort during hover. A range of less than one Volt was well within acceptable limits. Figure 6-6: Y Velocity Estimate (m/s) vs. Time (s) shows the noise levels on y velocity estimates, which along with the x velocity estimate make up the two noisiest states. Again, the small variation about the actual value is well within acceptable limits. Apart from the two ratio adjustments discussed, very little noise tuning was necessary. This is almost certainly due in part to the fairly robust way in which the SRSPF handles noise covariance matrices. Many different noise matrices were tried, and variations in entire orders of magnitude from the true values still resulted in a stable estimator without major differences in performance beyond those discussed.

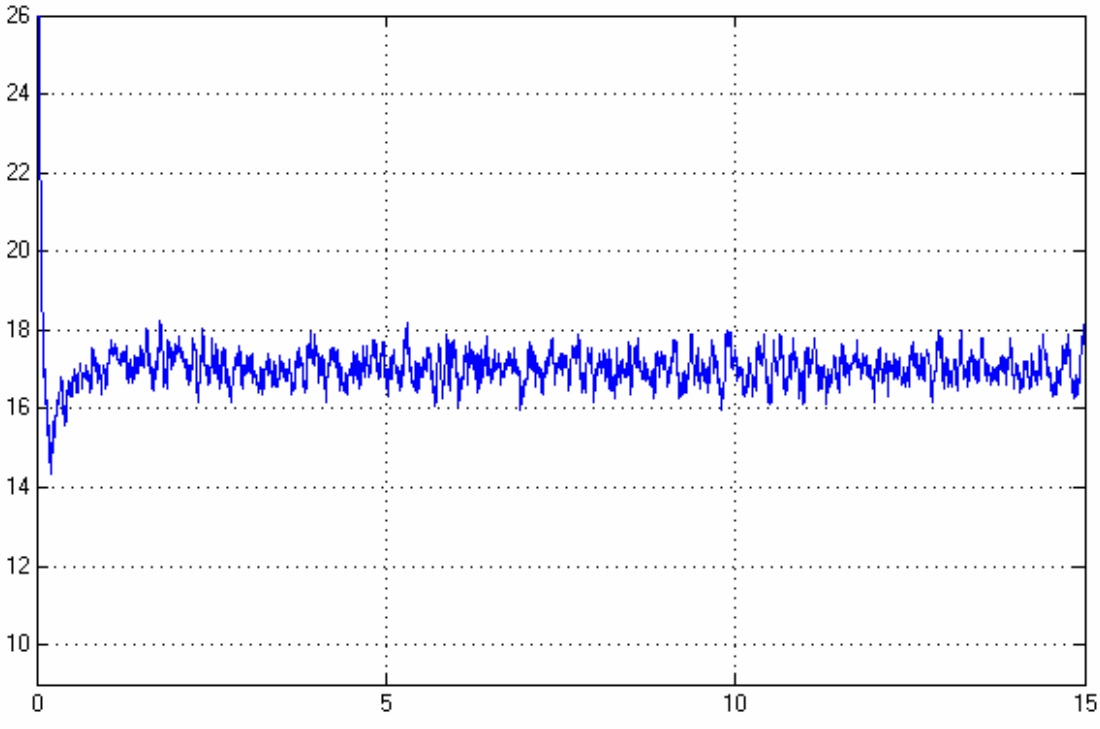


Figure 6-5: Motor 2 Voltage (V) vs. Time (s)

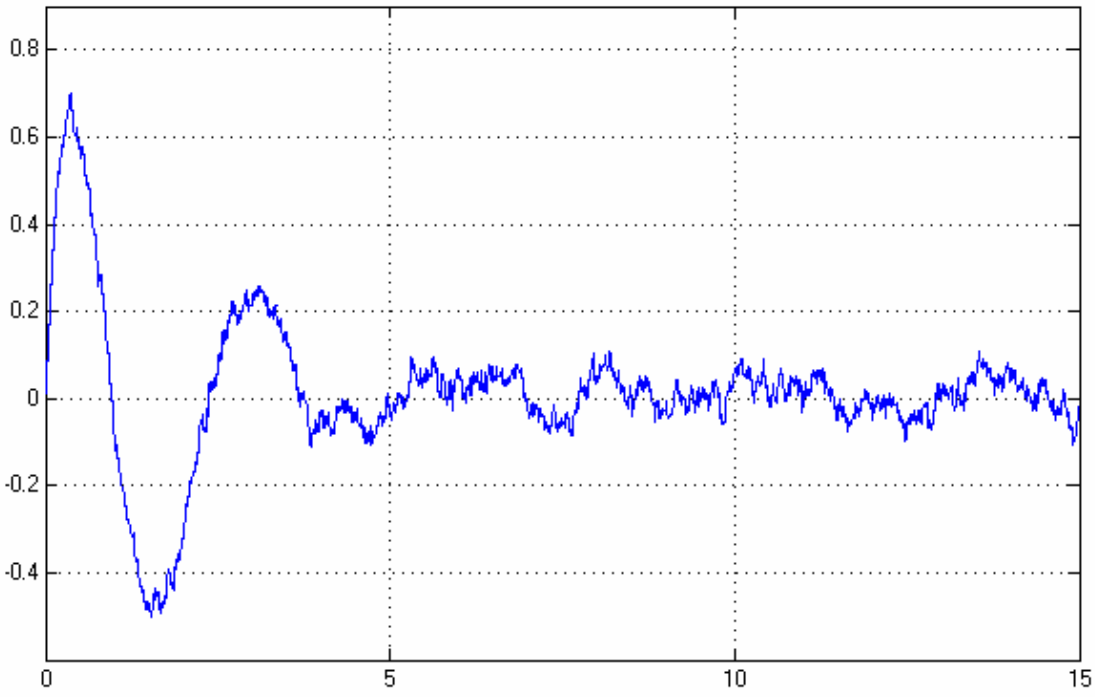


Figure 6-6: Y Velocity Estimate (m/s) vs. Time (s)

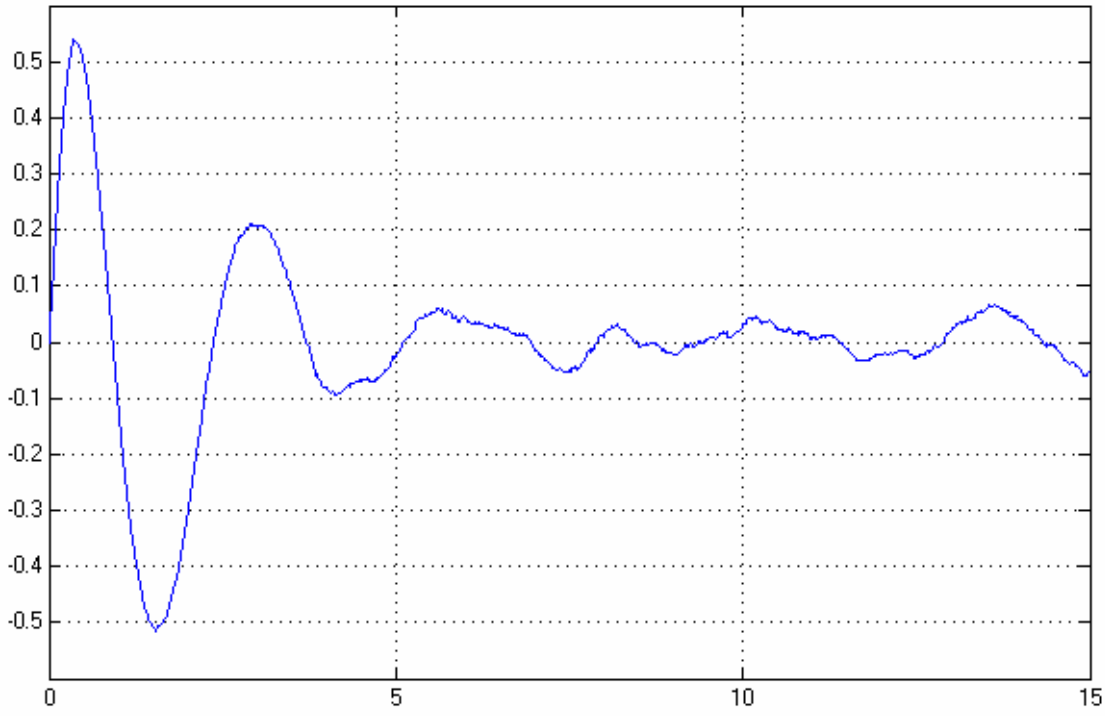


Figure 6-7: True Y Velocity (m/s) vs. Time (s)

Control

In order to simplify the control problem, two main control loops were implemented. The first lower level loop handled simple feedback control of each of the four thrust modules. Using input from the encoders located on the prop shafts, each motor control board performed simple PID control on prop speed by varying the voltage applied to the motor via PWM. The input to the control loop was a simple prop speed commanded by the main AFV processing module. Figure 6-8: Prop Local Control Loop contains a graphical representation of the loop. For details of tuning and design of the motor/prop control loop, please see the 2003 electronics documentation [1].

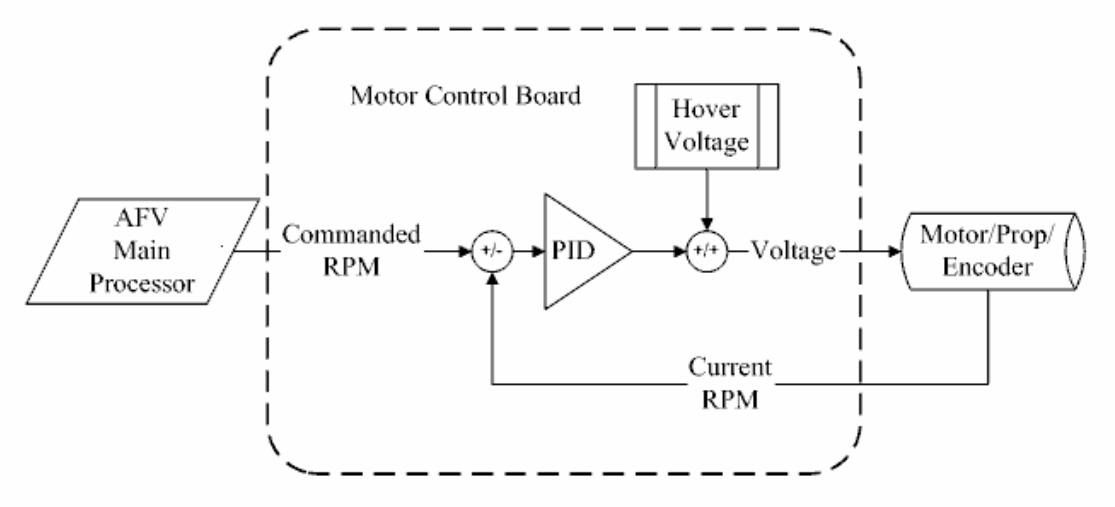


Figure 6-8: Prop Local Control Loop

The second outer control loop was more complex due to details of implementation and tuning. These details are discussed below. The general idea, however, was to stabilize the vehicle and permit the vehicle to follow simple velocity and yaw angle commands received from the PC base station via wireless. The loop would take these commands and compare them to the current best estimate of the state provided by the estimator. The resulting difference would feed the vehicle outer loop controller, which in turn produces four commanded prop RPM values. These four values are fed to the local prop control loop which presumably produces the necessary thrust and yaw torques via speed control of the props. Resulting angular rates and vehicle accelerations are sensed by the onboard IMU. These sensor measurements and the current prop speeds provided by the encoders are then fed into the state estimator to produce the next loop cycles best estimate of current state. The process, including the human in the loop configuration settled upon below, is summarized in Figure 6-9: Vehicle Control Loop.

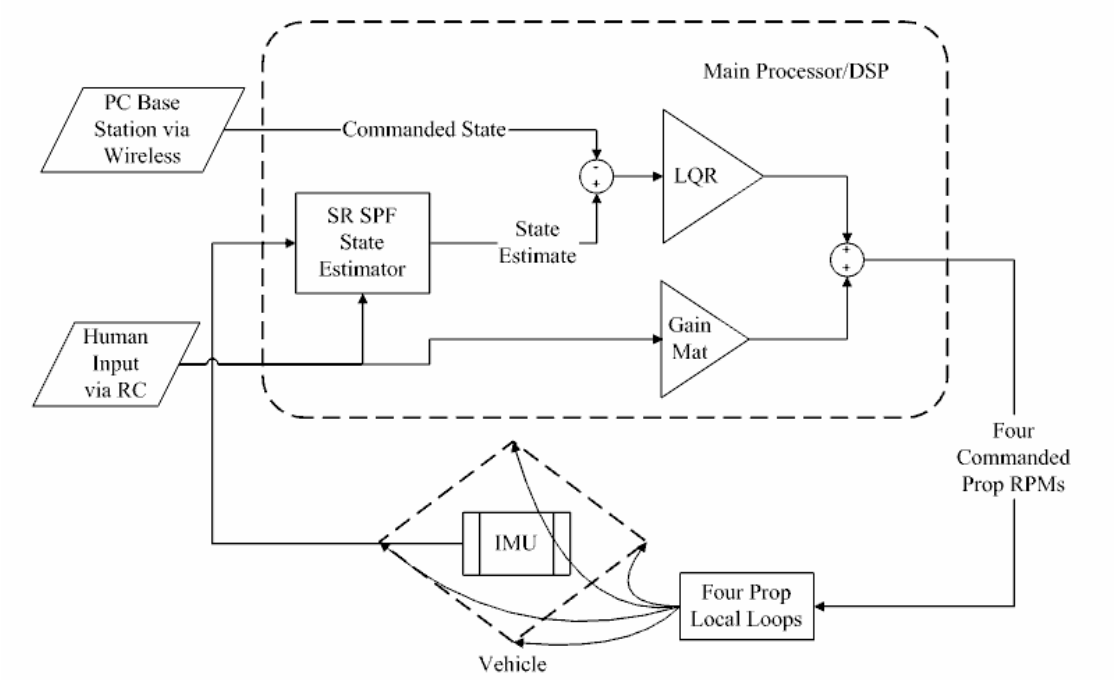


Figure 6-9: Vehicle Control Loop

Though the long term goals of the project involve eventually moving to nonlinear vehicle control, the current stage of the project requires only a controller that can accept velocity commands about hover. A simple LQR gain matrix was therefore created based on the linearized equations of motion about the hover state. In order to help reduce the nonlinearity of the system being controlled, the controller was designed to produce four thrust values. The necessary nonlinear transformation could then be applied to these four thrust values to get the desired prop RPM values. Since thrust varies with RPM squared, and the transformation could be easily backed out of the dynamics, this strategy helped produce a more linear controller. The four RPM values produced are then fed to the four local motor controllers which contain their own PID feedback loop on prop speed. In order to tune the LQR control weights, and thus controller performance, the simulation was started with some initial nonzero state,

such as 10 degrees of roll or 1 m/s z velocity, and the state response and voltage applied were observed. Saturation was expected to barely occur for nonzero states of the magnitude mentioned. Because of the coupling between roll and pitch angles and x and y velocities, the responses seen could not be simply tuned to produce a small amount of overshoot before settling to the final value. However, by tuning the appropriate nonzero states to just produce saturation, excellent performance was achieved.

As it turns out, the LQR controller generated by a uniform state weighting matrix performed almost exactly as desired. In fact, the gain matrix produced turned out to be fairly insensitive to changes to individual weights. This is likely due to the highly coupled nature of most of the states. The weighting for y velocity, for example, would have to be increased dramatically to see significant change in the gains used for roll or y position since y velocity is controlled by roll angle and directly influences y position. This left the control effort weight to be tuned. Depending on how much trade off between speed of response and disturbance rejection for potential saturation with large disturbances is desired, the weight can be raised or lowered. For the above mentioned disturbance situations a weight of one worked well. Figure 6-5: Motor 2 Voltage (V) vs. Time (s) shows the voltage history for motor 2 given an initial roll angle of 12.5 degrees. Note that the voltage saturates very briefly before falling. Figure 6-7: True Y Velocity (m/s) vs. Time (s) contains the true time history of y velocity. Note first that the state has almost completely settled in under five seconds. This includes zeroing of the initial roll angle as well as the y velocity produced by this roll angle and the y position error produced by this y velocity. Though the y velocity response may seem under damped, this behavior is in fact due to the need to zero the y position error

caused by the initial positive velocity error, which forces a nearly equal but opposite negative peak. The y velocity case is most representative of controller performance in general.

In addition to this nominal hover controller, some direct human control over vehicle behavior was desired. If the controller was to for some reason produce large errors quickly, ideally a human operator would be able to control the vehicle by hand, if only briefly. The current implementation allows the human operator to directly influence the voltages applied to each of the four motors. This provides the most robust vehicle response to human input, but it also has questionable effects on the stability of the total vehicle control. The gains used were therefore kept to approximately $1/10^{th}$ the gains seen in the LQR hover controller. The human therefore can have a noticeable influence on the vehicle by sending a strong signal, but most of the time the human input will only produce a slight tendency for the vehicle to return to a true zero state. Though this influence may seem minimal, the human input is also being used for measurement in the estimator, and the bulk of the zeroing of the true state is done in this manner. The relative gains on the various states were based on a pre-LQR version of the controller that commanded prop RPM values in the same way that the human input applies control commands. Alternatives, such as the use of the human input purely as a measurement, were considered and rejected for their lack of direct control over the vehicle.

The thorough simulation of the vehicle dynamics and measurement dynamics provided an effective test bed for extensive experimentation and filter tuning. Though the final estimator and control gains settled upon are fairly simple and straightforward, they are the product of trying a number of different control and estimation

configurations and noise matrices. Though the proposed new real time SRSPF performs much better than the original filtering scheme, especially with the addition of human input as measurement, it is at the expense of processing power. Fortunately this processing power should be available in the next iteration of the vehicle electronics.

CHAPTER 7:
CONCLUSION

In the past year and a half encompassing the current phase of the AFV project, much was accomplished. On the mechanical side, a fresh vehicle design combined with selection of robust and well-performing components resulted in a vehicle capable of stable hover flight. Though unverified by final vehicle tests, component testing and vehicle analysis suggests that the vehicle would be capable of both impressive maneuverability and significant endurance for a hovering electric powered vehicle. The largest improvement of the current version of the vehicle over legacy versions was the demonstration of stable hover flight despite the additional weight and design concerns caused by operation without tethered power supply and the addition of a functional fully self-contained onboard inertial navigation system. The addition of onboard power supply and sensing was not a trivial step as the batteries, IMU and its supporting electronics alone comprised approximately half of the total final vehicle weight. Of the remaining weight, a significant portion was due to the need to scale a vehicle up enough to enable it to carry this additional burden. The end product was a vehicle approximately six times the weight of previous incarnations that is capable of roughly double the total thrust of previous designs as normalized against vehicle weight. The complete lack of tethers for communication or power supply has brought the vehicle almost to the point where it can operate in real world situations without human intervention given the addition of appropriate global sensors and AI support provided by the PC base station.

In addition to the above mentioned accomplishments, a fair amount of work was also completed to aid in the further evolution of the project and vehicle. The simulation

designed is capable of highly accurate simulation of vehicle and sensor dynamics while still providing a degree of flexibility that enables the user to test a variety of control systems, estimators, potential sensors and actuators, and modeled scenarios. Work has already been done with the simulation to both aid in the selection of a new lower cost IMU and develop a more accurate on board state estimator. Future near term work will likely include the development and testing of a nonlinear control system and the tuning of estimation and control to accommodate the addition of GPS or a comparable indoor positioning system.

Though initial goals established by the team seemed lofty and at times unattainable, the final product produced argues for the competency and persistence of the entire team. Though design of a unique kind of vehicle and implementation of concepts in ever problematic hardware offered up countless pitfalls and apparent dead ends, the team was able to demonstrate the feasibility of a highly maneuverable four rotor hovering unmanned vehicle through hard work, persistence, and the occasional break provided by the advent of new technologies.

APPENDIX A:

BRAINSTORMING NOTES

Configuration

- Use of thrust vectoring (solenoids that open and close thrust ports)
- Current four prop design – requires vehicle tilt
- Two coaxial counter rotating blades for primary thrust. Smaller "omni" blades (rotors axis parallel to ground) for maneuvering - Decoupled degrees of freedom.
- Does rotation matter? Could drive counter rotating blades with same motor….some variation in which blade gets more applied power to control yaw?
- Brushless main thrust motor, brushed for maneuver:
- Main CCW, four CW
- 1 main CW, 1 main CCW, four as current
- above, either side by side or coaxial
- Two stage air acceleration? 14x4 then 14x8? (parallel versus series/coaxial rotors)
- Want efficient design – helicopter tail fins throw away ~30% of power, counter-rotating props are much more efficient.

Structure

- Center of mass with respect to thrust props…..higher cg increases response by decreasing inertia, but also increases response to disturbance torques. Same goes for footprint of rotors… further apart increases torque, but decreases speed of rotation for set prop translation.
- Use of wire to tension AFV, increase stiffness without adding significant weight.
- Foam vibration damping at the motor/prop attachment (rubber washers)
- How can we increase passive stability? Parachutist idea… dangle batteries?

Props

- Use of helicopter rotors versus props. Rotors are less efficient due to symmetric blade design, but could allow more rapid thrust changes.
- Use of piezoelectric bimetal or bimorph in propeller blades to vary the angle of attack/airfoil shape during flight – Electric thrust control.
- Fewer the blades, the more efficient. Endurance flight competitions, competitors actually use 1 blade with a counterbalance.
- Lower the pitch, the more efficient the thrust per power. However, this reduces (fixed wing) top speed, and increases the RPM necessary to get a

given thrust as compared to higher pitch, less efficient versions. Only benefit is in reducing pitch until blade no longer stalls for zero free stream velocity.
- Larger the diameter, more efficient. Large diameter props accelerate a lot of air a little bit, which is more efficient than accelerating a little air a lot.
- APC props are best for any application not in need of downline braking (essentially drag on the vehicle when in glide, prop not turning – this is important for larger models). They have a narrower, more efficient tip and overall more efficient design than their competitors.
- Propellers work similar to gears in terms of thrust and max translation speed.
- Smaller propellers have smaller inertia, benefit for maneuvering when varying speed? Slowing and speeding up blades decreases agility. Vary speed rather than torque. Helicopters keep speed constant, vary torque by varying blade angle.
- Helicopters prefer high inertia rotors since maneuvering is controlled independent of rotor speed.
- Propeller in plate/duct to reduce tip vortices, increase prop efficiency.
- Have props under vehicle – motors/mounting don't interfere with stronger airflow exiting prop

General

- Power supply possibilities – prefer electric?
- Rapid altitude changes necessary?
- Cooling fins parallel to airflow for electronics?
- Gear encoder to increase resolution?
- Large gear ratio plus hall effect sensors may be sufficient for brushless motors

APPENDIX B:

COMPONENT CHARACTERISTICS

Motors

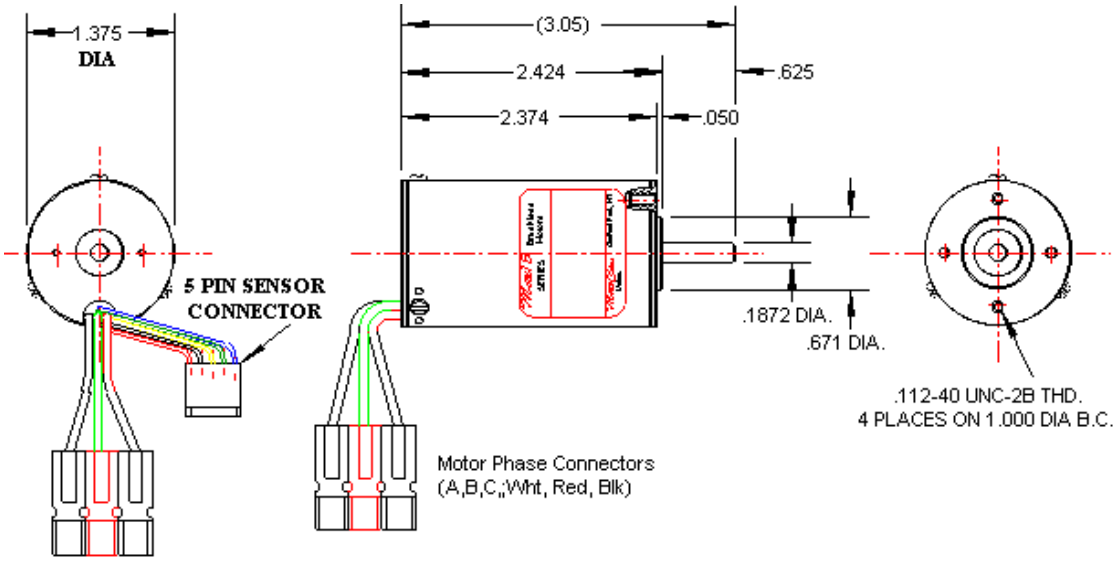


Figure B-1: MaxCim Motor Spec Sheet [14]

Table B-1: MaxCim Motor Parameters [14]

Motor Parameters	**(Units)**	**Model MaxN32-13D**
Torque Constant	(Oz.In./Amp)	0.548
Voltage Constant	(rpm/Volt)	2500
Motor Constant	(Oz.In./Sqrt(Watts))	3.96
Rated Power*	(Watts)	> 1200*
Line-Line Resistance	(Ohms)	0.022
Max. Current	(Amps)	70
Idle current Io - (7 cells)	(Amps)	2.5
Max. Operating Speed	(rpm) (cells)	50,000 (18)
Cogging Torque	(Oz.In.)	<0.3
Weight with connectors	(Oz.)	7.5

Props

Table B-2: Prop Constants

18x6 prop	RPM	N thrust	Model based	Nm Torque	Model based	kd	kt
PropSelector	4000	15.28	14.74	0.359	0.551	2.05E-06	8.71E-05
	4700	21.10	20.35	0.496	0.761	2.05E-06	8.71E-05
	5200	25.83	24.91	0.607	0.931	2.05E-06	8.71E-05
Test Data	1712	2.62	2.70	0.166	0.101	5.18E-06	8.17E-05
	2580	5.78	6.13	0.276	0.229	3.78E-06	7.92E-05
	3190	9.34	9.37	0.357	0.350	3.20E-06	8.37E-05
	4020	14.96	14.89	0.533	0.556	3.01E-06	8.44E-05
APC	4470	20.91	18.41	0.719	0.688	3.28E-06	9.54E-05
Modeled						3.14E-06	8.40E-05

Prop diameter: 18”
Prop pitch: 6”
Prop weight: 130 g
Hub diameter: 1.75”
Hub depth: 5/8”
Hub bore: 3/8”

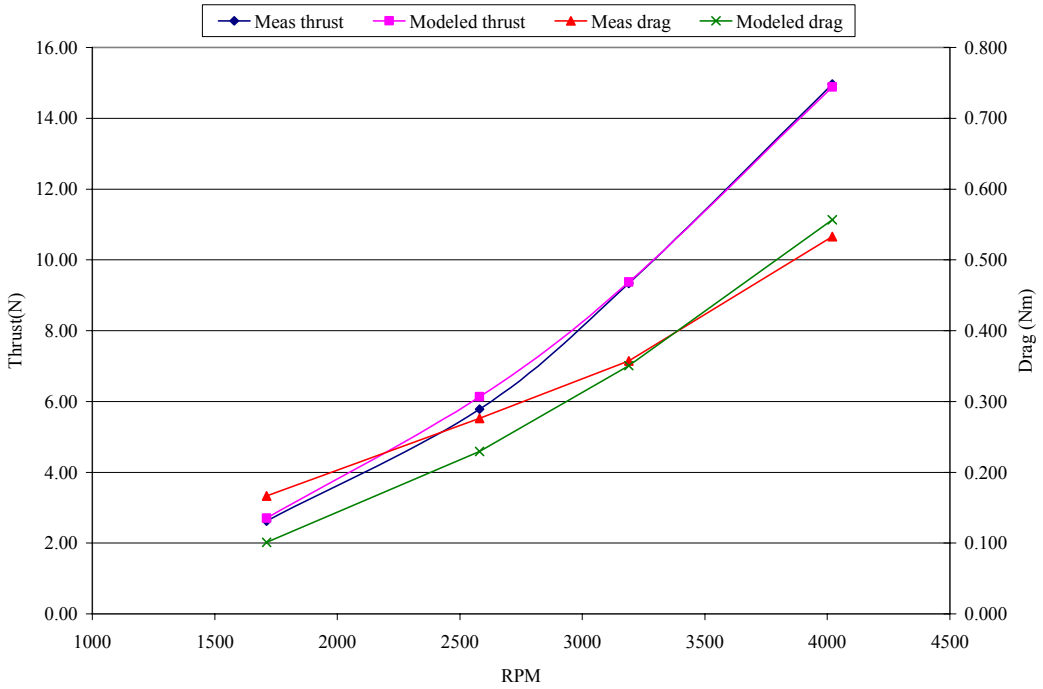


Figure B-2: Prop Testing Results

Gears and Belts

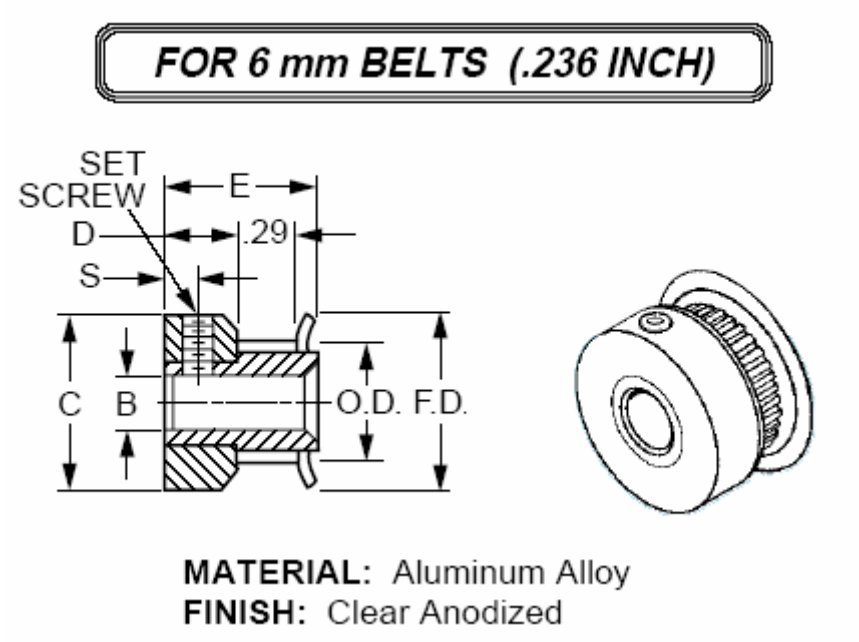


Figure B-3: Motor Timing Pulley Spec Sheet [15]

Part Number: A 6A51-015DF0606
Unit: Inch
Pitch: GT2 (2MM)
No. Of Grooves: 15
Material: Aluminum Alloy
Belt Width: .236 (6MM)
Bore Size (B) : 0.188"
Flange & Hub Configuration: 2 Flanges / With Hub
Pitch Dia.: 0.376"
Outside Dia. (O.D.): 0.356"
Overall Length (E): 0.563"
Hub Dia. (C): 0.555"
Hub Proj. (D): 15/64"
(S): 7/64"
Flange Dia. (F.D.): 0.555"

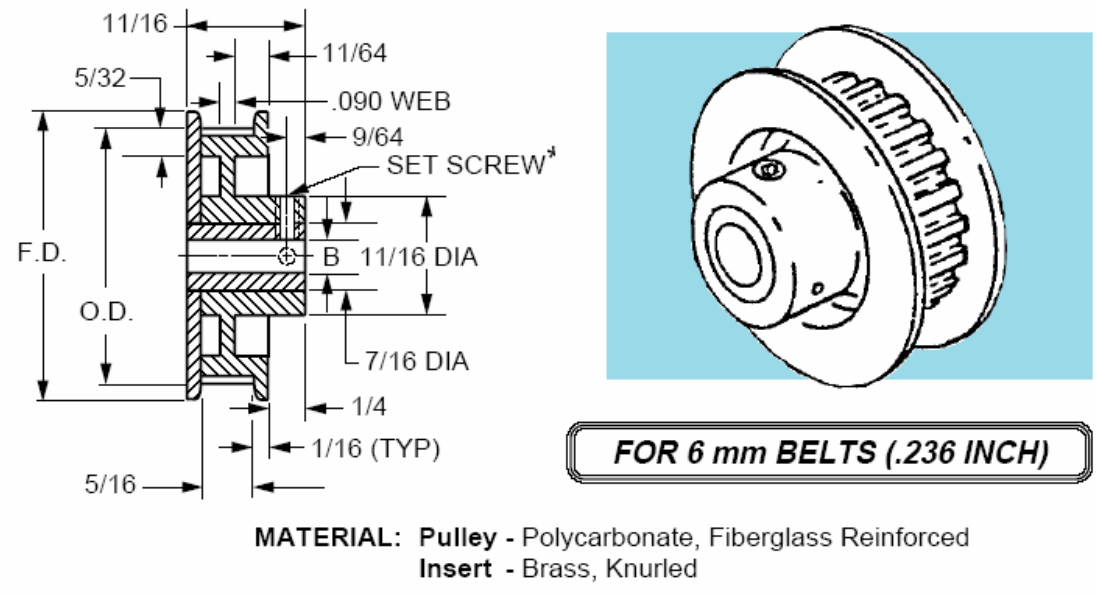
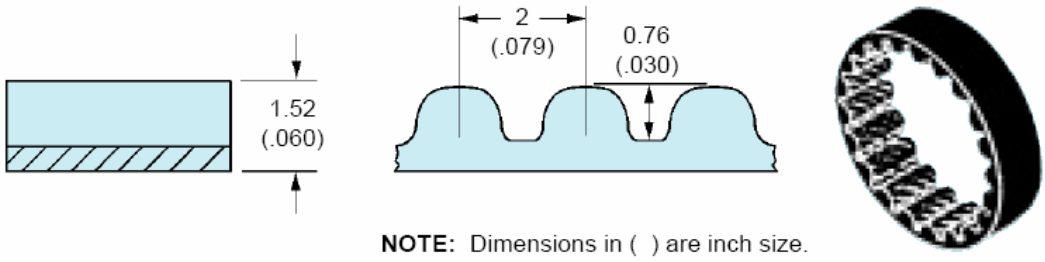


Figure B-4: Prop Timing Pulley Spec Sheet [15]

Part Number: A 6Z51-100DF0608
Unit: Inch
Pitch: GT (2mm)
No. Of Grooves: 100
Material: Polycarbonate
Belt Width: .236(6mm)"
Bore Size (B): 0.250"
Bore Config.: Brass Insert
Flange Config.: 2 Flanges / With Hub
Pitch Dia.: 2.506"
Outside Dia. (O.D.): 2.486"
Overall Length: 0.688"
Flange Dia. (F.D.): 2.71"



NOTE: Dimensions in ( ) are inch size.

MATERIAL: Nylon Covered, Fiberglass Reinforced, Neoprene
BREAKING STRENGTH: 86 N per 1 mm (62 lbs. per 1/8") Belt Width; not representative of the load-carrying capacity of the belt
WORKING TENSION: 111 N for 25.4 mm Belt (25 lbs. for 1" Belt). For more on Working Tension, see the technical section.
TEMPERATURE RANGE: -34°C to +85°C (-30°F to +185°F)

Figure B-5: Timing Belt Spec Sheet [15]

Part Number: A 6R51M116060
Unit: Metric
Belt Type: Single Sided
Pitch: GT (2MM)
No. Of Grooves: 116
Belt Width: 6.0 mm
Material: Neoprene
Tension Member(cords): Fiberglass
Pitch Length: 232 mm

Encoders

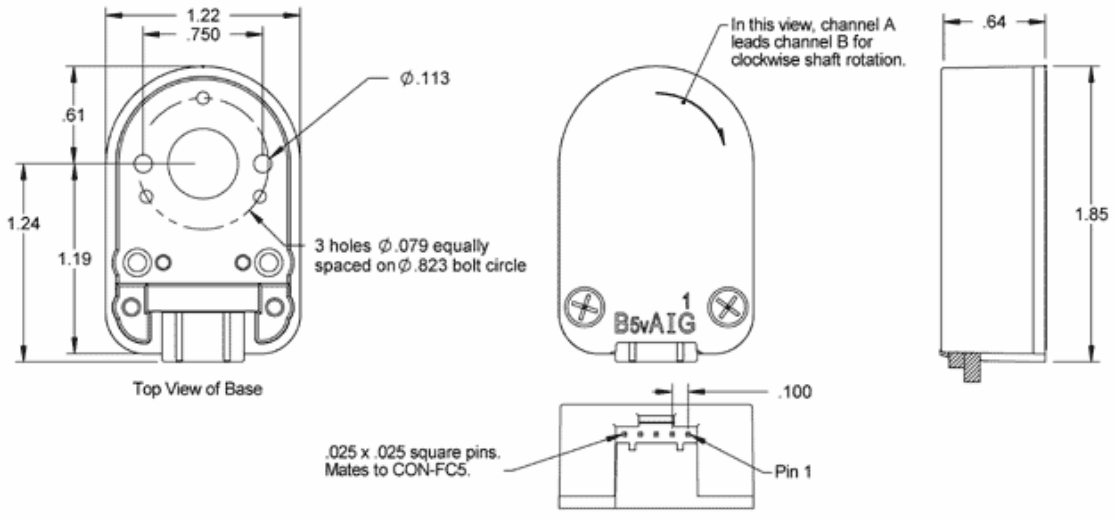


Figure B-6: Encoder Spec Sheet [16]

Table B-3: Encoder Parameters [16]

Parameter	**Dimension**	**Units**
Moment of Inertia	8.0 x 10^-6	oz-in-s²
Hub Set Screw	3-48 or 4-48	in.
Hex Wrench Size	.050	in.
Encoder Base Plate Thickness	.135	in.
3 Mounting Screw Size	0-80	in.
2 Mounting Screw Size	2-56 or 4-40	in.
3 Screw Bolt Circle Diameter	.823 ±.005	in.
2 Screw Bolt Circle Diameter	.750 ±.005	in.
Required Shaft Length With **E**-option With **H**-option	.445 to .570* .445 to .750* =>.445*	in. in. in.
Weight	.80	oz.

Batteries

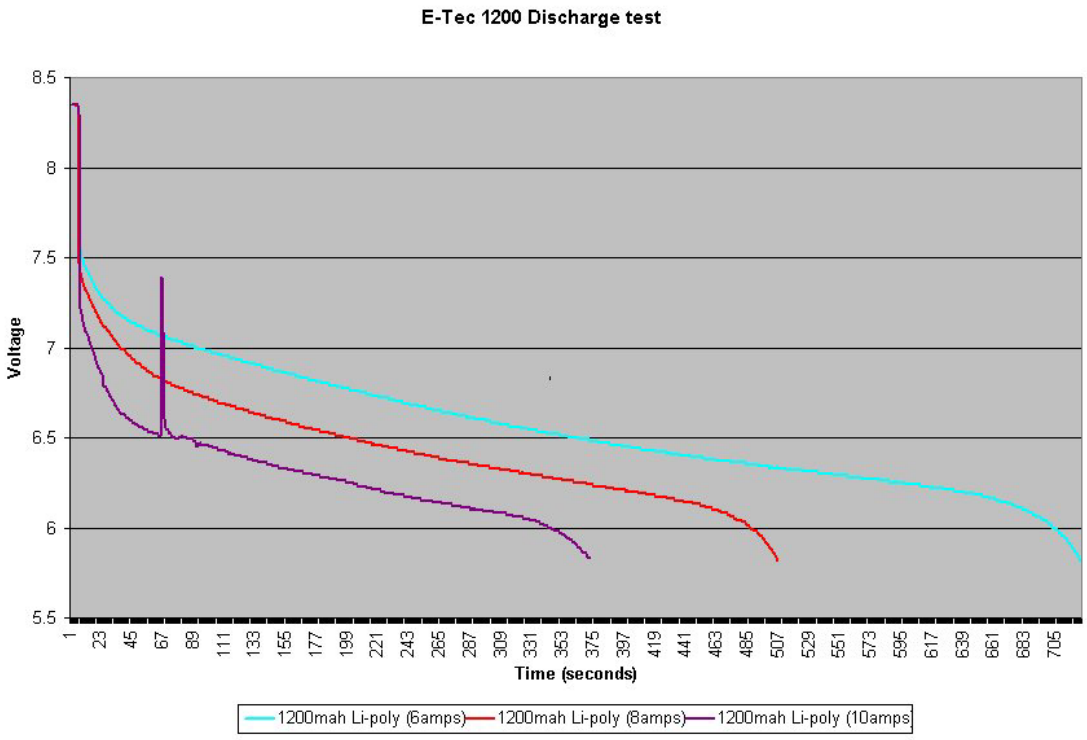


Figure B-7: Battery Discharge Test Results [17]

Unit: E-tec 1200
Capacity: 1200 mAh
Maximum continuous drain rate: 6 C
Maximum drain rate: 7.5C
Unpackaged weight: 24 g
Nominal Voltage: 4.2 V

Fabricated Parts and Misc Components

Table B-4: Parts and Components Information

Category	Part	Quantity	Supplier	Part #
Stand-alone components	IMU	1	Systron-Donner	
Structure	IMU mount		McMaster	8733K23
	Board mount		McMaster	8733K23
	Strut mount		McMaster	8733K38
	Threaded rod		McMaster	94435A357
	Threaded rod sleeves		McMaster	7237K17
	All struts		McMaster	9924K13
	Strut plugs		McMaster	9061K15
	Tension wire rope	30 feet	McMaster	3458T24
	Midget turnbuckles	3	McMaster	3003T144
	Wire rope clips	6	McMaster	3677T51
	Strut ends/pulley box mount		McMaster	89215K17
	Landing block/motor board mount		McMaster	8732K16
	Battery hangers		McMaster	8733K23
	Battery hanger retainer		McMaster	8732K11
	Strut ends/pulley box mount		McMaster	89215K17
	Landing gear springs	4	McMaster	1986K13
	Landing gear spring channel		McMaster	8538K19
Misc Fasteners	EE main board standoffs	8	McMaster	92745A320
	EE motor board standoffs	16	McMaster	92745A324
	Motor end strut spring pin	4	McMaster	92383A159
	Center end strut spring pin	4	McMaster	92383A155
	Battery hanger mount spring pin	8	McMaster	92383A157
	Landing gear mounting spring pin	12	McMaster	92383A157
	Pulley box joint spring pin	16	McMaster	92383A151
	Body threaded rod nuts		McMaster	95170A370

Table B-4 (Continued)

	Vibration Absorbtion washers		McMaster	90130A007
Batteries	4 series EE pack		Bishop Power Products	E-tec 1200 Li-Poly
	7 series x 2 parallel motor pack		Bishop Power Products	E-tec 1200 Li-Poly
Pulley box components	Motor control board	4		
	Pusher prop	2	APC	LP18060WP
	Tractor prop	2	APC	LP18060W
	Brushless motor	4	MaxCim	MaxN32-13D
	Encoder	4	US Digital	E5S-1024-375-IHA
	Encoder cable	4	US Digital	CA-3620-8IN
	Motor board/main board comm cable	4	US Digital	CA-3620-11IN
	15 tooth pulley	4	SDP	A 6A51-015DF0606
	100 tooth pulley	4	SDP	A 6Z51-100DF0608
	116 tooth belt	4	SDP	A 6R51M116060
	Prop shaft bearings	8	McMaster	57155K166
	Prop shaft collar	4	McMaster	6157K12
	Prop washer		McMaster	8974K711
	Prop shaft lock nut	4	McMaster	90101A240
	Pulleybox extension		McMaster	6023K193
	Pulleybox		McMaster	6546K11
	Prop shaft		McMaster	9061K15
Landing platform	Edge guard		McMaster	8451A55
	Platform sheet	1	McMaster	9232T221

Note: EXCEL file contains description column.

Table B-5: Supplier Information

APC	www.apcprop.com
McMaster	www.mcmaster.com
SDP	www.sdp-si.com
US Digital	www.usdigital.com
MaxCim	www.maxcim.com
Bishop Power Products	www.b-p-p.com

APPENDIX C:

DERIVATION OF AFV DYNAMICS

The following is a derivation of the equations of motion for the AFV assuming it is a rigid body acted on by thrust forces generated by the propellers, drag forces generated by the propellers, gravity, disturbance forces in the global frame and disturbance torques in the local frame. Analysis will include not only straightforward propeller thrust/drag effects, but will also take into account the change in propeller effective pitch with changes in free stream velocity as observed by the prop (due to vehicle translation and rotation). Advancing/retreating blade effects are specifically neglected due to the assumption that the vehicle will primarily operate with small lateral velocities. Following the derivation of the dynamics equations is a derivation of the measurement equations assuming an onboard strap down inertial measurement unit.

Bases and the Direction Cosines

The basis for the space coordinate system, which is fixed in space, is given by [x y z]'. The space coordinate system is a standard right-handed coordinate system with z pointing down. The basis for the body coordinate system, which is fixed to the AFV, is given by [n o a]'. The two coordinate systems are related by the relationship [x y z]' = A*[n o a]', where A is the rotation matrix.

Inversion of the rotation matrix yields

$$\begin{bmatrix} n \\ o \\ a \end{bmatrix} = A^{-1} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{C-1}$$

Prop location in body coordinates – rotation direction:
Prop1: [L 0 0]' -a
Prop2: [0 L 0]' a
Prop3: [-L 0 0]' -a
Prop4: [0 -L 0]' a

Euler Angles

The rotation matrix is defined using the Roll Pitch Yaw (RPY) Angles. These angles define the rotation matrix via successive rotations about the Roll, Pitch, and Yaw angles of the body coordinate system. Since we are rotating about the body coordinate system, successive rotations pre-multiply previous rotations.

$A(\phi,\theta,\psi)$ = Yaw*Pitch*Roll
$A(\phi,\theta,\psi)$ = $Rot(a,\psi)*Rot(o,\theta)*Rot(n,\phi)$

$$A = \begin{bmatrix} C\psi & -S\psi & 0 \\ S\psi & C\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C\theta & 0 & S\theta \\ 0 & 1 & 0 \\ -S\theta & 0 & C\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & C\phi & -S\phi \\ 0 & S\phi & C\phi \end{bmatrix} \tag{C-2}$$

$$A = \begin{bmatrix} C\psi C\theta & C\psi S\theta S\phi - S\psi C\phi & C\psi S\theta C\phi + S\psi S\phi \\ S\psi C\theta & S\psi S\theta S\phi + C\psi C\phi & S\psi S\theta C\phi - C\psi S\phi \\ -S\theta & C\theta S\phi & C\theta C\phi \end{bmatrix} \tag{C-3}$$

$$A^{-1} = \begin{bmatrix} C\theta C\phi & C\theta S\phi & -S\theta \\ S\psi S\theta C\phi - C\psi S\phi & S\psi S\theta S\phi + C\psi C\phi & S\psi C\theta \\ C\psi S\theta C\phi + S\psi S\phi & C\psi S\theta S\phi - S\psi C\phi & C\psi C\theta \end{bmatrix} \tag{C-4}$$

Euler time derivatives are related to body angular rates by the matrix M

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = M^{-1} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = M^{-1} A \begin{bmatrix} \omega_n \\ \omega_o \\ \omega_a \end{bmatrix} \tag{C-5}$$

Where

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} C\psi & -S\psi & 0 \\ S\psi & C\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C\theta & 0 & S\theta \\ 0 & 1 & 0 \\ -S\theta & 0 & C\theta \end{bmatrix} \begin{bmatrix} \dot{\psi} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} C\psi & -S\psi & 0 \\ S\psi & C\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\phi} \end{bmatrix} \tag{C-6}$$

yielding

$$M^{-1} = \begin{bmatrix} \dfrac{C\psi}{C\theta} & \dfrac{S\psi}{C\theta} & 0 \\ -S\psi & C\psi & 0 \\ C\psi T\theta & S\psi T\theta & 1 \end{bmatrix} \tag{C-7}$$

Differentiating yields the relationship between body torques and Euler rates

$$\begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \frac{\partial (M^{-1}A)}{\partial \phi} \begin{bmatrix} \omega_n \\ \omega_o \\ \omega_a \end{bmatrix} + \frac{\partial (M^{-1}A)}{\partial \theta} \begin{bmatrix} \omega_n \\ \omega_o \\ \omega_a \end{bmatrix} + \frac{\partial (M^{-1}A)}{\partial \psi} \begin{bmatrix} \omega_n \\ \omega_o \\ \omega_a \end{bmatrix} + M^{-1}A \begin{bmatrix} \dot{\omega}_n \\ \dot{\omega}_o \\ \dot{\omega}_a \end{bmatrix} \tag{C-8}$$

For body velocities and global velocities

$$\begin{bmatrix} \dot{n} \\ \dot{o} \\ \dot{a} \end{bmatrix} = A^{-1} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \tag{C-9}$$

since we are concerned only with the velocity of the center of mass of the vehicle, located at n=o=a=0.

Applied Forces

The nominal thrust produced by each prop varies with the prop angular velocity squared

$$T_{inom} = k_t \alpha_i^2 \tag{C-10}$$

where k_t is the coefficient of thrust of the prop

However, the thrust is affected by losses and gains due to prop motion relative to stationary air affecting the effective prop pitch. It is assumed that the prop produces zero thrust at the prop pitch speed, equal to prop pitch*rotations/second. It is also assumed that when the vehicle rotates the prop sees a linear velocity equal to L*body rotation radians/second

$$\begin{aligned}
T_1 &= T_{1nom}\left[1 - \frac{2\pi L}{P\alpha_1}\omega_o + \frac{2\pi}{P\alpha_1}(\dot{a} - w_a)\right] \\
T_2 &= T_{2nom}\left[1 + \frac{2\pi L}{P\alpha_2}\omega_n + \frac{2\pi}{P\alpha_2}(\dot{a} - w_a)\right] \\
T_3 &= T_{3nom}\left[1 + \frac{2\pi L}{P\alpha_3}\omega_o + \frac{2\pi}{P\alpha_3}(\dot{a} - w_a)\right] \\
T_4 &= T_{4nom}\left[1 - \frac{2\pi L}{P\alpha_3}\omega_n + \frac{2\pi}{P\alpha_3}(\dot{a} - w_a)\right]
\end{aligned} \tag{C-11}$$

where L is the radial distance of the prop center and P is the prop pitch in meters

The corresponding thrust vectors are

$$\overline{T_i} = -T_i \hat{a} \tag{C-12}$$

The wind loading disturbance forces are

$$\begin{aligned}
\overline{F_n} &= k_s(w_n - \dot{n})\hat{n} \\
\overline{F_o} &= k_s(w_o - \dot{o})\hat{o} \\
\overline{F_a} &= k_u(w_a - \dot{a})\hat{a}
\end{aligned} \tag{C-13}$$

However, wind forces are expected to act in the global frame

$$\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = A \begin{bmatrix} F_n \\ F_o \\ F_a \end{bmatrix} \tag{C-14}$$

$$\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = A \begin{bmatrix} k_s & 0 & 0 \\ 0 & k_s & 0 \\ 0 & 0 & k_u \end{bmatrix} \left(\begin{bmatrix} w_n \\ w_o \\ w_a \end{bmatrix} - \begin{bmatrix} \dot{n} \\ \dot{o} \\ \dot{a} \end{bmatrix} \right) \tag{C-15}$$

With

$$\begin{bmatrix} w_n \\ w_o \\ w_a \end{bmatrix} = A^{-1} \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} \tag{C-16}$$

The weight of the vehicle acts at the center of mass, and is given by

$$\overline{W} = mg\hat{z} \tag{C-17}$$

Linear momentum balance yields

$$m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = g \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} - \left(T_1 + T_2 + T_3 + T_4\right) A \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} \tag{C-18}$$

Applied Moments

The nominal drag produced by each prop is

$$D_i = k_d \alpha_i^2 \tag{C-19}$$

where k_d is the coefficient of drag of the prop

Drag moments are assumed to be affected by prop motion or free stream velocity in the same way thrust is affected.

$$\begin{aligned} D_1 &= k_d \alpha_1^2 \left(1 - \frac{2\pi L}{P\alpha_1} \omega_o + \frac{2\pi}{P\alpha_1} \left(\dot{a} - w_a \right) \right) \\ D_2 &= k_d \alpha_2^2 \left(1 + \frac{2\pi L}{P\alpha_2} \omega_n + \frac{2\pi}{P\alpha_2} \left(\dot{a} - w_a \right) \right) \\ D_3 &= k_d \alpha_3^2 \left(1 + \frac{2\pi L}{P\alpha_3} \omega_o + \frac{2\pi}{P\alpha_3} \left(\dot{a} - w_a \right) \right) \\ D_4 &= k_d \alpha_4^2 \left(1 - \frac{2\pi L}{P\alpha_4} \omega_n + \frac{2\pi}{P\alpha_4} \left(\dot{a} - w_a \right) \right) \end{aligned} \tag{C-20}$$

Drag moment vectors are then given by

$$\begin{aligned} \overline{D_1} &= D_1 \hat{a} \\ \overline{D_2} &= -D_2 \hat{a} \\ \overline{D_3} &= D_3 \hat{a} \\ \overline{D_4} &= -D_4 \hat{a} \end{aligned} \tag{C-21}$$

The moments about the center of mass generated by the thrust forces are given by

$$\begin{aligned}\overline{r_1}x\overline{T}_1 &= LT_1\hat{o}\\ \overline{r_2}x\overline{T}_2 &= -LT_2\hat{n}\\ \overline{r_3}x\overline{T}_3 &= -LT_3\hat{o}\\ \overline{r_4}x\overline{T}_4 &= LT_4\hat{n}\end{aligned} \tag{C-22}$$

The moment produced by temporary inequalities in the sum of the changes of angular momentum of the four individual props is

$$M_{ma} = J_t\left(\dot{\alpha}_1 + \dot{\alpha}_3 - \dot{\alpha}_2 - \dot{\alpha}_4\right) \tag{C-23}$$

$$\overline{M_{ma}} = M_{ma}\hat{a} \tag{C-24}$$

where J_t is the mass moment of inertia of a single prop (and geared motor rotor) about its rotation axis

The disturbance torques are

$$\begin{aligned}\overline{\tau_n} &= \tau_n\hat{n}\\ \overline{\tau_o} &= \tau_o\hat{o}\\ \overline{\tau_a} &= \tau_a\hat{a}\end{aligned} \tag{C-25}$$

Angular momentum balance,

$$J\begin{bmatrix}\dot{\omega}_n\\ \dot{\omega}_o\\ \dot{\omega}_a\end{bmatrix} = \begin{bmatrix}L\left(T_4 - T_2\right)\\ L\left(T_1 - T_3\right)\\ D_1 + D_3 - D_2 - D_4 + M_{ma}\end{bmatrix} + \begin{bmatrix}\tau_n\\ \tau_o\\ \tau_a\end{bmatrix} + \begin{bmatrix}\omega_n\\ \omega_o\\ \omega_a\end{bmatrix} xJ \begin{bmatrix}\omega_n\\ \omega_o\\ \omega_a\end{bmatrix} \tag{C-26}$$

$$\begin{bmatrix}\dot{\omega}_n\\ \dot{\omega}_o\\ \dot{\omega}_a\end{bmatrix} = J^{-1}\begin{bmatrix}L\left(T_4 - T_2\right)\\ L\left(T_1 - T_3\right)\\ D_1 + D_3 - D_2 - D_4 + M_{ma}\end{bmatrix} + J^{-1}\begin{bmatrix}\tau_n\\ \tau_o\\ \tau_a\end{bmatrix} + J^{-1}\begin{bmatrix}\omega_n\\ \omega_o\\ \omega_a\end{bmatrix} xJ \begin{bmatrix}\omega_n\\ \omega_o\\ \omega_a\end{bmatrix} \tag{C-27}$$

Where J is the full rotational mass moment of inertia matrix for the vehicle. Equation C-8 coupled with angular momentum balance results give us the full equations of motion in rotation.

Motor Dynamics

The motor torques are given by

$$\begin{aligned}\tau_{m1} &= \frac{k_i}{R}\left(V_1 - \alpha_1 k_v\right)\\ \tau_{m2} &= \frac{k_i}{R}\left(V_2 - \alpha_2 k_v\right)\\ \tau_{m3} &= \frac{k_i}{R}\left(V_3 - \alpha_3 k_v\right)\\ \tau_{m4} &= \frac{k_i}{R}\left(V_4 - \alpha_4 k_v\right)\end{aligned} \tag{C-28}$$

where k_i and k_v are motor torque and speed constants, appropriately transformed to take gearing into account, and R is the resistance of the motor

The rate of change of prop speed is simply torque minus drag divided by the total prop/gear/motor rotor mass moment of inertia

$$\begin{aligned}\dot{\alpha}_1 &= \frac{\tau_{m1} - D_1}{J_t}\\ \dot{\alpha}_2 &= \frac{\tau_{m2} - D_2}{J_t}\\ \dot{\alpha}_3 &= \frac{\tau_{m3} - D_3}{J_t}\\ \dot{\alpha}_4 &= \frac{\tau_{m4} - D_4}{J_t}\end{aligned} \tag{C-29}$$

Final Differential Equations of Motion, Summary

$$\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix} = \frac{g}{m}\begin{bmatrix}0\\0\\1\end{bmatrix} - \frac{\left(T_1 + T_2 + T_3 + T_4\right)}{m} A \begin{bmatrix}0\\0\\1\end{bmatrix} + \frac{1}{m}\begin{bmatrix}F_x\\F_y\\F_z\end{bmatrix} \tag{C-30}$$

$$\begin{bmatrix}\ddot{\phi}\\ \ddot{\theta}\\ \ddot{\psi}\end{bmatrix} = \frac{\partial(M^{-1}A)}{\partial\phi}\begin{bmatrix}\omega_n\\ \omega_o\\ \omega_a\end{bmatrix} + \frac{\partial(M^{-1}A)}{\partial\theta}\begin{bmatrix}\omega_n\\ \omega_o\\ \omega_a\end{bmatrix} + \frac{\partial(M^{-1}A)}{\partial\psi}\begin{bmatrix}\omega_n\\ \omega_o\\ \omega_a\end{bmatrix} + M^{-1}A\begin{bmatrix}\dot{\omega}_n\\ \dot{\omega}_o\\ \dot{\omega}_a\end{bmatrix} \tag{C-31}$$

$$\begin{aligned}\dot{\alpha}_1 &= \frac{\tau_{m1} - D_1}{J_t}\\ \dot{\alpha}_2 &= \frac{\tau_{m2} - D_2}{J_t}\\ \dot{\alpha}_3 &= \frac{\tau_{m3} - D_3}{J_t}\\ \dot{\alpha}_4 &= \frac{\tau_{m4} - D_4}{J_t}\end{aligned} \tag{C-32}$$

Subsets, in order of calculation:

$$\tau_{mi} = \frac{k_i}{R}\left(V_i - \alpha_i k_v\right) \tag{C-33}$$

$$\begin{bmatrix} w_n \\ w_o \\ w_a \end{bmatrix} = A^{-1}\begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} \tag{C-34}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = A\begin{bmatrix} \dot{n} \\ \dot{o} \\ \dot{a} \end{bmatrix} \tag{C-35}$$

$$\begin{bmatrix} \dot{n} \\ \dot{o} \\ \dot{a} \end{bmatrix} = A^{-1}\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \tag{C-36}$$

$$\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = A\begin{bmatrix} k_s & 0 & 0 \\ 0 & k_s & 0 \\ 0 & 0 & k_u \end{bmatrix}\left(\begin{bmatrix} w_n \\ w_o \\ w_a \end{bmatrix} - \begin{bmatrix} \dot{n} \\ \dot{o} \\ \dot{a} \end{bmatrix}\right) \tag{C-37}$$

$$\begin{bmatrix} \omega_n \\ \omega_o \\ \omega_a \end{bmatrix} = A^{-1}M\begin{bmatrix} \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} \tag{C-38}$$

$$\begin{aligned} T_1 &= T_{1nom}\left[1 - \frac{2\pi L}{P\alpha_1}\omega_o + \frac{2\pi}{P\alpha_1}\left(\dot{a} - w_a\right)\right] \\ T_2 &= T_{2nom}\left[1 + \frac{2\pi L}{P\alpha_2}\omega_n + \frac{2\pi}{P\alpha_2}\left(\dot{a} - w_a\right)\right] \\ T_3 &= T_{3nom}\left[1 + \frac{2\pi L}{P\alpha_3}\omega_o + \frac{2\pi}{P\alpha_3}\left(\dot{a} - w_a\right)\right] \\ T_4 &= T_{4nom}\left[1 - \frac{2\pi L}{P\alpha_3}\omega_n + \frac{2\pi}{P\alpha_3}\left(\dot{a} - w_a\right)\right] \end{aligned} \tag{C-39}$$

$$
\begin{aligned}
D_1 &= k_d\alpha_1^2\left(1-\frac{2\pi L}{P\alpha_1}\omega_o+\frac{2\pi}{P\alpha_1}\left(\dot{a}-w_a\right)\right)\\
D_2 &= k_d\alpha_2^2\left(1+\frac{2\pi L}{P\alpha_2}\omega_n+\frac{2\pi}{P\alpha_2}\left(\dot{a}-w_a\right)\right)\\
D_3 &= k_d\alpha_3^2\left(1+\frac{2\pi L}{P\alpha_3}\omega_o+\frac{2\pi}{P\alpha_3}\left(\dot{a}-w_a\right)\right)\\
D_4 &= k_d\alpha_4^2\left(1-\frac{2\pi L}{P\alpha_4}\omega_n+\frac{2\pi}{P\alpha_4}\left(\dot{a}-w_a\right)\right)
\end{aligned}
\tag{C-40}
$$

$$
M_{ma} = J_t\left(\dot{\alpha}_1+\dot{\alpha}_3-\dot{\alpha}_2-\dot{\alpha}_4\right) \tag{C-41}
$$

$$
\begin{bmatrix}\dot{\omega}_n\\ \dot{\omega}_o\\ \dot{\omega}_a\end{bmatrix} = J^{-1}\begin{bmatrix}L\left(T_4-T_2\right)\\ L\left(T_1-T_3\right)\\ D_1+D_3-D_2-D_4+M_{ma}\end{bmatrix}+J^{-1}\begin{bmatrix}\tau_n\\ \tau_o\\ \tau_a\end{bmatrix}+J^{-1}\begin{bmatrix}\omega_n\\ \omega_o\\ \omega_a\end{bmatrix}xJ\begin{bmatrix}\omega_n\\ \omega_o\\ \omega_a\end{bmatrix} \tag{C-42}
$$

These nonlinear equations are of the final form $\dot{q} = f\left(q,u,v\right)$ where

$$
q=\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\\ x\\ y\\ z\\ \dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\\ \phi\\ \theta\\ \psi\\ \alpha_1\\ \alpha_2\\ \alpha_3\\ \alpha_4\end{bmatrix}\quad u=\begin{bmatrix}V_1\\ V_2\\ V_3\\ V_4\end{bmatrix}\quad v=\begin{bmatrix}w_x\\ w_y\\ w_z\\ \tau_n\\ \tau_o\\ \tau_a\end{bmatrix} \tag{C-43}
$$

System Parameters Key

k_d= coefficient relating yaw torque caused by prop drag to prop RPM – prop drag coeff

k_t= coefficient relating prop thrust to prop RPM – prop thrust coeff

k_m= coefficient relating changes in prop angular momentum to vehicle yaw torque – single prop moment of inertia

k_q= coefficient defining the torque/speed curve of the prop motor – defines slope of torque/speed curve – see k_b for solving

k_b= coefficient relating input voltage to prop motor performance – k_b/k_q = Motor rad/s/volt – k_bk_m = torque/volt

k_r= coefficient defining how retreating/advancing blade effects produce moments on the props when the vehicle is moving in yaw – approx equal to Lk_a – Approaches exactly equal when L is sufficiently large to allow for negligible differences in prop blade velocity on the outside of the prop hub and on the inside when the vehicle is yawing

k_w= coefficient defining how prop thrust varies with changes in effective prop pitch due to vehicle roll and pitch – approx equal to Lk_p – Approaches exactly equal when L is sufficiently large to allow for negligible differences in prop blade velocity on the outside of the prop hub and on the inside when the vehicle is rolling or pitching

k_g= coefficient defining how prop thrust varies with advancing/retreating blade effects due to vehicle yaw – approx equal to L^2k_e - Exactly equal when L is sufficiently large to allow for negligible differences in prop blade velocity on the outside of the prop hub and on the inside when the vehicle is yawing

k_j= coefficient defining how prop thrust varies with advancing/retreating blade effects due to vehicle yaw – approx equal to L^2k_h - Exactly equal when L is sufficiently large to allow for negligible differences in prop blade velocity on the outside of the prop hub and on the inside when the vehicle is yawing - equal to k_gk_d/k_t?

k_s= coefficient relating lateral wind velocity to disturbance force (P/A) – air momentum at certain airspeed hitting certain vehicle cross sectional area, plus drag

k_u= coefficient relating vertical wind velocity to disturbance force (P/A) – air momentum at certain airspeed hitting certain vehicle cross sectional area, plus drag

J= vehicle mass moment of inertia matrix

m = mass of the vehicle

L= radial distance of prop centers from vehicle center of mass

g = gravity

State Variables Key

α_i= angular velocity of the four props

ω_i= vehicle angular velocity about body coordinate axis i – sensor input

V_i= voltage applied to motor i – control output

ψ= Euler yaw angle of body coordinate axis relative to space coordinate axis

θ= Euler pitch angle of body coordinate axis relative to space coordinate axis

ϕ= Euler roll angle of body coordinate axis relative to space coordinate axis

x,y,z = position of body coordinate axis relative to space coordinate axis origin

ndot, odot, adot= rate of change of body linear coordinates (body velocity relative to body coordinate axes)

τ_i= disturbance torque about body coordinate axis i

w_i= disturbance wind velocity in space coordinate axis i, in units of ndot, odot, and idot

Measurement Model

Ideally perfect measurements of the full state would be available for control. However, sensors instead produce slightly biased, noisy measurements of the vehicle acceleration and angular rates. It is assumed that these measurements have three primary sources of potential error:

- Non-ideal sensor placement
- Sensor drift
- White noise added to final sensor measurement

Scale factor error is another potential source of sensor error, but it will not be estimated due to vehicle operation in a relatively small range of measurement magnitudes and the typically relatively good sensor specs for this parameter.

Sensors are not perfectly placed on the vehicle's body. Ideally all three accelerometers were placed perfectly at the vehicle center of mass. However, in practice, all three accelerometers cannot be located at the same point within an IMU and even if they were, the IMU itself may not be mounted exactly at the vehicle CM. Instead, it is assumed that the accelerometers are offset in the direction they measure by some amount beta. This offset will cause rotational rates to add to the sensor measurement due to the measurement of centrifugal forces.

For generality, it will also be assumed that the IMU may be rotated from true alignment with the AFV principal axes. This simply adds a rotation matrix transforming the true AFV coordinates into the actual IMU coordinates. The measurement will also have a bias offset, DELTA, and white noise, w. The measurement equations then follow.

$$\begin{bmatrix} \omega_{nmeas} \\ \omega_{omeas} \\ \omega_{ameas} \end{bmatrix} = R \begin{bmatrix} \omega_{nactual} \\ \omega_{oactual} \\ \omega_{aactual} \end{bmatrix} + \begin{bmatrix} \Delta_{\omega n} \\ \Delta_{\omega o} \\ \Delta_{\omega a} \end{bmatrix} + \begin{bmatrix} w_{\omega n} \\ w_{\omega o} \\ w_{\omega a} \end{bmatrix} \tag{C-44}$$

solving for actual measurements yields:

$$\begin{bmatrix} \omega_{nactual} \\ \omega_{oactual} \\ \omega_{aactual} \end{bmatrix} = R^{-1} \left(\begin{bmatrix} \omega_{nmeas} \\ \omega_{omeas} \\ \omega_{ameas} \end{bmatrix} - \begin{bmatrix} \Delta_{\omega n} \\ \Delta_{\omega o} \\ \Delta_{\omega a} \end{bmatrix} - \begin{bmatrix} w_{\omega n} \\ w_{\omega o} \\ w_{\omega a} \end{bmatrix} \right) \tag{C-45}$$

Since noise will not be known in the course of measurement and estimation, the equation reduces to the following:

$$\begin{bmatrix} \omega_{nactual} \\ \omega_{oactual} \\ \omega_{aactual} \end{bmatrix} = R^{-1}\left(\begin{bmatrix} \omega_{nmeas} \\ \omega_{omeas} \\ \omega_{ameas} \end{bmatrix} - \begin{bmatrix} \Delta_{\omega n} \\ \Delta_{\omega o} \\ \Delta_{\omega a} \end{bmatrix} \right) \tag{C-46}$$

Having acquired a guess at the actual rates, this information can be backed out of acceleration measurements:

$$\begin{bmatrix} \ddot{n}_{meas} \\ \ddot{o}_{meas} \\ \ddot{a}_{meas} \end{bmatrix} = R\begin{bmatrix} \ddot{n}_{actual} + \beta_n\left(|\omega_{oactual}| + |\omega_{aactual}|\right) \\ \ddot{o}_{actual} + \beta_o\left(|\omega_{nactual}| + |\omega_{aactual}|\right) \\ \ddot{a}_{actual} + \beta_a\left(|\omega_{nactual}| + |\omega_{oactual}|\right) \end{bmatrix} + \begin{bmatrix} \Delta_n \\ \Delta_o \\ \Delta_a \end{bmatrix} + \begin{bmatrix} w_n \\ w_o \\ w_a \end{bmatrix} \tag{C-47}$$

solving for actual measurements yields:

$$\begin{bmatrix} \ddot{n}_{actual} \\ \ddot{o}_{actual} \\ \ddot{a}_{actual} \end{bmatrix} = R^{-1}\left(\begin{bmatrix} \ddot{n}_{meas} \\ \ddot{o}_{meas} \\ \ddot{a}_{meas} \end{bmatrix} - \begin{bmatrix} \Delta_n \\ \Delta_o \\ \Delta_a \end{bmatrix} - \begin{bmatrix} w_n \\ w_o \\ w_a \end{bmatrix} \right) - \begin{bmatrix} \beta_n\left(|\omega_{oactual}| + |\omega_{aactual}|\right) \\ \beta_o\left(|\omega_{nactual}| + |\omega_{aactual}|\right) \\ \beta_a\left(|\omega_{nactual}| + |\omega_{oactual}|\right) \end{bmatrix} \tag{C-48}$$

With noise assumed zero,

$$\begin{bmatrix} \ddot{n}_{actual} \\ \ddot{o}_{actual} \\ \ddot{a}_{actual} \end{bmatrix} = R^{-1}\left(\begin{bmatrix} \ddot{n}_{meas} \\ \ddot{o}_{meas} \\ \ddot{a}_{meas} \end{bmatrix} - \begin{bmatrix} \Delta_n \\ \Delta_o \\ \Delta_a \end{bmatrix} \right) - \begin{bmatrix} \beta_n\left(|\omega_{oactual}| + |\omega_{aactual}|\right) \\ \beta_o\left(|\omega_{nactual}| + |\omega_{aactual}|\right) \\ \beta_a\left(|\omega_{nactual}| + |\omega_{oactual}|\right) \end{bmatrix} \tag{C-49}$$

The offsets themselves have drift which can be modeled as driven by white noise. This modeling is only necessary for simulation of sensor corruption. There is a feedback term, kappa, which places a bound on the amount the parameter can drift from its initial value. In the following equations, Del tilda is a deviation from the initial parameter value. The bound placed on the drift is described by the ratio of kappa to the white noise power. If both are raised, the drift will be jagged, but will not go far. If both are lowered, the drift will appear smoother. If kappa is raised and the power lowered, the bounding range is decreased.

$$\begin{bmatrix} \dot{\Delta}_n \\ \dot{\Delta}_o \\ \dot{\Delta}_a \end{bmatrix} = -\kappa_{noa}\begin{bmatrix} \tilde{\Delta}_n \\ \tilde{\Delta}_o \\ \tilde{\Delta}_a \end{bmatrix} + \begin{bmatrix} r_n \\ r_o \\ r_a \end{bmatrix} \tag{C-50}$$

$$\begin{bmatrix} \dot{\Delta}_{\omega n} \\ \dot{\Delta}_{\omega o} \\ \dot{\Delta}_{\omega a} \end{bmatrix} = -\kappa_{\omega}\begin{bmatrix} \tilde{\Delta}_{\omega n} \\ \tilde{\Delta}_{\omega o} \\ \tilde{\Delta}_{\omega a} \end{bmatrix} + \begin{bmatrix} r_{\omega n} \\ r_{\omega o} \\ r_{\omega a} \end{bmatrix} \tag{C-51}$$

In order to convert the above measurements from local to global coordinates, the proper rotation given the current state must be applied:

$$\begin{bmatrix} x \\ y \\ y \end{bmatrix} = A \begin{bmatrix} n \\ o \\ a \end{bmatrix} \tag{C-52}$$

where A is as defined in the vehicle nonlinear dynamics.

The primary shortcoming of the measurements provided thus far is their failure to provide any absolute information about vehicle velocity and position. This information is unobservable given inertial measurements and the vehicle dynamics. While estimation of the above sensor parameters can reduce the accrual of errors in these states, there is no way to remove error from the system once it is accumulated. This will cause the vehicle to drift increasingly with time. A final version of the vehicle could utilize GPS to provide absolute position information. However, GPS restricts the testing of the vehicle to outdoors. Instead, there will be a human controlling a joystick attempting to simply cancel out drift. The signals sent from this human interface device will be treated both as an outer loop control signal (to be discussed later) and a measurement of absolute position and velocity, yaw and yaw rate.

The human observer will have two two-axis thumbsticks, allowing for the transmission of four data values each step. One thumbstick will be linked to vehicle x and y axes while the other thumbstick will control z and yaw. A typical human observer attempting to control a state can be modeled as a PD controller with variation in the PD gains from step to step. Since x and y are controlled from the same thumbstick and are identical in how they respond dynamically, it can be assumed that they have identical PD gains. The measurement obtained from the human observer can be modeled by the following relationship:

$$\begin{bmatrix} TS_x \\ TS_y \\ TS_z \\ TS_\psi \end{bmatrix} = \begin{bmatrix} D_{xy} & 0 & 0 & P_{xy} & 0 & 0 & 0 & 0 \\ 0 & D_{xy} & 0 & 0 & P_{xy} & 0 & 0 & 0 \\ 0 & 0 & D_z & 0 & 0 & P_z & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & D_\psi & P_\psi \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ x \\ y \\ z \\ \dot{\psi} \\ \psi \end{bmatrix} + \begin{bmatrix} nt_x \\ nt_y \\ nt_z \\ nt_\psi \end{bmatrix} \tag{C-53}$$

where nt_is are measurement noise. While there is no true measurement noise, the measurement noise nt_i can be used to handle the variations in PD gains inherent in a human operator.

APPENDIX D:

ASSEMBLY/DISASSEMBLY INSTRUCTIONS

While there should be no regular need for mechanical maintenance of the AFV, there may arise a need to disassemble the vehicle for the servicing of damaged parts in the event of a crash or component failure. If this becomes necessary, the following guide should help.

Pulley Box Removal/Replacement

The pulley box assembly is a self-contained unit. It can be swapped directly with another of the same prop type without adjustment of the tensioning wires. When removing a pulley box assembly, simply unscrew the four mounting screws. When re-mounting an assembly, replace and tighten these four screws. Be careful to tighten all four screws at roughly the same rate and to the same final tension. Insufficient tension will result in vibration of the pulley box assembly. Excess tension may shear the screw head off. Some minor adjustments to the angle at which the pulley box mates with the vehicle may be made by tightening the bottom two screws more or less than the top two screws. Two vibration isolation rings should be used between the strut end and the pulley box extension piece, slipped around the mounting screws.

IMU Removal/Replacement

In order to remove the IMU, first remove tension on all three wire turnbuckles until the wires are close to slack. Loosen the four wire clamp screws in the bottom center strut end and slip the wires off of the strut. The IMU should slide down and out. The IMU can now be removed from its plastic mount. In order to reassemble, follow the disassembly instructions in reverse. Re-tension all three wires before tightening the four wire clamp screws. It may be necessary to slacken the wires, adjust the bottom strut slightly, and re-tension the wires in order to ensure that the base strut is properly vertical. Once this is done, tension the wires as far as the turnbuckles allow without significant resistance. No additional readjustment should be necessary assuming no other wire clamp screws were touched.

Battery Replacement

Battery changing is a fairly straightforward task. The four motor battery packs are removed by loosening the retaining screw on top of the mounts and twisting the retaining bar parallel to the vehicle structural struts. The EE battery pack is held in place by a simple friction fit. It can be removed by pushing the pack out the one side of the mounting clip that does not have a retaining bump. When replacing the motor battery packs or hooking the motor boards up to the power tether, the smaller connectors should always be attached first, and contact verified, before connecting the large connectors. The small connector has a resistor in line to prevent sparking between connectors as the capacitors onboard the motor control boards charge rapidly in response to applied voltage.

Pulley Box Disassembly/Reassembly

If it becomes necessary to disassemble a pulley box assemble, the prop must first be removed. Once the retaining nut on the prop shaft has been removed, discard it. With the prop and the prop shaft washer beneath it removed, the encoder can be accessed and disassembled. LocTite removal solution should be applied to the large pulley set screw, and the set screw removed. The shaft collar at the base of the prop shaft is then removed. The prop shaft can now be pulled upwards out of the pulley box assembly. From here, the large pulley or the prop shaft bearings can be replaced or serviced. Removal or service of the motor, small pulley, belt, or motor shaft bearing requires removal of the motor. To dismount the motor, first loosen the small pulley set screw using the LocTite removal solution. Slowly unscrew the motor mounting screws, keeping the unscrewed length equal across both. Keep the motor pulled as far down from the pulley box as possible to avoid jamming a mounting screw against the small pulley. Re-assemble by following the above directions in reverse. Some slight sanding of the motor and prop shaft may be necessary to remove burs from set screw marring. Clean pulley set screw holes thoroughly as residual removal solution may prevent fresh LocTite from setting properly. Replacement of the large plastic pulley may be necessary if the removal solution has degraded the plastic of the pulley. Apply fresh LocTite to set screws when they are replaced, ensuring that the set screws line up with any flats on the shafts they mount to. Do not run the motor until the LocTite has set completely. Be sure to use a fresh locknut on the prop shaft when assembly is complete as the deformable nylon insert is not reusable.

Should any disassembly beyond what is described above be required, some freshly machined parts may be necessary. Parts joined with spring pins are likely unable to be disassembled and reassembled without damage to the involved parts. Loose connections should not be tolerated as they will affect the integrity and resonant frequency of the structure.

APPENDIX E:

ELECTRONIC CONTENT

Data CD Contents

The AFVMechECD contains documentation and files relating to the mechanical design and simulation aspects of the Cornell AFV project. Software packages utilized included ANSYS for finite element analysis, MATLAB 6.5 for analysis and simulation, Simulink for simulation, Microsoft Excel 2002 for analysis and documentation, and Microsoft Word 2002 for documentation. CD navigation should be self-explanatory.

- AFVMechECD
  - Analysis&Simulation
    - 2dsim
      - *AFVmodelconsts.m*
      - *TwoDmodstatederv.mdl*
    - 3dsim
      - afv animation
        - *afv4.bmp*
        - *animate_afv.m*
        - *display_afv.m*
        - *make_afv.m*
        - *myrot.m*
        - *rotobj.m*
      - estimation
        - measurement
          - *hfunmine.m*
          - *IMUgeometry_f.m*
        - prediction
          - *ffunmine.m*
          - *ThreeDAFVstatedervNoVolts*
        - *currfilterest.m*
        - *estimatestate.m*
        - *srspf.m*
      - initialization
        - *ThreeDAFVmodelconsts.m*
        - *ThreeDAFVstatedervThrust.m*
      - old controllers
        - *misc controllers, number corresponds to data log number*
      - old versions
        - *Misc old versions not necessarily working.zip*
        - *Working 3dsim_EKF without LQR.zip*

- simulation
  - *IMUgeometry.m*
  - *ThreeDAFVstatedervNew.m*
- *sensortuning.mdl*
- *Sim File Hierarchy.xls*
- *ThreeDAFVsimworkingvelocity.mdl*
- *thrusttorads.m*

- Structure
  - 4 prop
    - *FourPropsStructureAnalysis.m*
  - 8 prop
    - *EightPropAFVPlotted3d.m*
    - *EightPropsStructureAnalysis.m*
    - *Structure.db/b*
- *motor analysis.xls*

Documentation

- 2001
  - *AFV Vision System.doc*
  - *Mechanical and Aerodynamic System Design.doc*
  - *Design&ImpofControl&SensingforCUAFV.doc*
- 2003-2004

  - Designof4RotHoverVehicle

    - Figures
      - Components Appendix
        - *Misc components info figures*
      - Simulation Plots
        - *Misc simulation plot BMPs*
      - *Misc document figure BMPs*

    - Part Data
      - Prop Data
        - *18X6 APC DATA.DAT*
        - *prop thrust-drag tests.xls*
      - Pulley Data
        - *Misc pulley/belt info*

    - Preliminary Documents
      - *AFV brainstorming.doc*
      - *AFV eqns motion and measurement.doc*
      - *Flowcharts for contro-estimation.vsd*

Designof4RotHoverVehicle
Defense.ppt
DocsTestCase.avi
ActualHoverTest.avi

AFV Electronics Documentation.doc

Machining Drawings
Misc Machining Spec Sheet BMPs

ProE
Misc Pro/E model files

Prop Programs
PropSelector.zip
Thrusthpv20d.zip

A printout of all code utilized for design or simulation follows.

4-prop Structure Analysis Code

```
% FourPropsStructureAnalysis.m
% Author: Eryk Nice

% This m file will perform a simplified analysis of the natural frequency
% of structure flexible modes given the four-prop wire stiffened design.
% These computed values for natural frequency must be significantly higher
% than the highest expected prop operating frequency to avoid problems with
% structural interaction with any cyclic prop forces.

% set to 1 if you wish to display prop coordinates
dispropco = 0;

%Tip to tip prop clearence - defines minimum radius of props from vehicle
%center
tttpc = 2*25.4/1000;

%Thrust prop radius
tpropr = 9*25.4/1000;

%width of thrust motor mount block from center - affects rotational prop
%mode
ttzwidth = 2*25.4/1000;

% minimum distance from prop tip to wire - vertical clearence of prop above
% wire
pttwdist = 2*25.4/1000;
% height of vertical strut above center - affects up and down mode of prop
zheight = 5*25.4/1000;
[0 0 zheight]';
% height of thrust motor mount block above center
tmzheight = 0.5*25.4/1000;

% computer the x,y coordinates of four props given above parameters
disp('Thrust prop coordinates')
tprop1=[tpropr+tttpc/2, tpropr+tttpc/2];
tprop2=[tpropr+tttpc/2, -tpropr-tttpc/2];
tprop3=[-tpropr-tttpc/2, -tpropr-tttpc/2];
tprop4=[-tpropr-tttpc/2, tpropr+tttpc/2];

% if flag is true, display coordinates
if dispropco == 1
[tpropr+tttpc/2  tpropr+tttpc/2 0]'
[tpropr+tttpc/2  -tpropr-tttpc/2 0]'
[-tpropr-tttpc/2  -tpropr-tttpc/2 0]'
[-tpropr-tttpc/2  tpropr+tttpc/2 0]'

[tpropr+tttpc/2  tpropr+tttpc/2 tmzheight/2]'
[tpropr+tttpc/2  -tpropr-tttpc/2 tmzheight/2]'
[-tpropr-tttpc/2  -tpropr-tttpc/2 tmzheight/2]'
[-tpropr-tttpc/2  tpropr+tttpc/2 tmzheight/2]'

[tpropr+tttpc/2  tpropr+tttpc/2 -tmzheight/2]'
```

```
[tpropr+tttpc/2  -tpropr-tttpc/2 -tmzheight/2]'
[-tpropr-tttpc/2  -tpropr-tttpc/2 -tmzheight/2]'
[-tpropr-tttpc/2  tpropr+tttpc/2 -tmzheight/2]'
end

%Thrust prop radial distance
tproprdist = (2*(tpropr+tttpc/2)^2)^0.5

%Circles for plotting
tproppts1 = mkcirc(tprop1(1),tprop1(2),tpropr);
tproppts2 = mkcirc(tprop2(1),tprop2(2),tpropr);
tproppts3 = mkcirc(tprop3(1),tprop3(2),tpropr);
tproppts4 = mkcirc(tprop4(1),tprop4(2),tpropr);


%Plot props
figure(1)
plot(tproppts1(:,1),tproppts1(:,2))
hold on
plot(tproppts2(:,1),tproppts2(:,2))
plot(tproppts3(:,1),tproppts3(:,2))
plot(tproppts4(:,1),tproppts4(:,2))

%Plot IMU
plot([0 2.5*25.4/1000 0 -2.5*25.4/1000 0],[2.5*25.4/1000 0 -2.5*25.4/1000 0 2.5*25.4/1000])

%Plot struts to prop centers
plot([0 tprop1(1)],[0 tprop1(2)])
plot([0 tprop2(1)],[0 tprop2(2)])
plot([0 tprop3(1)],[0 tprop3(2)])
plot([0 tprop4(1)],[0 tprop4(2)])

disp('plot thrust motor mounts')
plot([tprop1(1)-ttzwidth*sin(pi/4) tprop1(1)+ttzwidth*sin(pi/4)],[tprop1(2)+ttzwidth*sin(pi/4)
tprop1(2)-ttzwidth*sin(pi/4)])
plot([tprop2(1)+ttzwidth*sin(pi/4) tprop2(1)-ttzwidth*sin(pi/4)],[tprop2(2)+ttzwidth*sin(pi/4)
tprop2(2)-ttzwidth*sin(pi/4)])
plot([tprop3(1)+ttzwidth*sin(pi/4) tprop3(1)-ttzwidth*sin(pi/4)],[tprop3(2)-ttzwidth*sin(pi/4)
tprop3(2)+ttzwidth*sin(pi/4)])
plot([tprop4(1)-ttzwidth*sin(pi/4) tprop4(1)+ttzwidth*sin(pi/4)],[tprop4(2)-ttzwidth*sin(pi/4)
tprop4(2)+ttzwidth*sin(pi/4)])

if dispropco == 1
[tprop1(1)-ttzwidth*sin(pi/4) tprop1(1)+ttzwidth*sin(pi/4);tprop1(2)+ttzwidth*sin(pi/4) tprop1(2)-
ttzwidth*sin(pi/4);tmzheight/2 tmzheight/2]
[tprop2(1)+ttzwidth*sin(pi/4) tprop2(1)-ttzwidth*sin(pi/4);tprop2(2)+ttzwidth*sin(pi/4) tprop2(2)-
ttzwidth*sin(pi/4);tmzheight/2 tmzheight/2]
[tprop3(1)+ttzwidth*sin(pi/4) tprop3(1)-ttzwidth*sin(pi/4);tprop3(2)-ttzwidth*sin(pi/4)
tprop3(2)+ttzwidth*sin(pi/4);tmzheight/2 tmzheight/2]
[tprop4(1)-ttzwidth*sin(pi/4) tprop4(1)+ttzwidth*sin(pi/4);tprop4(2)-ttzwidth*sin(pi/4)
tprop4(2)+ttzwidth*sin(pi/4);tmzheight/2 tmzheight/2]
[tprop1(1)-ttzwidth*sin(pi/4) tprop1(1)+ttzwidth*sin(pi/4);tprop1(2)+ttzwidth*sin(pi/4) tprop1(2)-
ttzwidth*sin(pi/4);-tmzheight/2 -tmzheight/2]
[tprop2(1)+ttzwidth*sin(pi/4) tprop2(1)-ttzwidth*sin(pi/4);tprop2(2)+ttzwidth*sin(pi/4) tprop2(2)-
ttzwidth*sin(pi/4);-tmzheight/2 -tmzheight/2]
```

```
[tprop3(1)+ttzwidth*sin(pi/4) tprop3(1)-ttzwidth*sin(pi/4);tprop3(2)-ttzwidth*sin(pi/4)
tprop3(2)+ttzwidth*sin(pi/4);-tmzheight/2 -tmzheight/2]
[tprop4(1)-ttzwidth*sin(pi/4) tprop4(1)+ttzwidth*sin(pi/4);tprop4(2)-ttzwidth*sin(pi/4)
tprop4(2)+ttzwidth*sin(pi/4);-tmzheight/2 -tmzheight/2]
end


%Plot Circumferential wire
plot([tprop1(1)+ttzwidth*sin(pi/4) tprop2(1)+ttzwidth*sin(pi/4)],[tprop1(2)-ttzwidth*sin(pi/4)
tprop2(2)+ttzwidth*sin(pi/4)])
plot([tprop2(1)-ttzwidth*sin(pi/4) tprop3(1)+ttzwidth*sin(pi/4)],[tprop2(2)-ttzwidth*sin(pi/4)
tprop3(2)-ttzwidth*sin(pi/4)])
plot([tprop3(1)-ttzwidth*sin(pi/4) tprop4(1)-ttzwidth*sin(pi/4)],[tprop3(2)+ttzwidth*sin(pi/4)
tprop4(2)-ttzwidth*sin(pi/4)])
plot([tprop4(1)+ttzwidth*sin(pi/4) tprop1(1)-ttzwidth*sin(pi/4)],[tprop4(2)+ttzwidth*sin(pi/4)
tprop1(2)+ttzwidth*sin(pi/4)])

%Compute length of circumferencial wire for weight purposes
circumfwirelength = distance(tprop1(1)+ttzwidth*sin(pi/4), tprop2(1)+ttzwidth*sin(pi/4), tprop1(1)-
ttzwidth*sin(pi/4), tprop1(2)+ttzwidth*sin(pi/4));
circumftotalwirelength = 4*circumfwirelength;

flatanglethrustprops = pi/4;

axis([-0.4 0.4 -0.4 0.4])
axis square
grid on
title('Prop layout (meters)')
hold off

%Height of thrust prop above strut - function of clearence and wire angle
tpzheight = tpropr*(zheight-tmzheight)/tproprdist + pttwdist + tmzheight;


figure(2)
%plot thrust props
plot([(-(tpropr+tttpc/2)-tpropr) (-(tpropr+tttpc/2)+tpropr)],[tpzheight tpzheight])
hold on
plot([((tpropr+tttpc/2)-tpropr) ((tpropr+tttpc/2)+tpropr)],[tpzheight tpzheight])
%plot IMU
plot([3.53/2*25.4/1000 -3.53/2*25.4/1000 -3.53/2*25.4/1000 3.53/2*25.4/1000 3.53/2*25.4/1000],[-
1.77*25.4/1000 -1.77*25.4/1000 (3.84-1.77)*25.4/1000 (3.84-1.77)*25.4/1000 -1.77*25.4/1000]);

%plot thrust prop verts
plot([-(tpropr+tttpc/2) -(tpropr+tttpc/2)],[ 0 tpzheight])
plot([(tpropr+tttpc/2) (tpropr+tttpc/2)],[0 tpzheight])
%plot motor mount block
plot([-(tpropr+tttpc/2) -(tpropr+tttpc/2)],[-tmzheight  tmzheight])
plot([(tpropr+tttpc/2) (tpropr+tttpc/2)],[-tmzheight  tmzheight])
%plot thrust struts
plot([-(tpropr+tttpc/2) (tpropr+tttpc/2)],[0 0])
%plot manuever wires
plot([-(tpropr+tttpc/2) 0 (tpropr+tttpc/2)],[tmzheight zheight tmzheight])
plot([(tpropr+tttpc/2) 0 -(tpropr+tttpc/2)],[-tmzheight -zheight -tmzheight])
%plot verts
```

```
plot([0 0],[zheight -zheight])
anglethrustprops = atan((zheight-tmzheight)/tproprdist);

%Compute length of vertical stiffening wires for weight purposes
thrustwirelength = distance(0, zheight, tproprdist, tmzheight);
thrusttotalwirelength = 8*thrustwirelength;

axis([-0.4 0.4 -0.4 0.4])
axis square
grid on
title('Thrust prop side views (meters)')
hold off

totalwirelength=thrusttotalwirelength+circumftotalwirelength + 8*ttzwidth
totalstrutlength=zheight*2+tproprdist*4;


%wire info
%wirer=0.000865;  %5/64 1-19
wirer=0.000692;  %1/16 1-19

%compute wire weight
wiredens = 7920;
wireA=3.14159*wirer*wirer;
wirev=wireA*totalwirelength;
wireweight = wiredens*wirev

wireE = 190000000000;
freqthrustprop = 7450/60

%motor & prop weight
tmasskg = .5;

%manuever motor & prop rotational inertia
tmassI = 0.5*0.03^2;


%Beam info (strut tubing)
beamE = 70000000000;
beamG = 26000000000;
beamor = 3/5*25.4/1000;
beamwallthickness = .028*25.4/1000;
beamir = beamor-beamwallthickness;
beamI = 0.25*pi*(beamor^4-beamir^4);
beamJ = 0.5*pi*(beamor^4-beamir^4);
beamdens = 2710;
beamA = pi*(beamor^2-beamir^2);
beamv = beamA*totalstrutlength;
beamweight = beamv*beamdens

thrustbeamL = tproprdist;

beamstrength = 120000000;

% assume we load the beam in compression to half of its max strength, and
```

```
% use this to compute the tension of the stiffening wire
beamload = beamstrength*beamA/2;
verttension = beamload/2/cos(anglethrustprops);
circumftension = beamload/2/cos(0.6172);

% The following computations examine a single cantelevered prop
% configuration assuming the center end of the strut is a fixed constraint
% and the four wires attached to the gear box are fixed at their other end.
%  The frequency is approximated by displacing the motor/prop combination
%  in the vertical, tangential, or radial direction or rotating about these
%  three directions.  The force produced by the spring of the stiffening
%  wires or the spring of the strut is used to compute an effective spring
%  constant.  This spring constant is them combined with the mass of the
%  motor/prop combination and a natural frequency is obtained.  The
%  assumption is made that the bulk of the relevant weight is found in the
%  motor/prop combo.

% VERTICAL

kzthrustwireflat = (2*circumftension/circumfwirelength);
kzthrustwirevert = 2*wireE*wireA*sin(anglethrustprops)*sin(anglethrustprops)/thrustwirelength;
kzthrustwire = kzthrustwireflat + kzthrustwirevert;
kzthruststrut = 3*beamE*beamI/(thrustbeamL)^3;

kzthrust = kzthrustwire + kzthruststrut;

znaturalfreqthruststrut = (kzthrust/tmasskg)^0.5/2/pi

% TANGENTIAL

ktthrustwireflat =
(2*wireE*wireA*sin(flatanglethrustprops)*sin(flatanglethrustprops)/thrustwirelength);
ktthrustwirevert = 2*verttension/thrustwirelength;
ktthrustwire =  ktthrustwireflat + ktthrustwirevert;
ktthruststrut = 3*beamE*beamI/(thrustbeamL)^3;

ktthrust = ktthrustwire + ktthruststrut;

tnaturalfreqthruststrut = (ktthrust/tmasskg)^0.5/2/pi

% RADIAL

krthrustwireflat =
(2*wireE*wireA*cos(flatanglethrustprops)*cos(flatanglethrustprops)/circumfwirelength);
krthrustwirevert = 2*wireE*wireA*cos(anglethrustprops)*cos(anglethrustprops)/thrustwirelength;
krthrustwire = krthrustwireflat + krthrustwirevert;
krthruststrut = beamE*beamA/thrustbeamL;

krthrust = krthrustwire + krthruststrut;

rnaturalfreqthruststrut = (krthrust/tmasskg)^0.5/2/pi

%Torsional about radius

tkrthrustwireflat = 0.5*(2*circumftension/circumfwirelength)*ttzwidth^2;
```

```
tkrthrustwirevert = 0.5*2*verttension/thrustwirelength*tmzheight^2;
tkrthrustwire = tkrthrustwireflat + tkrthrustwirevert;
tkrthruststrut = beamG*beamJ/thrustbeamL;

tkrthrust = tkrthrustwire + tkrthruststrut;

trnaturalfreqthruststrut = (tkrthrust/tmassI)^0.5/2/pi

%Torsional about tangent

tktthrustwireflat =
0.5*(2*wireE*wireA*cos(flatanglethrustprops)*cos(flatanglethrustprops)/circumfwirelength)*ttzwidth^
2;
tktthrustwirevert =
0.5*2*wireE*wireA*cos(anglethrustprops)*cos(anglethrustprops)/thrustwirelength*tmzheight^2;
tktthrustwire = tktthrustwireflat + tktthrustwirevert;
tktthruststrut = beamE*beamI/thrustbeamL;

tktthrust = tktthrustwire + tktthruststrut;

ttnaturalfreqthruststrut = (tktthrust/tmassI)^0.5/2/pi

%Torsional about vertical

tkzthrustwireflat =
0.5*(2*wireE*wireA*cos(flatanglethrustprops)*cos(flatanglethrustprops)/circumfwirelength)*ttzwidth^
2;
tkzthrustwirevert = 0;
tkzthrustwire = tkzthrustwireflat + tkzthrustwirevert;
tkzthruststrut = beamE*beamI/thrustbeamL;

tkzthrust = tkzthrustwire + tkzthruststrut;

tznaturalfreqthruststrut = (tkzthrust/tmassI)^0.5/2/pi
```

8-prop Structure Analysis Code

```
% EightPropsStructureAnalysis.m

dispropco = 0;


%Tip to tip prop clearence
tttpc = 2*25.4/1000;

%Thrust prop radius
tpropr = 10*25.4/1000;

%Manuever prop radius
mpropr = 7*25.4/1000;

%width of manuever motor mount block from center
mtzwidth = 3*25.4/1000;
%width of thrust motor mount block from center
ttzwidth = 4*25.4/1000;

%distance from prop tip to wire
pttwdist = 2*25.4/1000;
%height of vertical strut above center
zheight = 12*25.4/1000;
[0 0 zheight]'
%height of manuever motor mount block above center
mmzheight = 0.5*25.4/1000;
%height of thrust motor mount block above center
tmzheight = 0.5*25.4/1000;

disp('Thrust prop coordinates')
tprop1=[tpropr+tttpc/2, tpropr+tttpc/2];
tprop2=[tpropr+tttpc/2, -tpropr-tttpc/2];
tprop3=[-tpropr-tttpc/2, -tpropr-tttpc/2];
tprop4=[-tpropr-tttpc/2, tpropr+tttpc/2];

if dispropco == 1
[tpropr+tttpc/2  tpropr+tttpc/2 0]'
[tpropr+tttpc/2  -tpropr-tttpc/2 0]'
[-tpropr-tttpc/2  -tpropr-tttpc/2 0]'
[-tpropr-tttpc/2  tpropr+tttpc/2 0]'

[tpropr+tttpc/2  tpropr+tttpc/2 tmzheight/2]'
[tpropr+tttpc/2  -tpropr-tttpc/2 tmzheight/2]'
[-tpropr-tttpc/2  -tpropr-tttpc/2 tmzheight/2]'
[-tpropr-tttpc/2  tpropr+tttpc/2 tmzheight/2]'

[tpropr+tttpc/2  tpropr+tttpc/2 -tmzheight/2]'
[tpropr+tttpc/2  -tpropr-tttpc/2 -tmzheight/2]'
[-tpropr-tttpc/2  -tpropr-tttpc/2 -tmzheight/2]'
[-tpropr-tttpc/2  tpropr+tttpc/2 -tmzheight/2]'
end
```

```
%Thrust prop radial distance
tproprdist = (2*(tpropr+tttpc/2)^2)^0.5;

%Manuever prop radial distance
mproprdist = ((tpropr+mpropr+tttpc)^2 - (tpropr+tttpc/2)^2)^0.5 + tpropr+tttpc/2;


disp('Manuever prop coordinates')
mprop1=[0, mproprdist];
mprop2=[mproprdist, 0];
mprop3=[0, -mproprdist];
mprop4=[-mproprdist, 0];

if dispropco == 1
[0 mproprdist 0]'
[mproprdist 0 0]'
[0 -mproprdist 0]'
[-mproprdist 0 0]'

[0 mproprdist mmzheight/2]'
[mproprdist 0 mmzheight/2]'
[0 -mproprdist mmzheight/2]'
[-mproprdist 0 mmzheight/2]'

[0 mproprdist -mmzheight/2]'
[mproprdist 0 -mmzheight/2]'
[0 -mproprdist -mmzheight/2]'
[-mproprdist 0 -mmzheight/2]'
end

%Circles for plotting
tproppts1 = mkcirc(tprop1(1),tprop1(2),tpropr);
tproppts2 = mkcirc(tprop2(1),tprop2(2),tpropr);
tproppts3 = mkcirc(tprop3(1),tprop3(2),tpropr);
tproppts4 = mkcirc(tprop4(1),tprop4(2),tpropr);

mproppts1 = mkcirc(mprop1(1),mprop1(2),mpropr);
mproppts2 = mkcirc(mprop2(1),mprop2(2),mpropr);
mproppts3 = mkcirc(mprop3(1),mprop3(2),mpropr);
mproppts4 = mkcirc(mprop4(1),mprop4(2),mpropr);


%Plot props
figure(1)
plot(tproppts1(:,1),tproppts1(:,2))
hold on
plot(tproppts2(:,1),tproppts2(:,2))
plot(tproppts3(:,1),tproppts3(:,2))
plot(tproppts4(:,1),tproppts4(:,2))

plot(mproppts1(:,1),mproppts1(:,2))
plot(mproppts2(:,1),mproppts2(:,2))
plot(mproppts3(:,1),mproppts3(:,2))
plot(mproppts4(:,1),mproppts4(:,2))
```

```
%Plot IMU
plot([0 2.5*25.4/1000 0 -2.5*25.4/1000 0],[2.5*25.4/1000 0 -2.5*25.4/1000 0 2.5*25.4/1000])

%Plot struts to prop centers
plot([0 tprop1(1)],[0 tprop1(2)])
plot([0 tprop2(1)],[0 tprop2(2)])
plot([0 tprop3(1)],[0 tprop3(2)])
plot([0 tprop4(1)],[0 tprop4(2)])

plot([0 mprop1(1)],[0 mprop1(2)])
plot([0 mprop2(1)],[0 mprop2(2)])
plot([0 mprop3(1)],[0 mprop3(2)])
plot([0 mprop4(1)],[0 mprop4(2)])

disp('plot thrust motor mounts')
plot([tprop1(1)-ttzwidth*sin(pi/4) tprop1(1)+ttzwidth*sin(pi/4)],[tprop1(2)+ttzwidth*sin(pi/4)
tprop1(2)-ttzwidth*sin(pi/4)])
plot([tprop2(1)+ttzwidth*sin(pi/4) tprop2(1)-ttzwidth*sin(pi/4)],[tprop2(2)+ttzwidth*sin(pi/4)
tprop2(2)-ttzwidth*sin(pi/4)])
plot([tprop3(1)+ttzwidth*sin(pi/4) tprop3(1)-ttzwidth*sin(pi/4)],[tprop3(2)-ttzwidth*sin(pi/4)
tprop3(2)+ttzwidth*sin(pi/4)])
plot([tprop4(1)-ttzwidth*sin(pi/4) tprop4(1)+ttzwidth*sin(pi/4)],[tprop4(2)-ttzwidth*sin(pi/4)
tprop4(2)+ttzwidth*sin(pi/4)])

if dispropco == 1
[tprop1(1)-ttzwidth*sin(pi/4) tprop1(1)+ttzwidth*sin(pi/4);tprop1(2)+ttzwidth*sin(pi/4) tprop1(2)-
ttzwidth*sin(pi/4);tmzheight/2 tmzheight/2]
[tprop2(1)+ttzwidth*sin(pi/4) tprop2(1)-ttzwidth*sin(pi/4);tprop2(2)+ttzwidth*sin(pi/4) tprop2(2)-
ttzwidth*sin(pi/4);tmzheight/2 tmzheight/2]
[tprop3(1)+ttzwidth*sin(pi/4) tprop3(1)-ttzwidth*sin(pi/4);tprop3(2)-ttzwidth*sin(pi/4)
tprop3(2)+ttzwidth*sin(pi/4);tmzheight/2 tmzheight/2]
[tprop4(1)-ttzwidth*sin(pi/4) tprop4(1)+ttzwidth*sin(pi/4);tprop4(2)-ttzwidth*sin(pi/4)
tprop4(2)+ttzwidth*sin(pi/4);tmzheight/2 tmzheight/2]
[tprop1(1)-ttzwidth*sin(pi/4) tprop1(1)+ttzwidth*sin(pi/4);tprop1(2)+ttzwidth*sin(pi/4) tprop1(2)-
ttzwidth*sin(pi/4);-tmzheight/2 -tmzheight/2]
[tprop2(1)+ttzwidth*sin(pi/4) tprop2(1)-ttzwidth*sin(pi/4);tprop2(2)+ttzwidth*sin(pi/4) tprop2(2)-
ttzwidth*sin(pi/4);-tmzheight/2 -tmzheight/2]
[tprop3(1)+ttzwidth*sin(pi/4) tprop3(1)-ttzwidth*sin(pi/4);tprop3(2)-ttzwidth*sin(pi/4)
tprop3(2)+ttzwidth*sin(pi/4);-tmzheight/2 -tmzheight/2]
[tprop4(1)-ttzwidth*sin(pi/4) tprop4(1)+ttzwidth*sin(pi/4);tprop4(2)-ttzwidth*sin(pi/4)
tprop4(2)+ttzwidth*sin(pi/4);-tmzheight/2 -tmzheight/2]
end

disp('plot manuever motor mounts')
plot([mprop1(1)-mtzwidth mprop1(1)+mtzwidth],[mprop1(2) mprop1(2)])
plot([mprop2(1) mprop2(1)],[mprop2(2)+mtzwidth mprop2(2)-mtzwidth])
plot([mprop3(1)+mtzwidth mprop3(1)-mtzwidth],[mprop3(2) mprop3(2)])
plot([mprop4(1) mprop4(1)],[mprop4(2)-mtzwidth mprop4(2)+mtzwidth])

if dispropco == 1
[mprop1(1)-mtzwidth mprop1(1)+mtzwidth;mprop1(2) mprop1(2);mmzheight/2 mmzheight/2]
[mprop2(1) mprop2(1);mprop2(2)+mtzwidth mprop2(2)-mtzwidth;mmzheight/2 mmzheight/2]
[mprop3(1)+mtzwidth mprop3(1)-mtzwidth;mprop3(2) mprop3(2);mmzheight/2 mmzheight/2]
[mprop4(1) mprop4(1);mprop4(2)-mtzwidth mprop4(2)+mtzwidth;mmzheight/2 mmzheight/2]
```

```
[mprop1(1)-mtzwidth mprop1(1)+mtzwidth;mprop1(2) mprop1(2);-mmzheight/2 -mmzheight/2]
[mprop2(1) mprop2(1);mprop2(2)+mtzwidth mprop2(2)-mtzwidth;-mmzheight/2 -mmzheight/2]
[mprop3(1)+mtzwidth mprop3(1)-mtzwidth;mprop3(2) mprop3(2);-mmzheight/2 -mmzheight/2]
[mprop4(1) mprop4(1);mprop4(2)-mtzwidth mprop4(2)+mtzwidth;-mmzheight/2 -mmzheight/2]
end

%Plot Circumferencial wire
plot([mprop1(1)+mtzwidth tprop1(1)-ttzwidth*sin(pi/4)],[mprop1(2) tprop1(2)+ttzwidth*sin(pi/4)])
plot([tprop1(1)+ttzwidth*sin(pi/4) mprop2(1)],[tprop1(2)-ttzwidth*sin(pi/4) mprop2(2)+mtzwidth])
plot([mprop2(1) tprop2(1)+ttzwidth*sin(pi/4)],[mprop2(2)-mtzwidth tprop2(2)+ttzwidth*sin(pi/4)])
plot([tprop2(1)-ttzwidth*sin(pi/4) mprop3(1)+mtzwidth],[tprop2(2)-ttzwidth*sin(pi/4) mprop3(2)])
plot([mprop3(1)-mtzwidth tprop3(1)+ttzwidth*sin(pi/4)],[mprop3(2) tprop3(2)-ttzwidth*sin(pi/4)])
plot([tprop3(1)-ttzwidth*sin(pi/4) mprop4(1)],[tprop3(2)+ttzwidth*sin(pi/4) mprop4(2)-mtzwidth])
plot([mprop4(1) tprop4(1)-ttzwidth*sin(pi/4)],[mprop4(2)+mtzwidth tprop4(2)-ttzwidth*sin(pi/4)])
plot([tprop4(1)+ttzwidth*sin(pi/4) mprop1(1)-mtzwidth],[tprop4(2)+ttzwidth*sin(pi/4) mprop1(2)])

circumfwirelength = distance(mprop1(1)+mtzwidth, mprop1(2), tprop1(1)-ttzwidth*sin(pi/4),
tprop1(2)+ttzwidth*sin(pi/4));
circumftotalwirelength = 8*circumfwirelength;

flatanglemanueverprops = acos((mproprdist-(tproprdist+ttzwidth)*sin(pi/4))/circumfwirelength);
flatanglethrustprops = pi/2-(asin((mproprdist-(tproprdist+ttzwidth)*sin(pi/4))/circumfwirelength) - pi/4);

axis([-40*25.4/1000 40*25.4/1000 -40*25.4/1000 40*25.4/1000])
axis square
grid on
title('Prop layout (meters)')
hold off

%Height of manuever prop above strut
mpzheight = mpropr*(zheight-mmzheight)/mproprdist + pttwdist + mmzheight;

%Height of thrust prop above strut
tpzheight = tpropr*(zheight-tmzheight)/tproprdist + pttwdist + tmzheight;

figure(2)
%plot manuever props
plot([(-mproprdist-mpropr) (-mproprdist+mpropr)],[-mpzheight -mpzheight])
hold on
plot([(mproprdist-mpropr) (mproprdist+mpropr)],[-mpzheight -mpzheight])
%plot manuever prop verts
plot([-mproprdist -mproprdist],[ 0 -mpzheight])
plot([mproprdist mproprdist],[0 -mpzheight])
%plot motor mount block
plot([-mproprdist -mproprdist],[-mmzheight  mmzheight])
plot([mproprdist mproprdist],[-mmzheight  mmzheight])
%plot manuever struts
plot([-mproprdist mproprdist],[0 0])
%plot manuever wires
plot([-mproprdist 0 mproprdist],[mmzheight zheight mmzheight])
plot([mproprdist 0 -mproprdist],[-mmzheight -zheight -mmzheight])
%plot verts
plot([0 0],[zheight -zheight])
anglemanueverprops = atan((zheight-mmzheight)/mproprdist);
```

```
manueverwirelength = distance(0, zheight, mproprdist, mmzheight);
manuevertotalwirelength = 8*manueverwirelength;

axis([-40*25.4/1000 40*25.4/1000 -40*25.4/1000 40*25.4/1000])
axis square
grid on
title('Manuever prop side view (meters)')
hold off


figure(3)
%plot thrust props
plot([(-tproprdist-tpropr) (-tproprdist+tpropr)],[tpzheight tpzheight])
hold on
plot([(tproprdist-tpropr) (tproprdist+tpropr)],[tpzheight tpzheight])
%plot IMU
plot([3.53/2*25.4/1000 -3.53/2*25.4/1000 -3.53/2*25.4/1000 3.53/2*25.4/1000 3.53/2*25.4/1000],[-
1.77*25.4/1000 -1.77*25.4/1000 (3.84-1.77)*25.4/1000 (3.84-1.77)*25.4/1000 -1.77*25.4/1000]);

%plot thrust prop verts
plot([-tproprdist -tproprdist],[ 0 tpzheight])
plot([tproprdist tproprdist],[0 tpzheight])
%plot motor mount block
plot([-tproprdist -tproprdist],[-tmzheight  tmzheight])
plot([tproprdist tproprdist],[-tmzheight  tmzheight])
%plot thrust struts
plot([-tproprdist tproprdist],[0 0])
%plot manuever wires
plot([-tproprdist 0 tproprdist],[tmzheight zheight tmzheight])
plot([tproprdist 0 -tproprdist],[-tmzheight -zheight -tmzheight])
%plot verts
plot([0 0],[zheight -zheight])
anglethrustprops = atan((zheight-tmzheight)/tproprdist);

thrustwirelength = distance(0, zheight, tproprdist, tmzheight);
thrusttotalwirelength = 8*thrustwirelength;

axis([-40*25.4/1000 40*25.4/1000 -40*25.4/1000 40*25.4/1000])
axis square
grid on
title('Thrust prop side views (meters)')
hold off

totalwirelength=thrusttotalwirelength+manuevertotalwirelength+circumftotalwirelength*2;
totalstrutlength=zheight*2+tproprdist*4+mproprdist*4;


%wire info
wirer=0.0006;
wiredens = 7920
wireA=3.14159*wirer*wirer
wirev=wireA*totalwirelength;
wireweight = wiredens*wirev;

wireE = 190000000000
```

```
freqmanueverprop = 3840/60
freqthrustprop = 2980/60

%manuever motor & prop weight
mmasskg = 0.25
tmasskg = 0.5

%manuever motor & prop rotational inertia
mmassI = 0.25*0.03^2
tmassI = 0.5*0.03^2


%Beam info
beamE = 70000000000
beamG = 26000000000
beamor = 10/32*25.4/1000;
beamwallthickness = 1/64*25.4/1000;
beamir = beamor-beamwallthickness;
beamI = 0.25*pi*(beamor^4-beamir^4)
beamJ = 0.5*pi*(beamor^4-beamir^4)
beamdens = 2710
beamA = pi*(beamor^2-beamir^2)
beamv = beamA*totalstrutlength;
beamweight = beamv*beamdens

manueverbeamL = mproprdist;
thrustbeamL = tproprdist;

beamstrength = 120000000;
beamload = beamstrength*beamA/2;
verttension = beamload/2/cos(anglemanueverprops);
circumftension = beamload/2/cos(0.6172);

% VERTICAL
kzmanueverwireflat =  2*(2*circumftension/circumfwirelength);
kzmanueverwirevert =
2*wireE*wireA*sin(anglemanueverprops)*sin(anglemanueverprops)/manueverwirelength;
kzmanueverwire = kzmanueverwireflat + kzmanueverwirevert;
kzmanueverstrut = 3*beamE*beamI/(manueverbeamL)^3;

kzthrustwireflat = 2*(2*circumftension/circumfwirelength);
kzthrustwirevert = 2*wireE*wireA*sin(anglethrustprops)*sin(anglethrustprops)/thrustwirelength;
kzthrustwire = kzthrustwireflat + kzthrustwirevert;
kzthruststrut = 3*beamE*beamI/(thrustbeamL)^3;

kzmanuever = kzmanueverwire + kzmanueverstrut;
kzthrust = kzthrustwire + kzthruststrut;

znaturalfreqmanueverstrut = (kzmanuever/mmasskg)^0.5/2/pi
znaturalfreqthruststrut = (kzthrust/tmasskg)^0.5/2/pi

% TANGENTIAL
ktmanueverwireflat =
2*(2*wireE*wireA*sin(flatanglemanueverprops)*sin(flatanglemanueverprops)/circumfwirelength);
ktmanueverwirevert = 2*verttension/manueverwirelength;
```

```
ktmanueverwire = ktmanueverwireflat + ktmanueverwirevert;
ktmanueverstrut = 3*beamE*beamI/(manueverbeamL)^3;

ktthrustwireflat =
2*(2*wireE*wireA*sin(flatanglethrustprops)*sin(flatanglethrustprops)/thrustwirelength);
ktthrustwirevert = 2*verttension/thrustwirelength;
ktthrustwire =  ktthrustwireflat + ktthrustwirevert;
ktthruststrut = 3*beamE*beamI/(thrustbeamL)^3;

ktmanuever = ktmanueverwire + ktmanueverstrut;
ktthrust = ktthrustwire + ktthruststrut;

tnaturalfreqmanueverstrut = (ktmanuever/mmasskg)^0.5/2/pi
tnaturalfreqthruststrut = (ktthrust/tmasskg)^0.5/2/pi

% RADIAL
krmanueverwireflat =
2*(2*wireE*wireA*cos(flatanglemanueverprops)*cos(flatanglemanueverprops)/circumfwirelength);
krmanueverwirevert =
2*wireE*wireA*cos(anglemanueverprops)*cos(anglemanueverprops)/manueverwirelength;
krmanueverwire = krmanueverwireflat + krmanueverwirevert;
krmanueverstrut = beamE*beamA/manueverbeamL;

krthrustwireflat =
2*(2*wireE*wireA*cos(flatanglethrustprops)*cos(flatanglethrustprops)/circumfwirelength);
krthrustwirevert = 2*wireE*wireA*cos(anglethrustprops)*cos(anglethrustprops)/thrustwirelength;
krthrustwire = krthrustwireflat + krthrustwirevert;
krthruststrut = beamE*beamA/thrustbeamL;

krmanuever = krmanueverwire + krmanueverstrut;
krthrust = krthrustwire + krthruststrut;

rnaturalfreqmanueverstrut = (krmanuever/mmasskg)^0.5/2/pi
rnaturalfreqthruststrut = (krthrust/tmasskg)^0.5/2/pi

%Torsional about radius
tkrmanueverwireflat = 0.5*2*(2*circumftension/circumfwirelength)*mtzwidth^2;
tkrmanueverwirevert = 0.5*2*verttension/manueverwirelength*mtzwidth^2;
tkrmanueverwire = tkrmanueverwireflat + tkrmanueverwirevert;
tkrmanueverstrut = beamG*beamJ/manueverbeamL;

tkrthrustwireflat = 0.5*2*(2*circumftension/circumfwirelength)*ttzwidth^2;
tkrthrustwirevert = 0.5*2*verttension/thrustwirelength*ttzwidth^2;
tkrthrustwire = tkrthrustwireflat + tkrthrustwirevert;
tkrthruststrut = beamG*beamJ/thrustbeamL;

tkrmanuever = tkrmanueverwire + tkrmanueverstrut;
tkrthrust = tkrthrustwire + tkrthruststrut;

trnaturalfreqmanueverstrut = (tkrmanuever/mmassI)^0.5/2/pi
trnaturalfreqthruststrut = (tkrthrust/tmassI)^0.5/2/pi

%Torsional about tangent
```

```
tktmanueverwireflat =
0.5*2*(2*wireE*wireA*cos(flatanglemanueverprops)*cos(flatanglemanueverprops)/circumfwirelength
)*mtzwidth^2;
tktmanueverwirevert =
0.5*2*wireE*wireA*cos(anglemanueverprops)*cos(anglemanueverprops)/manueverwirelength*mtzwi
dth^2;
tktmanueverwire = tktmanueverwireflat + tktmanueverwirevert;
tktmanueverstrut = beamE*beamI/manueverbeamL;

tktthrustwireflat =
0.5*2*(2*wireE*wireA*cos(flatanglethrustprops)*cos(flatanglethrustprops)/circumfwirelength)*ttzwidt
h^2;
tktthrustwirevert =
0.5*2*wireE*wireA*cos(anglethrustprops)*cos(anglethrustprops)/thrustwirelength*ttzwidth^2;
tktthrustwire = tktthrustwireflat + tktthrustwirevert;
tktthruststrut = beamE*beamI/thrustbeamL;

tktmanuever = tktmanueverwire + tktmanueverstrut;
tktthrust = tktthrustwire + tktthruststrut;

ttnaturalfreqmanueverstrut = (tktmanuever/mmassI)^0.5/2/pi
ttnaturalfreqthruststrut = (tktthrust/tmassI)^0.5/2/pi

%Torsional about vertical
tkzmanueverwireflat =
0.5*2*(2*wireE*wireA*cos(flatanglemanueverprops)*cos(flatanglemanueverprops)/circumfwirelength
)*mtzwidth^2;
tkzmanueverwirevert = 0;
tkzmanueverwire = tkzmanueverwireflat + tkzmanueverwirevert;
tkzmanueverstrut = beamE*beamI/manueverbeamL;

tkzthrustwireflat =
0.5*2*(2*wireE*wireA*cos(flatanglethrustprops)*cos(flatanglethrustprops)/circumfwirelength)*ttzwidt
h^2;
tkzthrustwirevert = 0;
tkzthrustwire = tkzthrustwireflat + tkzthrustwirevert;
tkzthruststrut = beamE*beamI/thrustbeamL;

tkzmanuever = tkzmanueverwire + tkzmanueverstrut;
tkzthrust = tkzthrustwire + tkzthruststrut;

tznaturalfreqmanueverstrut = (tkzmanuever/mmassI)^0.5/2/pi
tznaturalfreqthruststrut = (tkzthrust/tmassI)^0.5/2/pi
```

Simulation Files

Table E-6: Simulation File Relationships

	File	Called by
Simulink model	ThreeDAFVsimworking velocity.mdl	
Simulink m files	currfilterest.m	ThreeDAFVsimworkingvelocity.mdl
	estimatestate.m	ThreeDAFVsimworkingvelocity.mdl
	ffunmine.m	srspf.m
	hfunmine.m	srspf.m
	IMUgeometry.m	ThreeDAFVsimworkingvelocity.mdl
	IMUgeometry_f.m	hfunmine.m
	srspf.m	estimatestate.m
	ThreeDAFVmodelconsts.m	MATLAB user (workspace)
	ThreeDAFVstatedervNew.m	ThreeDAFVsimworkingvelocity.mdl
	ThreeDAFVstatedervNoVolts.m	ffunmine.m
	ThreeDAFVstatedervThrust.m	ThreeDAFVmodelconsts.m
	thrusttorads.m	ThreeDAFVsimworkingvelocity.mdl
post-processing m files	animate_afv.m	MATLAB user (workspace)
	display_afv.m	animate_afv.m
	make_afv.m	animate_afv.m
	myrot.m	rotobj.m
	rotobj.m	make_afv.m

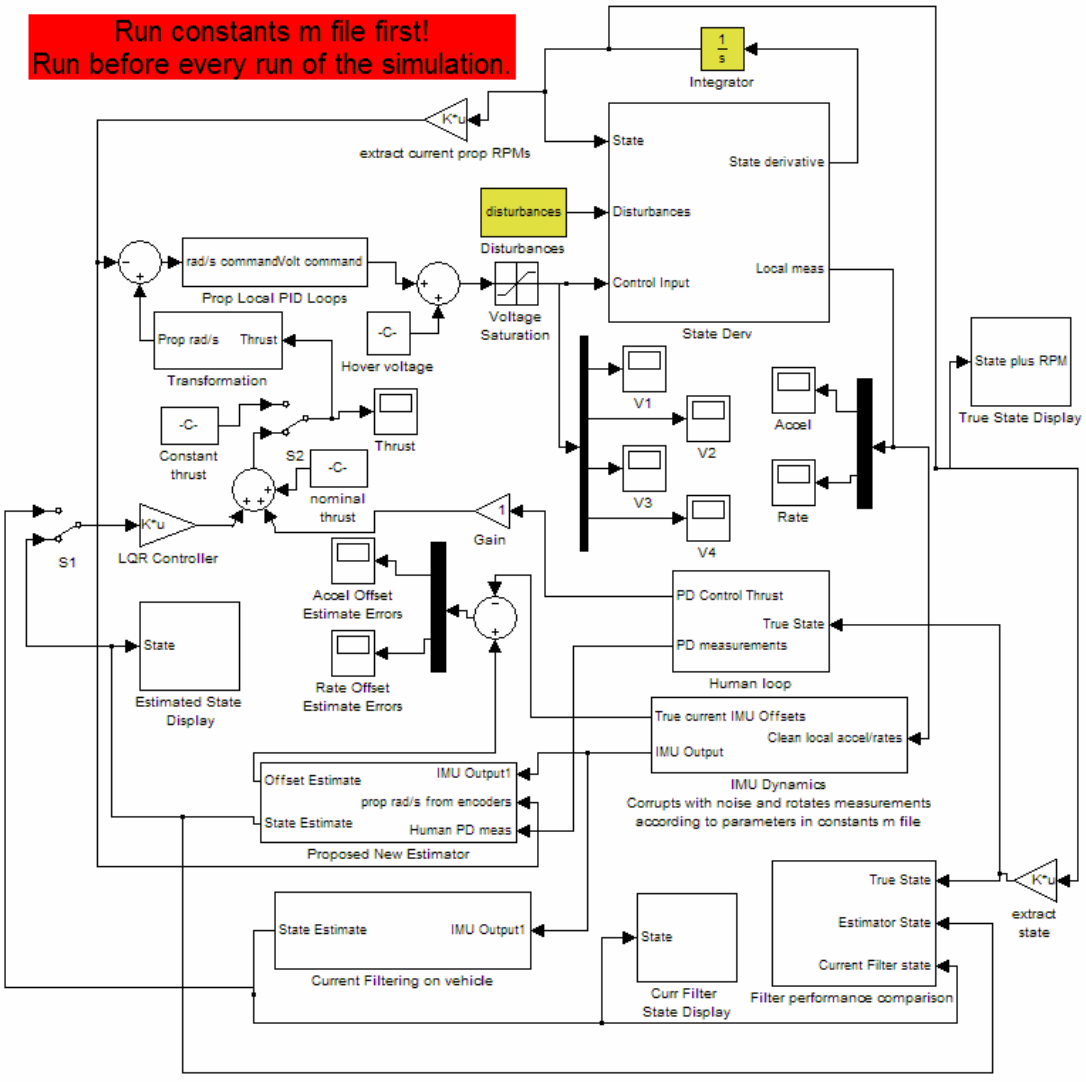


Figure E-8: ThreeDAFVsimworkingvelocity.mdl

```
% currfilterest.m
% Author: Eryk Nice

% this m file replicates the current filtering scheme onboard the vehicle
% using code taken directly from the current DSP code.  Filters, etc are
% initialized in the constants m file.

function intedstate = currfilterest(inputvec)

I_accel = inputvec(1);
J_accel = inputvec(2);
K_accel = inputvec(3);
I_rate = inputvec(4);
J_rate = inputvec(5);
K_rate = inputvec(6);

global X_angle Y_angle Z_angle X_vel Y_vel Z_vel X_pos Y_pos Z_pos sample_rate g_local
global X_angle_p Y_angle_p X_angle_ref Y_angle_ref X_angle_lpf Y_angle_lpf
global I_lpf J_lpf K_lpf X_r_f Y_r_f Z_r_f last_X_r last_Y_r last_Z_r
global I_rate_offset J_rate_offset K_rate_offset I_accel_offset J_accel_offset K_accel_offset

Cpsi=cos(Z_angle); % Precompute trig functions
Spsi=sin(Z_angle);
Cphi=cos(X_angle);
Sphi=sin(X_angle);
Ctheta=cos(Y_angle);
Stheta=sin(Y_angle);

% Subtract off rate offsets
I_r_remain=I_rate-I_rate_offset;
J_r_remain=J_rate-J_rate_offset;
K_r_remain=K_rate-K_rate_offset;

% Compute square of local rates (not currently used)
% I_rate_sqr = I_r_remain*I_r_remain;
% J_rate_sqr = J_r_remain*J_r_remain;
% K_rate_sqr = K_r_remain*K_r_remain;

% Centrifugal acceleration correction
% Note that I am not sure on the signs of these constants because of incomplete data from
% Systron
%I_accel -= (K_r_remain*K_r_remain+J_r_remain*J_r_remain)*0.01424;
%J_accel += (K_r_remain*K_r_remain+I_r_remain*I_r_remain)*0.01432;
%K_accel += (I_r_remain*I_r_remain+J_r_remain*J_r_remain)*0.01432;

% Apply rotation matrix to local rates to get global rates
X_rate=(I_r_remain*(Cpsi*Ctheta)+J_r_remain*(Cpsi*Stheta*Sphi-
Spsi*Cphi)+K_r_remain*(Cpsi*Stheta*Cphi+Spsi*Sphi));
Y_rate=(I_r_remain*(Spsi*Ctheta)+J_r_remain*(Spsi*Stheta*Sphi+Cpsi*Cphi)+K_r_remain*(Spsi*St
heta*Cphi-Cpsi*Sphi));
Z_rate=(I_r_remain*(-1*Stheta)+J_r_remain*Ctheta*Sphi+K_r_remain*Ctheta*Cphi);

% Try to avoid actual singularity by not allowing Ctheta to equal zero
if (abs(Ctheta) < 0.000001)
```

```
            if (Ctheta > 0)
        Ctheta = 0.000001;
            else Ctheta = -0.000001;
    end
end
% Compute Euler angle derivatives from global rates
dX_angle = (Cpsi/Ctheta)*(X_rate)+(Spsi/Ctheta)*(Y_rate);
dY_angle = -Spsi*(X_rate)+Cpsi*(Y_rate);
dZ_angle = (Cpsi*Stheta/Ctheta)*(X_rate)+(Spsi*Stheta/Ctheta)*(Y_rate)+Z_rate;

% Integrate euler angle derivatives. This is all we do with Z angle (Yaw)
% For X and Y (phi and theta), we do this but then we later subtract off the
% low pass filtered angles to combine the gyro data with accelerometer
% estimated tilt angles
X_angle_p = X_angle_p + dX_angle/sample_rate;
Y_angle_p = Y_angle_p + dY_angle/sample_rate;
Z_angle = Z_angle + dZ_angle/sample_rate;

% Low pass filter X and Y angles with a 10 sec time constant
X_angle_lpf=(1.0-1.0/(10.0*sample_rate))*X_angle_lpf+(1.0/(10.0*sample_rate))*X_angle_p;
Y_angle_lpf=(1.0-1.0/(10.0*sample_rate))*Y_angle_lpf+(1.0/(10.0*sample_rate))*Y_angle_p;

% Compute angle estimate as sum of tilt angle reference and gyro estimate
% minus the 10 second average of the gyro data
% This makes the low frequency angle estimate come from accelerometer tilt estimate
% and the high freq info from gyros
X_angle = X_angle_p + X_angle_ref - X_angle_lpf;
Y_angle = Y_angle_p + Y_angle_ref - Y_angle_lpf;

% restore Ctheta to true value because it no longer matters if it becomes 0
Ctheta=cos(Y_angle);

% Subtract off accelerometer offsets
I_remain = I_accel-I_accel_offset;
J_remain = J_accel-J_accel_offset;
K_remain = K_accel-K_accel_offset;

% LPF the accelerometer outpus (without subtracting offsets) to try to estimate
% angle
I_lpf=(1.0-1.0/(10.0*sample_rate))*I_lpf+(1.0/(10.0*sample_rate))*I_accel;
J_lpf=(1.0-1.0/(10.0*sample_rate))*J_lpf+(1.0/(10.0*sample_rate))*J_accel;
K_lpf=(1.0-1.0/(10.0*sample_rate))*K_lpf+(1.0/(10.0*sample_rate))*K_accel;

% Estimate angle from long term average of I,J,K accels
X_angle_ref=atan(J_lpf/(K_lpf));
Y_angle_ref=-atan(I_lpf/(J_lpf*sin(X_angle_ref)+(K_lpf)*cos(X_angle_ref)));

% Apply rotation matrix to accelerations to get global accels
X_r=(I_remain*(Cpsi*Ctheta)+J_remain*(Cpsi*Stheta*Sphi-
Spsi*Cphi)+K_remain*(Cpsi*Stheta*Cphi+Spsi*Sphi));
Y_r=(I_remain*(Spsi*Ctheta)+J_remain*(Spsi*Stheta*Sphi+Cpsi*Cphi)+K_remain*(Spsi*Stheta*Cph
i-Cpsi*Sphi));
Z_r=g_local+(I_remain*(-1*Stheta)+J_remain*Ctheta*Sphi+K_remain*Ctheta*Cphi);

% High-pass filter the global acceleration
```

```
X_r_f=(1.0-1.0/(25.0*sample_rate))*X_r_f+(X_r-last_X_r);
Y_r_f=(1.0-1.0/(25.0*sample_rate))*Y_r_f+(Y_r-last_Y_r);
Z_r_f=(1.0-1.0/(15.0*sample_rate))*Z_r_f+(Z_r-last_Z_r);

last_X_r=X_r; % used to HPF the global acceleration
last_Y_r=Y_r;
last_Z_r=Z_r;

% Integrate the HPF of the global accelerations to get velocity
X_vel = X_vel + X_r_f/sample_rate;
Y_vel = Y_vel + Y_r_f/sample_rate;
Z_vel = Z_vel + Z_r_f/sample_rate;

% Could try to get position this way but we don't bother
X_pos = X_pos + X_vel/sample_rate;
Y_pos = Y_pos + Y_vel/sample_rate;
Z_pos = Z_pos + Z_vel/sample_rate;

intedstate = [X_vel Y_vel Z_vel X_pos Y_pos Z_pos dX_angle dY_angle dZ_angle X_angle Y_angle
Z_angle]';
```

```
% estimatestate.m
% Author: Eryk Nice

% This function takes the current state estimate and
% performs a dynamics propogation and measurement update to get a new
% updated state estimate each time it recieves a measurement input.

function outputvecest = estimatestate(inputvec)

zkp1 = inputvec(1:10);
avecu = inputvec(11:14);
adotvecu = inputvec(15:18);

Uvec = [avecu; adotvecu];

global delt xkgk QkSR PkgkSR Rkp1SR count sig_fact dimen

% the current filtering scheme used is a square root implementation of a
% sigma point filter.  The m files ffunmine and hfunmine take a state
% estimate and prop RPM information and generate a discrete dynamics update
% or expected measurement, respectively.
[xkp1gkp1,Pkp1gkp1SR,xkp1gk,zkp1gk,nu]=srspf(xkgk,PkgkSR,Uvec,QkSR,'ffunmine',zkp1,Rkp1SR,
'hfunmine',delt,sig_fact,count,dimen);

% prepare global variables for next step.  Have option of creating vectors
% for storage here using the count variable or using logging in simulink.
xkgk = xkp1gkp1;
PkgkSR = Pkp1gkp1SR;
outputvecest = xkgk;
```

```
% ffunmine.m
% Author: Eryk Nice

function nextx = ffunmine(xcurrkgk,uvec,delt)

% generate a set of discrete dynamics update vectors given an array of state estimates
M = size(xcurrkgk,2);
for jj=1:M
   % returns a continuous time derivitive given state and prop RPMs
   [statedervs, localmeas]=ThreeDAFVstatedervNoVolts([xcurrkgk(1:12,jj);uvec(:,jj)]);
   % conversion to a discrete step
   nextx(:,jj) = xcurrkgk(:,jj) + [statedervs*delt; zeros(6,1)];
end
```

```
% hfunmine.m
% Author: Eryk Nice

function nextz = hfunmine(xcurrkp1gk,uvec,delt)

% produce an expected measurement given the current state estimate
% (including offset estimates) and prop RPMs

% Note: the same variables for corruption of the noise and estimate of the
% measurements are used in both the IMU Dynamics block and the estimator.
% To examine the effects of having different values for the estimator, new
% variables will have to be created and sent to the code used in
% estimation.  The same goes for dynamics parameters.

global initgainmat

M = size(xcurrkp1gk,2);
nextz = zeros(10,M);
for jj=1:M
   [statedervs, localmeas]=ThreeDAFVstatedervNoVolts([xcurrkp1gk(1:12,jj);uvec(:,jj)]);
   nextz(1:6,jj) = IMUgeometry_f(localmeas) + xcurrkp1gk(13:18,jj); % add offsets
   nextz(7:10,jj) = initgainmat*[xcurrkp1gk(1:6,jj); xcurrkp1gk(9,jj); xcurrkp1gk(12,jj)];

end
```

```
% IMUgeometry.m
% Author: Eryk Nice

function shiftedmeas=IMUgeometry(inputvec)

% rotate measurements if IMU is not aligned with vehicle axes, and add
% centrifugal force terms to accel measurements

global betan betao betaa rhon rhoo rhoa

% Current state
% true local accels
noadbldot = inputvec(1:3);
% true local rates
omega = inputvec(4:6);

% Compute trig values once
sinrn = sin(rhon);
sinro = sin(rhoo);
sinra = sin(rhoa);

cosrn = cos(rhon);
cosro = cos(rhoo);
cosra = cos(rhoa);


% Rotation/Translation matrices
% rotation matrix from IMU to vehicle coordinates
R = [cosra*cosro cosra*sinro*sinrn-sinra*cosrn cosra*sinro*cosrn+sinra*sinrn; sinra*cosro
sinra*sinro*sinrn+cosra*cosrn sinra*sinro*cosrn-cosra*sinrn; -sinro cosro*sinrn cosro*cosrn];

% force terms the accels will see due to centrifugal force and vehicle
% rotation
accelcentripvec = [betan*(abs(omega(2))+abs(omega(3)))  betao*(abs(omega(1))+abs(omega(3)))
betaa*(abs(omega(1))+abs(omega(2)))]';

accelmeas = R*(noadbldot + accelcentripvec);
ratemeas = R*omega;

shiftedmeas = [accelmeas; ratemeas];
```

```
% IMUgeometry_f.m
% Author: Eryk Nice

function shiftedmeas=IMUgeometry_f(inputvec)

% rotate measurements if IMU is not aligned with vehicle axes, and add
% centrifugal force terms to accel measurements.  Use _f parameters to
% allow constants used in filter to differ from true values.

global betan_f betao_f betaa_f rhon_f rhoo_f rhoa_f

betan = betan_f;
betao = betao_f;
betaa = betaa_f;
rhon = rhon_f;
rhoo = rhoo_f;
rhoa = rhoa_f;

% Current state
% true local accels
noadbldot = inputvec(1:3);
% true local rates
omega = inputvec(4:6);

% Compute trig values once
sinrn = sin(rhon);
sinro = sin(rhoo);
sinra = sin(rhoa);

cosrn = cos(rhon);
cosro = cos(rhoo);
cosra = cos(rhoa);


% Rotation/Translation matrices
% rotation matrix from IMU to vehicle coordinates
R = [cosra*cosro cosra*sinro*sinrn-sinra*cosrn cosra*sinro*cosrn+sinra*sinrn; sinra*cosro
sinra*sinro*sinrn+cosra*cosrn sinra*sinro*cosrn-cosra*sinrn; -sinro cosro*sinrn cosro*cosrn];

% force terms the accels will see due to centrifugal force and vehicle
% rotation
accelcentripvec = [betan*(abs(omega(2))+abs(omega(3)))  betao*(abs(omega(1))+abs(omega(3)))
betaa*(abs(omega(1))+abs(omega(2)))]';

accelmeas = R*(noadbldot + accelcentripvec);
ratemeas = R*omega;

shiftedmeas = [accelmeas; ratemeas];
```

```
% srspf.m

function
[xEst,SxEst,xPred,zPred,innovation]=srspf(xEst,SxEst,U,Qsq,ffun,z,Rsq,hfun,dt,sig_fact,k,dimen);
%
% SQUARE ROOT SPF                SIGMA POINT FILTER
% One iteration of SPF, including prediction and correction.
%
% [xEst,SxEst,xPred,zPred,innovation]=srspf(xEst,SxEst,U,Qsq,ffun,z,Rsq,hfun,dt,sig_fact,k,dimen);
%
% INPUTS   :  - xEst            : state mean estimate at time k
%           - SxEst           : square root state covariance at time k
%           - U               : vector of control inputs
%           - Qsq             : square root process noise covariance at time k
%           - ffun            : process model function
%           - z               : observation at k+1
%           - Rsq             : square root measurement noise covariance at k+1
%           - hfun            : observation model function
%           - dt              : time step (passed to ffun/hfun)
%                     - sig_fact        : sigma point scaling factor. Defaults to 0.5.
%           - k               : current iteration
%           - dimen           : number of states, total number of sigma points inc noise, number of outputs
%
% OUTPUTS  :  - xEst            : updated estimate of state mean at time k+1
%                   - PEst            : updated state covariance at time k+1
%           - xPred           : prediction of state mean at time k+1
%           - PPred           : prediction of state covariance at time k+1
%                   - innovation      : innovation vector
%
% CALLS                      : - ScaledSigmaPts.m
%
% AUTHORS  :  Simon J. Julier      (sjulier@erols.com)    1998-2000
%                                  Shelby Brunke                    (sbrunke@u.washington.edu) 2000 -
2001
%                                  Mark Campbell                    (mc288@cornell.edu) 2003
% DATE     :  15 Oct 2003
%
% NOTES    :
%           This code was written to be readable. There is significant
%           scope for optimisation even in Matlab.
%

n           = dimen(1);     %number of states, not counting noise
nsp = dimen(2);     %number of sigma points
nxsp=2*n+1;         %number of state sigma points
ny = dimen(3);      %number of outputs
nn=n+n+ny;          %total number of states and noises

%%matrices of all ones that are helpful - could embed in the code below
ensp=ones(1,nsp);
exnsp=ensp(1:nxsp);
e2n=ensp(1:2*n);
e2ny=ensp(1:2*ny);
```

```
Psqrtm = sig_fact*SxEst';
xSigmaPts=[zeros(n,1) -Psqrtm Psqrtm];
xSigmaPts = xSigmaPts + xEst*exnsp;
%--------------------------------------------------------------------------

%-----GENERATE WEIGHTING MATRICES----------------------------------------
Wi=0.5/sig_fact^2;
W0M=(sig_fact^2-nn)/sig_fact^2;
W0C=(sig_fact^2-nn)/sig_fact^2+3-sig_fact^2/nn;
%--------------------------------------------------------------------------


%-----TIME UPDATE (PROPAGATE SIGMAPOINTS)---------------------------------
xPredSigmaPts = feval(ffun,xSigmaPts,U(:)*exnsp,dt);
xwPredSigmaPts = xPredSigmaPts(:,1)*e2n + sig_fact*[+Qsq' -Qsq'];
xvPredSigmaPts = xPredSigmaPts(:,1)*e2ny;
%%%%%% Calculate Mean (a priori)
xPred = W0M*xPredSigmaPts(:,1) + Wi*sum(xPredSigmaPts(:,2:nxsp),2) +
Wi*(2*n+2*ny)*xPredSigmaPts(:,1); %%Last term due to noise sigma points
%%%%%% Calculate (central) Covariance Square Root (a priori)
exSigmaPts = [xPredSigmaPts xwPredSigmaPts xvPredSigmaPts] - xPred*ensp; %%Eqn 10

%-----MEASUREMENT UPDATE (PROPAGATE SIGMAPOINTS)--------------------------------
zPredSigmaPts = feval(hfun,xPredSigmaPts,U(:)*exnsp,dt);
zwPredSigmaPts = feval(hfun,xwPredSigmaPts,U(:)*exnsp,dt);
zvPredSigmaPts = zPredSigmaPts(:,1)*e2ny+sig_fact*[+Rsq' -Rsq'];
%%%%%% Calculate Mean
zPred = W0M*zPredSigmaPts(:,1) + Wi*sum(zPredSigmaPts(:,2:nxsp),2) +
Wi*sum(zwPredSigmaPts,2) + Wi*(2*ny)*zPredSigmaPts(:,1); %%Eqn 9
%%%%%% Calculate (central) Covariance Square Root (a priori)
ezSigmaPts = [zPredSigmaPts zwPredSigmaPts zvPredSigmaPts] - zPred*ensp; %%Eqn 10

%%%%%% Calculate Kalman Gain
PxzPred = exSigmaPts(:,2:nsp)*ezSigmaPts(:,2:nsp)' + W0C/Wi*exSigmaPts(:,1)*ezSigmaPts(:,1)';
Pyy = ezSigmaPts(:,2:nsp)*ezSigmaPts(:,2:nsp)' + W0C/Wi*ezSigmaPts(:,1)*ezSigmaPts(:,1)';
K = (PxzPred)*inv(Pyy);

%%%%%% Orthogonalize Square Root Matrix
[tmp,Sx_bar] = qr([exSigmaPts(:,2:nsp) - K*ezSigmaPts(:,2:nsp)]',0);
%%%%%% Negative weight is handled below using Cholesky update
if W0C < 0
  Sx = cholupdate(sqrt(Wi)*Sx_bar,[exSigmaPts(:,1) - K*ezSigmaPts(:,1)]*sqrt(abs(W0C)),'-');
else
  Sx = cholupdate(sqrt(Wi)*Sx_bar,[exSigmaPts(:,1) - K*ezSigmaPts(:,1)]*sqrt(W0C),'+');
end
SxEst=Sx;

%%%%%% Calculate Innovation
innovation = z - zPred;
%%%%%% Update mean
xEst = xPred + K*innovation;
```

```
% ThreeDAFVmodelconsts.m
% Author: Eryk Nice

%3D model and simulation constants - contains vehicle parameters, estimator
%parameters, LQR control weighting matrices, sensor and process noise
%parameters, etc.  Must be run before every simulation run in simulink to
%re-initialize estimators, etc.

clear all

global delt estimatorHz
estimatorHz = 150;
delt = 1/estimatorHz;

% variable to control decimation of stored data for video generation.
% Alter to get video time to reflect real time.
decimatestore = 66;

% process noise power
torquedistpow = 3e-5;
winddistpow = 3e-4;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% vehicle parameters
%%%%%%%%%%%%%%%%%%%%
% Suffix of _f is the variable used in all filtering code.  Change _f to be
% different from normal variable to see effect of modelling error.  Only
% used for those values that are likely to possibly be different from
% modelled, eg prop constants, IMU rotation or position parameters.
% Not completely implemented - if you wish to make more constants different
% for filtered versions, you need only add them below and change them in
% ThreeDAFVstatedervNoVoltsposRPMs and ThreeDAFVstatedervNoVolts as kd and
% kt are handled both here and there.  _posRPMs is used in LQR controller
% generation, other is used in filter estimation.

%radial dist of prop [m]
global L
L = 14.955*2.54/100;
%wind to force coeff - [N/wind velocity m/s]
% x y side
global ku
ku = 1;
% z bottom
global ks
ks = 1;

%Rotational inertia of vehicle [kg*m^2]
Jn=6.0513200e02/2.2*2.54^2/10000;
Jo=6.0190021e02/2.2*2.54^2/10000;
Ja=1.1417148e03/2.2*2.54^2/10000;

% have option of replacing this with full rotational inertia matrix
global J
J = [Jn 0 0;0 Jo 0;0 0 Ja];
```

```
%mass [kg]
global m
m = 6; %6.22;
%gravity [m/s^2]
global g
g = 9.8028737;

% nominal thrust per prop [N]
global Tnom
Tnom = m/4*g;

%prop [m]
%diamet = 18*2.54/100;
global pitch
pitch = 6*2.54/100;
%coeff thrust
global kt kt_f
kt = 8.4e-5;
kt_f = 8.4e-5;
%coeff drag
global kd kd_f
kd = 3.14e-6;
kd_f = 3.14e-6;
%moment inertia prop [kg*m^2]
Jp = 5/2.2*2.54^2/10000;

%motor
%gearing
G = 80/12;
%voltage constant [volts/rad/s]
global kv
kv = 1/2500*60/2/pi*G;
%current constant [Nm/Amp]
global ki
ki = 0.548*0.00706*G;
%Motor resistance [Ohm]
global R
R = 0.3;
% moment inertia motor rotor [kg*m^2]
Jmot = 0;
%total MOI of prop and motor as seen by prop [kg*m^2]
global Jt
Jt = Jmot*G^2+Jp;

% nominal prop rad/s
global radssecnom
radssecnom = (Tnom/kt)^0.5;

% nominal voltage
global Vnom
Vnom = kv*radssecnom + R*kd/ki*radssecnom^2;

% prop local control PID loop
PropP = 0.62;
```

```
PropI = 3.35;
PropD = 0;

PropN = 1000;

%initial state vector
initstatevec = radssecnom*[zeros(1,12) 1 1 1 1]' + [0;  % xdot
                                                    0;  % ydot
                                                    0;  % zdot
                                                    0;  % x
                                                    0;  % y
                                                    0;  % z
                                                    0;  % phidot
                                                    0;  % thetadot
                                                    0;  % psidot
                                                    pi/15;  % phi
                                                    0;  % theta
                                                    0;  % psi
                                                    zeros(4,1)];


%%%%%%%%%%%%%%%%% Measurement parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global kappanoa kappaomega noaoffsetdev omegaoffsetdev accelinitbias rateinitbias

% initial bias parameters
accelinitbias = 8.5e-3*g;%m/s^2
rateinitbias = 1*pi/180; %rad/s

% First guess for estimators - produced from initial IMU calibration
% routine.  Note, we use initial accels to get angle, so assume initial
% accel offsets are zero.
accelinitbiasvec0 = accelinitbias*randn(3,1);%m/s^2
rateinitbiasvec0 = rateinitbias*randn(3,1); %rad/s

% Controls the drift rate of accelerometer offsets
kappanoa = 5e-6;
% Controls the drift rate of gyro offsets
kappaomega = 1e-13;

% std dev of white noise driving accel drift
noaoffsetdev = 5e-5;
% std dev of white noise driving rate gyro drift
omegaoffsetdev = 5e-10;

% actual initial bias vectors
accelinitbiasvec = accelinitbiasvec0 + noaoffsetdev*randn(3,1);%m/s^2
rateinitbiasvec = rateinitbiasvec0 + omegaoffsetdev*randn(3,1); %rad/s


global betan betao betaa rhon rhoo rhoa accelnoisedev ratenoisedev
% variables used in filter meas update step - change values below to
% examine impact of modelling errors.
global betan_f betao_f betaa_f rhon_f rhoo_f rhoa_f
```

```
% linear offsets of accelerometers from vehicle CM
betan = 0;
betan_f = 0;
betao = 0;
betao_f = 0;
betaa = 0;
betaa_f = 0;

%rotation of IMU from vehicle coordinates
rhon = 0;
rhon_f = 0;
rhoo = 0;
rhoo_f = 0;
rhoa = 0;
rhoa_f = 0;

% standard dev of accelerometer white noise
accelnoisedev = 60e-6*g*30^.5;
% standard dev of rate gyro white noise
ratenoisedev = 1e-10;

% standard dev of local torque and global wind force process noise
torqueprocnoisedev = torquedistpow^.5;
forceprocnoisedev = winddistpow^.5;


%%%%%%%%%%%%%%%%% HUMAN OUTER CONTROL LOOP
%%%%%%%%%%%%%%%%%%%%%%%%%
% initial PD gains
Dxy0 = .3;
Dz0 = .4;
Pxy0 = .5;
Pz0 = .5;
Dyaw0 = .22;
Pyaw0 = .8;

global initgainvec initgainmat
initgainvec = 1*[Dxy0 Dz0 Pxy0 Pz0 Dyaw0 Pyaw0]';
% used in meas prediction step in filter
initgainmat = 1*[Dxy0 0 0 Pxy0 0 0 0 0;
                 0 Dxy0 0 0 Pxy0 0 0 0;
                 0 0 Dz0 0 0 Pz0 0 0;
                 0 0 0 0 0 0 Dyaw0 Pyaw0];

% feedback matrix to control gain drift
kappaDxy = 5e-1;
kappaDz = 5e-1;
kappaPxy = 5e-1;
kappaPz = 5e-1;
kappaDyaw = 5e-1;
kappaPyaw = 5e-1;
kappagainmat = -diag([kappaDxy kappaDz kappaPxy kappaPz kappaDyaw kappaPyaw]);

 %noise power driving gain drifts
```

```
Dxypow = 1e-3;
Dzpow = 1e-3;
Pxypow = 1e-3;
Pzpow = 1e-3;
Dyawpow = 1e-3;
Pyawpow = 1e-3;
gaindriftdrivingnoisevec = 1*[Dxypow Dzpow Pxypow Pzpow Dyawpow Pyawpow]';


%%%%%%%%%%%%%%%%%%%%%% Filter variables
%%%%%%%%%%%%%%%%%%%%%%
global xkgk QkSR PkgkSR Rkp1SR count sig_fact dimen

% state estimate - added term is errors introduced in initial guess
xkgk = [initstatevec(1:12); zeros(6,1)] +  [0;  % xdot
                                0;  % ydot
                                0;  % zdot
                                0;  % x
                                0;  % y
                                0;  % z
                                0;  % phidot
                                0;  % thetadot
                                0;  % psidot
                                0;  % phi
                                0;  % theta
                                0;  % psi
                                0*accelinitbiasvec0;  % Dnoa accel offset vec -
                                                % zeros since we use accel bias for
                                                % initial angle in calibration
                                rateinitbiasvec0];  % Dwnoa rate offset vec


% assumed process noise standard devs
delvels = forceprocnoisedev; % m/s/s
delposs = 1e-8;%forceprocnoisedev/estimatorHz*10; % m/s
delws = torqueprocnoisedev; % rad/s/s
delangs = 1e-9;%torqueprocnoisedev/estimatorHz*10; % rad/s
delacceloffs = noaoffsetdev/1e4; % m/s^2/s
delwoffs = omegaoffsetdev; % rad/s/s

procnoisevec = [delvels/10*[ku ku ks] delposs*[ku ku ks] delws*[1 1 1] delangs*[1 1 1]
delacceloffs*[1 1 1] delwoffs*[1 1 1]]';
% process noise covariance square root matrix
QkSRnom = diag(procnoisevec);%./estimatorHz;
QkSR = 1/1*QkSRnom;

% initial state covariance square root matrix
PkgkSR = QkSR + diag([zeros(12,1); accelinitbias*[1 1 1]'; zeros(3,1)]);
PkgkSR = 5*PkgkSR;


xynoisedev = (Dxypow + Pxypow)^0.5;
znoisedev = (Dzpow + Pzpow)^0.5;
yawnoisedev = (Dyawpow + Pyawpow)^0.5;
% measurement noise covariance square root matrix
```

```
Rkp1SRnom = diag([10*accelnoisedev*[1 1 1] ratenoisedev*[1 1 1] 1/2*[xynoisedev xynoisedev
znoisedev yawnoisedev]]);
Rkp1SR = 1/1*Rkp1SRnom;

count = 1;

% sigma factor used in sqare root sigma point filter
sig_fact = 0.5;

% ESTABLISH DIMENSIONS OF SYSTEM
n = 12;   % order of system -- '1' returns row dimension
no = 10;              % number of outputs -- '1' returns row dimension
ni = 8;    % number of inputs -- '1' returns row dimension
np = 6;     % number of parameters to estimate (6 aero forces/moments)
nsp = 2*(n+np)+1; % number of "state" sigma points
% Set up a vector of the important dimensions to pass to the SPF function
dimen=[n+np 2*(n+np+n+np+no)+1 no]; % number of states, total number of sigma points inc noise,
number of outputs

%%%%%%%%%%%%%%%%%%%%%%%%% Current filtering scheme variables
%%%%%%%%%%%%%%%%%%%%%%%%%%
% code yanked directly from DSP, but requires initialization below.
% Initialization is the equivalent of the calibration step on the vehicle.
global X_angle Y_angle Z_angle X_vel Y_vel Z_vel X_pos Y_pos Z_pos sample_rate g_local
global X_angle_p Y_angle_p X_angle_ref Y_angle_ref X_angle_lpf Y_angle_lpf
global I_lpf J_lpf K_lpf X_r_f Y_r_f Z_r_f last_X_r last_Y_r last_Z_r
global I_rate_offset J_rate_offset K_rate_offset I_accel_offset J_accel_offset K_accel_offset

sample_rate = 600;
g_local = g;
X_vel = xkgk(1);
Y_vel = xkgk(2);
Z_vel = xkgk(3);
X_pos = xkgk(4);
Y_pos = xkgk(5);
Z_pos = xkgk(6);
X_angle = xkgk(10);
X_angle_p = X_angle;
X_angle_ref = X_angle;
X_angle_lpf = 0;
Y_angle = xkgk(11);
Y_angle_p = Y_angle;
Y_angle_ref = Y_angle;
Y_angle_lpf = 0;
Z_angle = xkgk(12);

I_accel_offset = xkgk(13);
J_accel_offset = xkgk(14);
K_accel_offset = xkgk(15);
I_rate_offset = xkgk(16);
J_rate_offset = xkgk(17);
K_rate_offset = xkgk(18);

I_lpf = 0;
J_lpf = 0;
```

```
K_lpf = g_local;
X_r_f = 0;
Y_r_f = 0;
Z_r_f = 0;
last_X_r = 0;
last_Y_r = 0;
last_Z_r = g_local;


%%%%%%%%%%%%%%%%%%%% LQRY Controller
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

syms xdot ydot zdot x y z phidot thetadot psidot phi theta psi T1 T2 T3 T4 real
sendvec = [xdot ydot zdot x y z phidot thetadot psidot phi theta psi T1 T2 T3 T4]';


[fsym, htruesym]=ThreeDAFVstatedervThrust(sendvec);

Amatsym = jacobian(fsym,sendvec(1:12));

Bmatsym = jacobian(fsym,sendvec(13:16));

% Once symbolic differentiation is done, evaluate jacobians at state to be
% linearized about.
xdot = 0;
ydot = 0;
zdot = 0;
x = 0;
y = 0;
z = 0;
phidot = 0;
thetadot = 0;
psidot = 0;
phi = 0;
theta = 0;
psi = 0;


T1 = Tnom;
T2 = Tnom;
T3 = Tnom;
T4 = Tnom;

Amat = eval(Amatsym);
Bmat = eval(Bmatsym);

% gains from original simulink controller
% xydotweight = 4.5;
% zdotweight = 6;
% xyweight = 0;
% zweight = 0;
% rollpitchdotweight = 3.9;
% yawdotweight = 3.3;
% rollpitchweight = 18;
% yawweight = 12;
```

```
% weights for LQR control design
xydotweight = 1;
zdotweight = 1;
xyweight = 1;
zweight = 1;
rollpitchdotweight = 1;
yawdotweight = 1;
rollpitchweight = 1;
yawweight = 1;

thrustweight = 1;

Rxxweight = diag([xydotweight*[1 1] zdotweight xyweight*[1 1] zweight rollpitchdotweight*[1 1]
yawdotweight rollpitchweight*[1 1] yawweight]);
Ruuweight = diag(thrustweight*[1 1 1 1]);

[Kmat,Smat,Emat] = lqr(Amat,Bmat,Rxxweight,Ruuweight);
```

```
% ThreeDAFVstatedervNew.m
% Author: Eryk Nice

function outputvec=ThreeDAFVstatedervNew(inputvec)

global L ku ks J m g pitch kt kd kv ki R Jt

% Current state
% global velocities
xdot = inputvec(1);
ydot = inputvec(2);
zdot = inputvec(3);
% global position
x = inputvec(4);
y = inputvec(5);
z = inputvec(6);
% Euler angle rates
phidot = inputvec(7);
thetadot = inputvec(8);
psidot = inputvec(9);
% Euler angles
phi = inputvec(10);
theta = inputvec(11);
psi = inputvec(12);
% prop rad/s
a1 = inputvec(13);
a2 = inputvec(14);
a3 = inputvec(15);
a4 = inputvec(16);

% Disturbances
% global wind
wx = inputvec(17);
wy = inputvec(18);
wz = inputvec(19);
% local torques
tn = inputvec(20);
to = inputvec(21);
ta = inputvec(22);

% Control input
% motor voltages
V1 = inputvec(23);
V2 = inputvec(24);
V3 = inputvec(25);
V4 = inputvec(26);

% prevent div/0 error
if a1 == 0
   a1 = 1e-100;
end
if a2 == 0
   a2 = 1e-100;
end
```

```
if a3 == 0
   a3 = 1e-100;
end
if a4 == 0
   a4 = 1e-100;
end

% Assemble variables in to vectors for manipulation
xyzdotvec = [xdot ydot zdot]';
xyzvec = [x y z]';
eulerdotvec = [phidot thetadot psidot]';
eulervec = [phi theta psi]';
avec = [a1 a2 a3 a4]';
wxyzvec = [wx wy wz]';
tnoavec = [tn to ta]';
Vvec = [V1 V2 V3 V4]';
onevec = ones(4,1);

% Compute trig values once
sinphi = sin(phi);
sintheta = sin(theta);
sinpsi = sin(psi);

cosphi = cos(phi);
costheta = cos(theta);
cospsi = cos(psi);


% Rotation/Translation matrices
% rotation matrix from local to global coordinates and its inverse
A = [cospsi*costheta cospsi*sintheta*sinphi-sinpsi*cosphi cospsi*sintheta*cosphi+sinpsi*sinphi;
sinpsi*costheta sinpsi*sintheta*sinphi+cospsi*cosphi sinpsi*sintheta*cosphi-cospsi*sinphi; -sintheta
costheta*sinphi costheta*cosphi];
Ainv = inv(A);

% matrix transformation from euler angular rates to global angular rates
% and its inverse
Minv = [cospsi/costheta sinpsi/costheta 0; -sinpsi cospsi 0; cospsi*sintheta/costheta
sinpsi*sintheta/costheta  1];
M = inv(Minv);
% partial derivative of Minv with respect to psi and theta, used in
% computing euler angle second derivatives
diffMinvAphi = [0,
cospsi/costheta*(cospsi*sintheta*cosphi+sinpsi*sinphi)+sinpsi/costheta*(sinpsi*sintheta*cosphi-
cospsi*sinphi),  cospsi/costheta*(-cospsi*sintheta*sinphi+sinpsi*cosphi)+sinpsi/costheta*(-
sinpsi*sintheta*sinphi-cospsi*cosphi);
   0,  -sinpsi*(cospsi*sintheta*cosphi+sinpsi*sinphi)+cospsi*(sinpsi*sintheta*cosphi-cospsi*sinphi),  -
sinpsi*(-cospsi*sintheta*sinphi+sinpsi*cosphi)+cospsi*(-sinpsi*sintheta*sinphi-cospsi*cosphi);
   0,
cospsi*sintheta/costheta*(cospsi*sintheta*cosphi+sinpsi*sinphi)+sinpsi*sintheta/costheta*(sinpsi*sinth
eta*cosphi-cospsi*sinphi)+costheta*cosphi,  cospsi*sintheta/costheta*(-
cospsi*sintheta*sinphi+sinpsi*cosphi)+sinpsi*sintheta/costheta*(-sinpsi*sintheta*sinphi-
cospsi*cosphi)-costheta*sinphi];
diffMinvAtheta = [0,  cospsi/costheta^2*(cospsi*sintheta*sinphi-
sinpsi*cosphi)*sintheta+cospsi^2*sinphi+sinpsi/costheta^2*(sinpsi*sintheta*sinphi+cospsi*cosphi)*sin
```

```
theta+sinpsi^2*sinphi,
cospsi/costheta^2*(cospsi*sintheta*cosphi+sinpsi*sinphi)*sintheta+cospsi^2*cosphi+sinpsi/costheta^2
*(sinpsi*sintheta*cosphi-cospsi*sinphi)*sintheta+sinpsi^2*cosphi;
   0,  0,  0;
   cospsi^2*costheta+sinpsi^2*costheta-costheta,  cospsi*(cospsi*sintheta*sinphi-
sinpsi*cosphi)+cospsi*sintheta^2/costheta^2*(cospsi*sintheta*sinphi-
sinpsi*cosphi)+cospsi^2*sintheta*sinphi+sinpsi*(sinpsi*sintheta*sinphi+cospsi*cosphi)+sinpsi*sinthet
a^2/costheta^2*(sinpsi*sintheta*sinphi+cospsi*cosphi)+sinpsi^2*sintheta*sinphi-sintheta*sinphi,
cospsi*(cospsi*sintheta*cosphi+sinpsi*sinphi)+cospsi*sintheta^2/costheta^2*(cospsi*sintheta*cosphi+
sinpsi*sinphi)+cospsi^2*sintheta*cosphi+sinpsi*(sinpsi*sintheta*cosphi-
cospsi*sinphi)+sinpsi*sintheta^2/costheta^2*(sinpsi*sintheta*cosphi-
cospsi*sinphi)+sinpsi^2*sintheta*cosphi-sintheta*cosphi];
diffMinvApsi = [0,  cospsi/costheta*(-sinpsi*sintheta*sinphi-
cospsi*cosphi)+cospsi/costheta*(sinpsi*sintheta*sinphi+cospsi*cosphi),   cospsi/costheta*(-
sinpsi*sintheta*cosphi+cospsi*sinphi)+cospsi/costheta*(sinpsi*sintheta*cosphi-cospsi*sinphi);
   0,  -sinpsi*(-sinpsi*sintheta*sinphi-cospsi*cosphi)-sinpsi*(sinpsi*sintheta*sinphi+cospsi*cosphi),   -
sinpsi*(-sinpsi*sintheta*cosphi+cospsi*sinphi)-sinpsi*(sinpsi*sintheta*cosphi-cospsi*sinphi);
   0,  cospsi*sintheta/costheta*(-sinpsi*sintheta*sinphi-
cospsi*cosphi)+cospsi*sintheta/costheta*(sinpsi*sintheta*sinphi+cospsi*cosphi),
cospsi*sintheta/costheta*(-
sinpsi*sintheta*cosphi+cospsi*sinphi)+cospsi*sintheta/costheta*(sinpsi*sintheta*cosphi-
cospsi*sinphi)];


% vector of torque generated by each of the four motors based on control
% input, voltage
tmvec = ki/R*(Vvec-kv*avec);

% global wind disturbance translated into local wind
wnoavec = Ainv*wxyzvec;

% global vehicle velocity translated into local velocity
noadotvec = Ainv*xyzdotvec;

% drag loading due to difference between local wind and local velocity
% simulates either the buffeting caused by a strong wind or the minor
% amounts of damping caused by drag at high velocities
Fnoarelativevec = (wnoavec-noadotvec).*[ks ks ku]';

% drag loading translated into global coordinates
Fxyzrelativevec = A*Fnoarelativevec;

% global angular rate vector obtained from euler angle rates - euler rates
% are NOT the same as global rates since two of the three euler rates are
% rotated by the other euler angles
omegaxyzvec = M*eulerdotvec;
% global angular rate vector translated into local angular rate vector
omeganoavec = Ainv*omegaxyzvec;

% vector of variables that take into account the pitch speed of a prop -
% the translation rate at which the prop spinning at its current RPM fails
% to produce thrust.  Picture a windmill spinning freely in a blowing wind
kpvec = onevec./(avec/2/pi*pitch);

% vector of variables handling the above described situation as caused by
```

```
% roll or pitch of the vehicle
kwvec = L*kpvec;

% vector of the absolute values of the thrust forces generated by each of
% the four props, assuming they are rotating in their intended direction
Tvec = kt*abs(avec).*avec.*(onevec+kwvec.*[-omeganoavec(2) omeganoavec(1) omeganoavec(2) -
omeganoavec(1)]'+kpvec*(noadotvec(3)-wnoavec(3)));

% vector of the absolute values of the drag (torque) forces produced by
% each of the four props, assuming they are rotating in their intended
% direction
Dvec = kd*abs(avec).*avec.*(onevec+kwvec.*[-omeganoavec(2) omeganoavec(1) omeganoavec(2) -
omeganoavec(1)]'+kpvec*(noadotvec(3)-wnoavec(3)));

% vector of the acceleration of each of the four props caused by motor
% torque minus prop drag
adotvec = (tmvec-Dvec)/Jt;

% vehicle torque caused by temporary inequalities in the amount the props
% are being accelerated or decelerated.  Generally small.
Mma = Jt*[1 -1 1 -1]*adotvec;

% total local torque exerted on the vehicle from thrust differentials
% causing roll and pitch moments, prop drag and prop accel/deccel causing
% yaw torque, and local disturbance torques.
torquenoatotalvec = [L*(Tvec(4)-Tvec(2)) L*(Tvec(1)-Tvec(3)) [1 -1 1 -1]*Dvec+Mma]'+tnoavec;

% local torques and angular rate coupling (precessive effects) yield total
% rate of change of local rates
omeganoadotvec = inv(J)*torquenoatotalvec + inv(J)*cross(omeganoavec,J*omeganoavec);

% total global forces exerted on the vehicle from gravity, prop thrust, and
% global wind loading cause global accelerations
xyzdbldotvec = [0 0 g]'-1/m*A*[0 0 onevec'*Tvec]'-Fxyzrelativevec;

noadbldotvec = Ainv*xyzdbldotvec;

% rate of change of euler rates is determined by differentiation of the
% base relationship between euler angle rates and global angular rates.
eulerdbldotvec = phidot*diffMinvAphi*omeganoavec + thetadot*diffMinvAtheta*omeganoavec +
psidot*diffMinvApsi*omeganoavec + Minv*A*omeganoadotvec;

% state derivative returned
statedervs = [xyzdbldotvec; xyzdotvec; eulerdbldotvec; eulerdotvec; adotvec];
localmeas = [noadbldotvec; omeganoavec];
outputvec = [statedervs; localmeas];
```

```
% ThreeDAFVstatedervNoVolts.m
% Author: Eryk Nice

function [statedervs, localmeas]=ThreeDAFVstatedervNoVolts(inputvec)

% This version of the state dynamics accepts prop rad/s information rather
% than Voltage input.

global L ku ks J m g pitch kt_f kd_f kv ki R Jt

kt = kt_f;
kd = kd_f;

% Current state
% global velocities
xdot = inputvec(1);
ydot = inputvec(2);
zdot = inputvec(3);
% global position
x = inputvec(4);
y = inputvec(5);
z = inputvec(6);
% Euler angle rates
phidot = inputvec(7);
thetadot = inputvec(8);
psidot = inputvec(9);
% Euler angles
phi = inputvec(10);
theta = inputvec(11);
psi = inputvec(12);
% prop rad/s
a1 = inputvec(13);
a2 = inputvec(14);
a3 = inputvec(15);
a4 = inputvec(16);
% prop rad/s
a1dot = inputvec(17);
a2dot = inputvec(18);
a3dot = inputvec(19);
a4dot = inputvec(20);


% Disturbances
% global wind
wx = 0;
wy = 0;
wz = 0;
% local torques
tn = 0;
to = 0;
ta = 0;

% prevent div/0 error
if a1 == 0
```

```
   a1 = 1e-100;
end
if a2 == 0
   a2 = 1e-100;
end
if a3 == 0
   a3 = 1e-100;
end
if a4 == 0
   a4 = 1e-100;
end

% Assemble variables in to vectors for manipulation
xyzdotvec = [xdot ydot zdot]';
xyzvec = [x y z]';
eulerdotvec = [phidot thetadot psidot]';
eulervec = [phi theta psi]';
avec = [a1 a2 a3 a4]';
adotvec = [a1dot a2dot a3dot a4dot]';
wxyzvec = [wx wy wz]';
tnoavec = [tn to ta]';
onevec = ones(4,1);

% Compute trig values once
sinphi = sin(phi);
sintheta = sin(theta);
sinpsi = sin(psi);

cosphi = cos(phi);
costheta = cos(theta);
cospsi = cos(psi);


% Rotation/Translation matrices
% rotation matrix from local to global coordinates and its inverse
A = [cospsi*costheta cospsi*sintheta*sinphi-sinpsi*cosphi cospsi*sintheta*cosphi+sinpsi*sinphi;
sinpsi*costheta sinpsi*sintheta*sinphi+cospsi*cosphi sinpsi*sintheta*cosphi-cospsi*sinphi; -sintheta
costheta*sinphi costheta*cosphi];
Ainv = inv(A);

% matrix transformation from euler angular rates to global angular rates
% and its inverse
Minv = [cospsi/costheta sinpsi/costheta 0; -sinpsi cospsi 0; cospsi*sintheta/costheta
sinpsi*sintheta/costheta  1];
M = inv(Minv);
% partial derivative of Minv with respect to psi and theta, used in
% computing euler angle second derivatives
diffMinvAphi = [0,
cospsi/costheta*(cospsi*sintheta*cosphi+sinpsi*sinphi)+sinpsi/costheta*(sinpsi*sintheta*cosphi-
cospsi*sinphi),  cospsi/costheta*(-cospsi*sintheta*sinphi+sinpsi*cosphi)+sinpsi/costheta*(-
sinpsi*sintheta*sinphi-cospsi*cosphi);
   0,  -sinpsi*(cospsi*sintheta*cosphi+sinpsi*sinphi)+cospsi*(sinpsi*sintheta*cosphi-cospsi*sinphi),  -
sinpsi*(-cospsi*sintheta*sinphi+sinpsi*cosphi)+cospsi*(-sinpsi*sintheta*sinphi-cospsi*cosphi);
   0,
cospsi*sintheta/costheta*(cospsi*sintheta*cosphi+sinpsi*sinphi)+sinpsi*sintheta/costheta*(sinpsi*sinth
```

```
eta*cosphi-cospsi*sinphi)+costheta*cosphi,  cospsi*sintheta/costheta*(-
cospsi*sintheta*sinphi+sinpsi*cosphi)+sinpsi*sintheta/costheta*(-sinpsi*sintheta*sinphi-
cospsi*cosphi)-costheta*sinphi];
diffMinvAtheta = [0,  cospsi/costheta^2*(cospsi*sintheta*sinphi-
sinpsi*cosphi)*sintheta+cospsi^2*sinphi+sinpsi/costheta^2*(sinpsi*sintheta*sinphi+cospsi*cosphi)*sin
theta+sinpsi^2*sinphi,
cospsi/costheta^2*(cospsi*sintheta*cosphi+sinpsi*sinphi)*sintheta+cospsi^2*cosphi+sinpsi/costheta^2
*(sinpsi*sintheta*cosphi-cospsi*sinphi)*sintheta+sinpsi^2*cosphi;
   0,  0,  0;
   cospsi^2*costheta+sinpsi^2*costheta-costheta,  cospsi*(cospsi*sintheta*sinphi-
sinpsi*cosphi)+cospsi*sintheta^2/costheta^2*(cospsi*sintheta*sinphi-
sinpsi*cosphi)+cospsi^2*sintheta*sinphi+sinpsi*(sinpsi*sintheta*sinphi+cospsi*cosphi)+sinpsi*sinthet
a^2/costheta^2*(sinpsi*sintheta*sinphi+cospsi*cosphi)+sinpsi^2*sintheta*sinphi-sintheta*sinphi,
cospsi*(cospsi*sintheta*cosphi+sinpsi*sinphi)+cospsi*sintheta^2/costheta^2*(cospsi*sintheta*cosphi+
sinpsi*sinphi)+cospsi^2*sintheta*cosphi+sinpsi*(sinpsi*sintheta*cosphi-
cospsi*sinphi)+sinpsi*sintheta^2/costheta^2*(sinpsi*sintheta*cosphi-
cospsi*sinphi)+sinpsi^2*sintheta*cosphi-sintheta*cosphi];
diffMinvApsi = [0,  cospsi/costheta*(-sinpsi*sintheta*sinphi-
cospsi*cosphi)+cospsi/costheta*(sinpsi*sintheta*sinphi+cospsi*cosphi),   cospsi/costheta*(-
sinpsi*sintheta*cosphi+cospsi*sinphi)+cospsi/costheta*(sinpsi*sintheta*cosphi-cospsi*sinphi);
   0,  -sinpsi*(-sinpsi*sintheta*sinphi-cospsi*cosphi)-sinpsi*(sinpsi*sintheta*sinphi+cospsi*cosphi),   -
sinpsi*(-sinpsi*sintheta*cosphi+cospsi*sinphi)-sinpsi*(sinpsi*sintheta*cosphi-cospsi*sinphi);
   0,  cospsi*sintheta/costheta*(-sinpsi*sintheta*sinphi-
cospsi*cosphi)+cospsi*sintheta/costheta*(sinpsi*sintheta*sinphi+cospsi*cosphi),
cospsi*sintheta/costheta*(-
sinpsi*sintheta*cosphi+cospsi*sinphi)+cospsi*sintheta/costheta*(sinpsi*sintheta*cosphi-
cospsi*sinphi)];


% global wind disturbance translated into local wind
wnoavec = Ainv*wxyzvec;

% global vehicle velocity translated into local velocity
noadotvec = Ainv*xyzdotvec;

% drag loading due to difference between local wind and local velocity
% simulates either the buffeting caused by a strong wind or the minor
% amounts of damping caused by drag at high velocities
Fnoarelativevec = (wnoavec-noadotvec).*[ks ks ku]';

% drag loading translated into global coordinates
Fxyzrelativevec = A*Fnoarelativevec;

% global angular rate vector obtained from euler angle rates - euler rates
% are NOT the same as global rates since two of the three euler rates are
% rotated by the other euler angles
omegaxyzvec = M*eulerdotvec;
% global angular rate vector translated into local angular rate vector
omeganoavec = Ainv*omegaxyzvec;

% vector of variables that take into account the pitch speed of a prop -
% the translation rate at which the prop spinning at its current RPM fails
% to produce thrust.  Picture a windmill spinning freely in a blowing wind
kpvec = onevec./(avec/2/pi*pitch);
```

```
% vector of variables handling the above described situation as caused by
% roll or pitch of the vehicle
kwvec = L*kpvec;

% vector of the absolute values of the thrust forces generated by each of
% the four props, assuming they are rotating in their intended direction
Tvec = kt*abs(avec).*avec.*(onevec+kwvec.*[-omeganoavec(2) omeganoavec(1) omeganoavec(2) -
omeganoavec(1)]'+kpvec*(noadotvec(3)-wnoavec(3)));

% vector of the absolute values of the drag (torque) forces produced by
% each of the four props, assuming they are rotating in their intended
% direction
Dvec = kd*abs(avec).*avec.*(onevec+kwvec.*[-omeganoavec(2) omeganoavec(1) omeganoavec(2) -
omeganoavec(1)]'+kpvec*(noadotvec(3)-wnoavec(3)));

% vehicle torque caused by temporary inequalities in the amount the props
% are being accelerated or decelerated.  Generally small.
Mma = Jt*[1 -1 1 -1]*adotvec;

% total local torque exerted on the vehicle from thrust differentials
% causing roll and pitch moments, prop drag and prop accel/deccel causing
% yaw torque, and local disturbance torques.
torquenoatotalvec = [L*(Tvec(4)-Tvec(2)) L*(Tvec(1)-Tvec(3)) [1 -1 1 -1]*Dvec+Mma]'+tnoavec;

% local torques and angular rate coupling (precessive effects) yield total
% rate of change of local rates
omeganoadotvec = inv(J)*torquenoatotalvec + inv(J)*cross(omeganoavec,J*omeganoavec);

% total global forces exerted on the vehicle from gravity, prop thrust, and
% global wind loading cause global accelerations
xyzdbldotvec = [0 0 g]'-1/m*A*[0 0 onevec'*Tvec]'-Fxyzrelativevec;

noadbldotvec = Ainv*xyzdbldotvec;

% rate of change of euler rates is determined by differentiation of the
% base relationship between euler angle rates and global angular rates.
eulerdbldotvec = phidot*diffMinvAphi*omeganoavec + thetadot*diffMinvAtheta*omeganoavec +
psidot*diffMinvApsi*omeganoavec + Minv*A*omeganoadotvec;

% state derivative returned
statedervs = [xyzdbldotvec; xyzdotvec; eulerdbldotvec; eulerdotvec];
localmeas = [noadbldotvec; omeganoavec];
```

```
% ThreeDAFVstatedervThrust.m
% Author: Eryk Nice

function [statedervs, localmeas]=ThreeDAFVstatedervThrust(inputvec)

% This version of state dynamics accepts thrust as an input

global L ku ks J m g pitch kt_f kd_f kv ki R Jt

kt = kt_f;
kd = kd_f;

% Current state
% global velocities
xdot = inputvec(1);
ydot = inputvec(2);
zdot = inputvec(3);
% global position
x = inputvec(4);
y = inputvec(5);
z = inputvec(6);
% Euler angle rates
phidot = inputvec(7);
thetadot = inputvec(8);
psidot = inputvec(9);
% Euler angles
phi = inputvec(10);
theta = inputvec(11);
psi = inputvec(12);
% prop rad/s
T1 = inputvec(13);
T2 = inputvec(14);
T3 = inputvec(15);
T4 = inputvec(16);


% Disturbances
% global wind
wx = 0;
wy = 0;
wz = 0;
% local torques
tn = 0;
to = 0;
ta = 0;

% Assemble variables in to vectors for manipulation
xyzdotvec = [xdot ydot zdot]';
xyzvec = [x y z]';
eulerdotvec = [phidot thetadot psidot]';
eulervec = [phi theta psi]';
Tvec = [T1 T2 T3 T4]';
wxyzvec = [wx wy wz]';
tnoavec = [tn to ta]';
```

```
onevec = ones(4,1);

% Compute trig values once
sinphi = sin(phi);
sintheta = sin(theta);
sinpsi = sin(psi);

cosphi = cos(phi);
costheta = cos(theta);
cospsi = cos(psi);


% Rotation/Translation matrices
% rotation matrix from local to global coordinates and its inverse
A = [cospsi*costheta cospsi*sintheta*sinphi-sinpsi*cosphi cospsi*sintheta*cosphi+sinpsi*sinphi;
sinpsi*costheta sinpsi*sintheta*sinphi+cospsi*cosphi sinpsi*sintheta*cosphi-cospsi*sinphi; -sintheta
costheta*sinphi costheta*cosphi];
Ainv = inv(A);

% matrix transformation from euler angular rates to global angular rates
% and its inverse
Minv = [cospsi/costheta sinpsi/costheta 0; -sinpsi cospsi 0; cospsi*sintheta/costheta
sinpsi*sintheta/costheta  1];
M = inv(Minv);
% partial derivative of Minv with respect to psi and theta, used in
% computing euler angle second derivatives
diffMinvAphi = [0,
cospsi/costheta*(cospsi*sintheta*cosphi+sinpsi*sinphi)+sinpsi/costheta*(sinpsi*sintheta*cosphi-
cospsi*sinphi),  cospsi/costheta*(-cospsi*sintheta*sinphi+sinpsi*cosphi)+sinpsi/costheta*(-
sinpsi*sintheta*sinphi-cospsi*cosphi);
   0,  -sinpsi*(cospsi*sintheta*cosphi+sinpsi*sinphi)+cospsi*(sinpsi*sintheta*cosphi-cospsi*sinphi),  -
sinpsi*(-cospsi*sintheta*sinphi+sinpsi*cosphi)+cospsi*(-sinpsi*sintheta*sinphi-cospsi*cosphi);
   0,
cospsi*sintheta/costheta*(cospsi*sintheta*cosphi+sinpsi*sinphi)+sinpsi*sintheta/costheta*(sinpsi*sinth
eta*cosphi-cospsi*sinphi)+costheta*cosphi,  cospsi*sintheta/costheta*(-
cospsi*sintheta*sinphi+sinpsi*cosphi)+sinpsi*sintheta/costheta*(-sinpsi*sintheta*sinphi-
cospsi*cosphi)-costheta*sinphi];
diffMinvAtheta = [0,  cospsi/costheta^2*(cospsi*sintheta*sinphi-
sinpsi*cosphi)*sintheta+cospsi^2*sinphi+sinpsi/costheta^2*(sinpsi*sintheta*sinphi+cospsi*cosphi)*sin
theta+sinpsi^2*sinphi,
cospsi/costheta^2*(cospsi*sintheta*cosphi+sinpsi*sinphi)*sintheta+cospsi^2*cosphi+sinpsi/costheta^2
*(sinpsi*sintheta*cosphi-cospsi*sinphi)*sintheta+sinpsi^2*cosphi;
   0,  0,  0;
   cospsi^2*costheta+sinpsi^2*costheta-costheta,  cospsi*(cospsi*sintheta*sinphi-
sinpsi*cosphi)+cospsi*sintheta^2/costheta^2*(cospsi*sintheta*sinphi-
sinpsi*cosphi)+cospsi^2*sintheta*sinphi+sinpsi*(sinpsi*sintheta*sinphi+cospsi*cosphi)+sinpsi*sinthet
a^2/costheta^2*(sinpsi*sintheta*sinphi+cospsi*cosphi)+sinpsi^2*sintheta*sinphi-sintheta*sinphi,
cospsi*(cospsi*sintheta*cosphi+sinpsi*sinphi)+cospsi*sintheta^2/costheta^2*(cospsi*sintheta*cosphi+
sinpsi*sinphi)+cospsi^2*sintheta*cosphi+sinpsi*(sinpsi*sintheta*cosphi-
cospsi*sinphi)+sinpsi*sintheta^2/costheta^2*(sinpsi*sintheta*cosphi-
cospsi*sinphi)+sinpsi^2*sintheta*cosphi-sintheta*cosphi];
diffMinvApsi = [0,  cospsi/costheta*(-sinpsi*sintheta*sinphi-
cospsi*cosphi)+cospsi/costheta*(sinpsi*sintheta*sinphi+cospsi*cosphi),   cospsi/costheta*(-
sinpsi*sintheta*cosphi+cospsi*sinphi)+cospsi/costheta*(sinpsi*sintheta*cosphi-cospsi*sinphi);
```

```
   0,  -sinpsi*(-sinpsi*sintheta*sinphi-cospsi*cosphi)-sinpsi*(sinpsi*sintheta*sinphi+cospsi*cosphi),   -
sinpsi*(-sinpsi*sintheta*cosphi+cospsi*sinphi)-sinpsi*(sinpsi*sintheta*cosphi-cospsi*sinphi);
   0,  cospsi*sintheta/costheta*(-sinpsi*sintheta*sinphi-
cospsi*cosphi)+cospsi*sintheta/costheta*(sinpsi*sintheta*sinphi+cospsi*cosphi),
cospsi*sintheta/costheta*(-
sinpsi*sintheta*cosphi+cospsi*sinphi)+cospsi*sintheta/costheta*(sinpsi*sintheta*cosphi-
cospsi*sinphi)];


% global wind disturbance translated into local wind
wnoavec = Ainv*wxyzvec;

% global vehicle velocity translated into local velocity
noadotvec = Ainv*xyzdotvec;

% drag loading due to difference between local wind and local velocity
% simulates either the buffeting caused by a strong wind or the minor
% amounts of damping caused by drag at high velocities
Fnoarelativevec = (wnoavec-noadotvec).*[ks ks ku]';

% drag loading translated into global coordinates
Fxyzrelativevec = A*Fnoarelativevec;

% global angular rate vector obtained from euler angle rates - euler rates
% are NOT the same as global rates since two of the three euler rates are
% rotated by the other euler angles
omegaxyzvec = M*eulerdotvec;
% global angular rate vector translated into local angular rate vector
omeganoavec = Ainv*omegaxyzvec;

% vector of the absolute values of the drag (torque) forces produced by
% each of the four props, assuming they are rotating in their intended
% direction
Dvec = Tvec/kt*kd;

% total local torque exerted on the vehicle from thrust differentials
% causing roll and pitch moments, prop drag and prop accel/deccel causing
% yaw torque, and local disturbance torques.
torquenoatotalvec = [L*(Tvec(4)-Tvec(2)) L*(Tvec(1)-Tvec(3)) [1 -1 1 -1]*Dvec]'+tnoavec;

% local torques and angular rate coupling (precessive effects) yield total
% rate of change of local rates
omeganoadotvec = inv(J)*torquenoatotalvec + inv(J)*cross(omeganoavec,J*omeganoavec);

% total global forces exerted on the vehicle from gravity, prop thrust, and
% global wind loading cause global accelerations
xyzdbldotvec = [0 0 g]'-1/m*A*[0 0 onevec'*Tvec]'-Fxyzrelativevec;

noadbldotvec = Ainv*xyzdbldotvec;

% rate of change of euler rates is determined by differentiation of the
% base relationship between euler angle rates and global angular rates.
eulerdbldotvec = phidot*diffMinvAphi*omeganoavec + thetadot*diffMinvAtheta*omeganoavec +
psidot*diffMinvApsi*omeganoavec + Minv*A*omeganoadotvec;
```

```
% state derivative returned
statedervs = [xyzdbldotvec; xyzdotvec; eulerdbldotvec; eulerdotvec];
localmeas = [noadbldotvec; omeganoavec];
```

```
% thrusttorads.m
% Author: Eryk Nice

function proprads = thrusttorads(thrust)

% This function converts a commanded thrust into a commanded prop rad/s

global kt

if thrust ==0
   proprads = 0;
elseif thrust > 0
   proprads = (thrust/kt)^0.5;
else
   proprads = -(-thrust/kt)^0.5;
end
```

Simulation Animation Files

```
% animate_afv.m
% Original Author: Sean Breheny
% Modified by: Eryk Nice

% AFV 3D animation
% NOTE: requires display_afv.m,make_afv.m,rotobj.m,myrot.m, and afv4.bmp

global cube_x cube_y cube_z rod0_x rod0_y rod0_z rod1_x rod1_y rod1_z
global rod2_x rod2_y rod2_z rod3_x rod3_y rod3_z
global motor0_x motor0_y motor0_z motor1_x motor1_y motor1_z
global motor2_x motor2_y motor2_z motor3_x motor3_y motor3_z
global prop0_x prop0_y prop0_z prop1_x prop1_y prop1_z
global prop2_x prop2_y prop2_z prop3_x prop3_y prop3_z
global l l2 X

% Read in texture map for AFV body (cube)
[X,map]=imread('afv4.bmp');

make_afv % Construct the matrices that describe the AFV (used by display_afv)

% Data to show motion of AFVs
t = xvecf.time;

% Create AFV file for output
% NOTE: compression is strange, if set to 'none', AVI plays back slowly and in a jerky manner (in
Windows Media Player)
% if set to CinePak, file size is the same but file plays smoother in Windows Media Player
% NOTE: Current frame size (1000x700) is large enough that you should play the video in full screen
to get best results
clear av
av=avifile('test11.avi','compression','CinePak');

% Frame loop
for q=1:1:length(t)
   h=figure(1);
   plot3(0,0,0) % Display axes
   hold on

   b=get(h,'CurrentAxes'); % Get handle to axes
   %set(b,'Visible','off'); % Turn them off
   set(h,'Renderer','zbuffer','MenuBar','none','Position',[10 10 1000 700]); % Last two numbers set
frame size (horiz vert)

   % Call display_afv once per AFV
   % Format is display_afv(x,y,z,psi,phi,theta)
   % Rotation is performed theta first, then phi, then psi
   % psi is rotation about x, phi about y, theta about z
   % follows right-hand rule
   display_afv(1/0.0254*xvecf.signals.values(1,1,q),-1/0.0254*yvecf.signals.values(1,1,q),-
1/0.0254*zvecf.signals.values(1,1,q),pi/180*rollvecf.signals.values(1,1,q),-
pi/180*pitchvecf.signals.values(1,1,q),-pi/180*yawvecf.signals.values(1,1,q))
```

```
%     display_afv(-2,0,0,psi(q),0,0)
%     display_afv(0,psi(q),0,0,0,0)
%     display_afv(2,bb(4,q),0,0,0,0)
%     display_afv(4,0,0,0,0,bb(5,q))
    % May need to change this axis setting to make the AFVs fit nicely in window
    % (i.e., this controls the correspondence between meters and pixels, each AFV is 1 meter wide
    axis(40*[-1 1 -1 1 -1 1])

    b=get(h,'CurrentAxes'); % Get handle to axes
    %set(b,'Visible','off'); % Turn them off

    % Add frame to AVI
    av = addframe(av,h);
    hold off % release figure so it will be cleared at beginning of loop
end
av=close(av);
```

```
% display_afv.m
% Original Author: Sean Breheny
% Modified by: Eryk Nice

function display_afv(x,y,z,psi,phi,theta)


global cube_x cube_y cube_z rod0_x rod0_y rod0_z rod1_x rod1_y rod1_z
global rod2_x rod2_y rod2_z rod3_x rod3_y rod3_z
global motor0_x motor0_y motor0_z motor1_x motor1_y motor1_z
global motor2_x motor2_y motor2_z motor3_x motor3_y motor3_z
global prop0_x prop0_y prop0_z prop1_x prop1_y prop1_z
global prop2_x prop2_y prop2_z prop3_x prop3_y prop3_z
global l l2 X

% Update each part according to position and orientation

[hcube_x,hcube_y,hcube_z]=rotobj(psi,phi,theta,cube_x,cube_y,cube_z,x,y,z);

[hrod0_x,hrod0_y,hrod0_z]=rotobj(psi,phi,theta,rod0_x,rod0_y,rod0_z,x,y,z);
[hrod1_x,hrod1_y,hrod1_z]=rotobj(psi,phi,theta,rod1_x,rod1_y,rod1_z,x,y,z);
[hrod2_x,hrod2_y,hrod2_z]=rotobj(psi,phi,theta,rod2_x,rod2_y,rod2_z,x,y,z);
[hrod3_x,hrod3_y,hrod3_z]=rotobj(psi,phi,theta,rod3_x,rod3_y,rod3_z,x,y,z);

[hmotor0_x,hmotor0_y,hmotor0_z]=rotobj(psi,phi,theta,motor0_x,motor0_y,motor0_z,x,y,z);
[hmotor1_x,hmotor1_y,hmotor1_z]=rotobj(psi,phi,theta,motor1_x,motor1_y,motor1_z,x,y,z);
[hmotor2_x,hmotor2_y,hmotor2_z]=rotobj(psi,phi,theta,motor2_x,motor2_y,motor2_z,x,y,z);
[hmotor3_x,hmotor3_y,hmotor3_z]=rotobj(psi,phi,theta,motor3_x,motor3_y,motor3_z,x,y,z);

[hprop0_x,hprop0_y,hprop0_z]=rotobj(psi,phi,theta,prop0_x,prop0_y,prop0_z,x,y,z);
[hprop1_x,hprop1_y,hprop1_z]=rotobj(psi,phi,theta,prop1_x,prop1_y,prop1_z,x,y,z);
[hprop2_x,hprop2_y,hprop2_z]=rotobj(psi,phi,theta,prop2_x,prop2_y,prop2_z,x,y,z);
[hprop3_x,hprop3_y,hprop3_z]=rotobj(psi,phi,theta,prop3_x,prop3_y,prop3_z,x,y,z);

[p q]=size(hcube_z);
l2=surf(hcube_x,hcube_y,hcube_z,0.65*ones(p,q));
set(l2,'CData',X,'FaceColor','texturemap')
hold on

[p q]=size(hrod0_z);
surf(hrod0_x,hrod0_y,hrod0_z,0.1*ones(p,q));
surf(hrod1_x,hrod1_y,hrod1_z,0.1*ones(p,q));
surf(hrod2_x,hrod2_y,hrod2_z,0.1*ones(p,q));
surf(hrod3_x,hrod3_y,hrod3_z,0.1*ones(p,q));

[p q]=size(hmotor0_z);
surf(hmotor0_x,hmotor0_y,hmotor0_z,0.55*ones(p,q))
surf(hmotor1_x,hmotor1_y,hmotor1_z,0.7*ones(p,q))
surf(hmotor2_x,hmotor2_y,hmotor2_z,0.7*ones(p,q))
surf(hmotor3_x,hmotor3_y,hmotor3_z,0.7*ones(p,q))

[p q]=size(hprop0_z);
surf(hprop0_x,hprop0_y,hprop0_z,0.55*ones(p,q))
surf(hprop1_x,hprop1_y,hprop1_z,0.7*ones(p,q))
```

```
surf(hprop2_x,hprop2_y,hprop2_z,0.7*ones(p,q))
surf(hprop3_x,hprop3_y,hprop3_z,0.7*ones(p,q))

shading flat
caxis([0 1])
b=[gray(32);hsv(32)];
colormap(b)
l=light;
```

```
% make_afv.m
% Original Author: Sean Breheny
% Modified by: Eryk Nice

% Construct 3D AFV graphics object

global cube_x cube_y cube_z rod0_x rod0_y rod0_z rod1_x rod1_y rod1_z
global rod2_x rod2_y rod2_z rod3_x rod3_y rod3_z
global motor0_x motor0_y motor0_z motor1_x motor1_y motor1_z
global motor2_x motor2_y motor2_z motor3_x motor3_y motor3_z
global prop0_x prop0_y prop0_z prop1_x prop1_y prop1_z
global prop2_x prop2_y prop2_z prop3_x prop3_y prop3_z


% Construct AFV

% Make components

% 9" R prop rotor with .5" thickness
[p_prop_x,p_prop_y,p_prop_z]=cylinder(ones(1,2),25);
p_prop_x=9*p_prop_x;
p_prop_y=9*p_prop_y;
p_prop_z=0.5*(p_prop_z)+.5;

% 1.5" dia motor cylinders with 2.5" height
[p_motor_x,p_motor_y,p_motor_z]=cylinder(ones(1,2),15);
p_motor_x=0.75*p_motor_x;
p_motor_y=0.75*p_motor_y;
p_motor_z=2.5*(p_motor_z)-2.5;

% 3/8" dia rod with 12" length
[p_rod_x,p_rod_y,p_rod_z]=cylinder(ones(1,2),10);
% 0.5745
p_rod_x=3/8*p_rod_x/2;
p_rod_y=3/8*p_rod_y/2;
p_rod_z=12*(p_rod_z);

% 3.5" cube
p_cube_x=3.5*[0 1 1 0;0 0 0 0;0 0 0 0;1 1 1 1;1 1 1 1];
p_cube_y=3.5*[zeros(5,2) ones(5,2)];
p_cube_z=3.5*[0 0 0 0;0 0 0 0;0 1 1 0;0 1 1 0;0 0 0 0];


% Assemble AFV

% Center cube at 0,0,0

cube_x=p_cube_x-3.5/2;
cube_y=p_cube_y-3.5/2;
cube_z=p_cube_z-3.5/2;

% Rod 0

psi=-pi/2;
```

```
phi=0;
theta=0*pi/180;

xoff=0;
yoff=3.5/2;
zoff=0;

[rod0_x,rod0_y,rod0_z]=rotobj(psi,phi,theta,p_rod_x,p_rod_y,p_rod_z,xoff,yoff,zoff);

% Rod 1

psi=-pi/2;
phi=pi/2;
theta=0*pi/180;

xoff=3.5/2;
yoff=0;
zoff=0;

[rod1_x,rod1_y,rod1_z]=rotobj(psi,phi,theta,p_rod_x,p_rod_y,p_rod_z,xoff,yoff,zoff);

% Rod 2

psi=-pi/2;
phi=pi;
theta=0*pi/180;

xoff=0;
yoff=-3.5/2;
zoff=0;

[rod2_x,rod2_y,rod2_z]=rotobj(psi,phi,theta,p_rod_x,p_rod_y,p_rod_z,xoff,yoff,zoff);


% Rod 3

psi=-pi/2;
phi=3*pi/2;
theta=0*pi/180;

xoff=-3.5/2;
yoff=0;
zoff=0;

[rod3_x,rod3_y,rod3_z]=rotobj(psi,phi,theta,p_rod_x,p_rod_y,p_rod_z,xoff,yoff,zoff);

% Motors

propraddist = 12+3.5/2;
motor0_x=p_motor_x+propraddist;
motor0_y=p_motor_y;
motor0_z=p_motor_z;

motor1_x=p_motor_x-propraddist;
motor1_y=p_motor_y;
```

```
motor1_z=p_motor_z;

motor2_x=p_motor_x;
motor2_y=p_motor_y+propraddist;
motor2_z=p_motor_z;

motor3_x=p_motor_x;
motor3_y=p_motor_y-propraddist;
motor3_z=p_motor_z;

% Props

prop0_x=p_prop_x+propraddist;
prop0_y=p_prop_y;
prop0_z=p_prop_z+0.1;

prop1_x=p_prop_x-propraddist;
prop1_y=p_prop_y;
prop1_z=p_prop_z+0.1;

prop2_x=p_prop_x;
prop2_y=p_prop_y+propraddist;
prop2_z=p_prop_z+0.1;

prop3_x=p_prop_x;
prop3_y=p_prop_y-propraddist;
prop3_z=p_prop_z+0.1;
```

```
% myrot.m
% Author: Sean Breheny

function outvect=myrot(psi,phi,theta,invect)

Rpsi=[1 0 0;0 cos(psi) -sin(psi);0 sin(psi) cos(psi)];
Rphi=[cos(phi) 0 sin(phi);0 1 0;-sin(phi) 0 cos(phi)];
Rtheta=[cos(theta) -sin(theta) 0;sin(theta) cos(theta) 0;0 0 1];

R=Rpsi*Rphi*Rtheta;

outvect=R*invect;
```

```
% rotobj.m
% Author: Sean Breheny

function [xout,yout,zout]=rotobj (psi,phi,theta,xin,yin,zin,xoff,yoff,zoff)
[p,q]=size(xin);

xout=xin;
yout=yin;
zout=zin;

for x1=1:1:p
   for x2=1:1:q
      V=[xin(x1,x2);yin(x1,x2);zin(x1,x2)];
      V=myrot(psi,phi,theta,V);
      xout(x1,x2)=V(1)+xoff;
      yout(x1,x2)=V(2)+yoff;
      zout(x1,x2)=V(3)+zoff;
   end
end
```

APPENDIX F:

Pro/E FILE INFORMATION AND MACHINING SPEC SHEETS

Table F-7: Pro/E Files Information

Name	**Parent assembly**
prt_imumount	asm_bodycent
prt_imu	asm_bodycent
prt_centbaseimusidestandoff	asm_bodycent
prt_strutmount	asm_bodycent
prt_strutplulong	asm_bodycent
prt_strutplug	asm_bodycent
prt_centbaseboardsidestandoff	asm_bodycent
prt_boardmount	asm_bodycent
prt_strutimuside	asm_bodycent
prt_strutlongprop	asm_bodycent
prt_strutshortprop	asm_bodycent
prt_strutbasewiremount	asm_bodycent
asm_eemain	asm_bodycent
prt_eemainboard	asm_eemain
prt_eestandoff	asm_eemain
prt_centthreadrod	asm_bodycent
asm_lipolypack2x2	asm_bodycent
prt_lipolybattcell	asm_lipolypack(config)
prt_eebattretainer	asm_bodycent
prt_recievermount	asm_bodycent
prt_reciever	asm_bodycent
prt_recieverclipbar	asm_bodycent
asm_lipolybatthungnew	asm_bodycent
prt_lipolybatthanger	asm_lipolybatthungnew
asm_lipolypack2x3	asm_lipolybatthungnew
asm_lipolypack2x4	asm_lipolybatthungnew
prt_lipolybatthangerretainerrod	asm_lipolybatthungnew
asm_pulleyboxmaxcim	asm_bodycent
prt_pulleyboxmaxcim	asm_pulleyboxmaxcim
prt_pulleyboxextension	asm_pulleyboxmaxcim
prt_maxcimmotor	asm_pulleyboxmaxcim
prt_quarterbearing	asm_pulleyboxmaxcim
prt_propshaft	asm_pulleyboxmaxcim
prt_encoder	asm_pulleyboxmaxcim
prt_18x6prop	asm_pulleyboxmaxcim
prt_propwasher	asm_pulleyboxmaxcim

Table F-7 (Continued)

prt_quartershaftcollar	asm_pulleyboxmaxcim
prt_15thpulley	asm_pulleyboxmaxcim
prt_100thpulley	asm_pulleyboxmaxcim
prt_strutend	asm_pulleyboxmaxcim
prt_strutplug	asm_pulleyboxmaxcim
asm_landinggear	asm_bodycent
prt_landinggearbase	asm_landinggear
prt_strutlanding	asm_landinggear
prt_eestandoff	asm_landinggear
prt_eemotorboard	asm_landinggear
prt_landingbaseplug	asm_landinggear
prt_landingspringchannel	asm_landinggear
asm_bodycent	

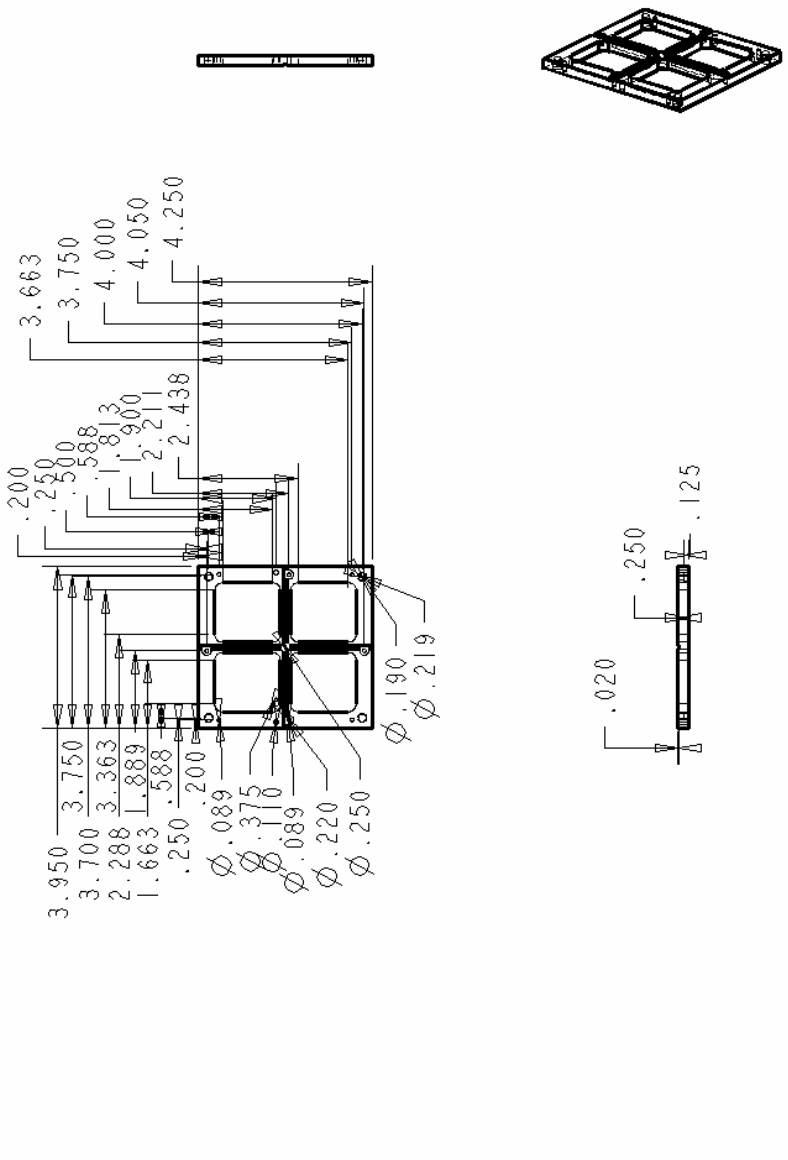
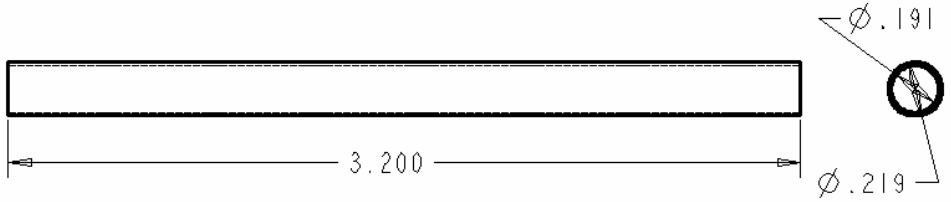


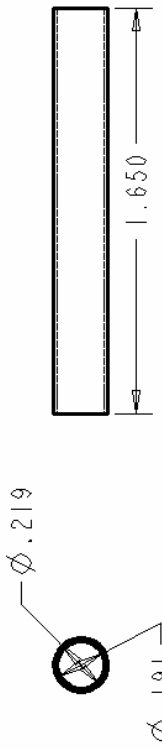
Figure F-9: drw_boardmount



For Educati

SCALE 2.000

Figure F-10: drw_centbaseboardsidestandoff



SCALE 2.000

For Educati

Figure F-11: drw_centbaseimusidestandoff

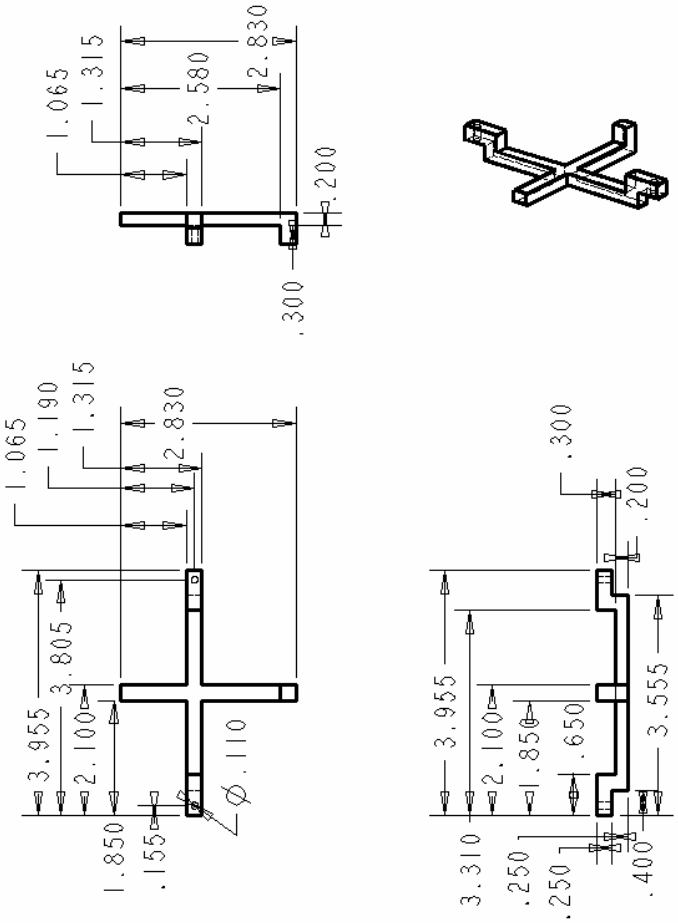


Figure F-12: drw_eebattretainer

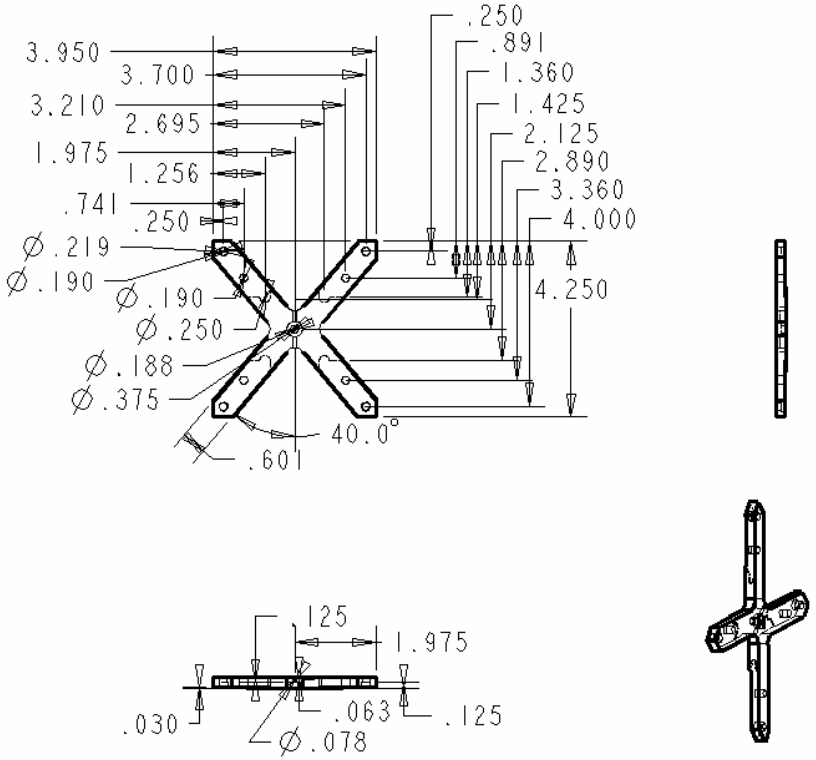


Figure F-13: drw_imumount

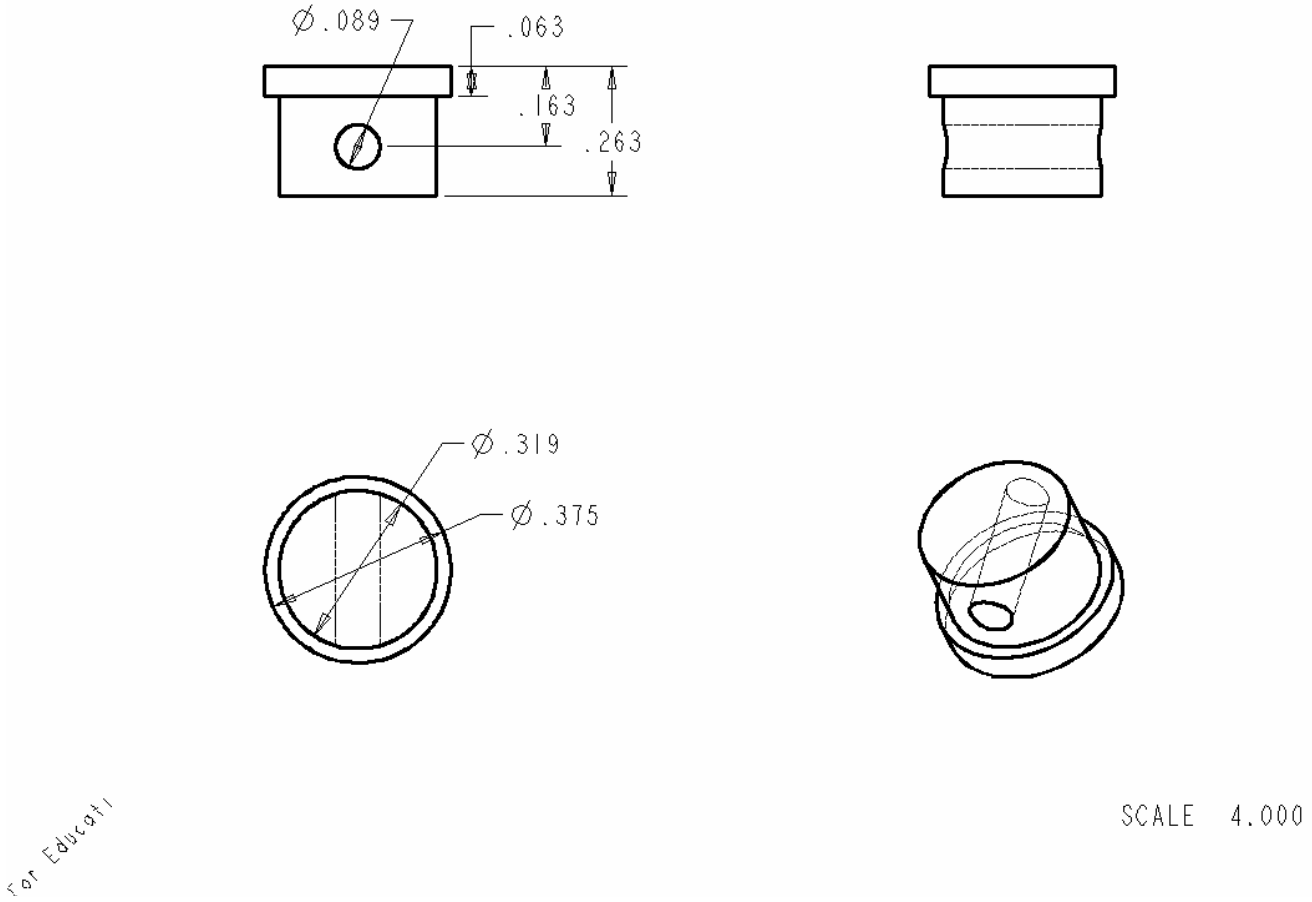


Figure F-14: drw_landingbaseplug

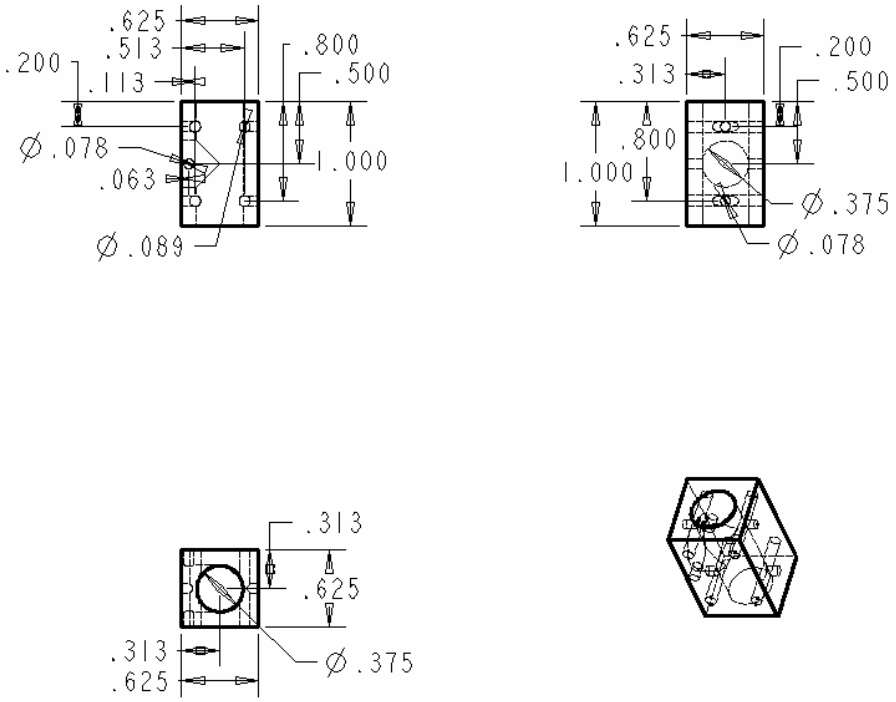


Figure F-15: drw_landinggearbase

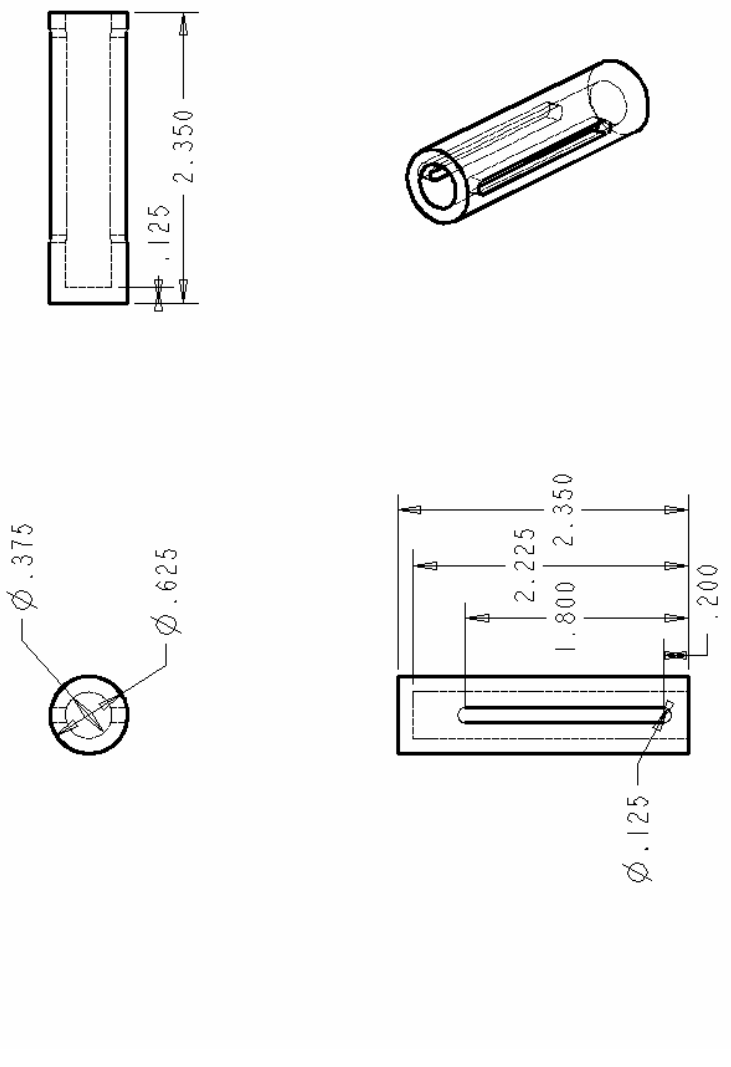


Figure F-16: drw_landingspringchannel

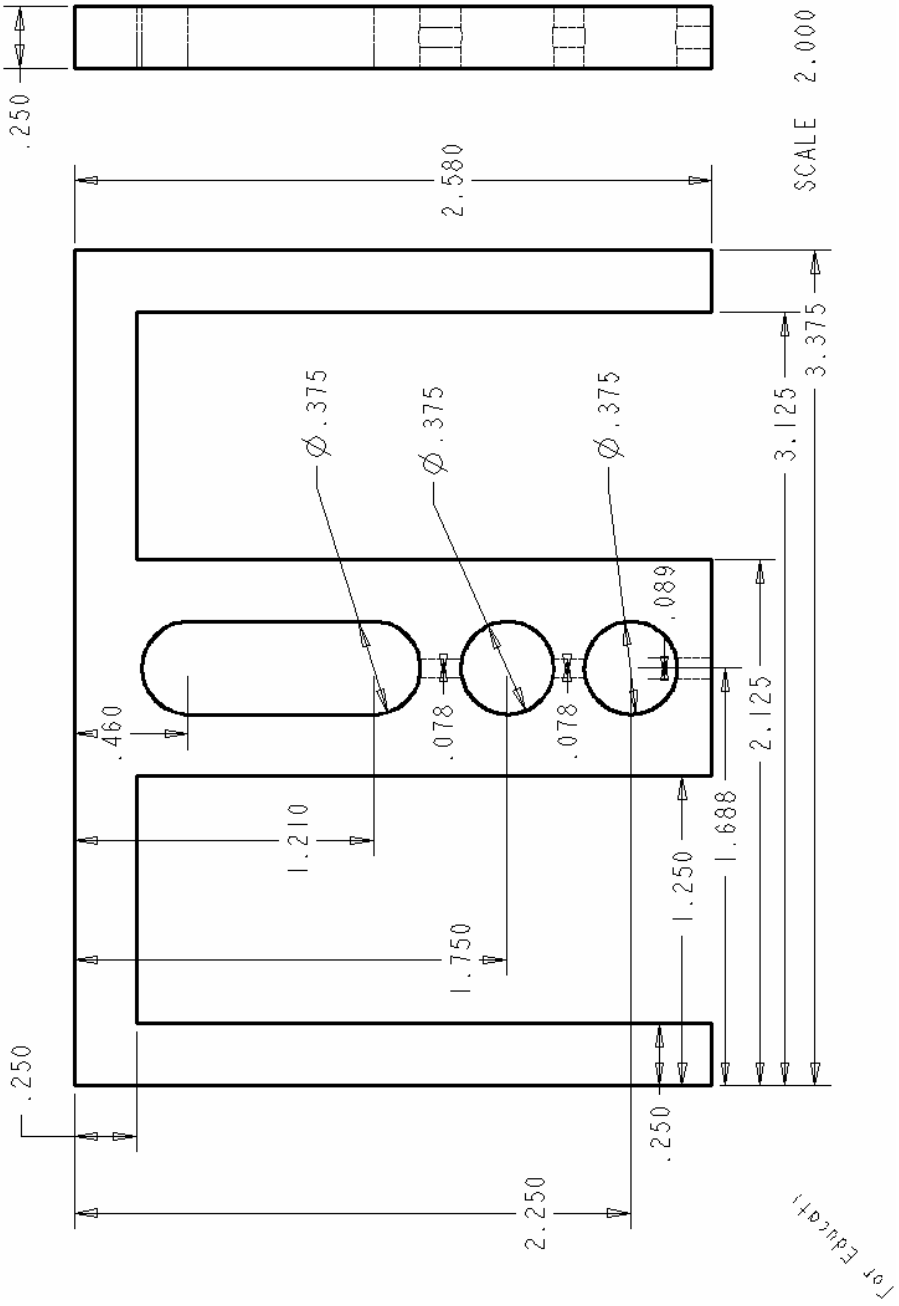


Figure F-17: drw_lipolybatthanger

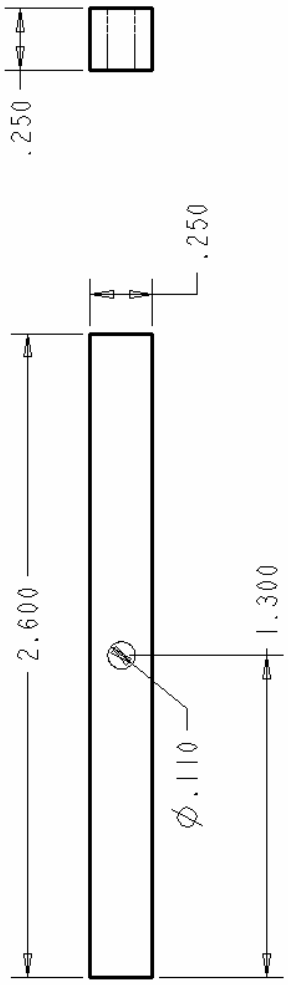


Figure F-18: drw_lipolybatthangerretainerrod

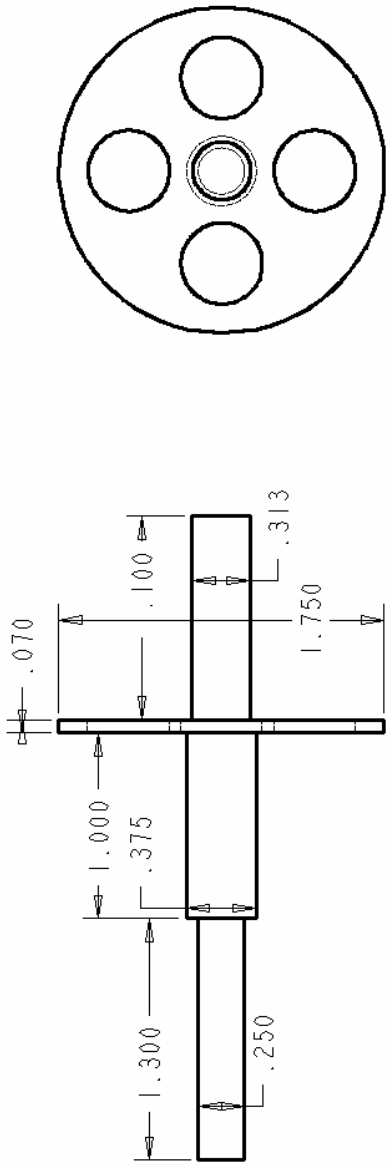


Figure F-19: drw_propshaft

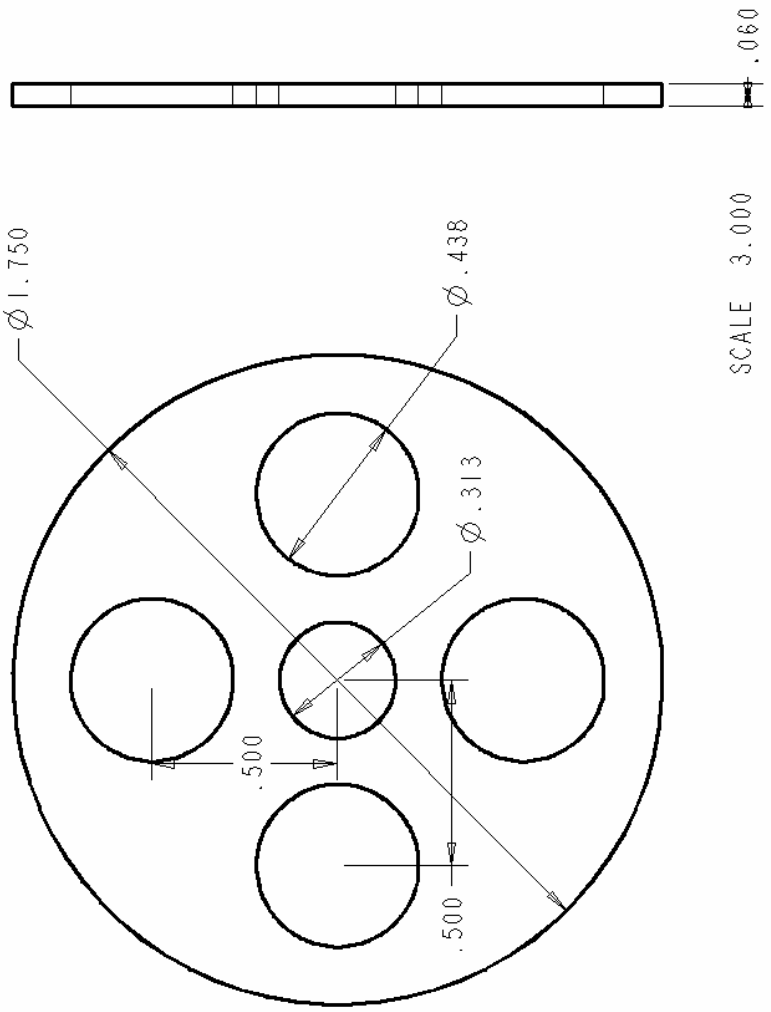


Figure F-20: drw_propwasher

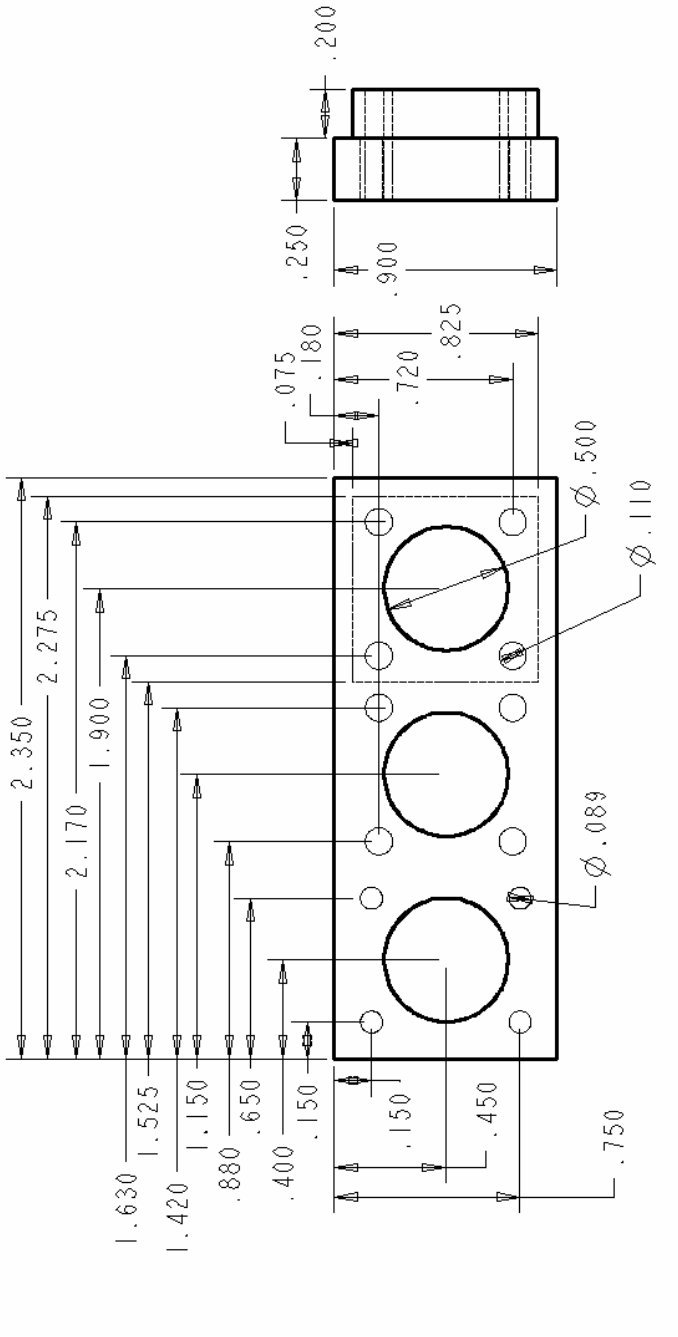


Figure F-21: drw_pulleyboxextension

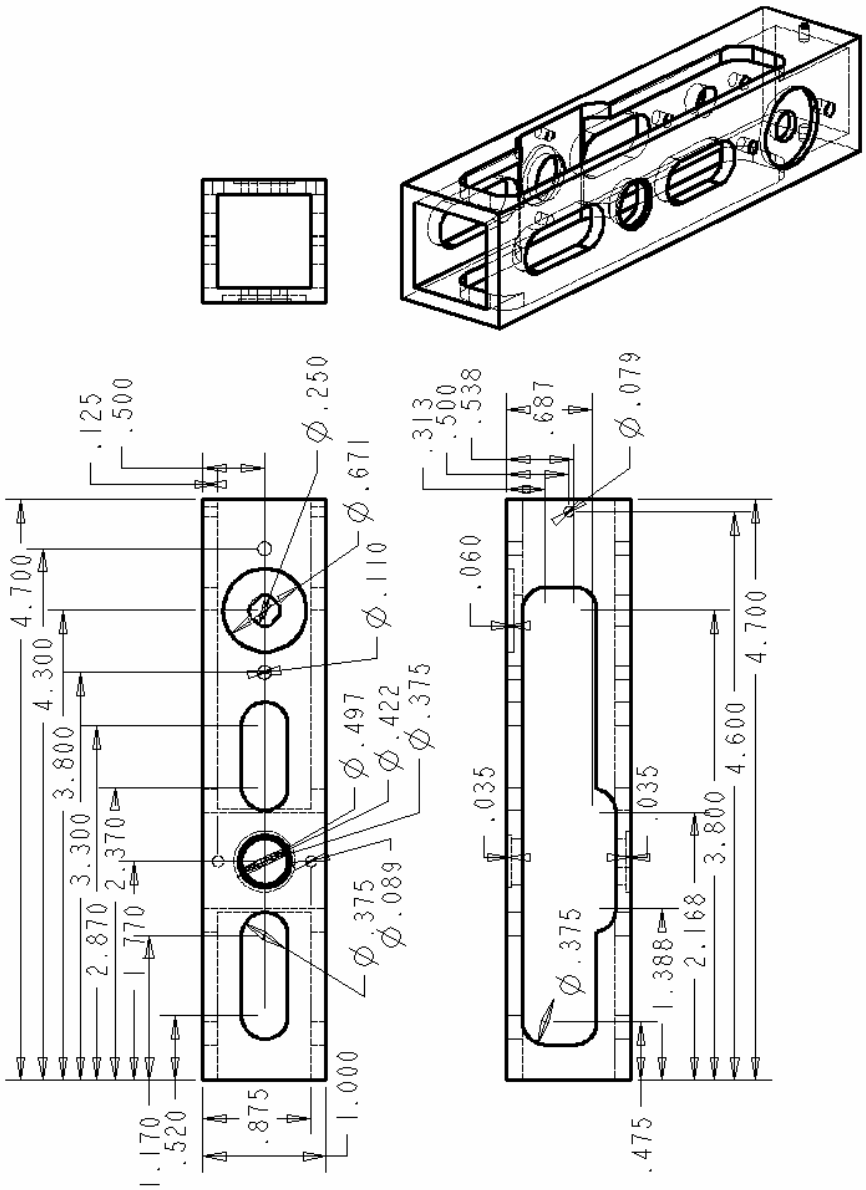
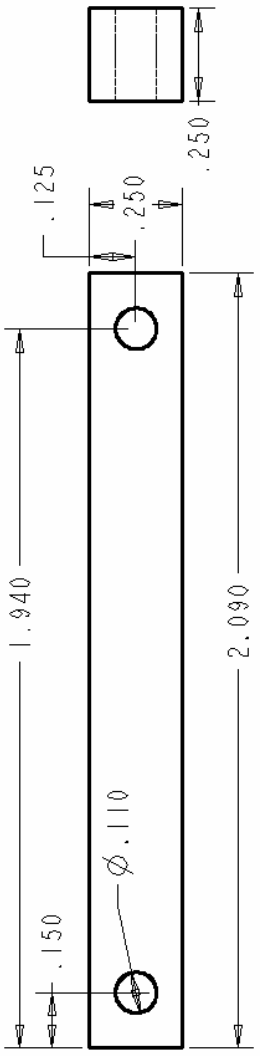


Figure F-22: drw_pulleyboxmaxcim



SCALE 3.000

Figure F-23: drw_recieverclipbar

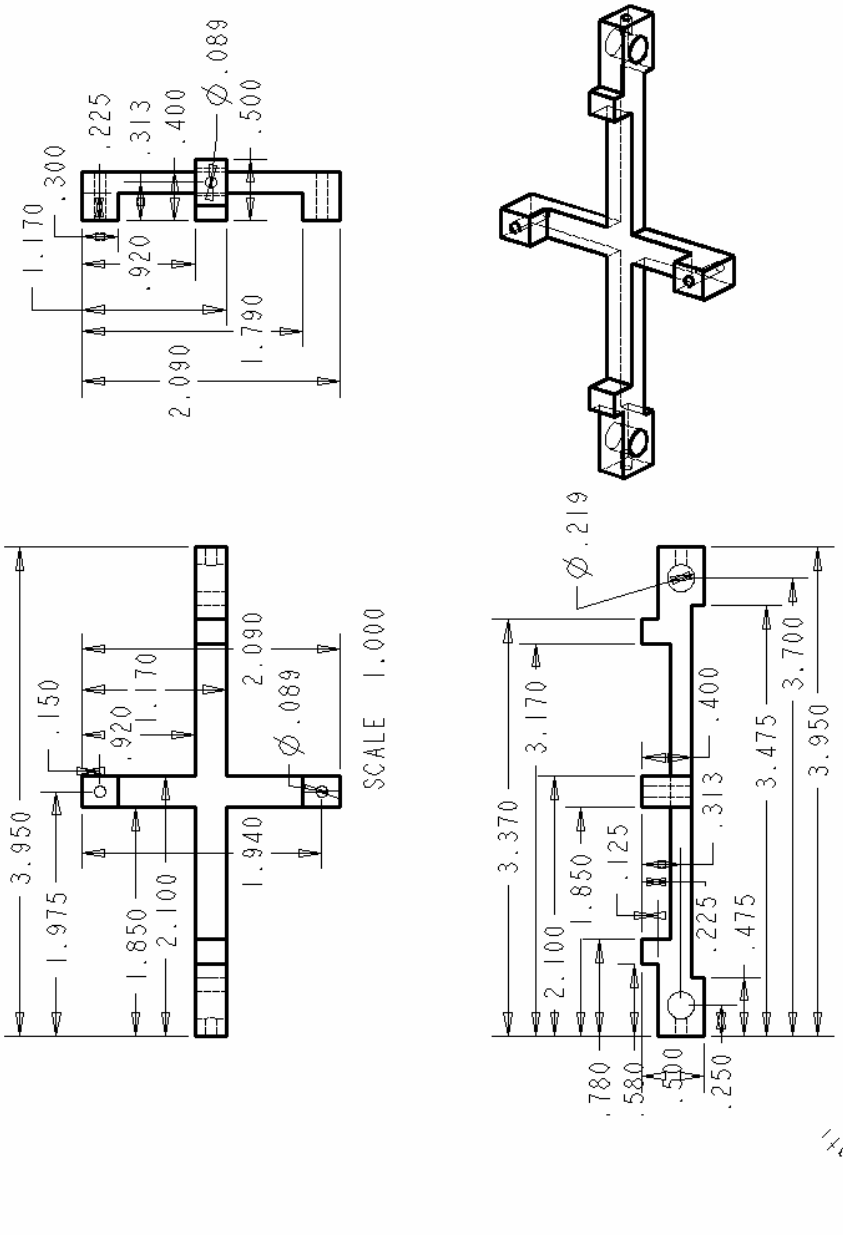


Figure F-24: drw_recievermount

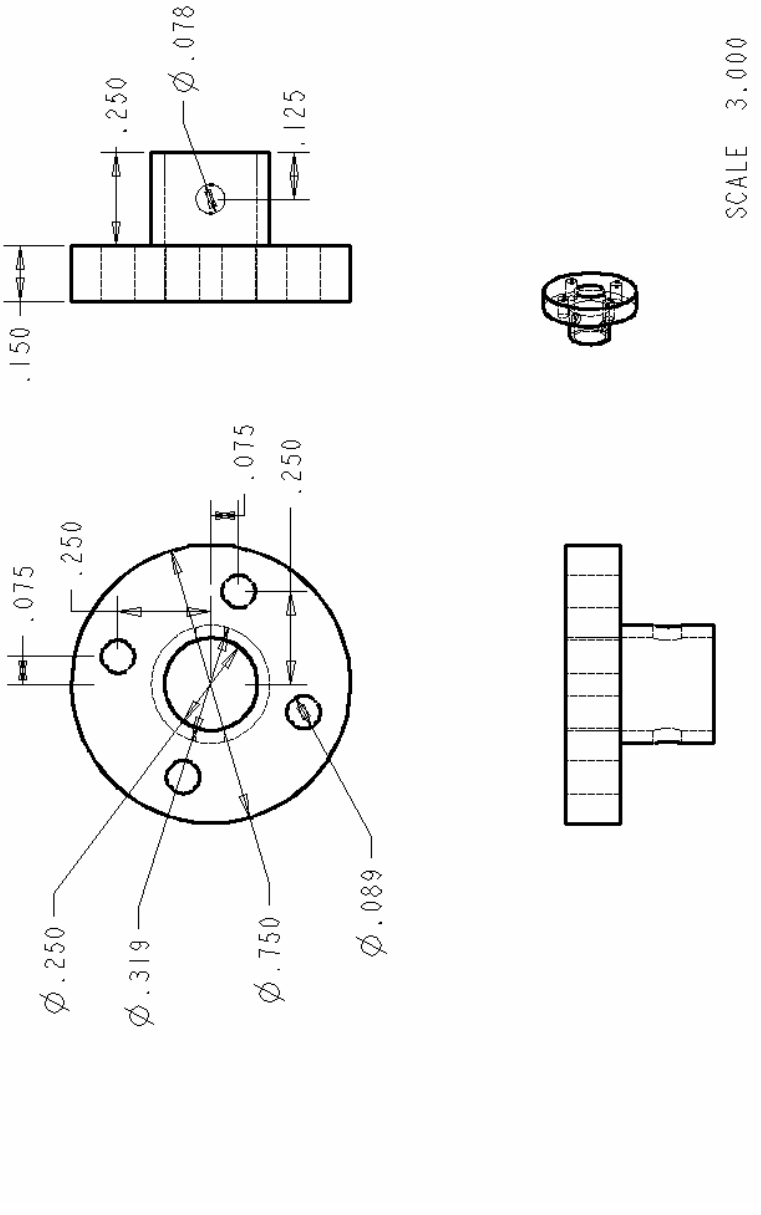


Figure F-25: drw_strutbasewiremount

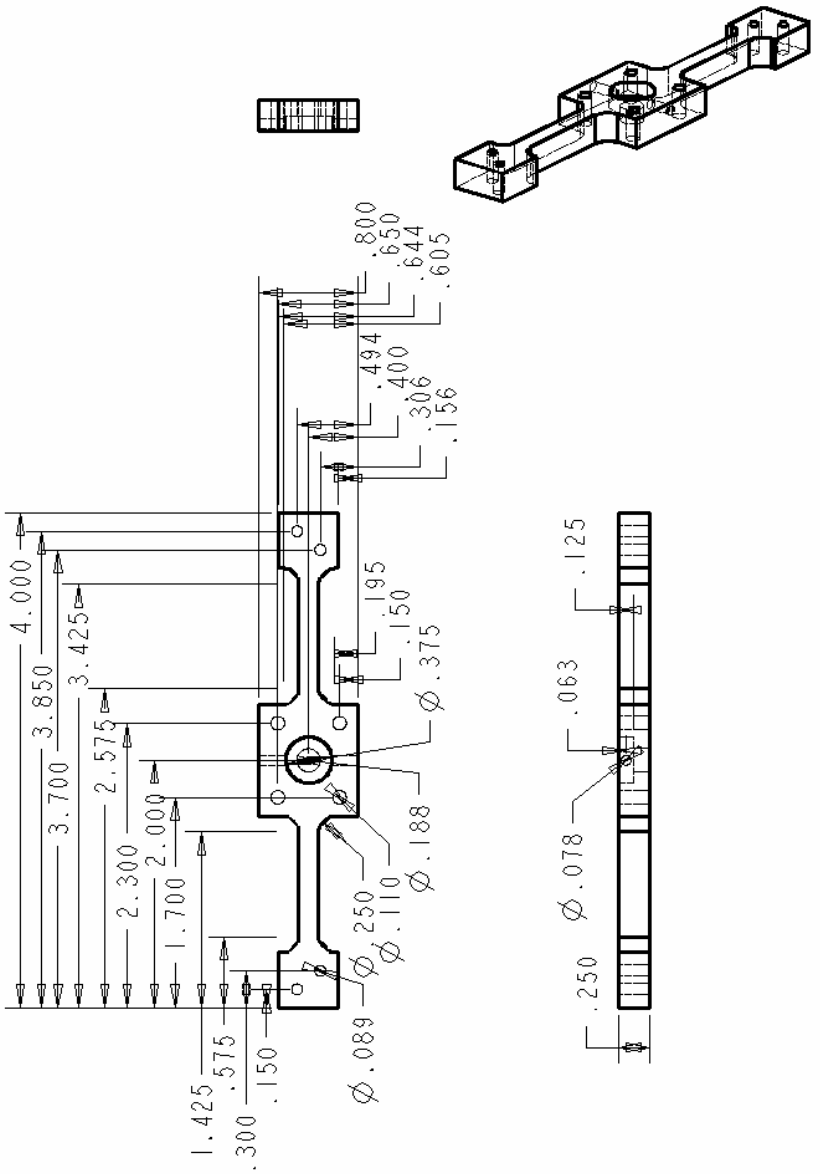


Figure F-26: drw_strutend

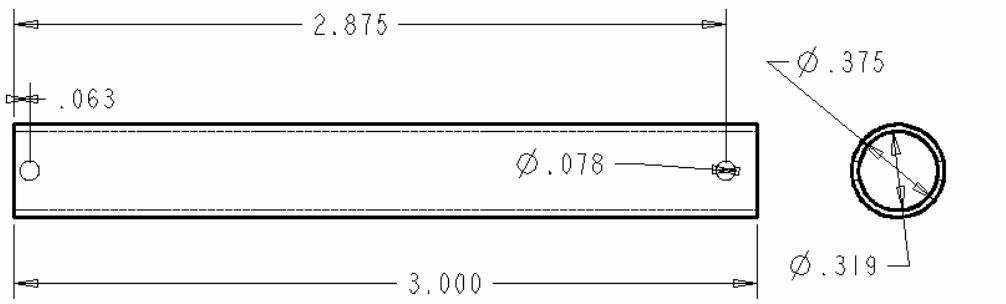


Figure F-27: drw_strutimuside

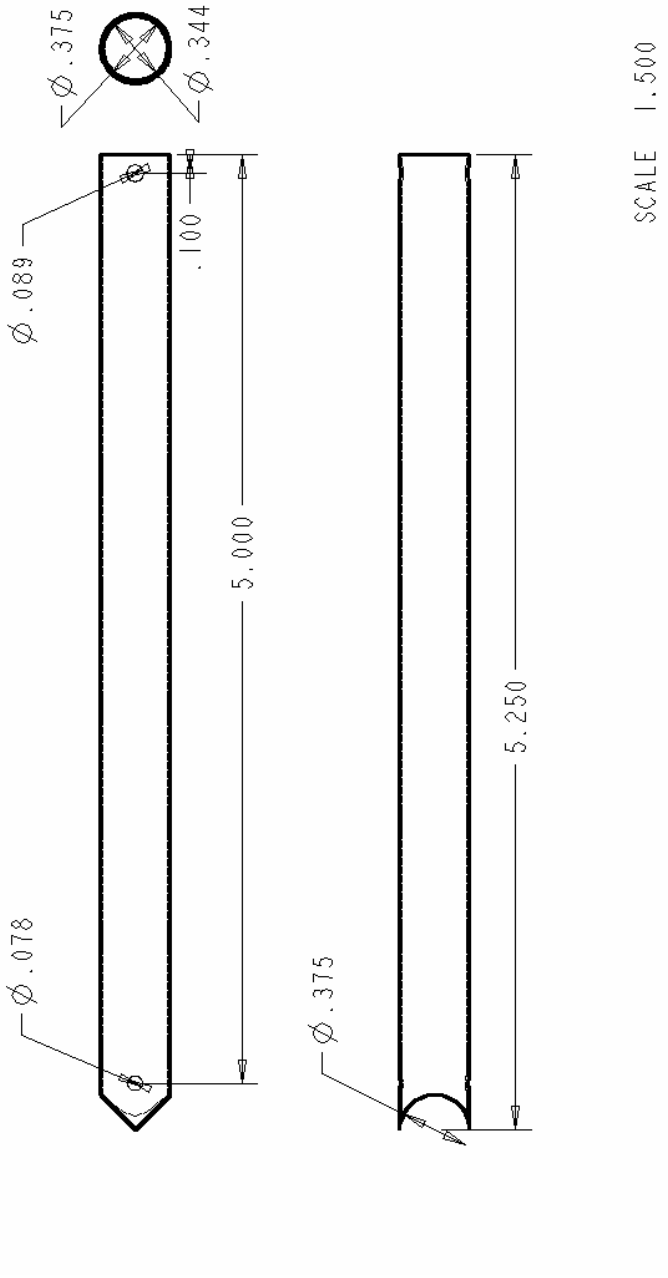


Figure F-28: drw_strutlanding

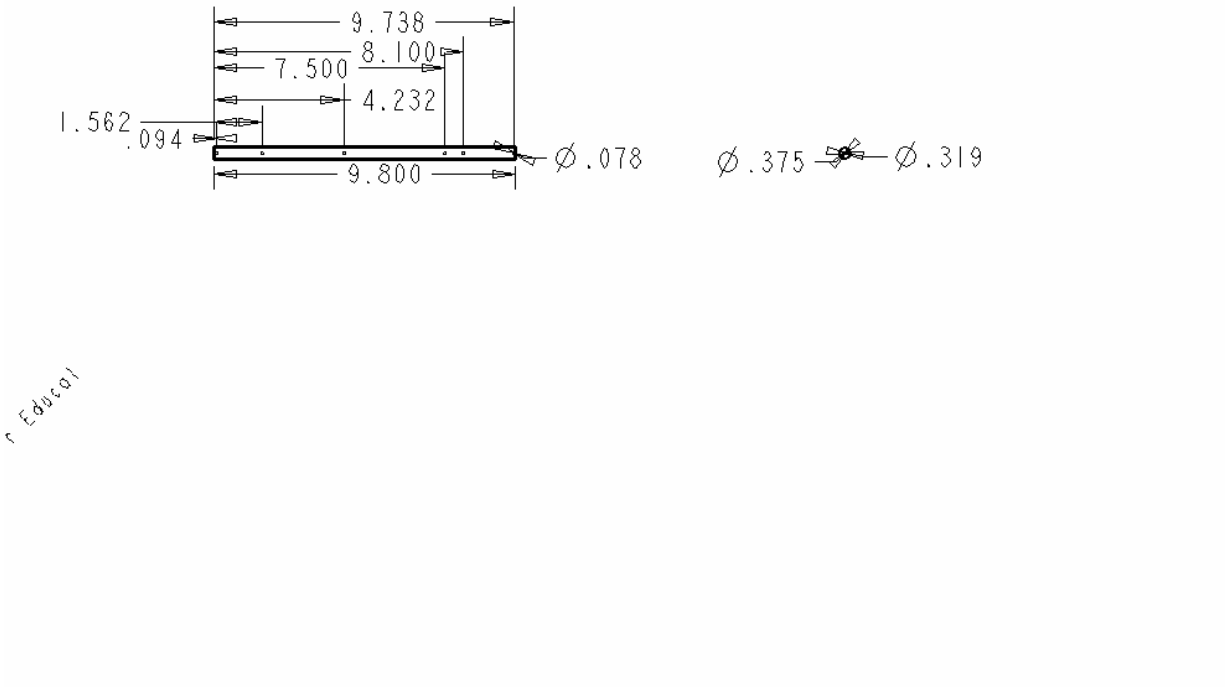


Figure F-29: drw_strutlongprop

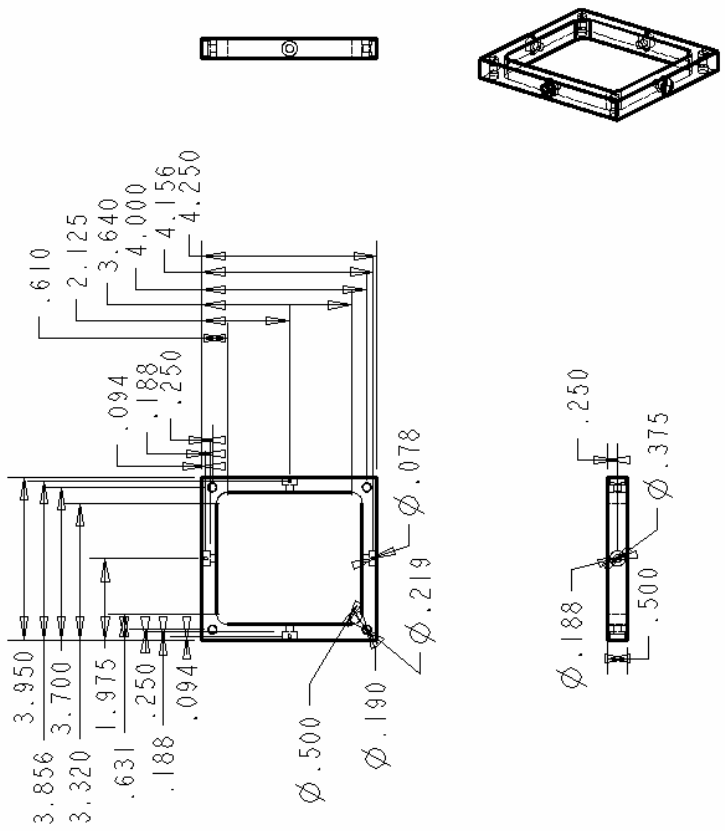
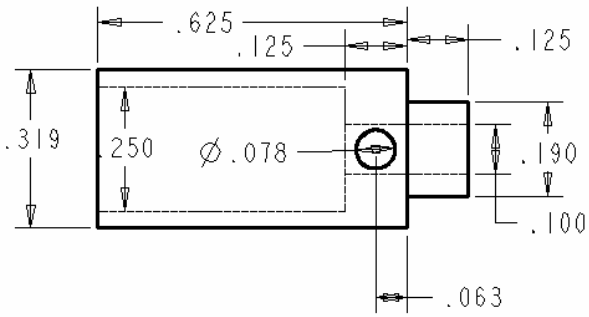
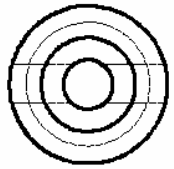


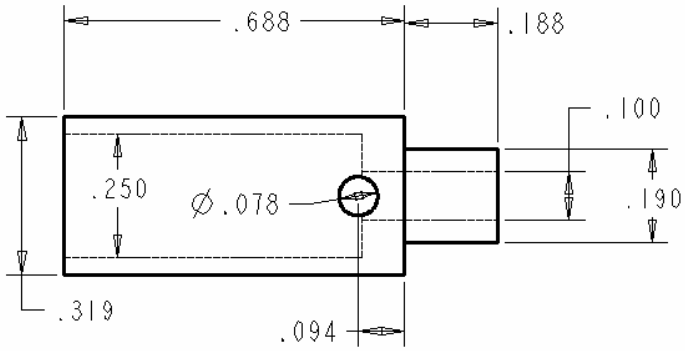
Figure F-30: drw_strutmount

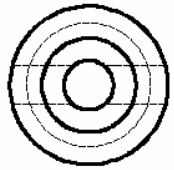




SCALE 4.000

Figure F-31: drw_strutplug





SCALE 4.000

Figure F-32: drw_strutpluglong

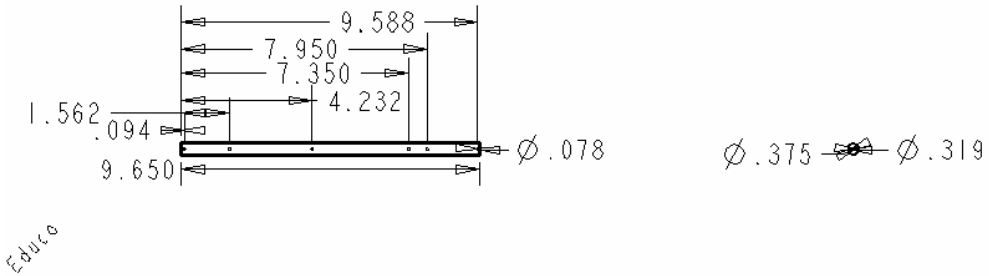


Figure F-33: drw_strutshortprop

REFERENCES

[1] S. Breheny, AFV Electronics Documentation, AFVMechECD:\Documentation\2003-2004\AFV Electronics Documentation.doc, 2003.

[2] S. Breheny and R. D'Andrea, Using Airborne Vehicle-Based Antenna Arrays to Improve Communications with UAV Clusters, *Conference on Decision and Control*, 4158-4162, 2003.

[3] S. Breheny, An Investigation of Vehicle-based Antenna Arrays for Unmanned Aerial Vehicle Communications, 2004.

[4] P. Castillo, A. Dzul and R. Lozano, Real-time Stabilization and Tracking of a Four Rotor Mini-Rotorcraft, http://www.hds.utc.fr/~castillo/publications/4rotor.pdf

[5] O. Fong and S. L. Huggins IV, Mechanical and Aerodynamic System Design, AFVMechECD:\Documentation\2001\ Mechanical and Aerodynamic System Design.doc, 2001.

[6] P. Pounds, R. Mahony, P. Hynes, and J. Roberts, Design of a Four-Rotor Aerial Robot, *Proc. 2002 Australasian Conference on Robotics and Automation*, http://www.araa.asn.au/acra/acra2002/Papers/Pounds-Mahony-Hynes-Roberts.pdf , 2002

[7] Association for Unmanned Vehicle Systems, International, Aerial Robotics Competition homepage, http://avdil.gtri.gatech.edu/AUVS/IARCLaunchPoint.html

[8] Draganfly Innovations Inc., Draganflyer IV homepage, http://www.rctoys.com/draganflyer4.php

[9] I. Kroo and F. Prinz, The Mesicopter: A Meso-Scale Flight Vehicle, Mesicopter homepage, http://aero.stanford.edu/mesicopter/

[10] Experimental Rocket Propulsion Society, Inc., The GizmoCopter Project homepage, http://gizmocopter.org/

[11] D. Fitzgerald, XUV-5 PipeDream Quad Project homepage, http://www.eese.bee.qut.edu.au/QUAV/index2.html , 2002

[12] J. Borenstein, The Hoverbot – An Electrically Powered Flying Robot, http://www-personal.engin.umich.edu/~johannb/hoverbot.htm

[13] S. Breheny, Design and Implementation of Control and Sensing Electronics for the Cornell Autonomous Flying Vehicle, AFVMechECD:\Documentation\2001\Design&ImpofControl&SensingforCUAFV.doc , 2001

[14] T. Cimato, MaxCim Motors Technical Specs, http://www.maxcim.com/technical.html

[15] Stock Drive Products/Sterling Instrument, Online Catalog, https://sdp-si.com/eStore/

[16] US Digital, Online Catalog, http://www.usdigital.com/products/e5s/

[17] Bishop Power Products, Online Catalog, http://www.b-p-p.com/Etec1200.htm