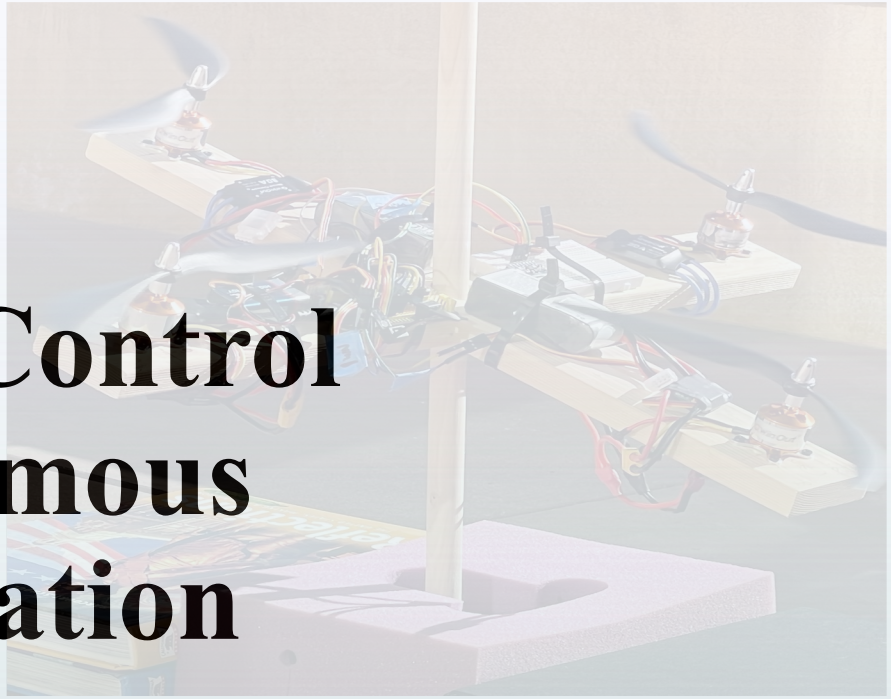
ETSD002



A Novel Dual Loop Control System for Autonomous Quadcopter Navigation

Anish Anand

Palos Verdes Peninsula High

Rancho Palos Verdes, CA, USA

No. of Cells:	2 - 3 Li-Poly 6 - 10 NiCd/NiMH
Kv:	1000 RPM/V
Max Efficiency:	80%
Max Efficiency Current:	4 - 10A (>75%)
No Load Current:	0.5A @10V
Resistance:	0.090 ohms
Max Current:	13A for 60S
Max Watts:	150W
Weight:	52.7 g / 1.86 oz
Size:	28 mm dia x 28 mm bell length

Introduction

Why Quadcopters?

- Unmanned aerial vehicles (UAVs) such as Quadcopters, are popular due to their lightweight simple structure, which makes them versatile and quick.
- Because of their versatility and speed, broad potential uses:
  - Transportation between hospitals (organs) or package delivery to homes
  - Search and rescue in remote locations
  - Wireless Communication Bay Stations (5G LTE)

The Problem: Autonomously controlled or "self-driving" UAVs are in early-stages of development/testing. Current research papers use Single Loop Control Systems, which have a low tolerance for errors in measurement, wind disturbances, and imbalances. Based on simulations, this leads to poor stability and a significant potential for crashing during autonomous navigation.

- Autonomous drones have complicated aerodynamics, due to six states of motion (positional x, y, z, and rotational f, q, y) controlled by only four motor inputs, making the system under-actuated. As a result, UAV stability when hovering or navigating is a challenge.

Solution: To tackle this problem of poor navigation stability, a Novel Dual Loop Control System for autonomous navigation was developed, based on Lyapunov's Theory of Stability.

The dual loop was implemented into four control algorithms and optimized via original MATLAB programs and parametric analysis. Further, two prototypes were built (single-axis and quad-axis) to implement the novel dual loop control system in real life and to validate simulation data.

Quadcopter Dynamics

- A Quadcopter moves by changing the **4 motor speeds** or angular velocities, w_N, in various configurations
- A Quadcopter **has 6 degrees of motion**
  - 3 translation {x,y,z}
  - 3 rotation {f,q,y}
- Controlling 6 degrees of freedom with 4 inputs is an **underactuated** system

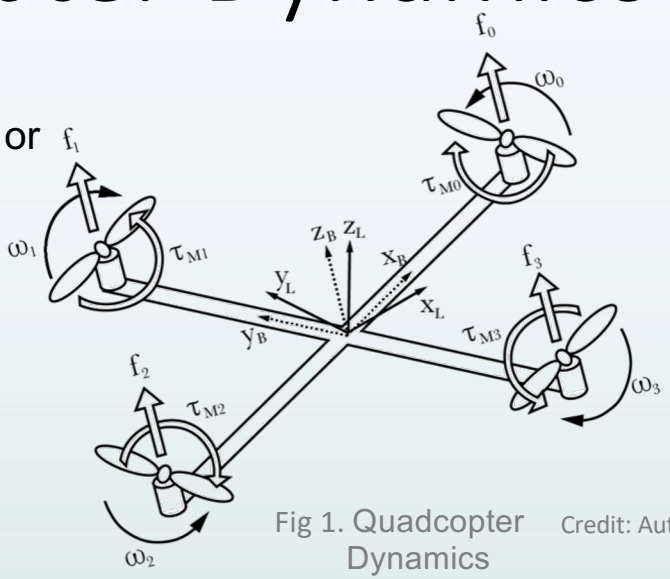
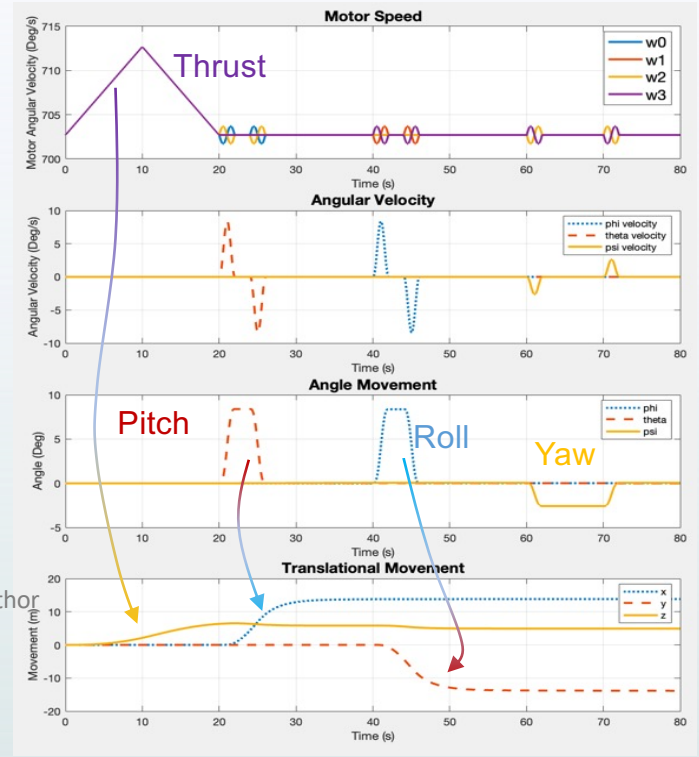


Fig 1. Quadcopter Dynamics

Credit: Author



Credit: Author

- Angular velocities of the motors determine the **thrust,** T_B, and **torque,** τ_B, of the quadcopter (quad arm length, l, lift constant. k, and drag constant, b)

$$T_B = \begin{bmatrix} 0 \\ 0 \\ T = \sum_{i=1}^{4} f_i = k\sum_{i=1}^{4} \omega_i^2 \end{bmatrix} \qquad \tau_B = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} lk(-\omega_2^2 + \omega_4^2) \\ lk(-\omega_1^2 + \omega_3^2) \\ \sum_{i=1}^{4} \tau_{Mi} = b\sum_{i=1}^{4} (-1)^i \omega_i^2 \end{bmatrix}$$

- The thrust, T_B, and torque, τ_B, of the quadcopter then determine the **acceleration**, a, and **angular velocity**, α_B, of the quadcopter (gravitiy, g, mass, m, and Rotaional Inertia, I)

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = -g\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \frac{T}{m}\begin{bmatrix} cos\psi sin\theta cos\phi + sin\psi sin\phi \\ sin\psi sin\theta cos\phi - cos\psi sin\phi \\ cos\theta cos\phi \end{bmatrix} \qquad \begin{bmatrix} \alpha_{B\phi} \\ \alpha_{B\theta} \\ \alpha_{B\psi} \end{bmatrix} = \begin{bmatrix} \tau_\phi / I_{xx} \\ \tau_\theta / I_{yy} \\ \tau_\psi / I_{zz} \end{bmatrix}$$

Current Feedback Control System

- A control system measures outputs, and then compares it to desired values, and adjusts using feedback information to minimize errors.

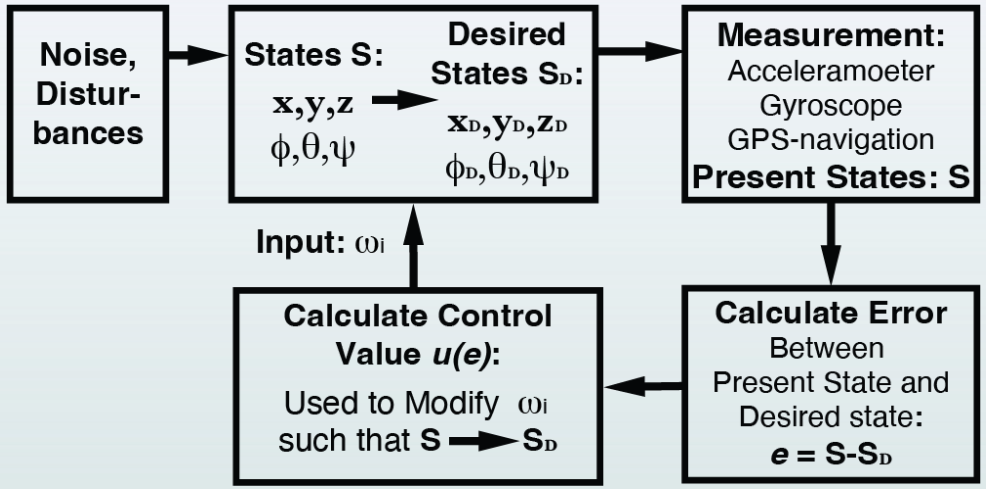
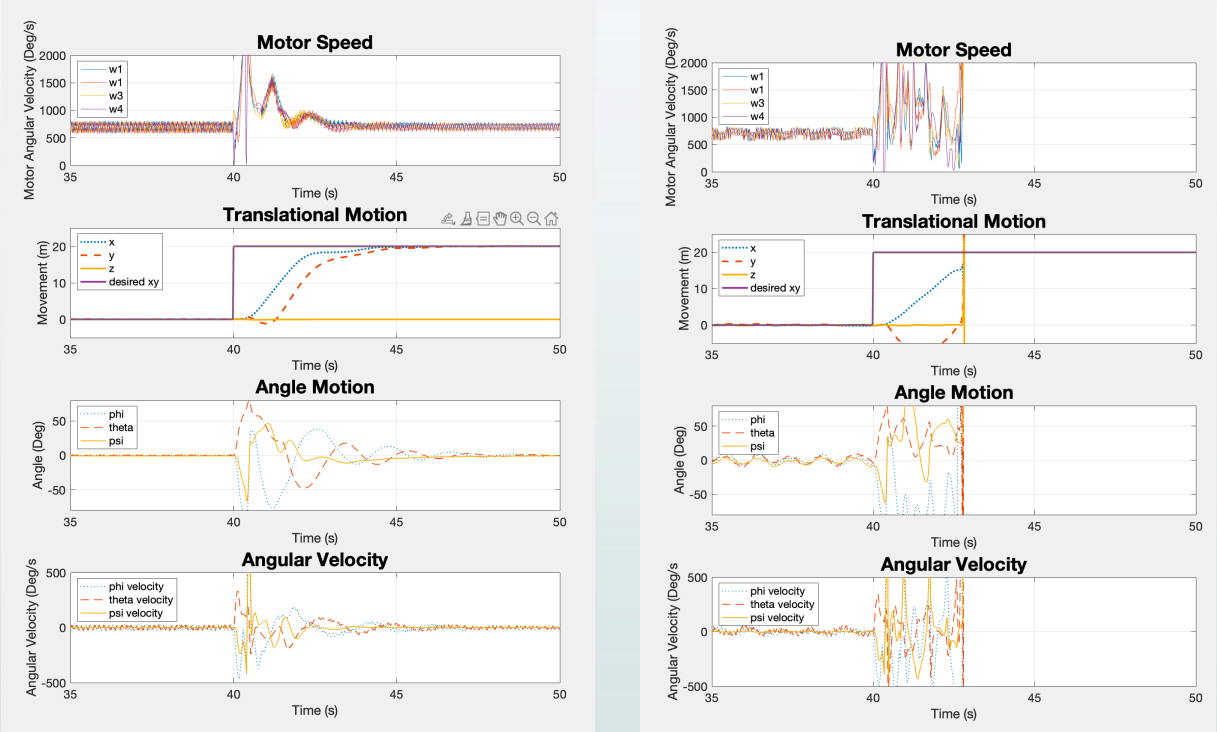


Fig 2. Feedback Control System for Autonomous Quadcopter Credit: Author



(a) (b)

Fig 3. Single Loop step Function (a) without and (b) with Noise, Disturbances, and Imbalance

- Quadcopters need a **feedback control system** to maintain **stability**, adjust flight trajectory, and achieve precise control during flight operations.
- A **single loop control system** works well to stabilize the quadcopter when **hovering** (staying in one location)
- During **navigation** of the quadcopter in MATLAB, the single loop control system is **difficult to stabilize** when challenged with errors in measurement, noise and disturbances, or large navigation step size (similar to stabilizing ball on parabolic hill)

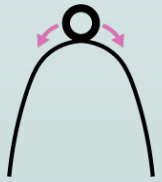
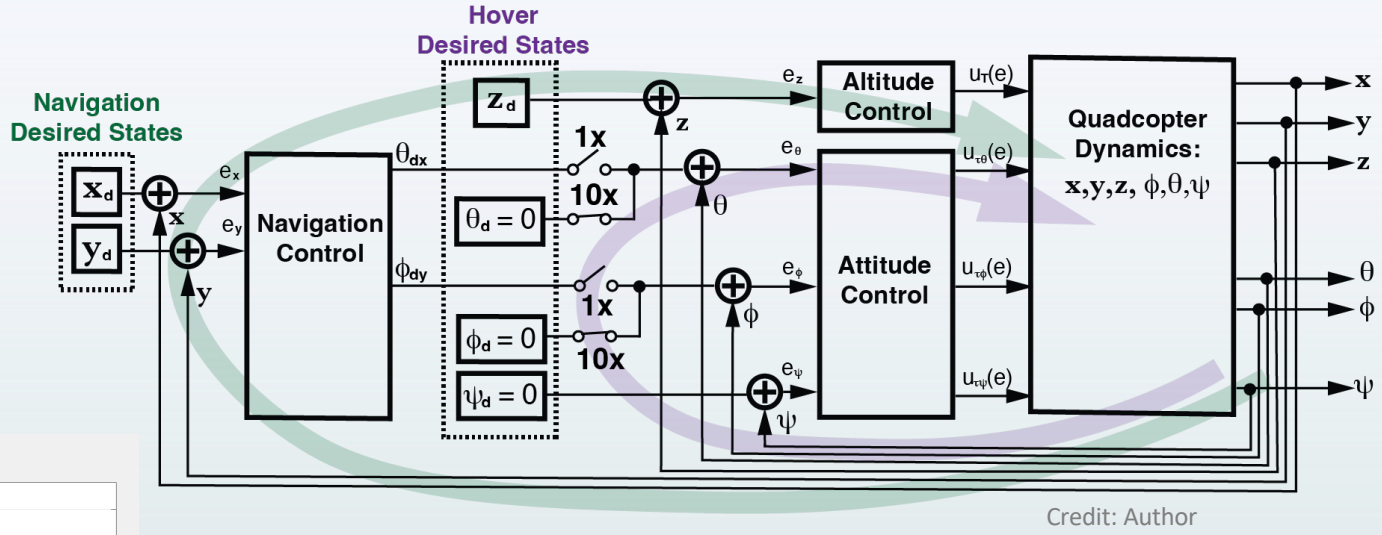


Fig 4. Parabolic hill with Ball on top representing Quadcopter stability
Credit: Author

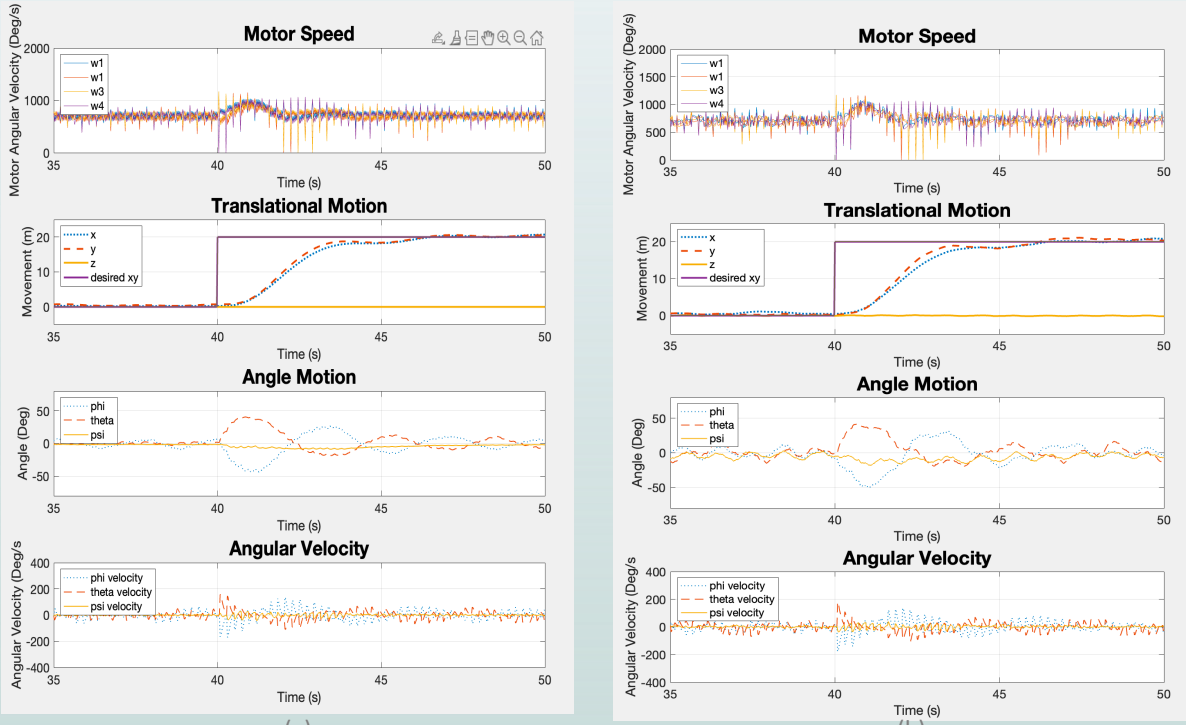
My Novel Dual Loop Control System

- **Two independent loops:** one loop is for navigation (green) and one loop (purple) is to return to hover state (creates stability for the angles and altitude in-between each navigation update). Hover loop runs 10 times (every 20 milliseconds) in between each navigation loop update (every 200 milliseconds)



Credit: Author

Fig 5 Novel Dual Loop Control System for Quadcopter

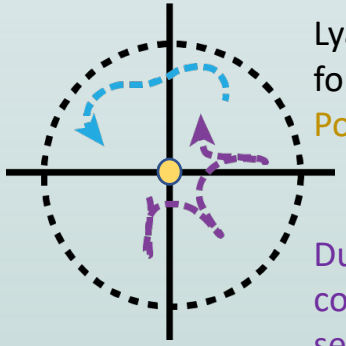


(a) (b)

Fig 6. Dual Loop Step Function (a) without and (b) with Noise, Disturbances, and Imbalance

- Novel control system with 2 independent loops is based on Lyapunov's Theory of Stability. The goal is to keep the quadcopter closer to Point of Equilibrium so it can be more stable when challenged with errors/noise

Single loop control states need to move slowly and precisely to stay within the region of stability



Lyapunov's region of stability for each control state and the Point of Equilibrium (o)

Dual Loop has one loop to determine control states for navigation and a second loop to force states closer to equilibrium point (hover state)

Fig 7. Diagram of control state stability

Credit: Author

Methods for Simulations

- Created Original Simulation environment to determine Quadcopter dynamics in MATLAB
- Wrote Original MATLAB programs to model 4 Control Algorithms using novel dual loop control system
  - Two Basic Algorithms: Proportional-Derivative (PD), Proportional-Integral-Derivative (PID)
  - Two Advanced Algorithms: Back-stepping Control (BSC), Sliding Method Control (SMC)

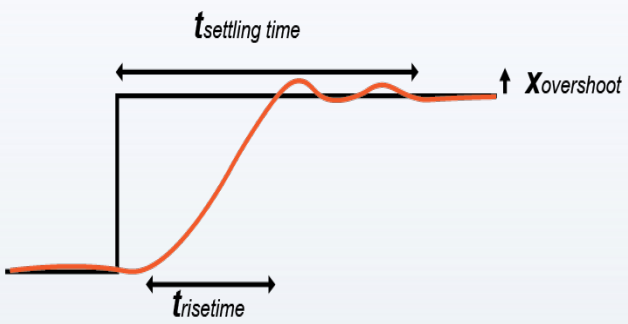


Fig 8. Step Function (black) and Step Response (orange) used to determine control system parameter optimization

Credit: Author

- Parametric analysis using a step function and response was simulated to generate data in CSV file for several variables
- Cost function based on settling time, robustness towards noise, required angular velocity range was used to determine optimum parameters and optimum algorithm

```
function [w1,w2,w3,w4] = xysmc1(m,l,k,b,Ixx,Iyy,Izz,Kx,Kx1,Kx2,Ky,Ky1,Ky2, Kz,Kz1,Kz2, Kphi,Kphi1, ...
    Kphi2, Ktheta,Ktheta1,Ktheta2,Kpsi,Kpsi1,Kpsi2,max_w,xvd,xd,xv,x,yvd,yd,yv,y,zvd,zd,zv,z,...
    phivd,phiv,phi,thetavd,thetav,theta,psivd,psid,psiv,psi)

g = 9.81;

% phid = 0;
% thetad =0;

sz = -(zvd - zv)-Kz*(zd - z);
sx = -(xvd - xv)-Kx*(xd - x);
sy = -(yvd - yv)-Ky*(yd - y);

T_PD = (g +(-Kz*Kz*(zd - z)-Kz1*sign(sz)-Kz2*sz))*m/cos(phi)/cos(theta);

thetad = m/T_PD *(-Kx*Kx*(xd - x)-Kx1*erf(real(sx))-Kx2*sx);
phid = - m/T_PD *(-Ky*Ky*(yd - y)-Ky1*erf(real(sy))-Ky2*sy);


sphi = -(phivd - phiv)-Kphi*(phid - phi);
stheta = -(thetavd - thetav)-Ktheta*(thetad - theta);
spsi = -(psivd - psiv)-Kpsi*(psid - psi);


torq_phi_PD = (-Kphi*Kphi*(phid - phi)-Kphi1*sign(sphi)-Kphi2*sphi)*Ixx;
torq_theta_PD = (-Ktheta*Ktheta*(thetad - theta)-Ktheta1*sign(stheta)-Ktheta2*stheta)*Iyy;
torq_psi_PD = (-Kpsi*Kpsi*(psid - psi)-Kpsi1*sign(spsi)-Kpsi2*spsi)*Izz;

w1 = sqrt(T_PD/4/k-torq_theta_PD/2/k/l+torq_psi_PD/4/b);
w2 = sqrt(T_PD/4/k-torq_phi_PD/2/k/l-torq_psi_PD/4/b);
w3 = sqrt(T_PD/4/k+torq_theta_PD/2/k/l+torq_psi_PD/4/b);
w4 = sqrt(T_PD/4/k+torq_phi_PD/2/k/l-torq_psi_PD/4/b);
```

Fig 9. MATLAB code for sliding methods control (Full code is linked in references)

startlc	Type	KphiD	KphiP	KphiI	Kthetа	Kthetа	Kthetа	OS_xy	ts_xy	xy_noi	z_nois	phi_da	phi_da	psi_da	phi_v_	phi_v_	psi_v_	w_noise_data	cost functior
44	**PD**	**4**	**8**	**4**	**4**	**8**	**4**	**1.54**	**9.23**	**0.33**	**0.08**	**53.11**	**14.18**	**1.85**	**200.5**	**35.7**	**10.94**	**26.4**	**6.491**
47	PD	2	8	2	2	8	2	1.79	8.685	0.295	0.07	53.75	14.14	1.89	203.3	35.87	11.13	28.48	6.556
48	PD	2	8	3	2	8	3	1.61	8.995	0.24	0.07	53.69	14.08	1.88	203.2	35.86	11.1	30.29	6.559
71	PD	2	6	1	2	6	1	0.25	12.66	0.34	0.07	43.73	11.59	1.33	160.4	31.57	7.88	42.22	6.625
51	PD	10	6	1	10	6	1	0.44	10.94	0.345	0.07	42.11	12.28	1.25	153.2	31.45	7.73	46.25	6.635
32	PD	8	8	2	8	8	2	2.07	8.58	0.3	0.07	52.22	14.71	1.81	195.6	35.57	10.79	29.65	6.654
62	PD	6	6	2	6	6	2	0.26	12.55	0.24	0.08	42.86	11.84	1.29	156.7	31.43	7.76	46.49	6.679
30	PD	10	8	5	10	8	5	1.68	8.955	0.345	0.07	51.55	14.72	1.77	192.7	35.46	10.6	31.28	6.692
33	PD	8	8	3	8	8	3	1.89	8.7	0.365	0.07	52.16	14.63	1.81	195.5	35.54	10.76	30.72	6.725
40	PD	6	8	5	6	8	5	1.5	9.405	0.35	0.09	52.54	14.29	1.82	197.9	35.57	10.79	32.02	6.735
60	PD	8	6	5	8	6	5	0.11	14.07	0.32	0.08	42.31	11.82	1.27	154.7	31.29	7.64	43.22	6.764
34	PD	8	8	4	8	8	4	1.73	8.855	0.37	0.08	52.1	14.56	1.8	195.4	35.52	10.72	32.85	6.784
66	PD	4	6	1	4	6	1	0.29	12.26	0.29	0.09	43.31	11.74	1.31	158.6	31.51	7.83	48.86	6.801
56	PD	8	6	1	8	6	1	0.39	11.46	0.255	0.07	42.51	12.09	1.27	155	31.45	7.76	51.94	6.82
29	PD	10	8	4	10	8	4	1.84	8.79	0.28	0.1	51.61	14.79	1.78	192.8	35.49	10.63	35.97	6.844
42	PD	4	8	2	4	8	2	1.87	8.665	0.395	0.08	53.23	14.31	1.86	200.7	35.73	11	33.21	6.855
63	PD	6	6	3	6	6	3	0.19	13.2	0.375	0.09	42.81	11.78	1.29	156.7	31.39	7.73	46.31	6.864
73	PD	2	6	3	2	6	3	0.12	13.9	0.325	0.08	43.63	11.48	1.33	160.3	31.51	7.83	46.06	6.875
72	PD	2	6	2	2	6	2	0.18	13.29	0.215	0.09	43.68	11.54	1.33	160.4	31.54	7.86	50.71	6.897
61	PD	6	6	1	6	6	1	0.34	11.88	0.3	0.08	42.91	11.91	1.29	156.8	31.47	7.79	52.1	6.907
52	PD	10	6	2	10	6	2	0.37	11.78	0.315	0.07	42.06	12.21	1.25	153.1	31.41	7.7	51.82	6.908
35	PD	8	8	5	8	8	5	1.58	8.965	0.425	0.08	52.04	14.49	1.8	195.3	35.5	10.69	35.38	6.911
50	PD	2	8	5	2	8	5	1.38	9.415	0.265	0.08	53.57	13.96	1.88	203	35.83	11.03	39.24	6.926

Fig 10. Part of parametric analysis used to determine the optimum parameters for each control algorithm (Full set of data is linked in references)

Results from Simulations (1/2)

Proportional-Derivative (PD)	Proportional-Integral-Derivative (PID)	Backstepping Control (BSC)	Sliding Method Control (SMC)
$e(t) = x_d(t) - x(t)$ $u(t) = K_P e(t) + K_D \frac{de(t)}{dt}$	$e(t) = x_d(t) - x(t)$ $u(t) = K_P e(t) + K_I \int e(t)dt + K_D \frac{de(t)}{dt}$	$\mathrm{e_s}(t) = (\dot{x}_d(t) - \dot{x}(t)) + K_1\big(x_d(t) - x(t)\big)$ $u(t) = K_1 \times \big(\mathrm{e_s}(t) + K_I e(t) - K_2 \mathrm{e_s}(t-1)\big)$	$\mathrm{e_s}(t) = \dot{x_d}(t) - \dot{x}(t) + K\big(x_d(t) - x(t)\big)$ $u(t) = K^2 e(t) + K_1 \mathrm{sgn}\big(\mathrm{e_s}(t)\big) + K_2 \mathrm{e_s}(t)$
A PD controller relies on the error of the present state, e(t), the proportional term, and the derivative of this error to create a control input, u(t) to the quadcopter to get the quadcopter to the desired state.	A PID controller is like a PD control but includes an integral term that can remove static errors like motor gain difference. Most common controller.	A BSC controller assumes a local stable point by changing the error term to be stable. Error term includes an error for the state and its derivative. Back-stepping over corrects and then removes overcorrection in the next step.	Similar to BSC, the SMC controller assumes a local stable point by changing the error term to be always stable. The sliding method uses a signum (or sign function) to always make the correction in the opposite direction of the error.
PROs: • Simple Control System to Implement • No integration is required only proportional and derivative in algorithm • Fast settling for angles **CONs:** • Cannot handle Steady State errors such as imbalance in motors gain or weight distribution • Cannot tolerate high levels of noise or disturbances	**PROs:** • Simple Control System to implement but requires integration • Can handle Steady State errors such as imbalance in motors gain or weight distribution **CONs:** • Cannot tolerate high levels of noise or disturbances • Slow settling for angles and angular velocity	**PROs:** • Very fast settling for angles to get back to hover state • Can tolerate high levels of noise or disturbances **CONs:** • Angular Velocity and Motor Speeds requirements are too high for changes in x,y direction in the real-world. • Cannot handle Steady State errors	**PROs:** • Very fast settling for angles to get back to hover state • Tolerates high levels of noise or disturbances • Maximum angle change and angular velocity is low during change in position • Tolerates steady state error (imbalance) well **CONs:** • Most complex algorithm to implement

Results from Simulations (2/2)

- Data for four Control algorithms were used to determine the settling time, standard deviation of the noise and maximum angular and angular velocity range for a 20-meter jump in both x- and y-direction

Proportional-Derivative Control Algorithm (PD)

	x,y	z	ϕ,θ	ϕ,θ vel	ψ
Settling Time	12s	3.5s	3.5s	3s	4s
Noise (St Dev)	1.5m	0.1m	4.4°	2.3°/s	4.4°
Maximum Range			35°	80°/s	15°

Proportional-Integral-Derivative (PID)

	x,y	z	ϕ,θ	ϕ,θ vel	ψ
Settling Time	11.5s	3.5s	12.5s	10s	10.5s
Noise (St Dev)	1.5m	0.1m	4.3°	2.1°/s	5.1°
Maximum Range			35°	80°/s	15°

Backstepping Control (BSC)

	x,y	z	ϕ,θ	ϕ,θ vel	ψ
Settling Time	8.5s	3s	1.3s	0.9s	1.8s
Noise (St Dev)	0.2m	0.1m	3.9°	5°/s	4.0°
Maximum Range			65°	850°/s	18°

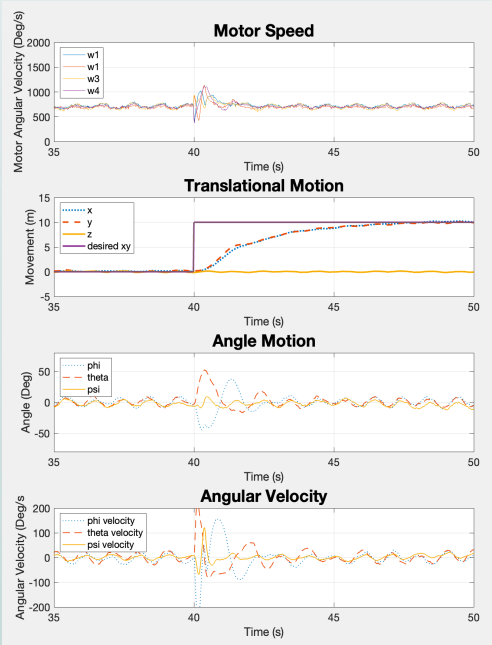
Sliding Method Control (SMC)

	x,y	z	ϕ,θ	ϕ,θ vel	ψ
Settling Time	8s	3s	1.7s	1.2s	2.3s
Noise (St Dev)	0.3m	0.1m	3.9°	7.5°/s	4.1°
Maximum Range			30°	80°/s	10°

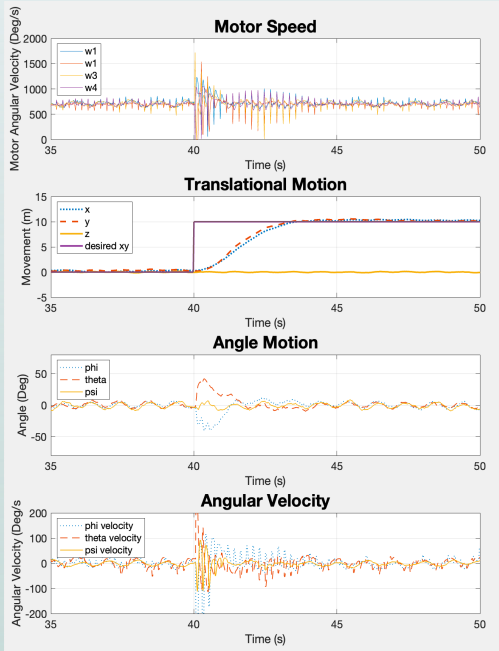
- Transient Data for Advanced Control algorithms shows improved translational speed for Dual Loop Algorithms when keeping Angular Range similar. Dual loop performs 2X faster than Single Loop control system.

Backstepping Control (BSC)

Single Loop

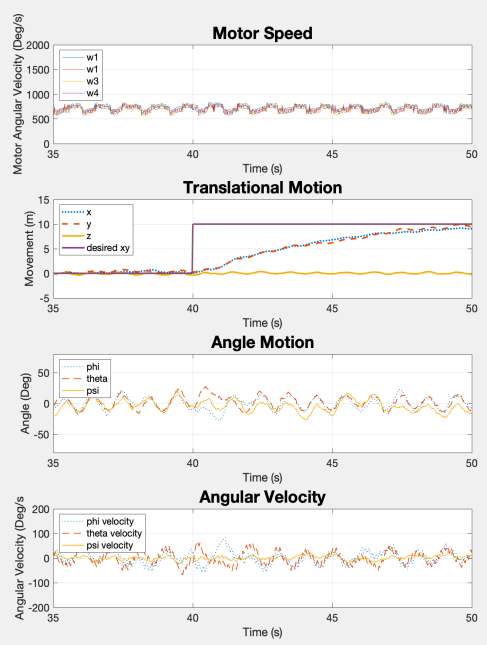


Dual Loop

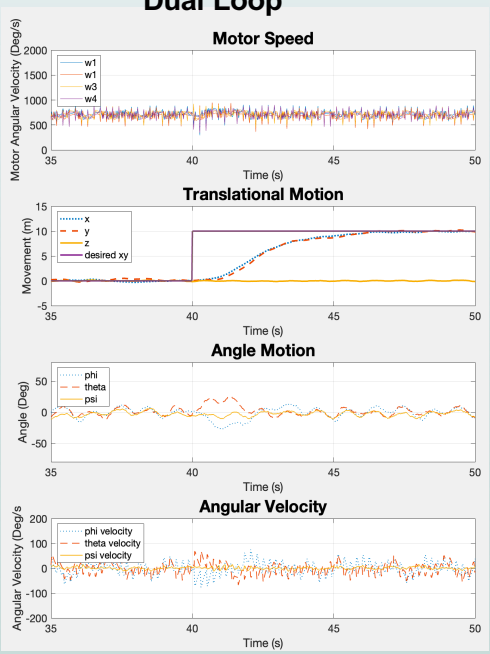


Sliding Method Control (SMC)

Single Loop



Dual Loop



Methods for Prototypes

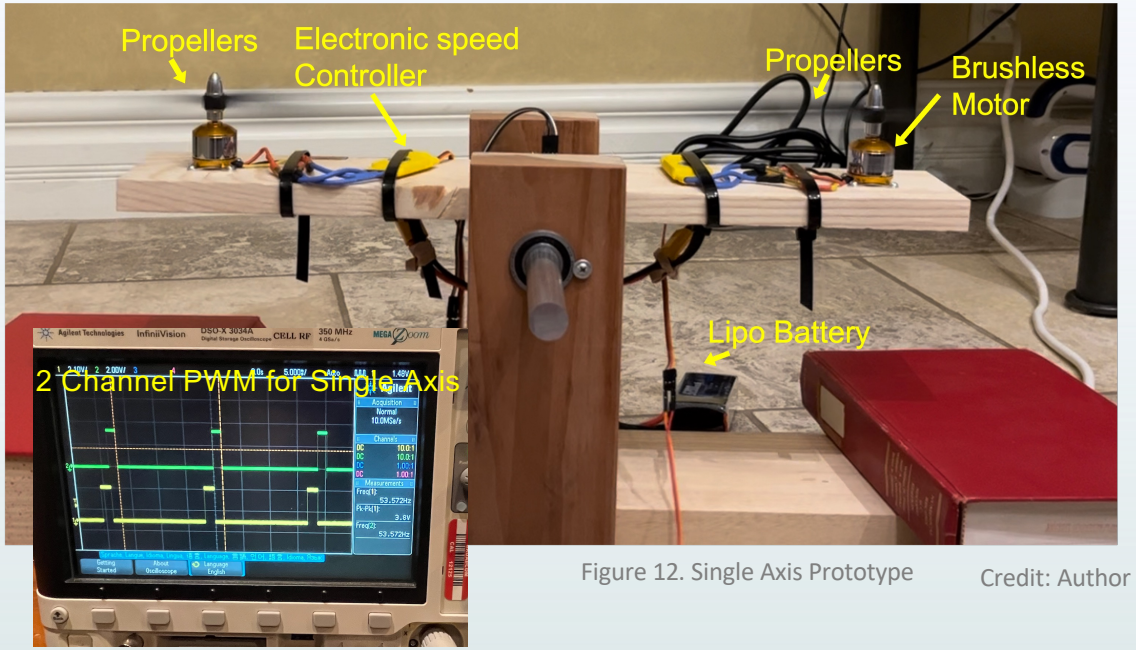


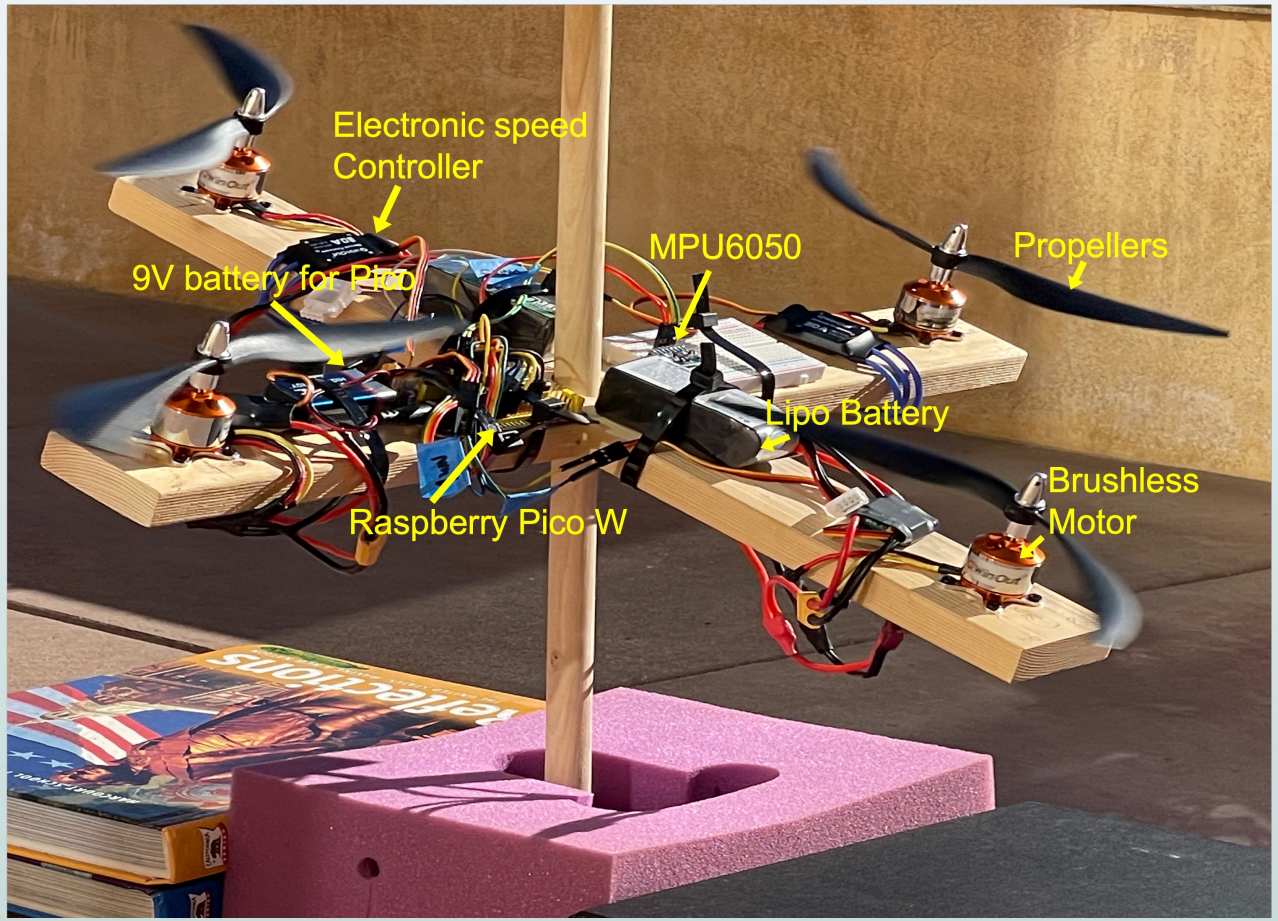
Figure 12. Single Axis Prototype

Credit: Author

Fig 11. Oscilloscope with PWM control signals required for the DC motors

- **Single Axis prototype** was built to study quadcopter control system dynamics in the θ state
- **Original Python code** to control the Gyroscope and Brushless motors through a Raspberry Pi
- Oscilloscope was used to validate the PWM signals are correct motor gains

Credit: Author

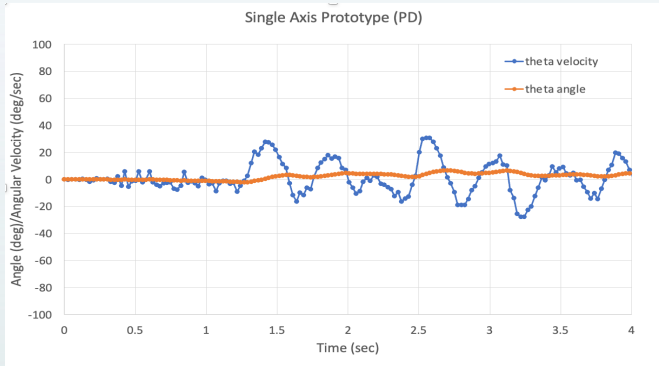


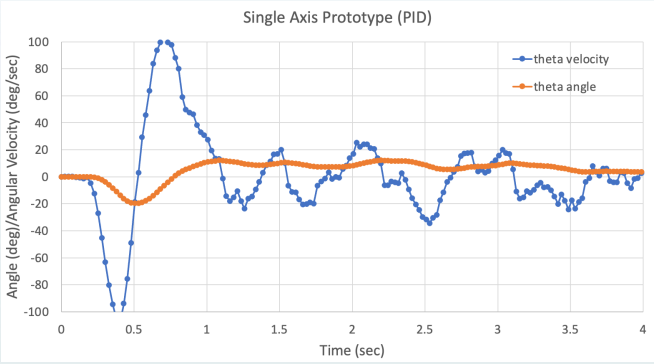
- **Quad Axis Prototype** was built to further study control theory with more states (z, ϕ, θ, ψ) in real world conditions
- All electronic components, microcontroller and batteries were secured to stand alone quad structure
- **Original MicroPython programs** were written to control the Quad axis prototype with a Pico microcontroller.
- HTML was used to create a webpage for autonomous control of the prototype.

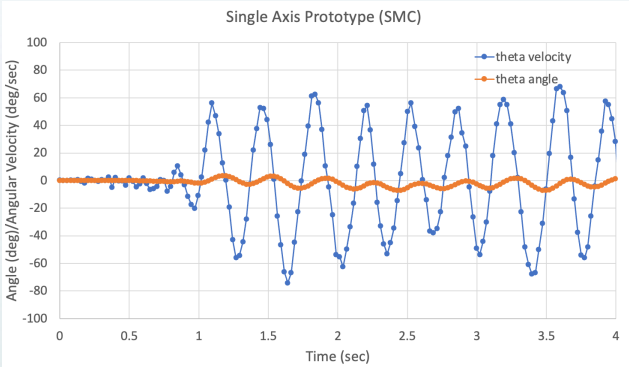
Figure 13. Quad Axis Prototype

Results from Prototypes

- **Single Axis Prototype Data:** Angular Velocity and Angular Data as a function of time generated for various parameters of the control Algorithm (saved in CSV data logging files on the Raspberry Pi Microprocessor)





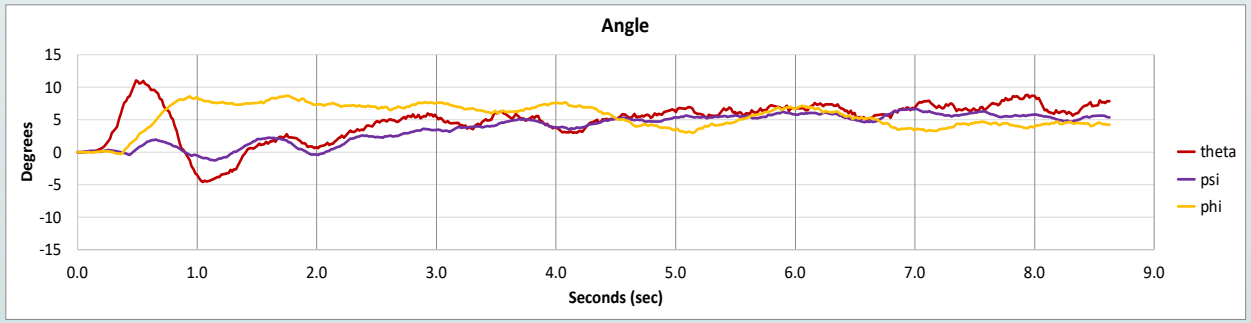


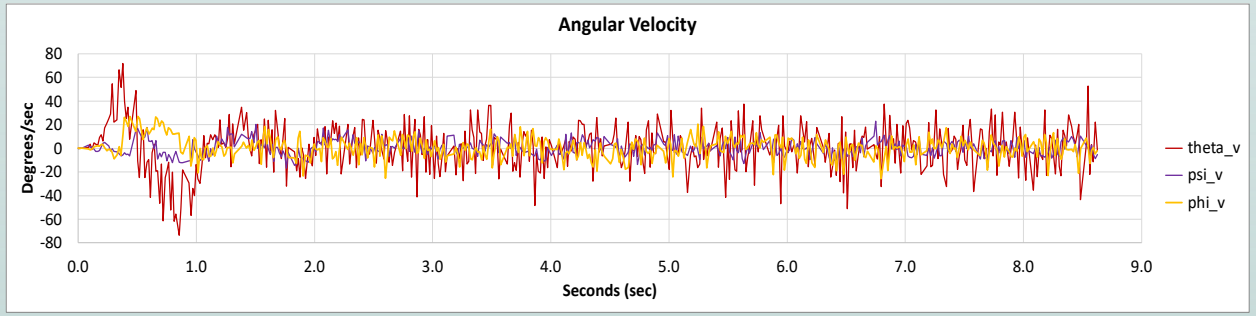
- Characteristics of the various algorithms are seen in the angular and angular velocity data thus **validating Simulation Data**

- **Quad Axis Prototype Data:** Angular Velocity and Angular Data as a function of time (saved in CSV data logging files on the Pico Micro-controller)

- The Angular Velocity sits very close to zero, while the angles has a slight offset due to PD algorithm unable to handle motor gain imbalance, **validating Simulation Data**

- These prototype validate the simulation environment, the quadcopter dynamics, and the basic control algorithms in its hover state.





Conclusions

- Single Loop Feedback Control Systems for Quadcopters in current literature are unstable when challenged with noise or large navigation steps (x-y direction) because the Control system is under-actuated: six degrees of freedom {x, y, z, f, q, y} controlled by only 4 inputs.
- A Novel Dual Loop Control System with 2 independent loops, hover loop (runs every cycle) and navigation loop (every 10th cycle), creates robustness to noise, disturbances and imbalances since each loop of the algorithm controls 4 states or less with 4 inputs.
- Novel Dual Loop Control System is **2-3x more Stable** (max measurement error tolerance and wind disturbance, max imbalance, max step size) and **2x Faster** than current Single Loop Control Systems (described in literature) for autonomous navigation.

Stability Variables (MATLAB Sim Data)	Single Loop	Dual Loop	Units
Maximum Step Size	15	45	Meters
Error Tolerance and Disturbance (x, y, z)	0.05	0.13	Standard deviation of error in meters
Error Tolerance and Disturbance (ϕ, θ, ψ)	0.03	0.05	Standard deviation of error in radians
Max Imbalance Tolerance	6%	12%	Imbalance in motor speed gain

- The Single Axis Prototype generated real world data for one degree of motion (q), which validated the MATLAB simulations. The Quad Axis Prototype tested four degrees of motion {z, f, q, y}, further validating the MATLAB simulations. These prototype validate the simulation environment, the quadcopter dynamics, and the basic control algorithms in its hover state.
- Sliding Method Control (SMC) Algorithm performs best in x-y position movements due to its fast-settling times as well as reasonable levels of angular velocity and motor speeds

$$Cost\ Function\ = \sum_{States} \frac{Settling\ time}{5} + \sum_{States} Noise + \sum_{Angles} \frac{Max\ Angle}{50}$$

	Cost Function
PD	12.07
PID	16.55
BSC	25.18
SMC	9.44

Supplemental Information

References

1. H. C. T. E. **Fernando**, A. T. A. De Silva, M. D. C. De Zoysa, K. A. D. C. Dilshan and S. R. Munasinghe, "Modelling, simulation and implementation of a quadrotor UAV," 2013 IEEE 8th International Conference on Industrial and Information Systems, Peradeniya, Sri Lanka, 2013, pp. 207-212.
2. S. A. **Suhail**, M. A. Bazaz and S. Hussain, "Altitude and Attitude Control of a Quadcopter Using Linear Active Disturbance Rejection Control," 2019 International Conference on Computing, Power and Communication Technologies (GUCON), New Delhi, India, 2019, pp. 281-286.
3. I. C. **Dikmen**, A. Arisoy and H. Temeltas, "Attitude control of a quadrotor," 2009 4th International Conference on Recent Advances in Space Technologies, Istanbul, Turkey, 2009, pp. 722-727.
4. T. **Luukkonen**, "Modelling and Control of Quadcopter", Research Project in Applied Mathematics, Aalto University School of Science, August 2011.
5. T. S. **Alderete**, "Simulator Aero Model Implementation", NASA Ames Research Center, Moffett Field, CAlifornia
6. B. **Zhang**, G. Yoon and H. Lim, "Attitude Control and Altitude Control of a Four-Rotor Flying Robot," 2018 14th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), Huangshan, China, 2018, pp. 1323-1328.
7. N. **Xuan-Mung** *et al*., "Quadcopter Precision Landing on Moving Targets via Disturbance Observer-Based Controller and Autonomous Landing Planner," in *IEEE Access*, vol. 10, pp. 83580-83590, 2022.
8. J. J. **Castillo-Zamora**, K. A. Camarillo-GóMez, G. I. PéRez-Soto and J. RodríGuez-ReséNdiz, "Comparison of PD, PID and Sliding-Mode Position Controllers for V–Tail Quadcopter Stability," in IEEE Access, vol. 6, pp. 38086-38096, 2018.
9. V. K. **Tripathi**, L. Behera and N. Verma, "Design of sliding mode and backstepping controllers for a quadcopter," 2015 39th National Systems Conference (NSC), Greater Noida, India, 2015, pp. 1-6
10. P. **Castillo**, R. Lozano and A. Dzul, "Stabilization of a mini-rotorcraft having four rotors," 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems
11. S. **Bouabdallah** and R. Siegwart, "Backstepping and Sliding-mode Techniques Applied to an Indoor Micro Quadrotor," Proceedings of the 2005 IEEE International Conference on Robotics and Automation, Barcelona, Spain, 2005, pp. 2247-2252.