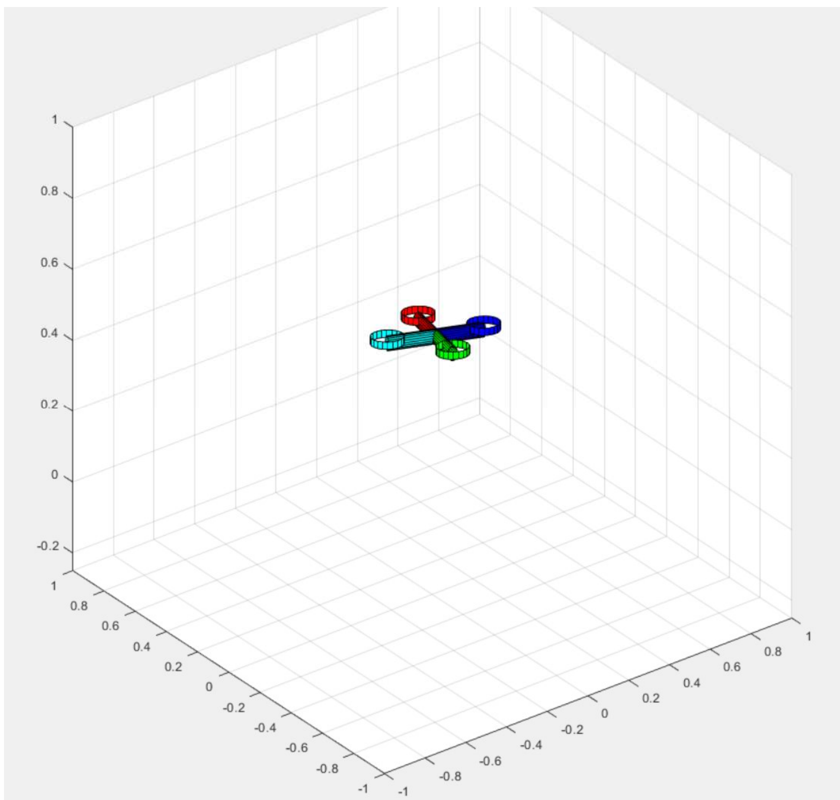
Drone Trajectory Planning

Jared Ticotin



Part A: Simulate the dynamics of the drone

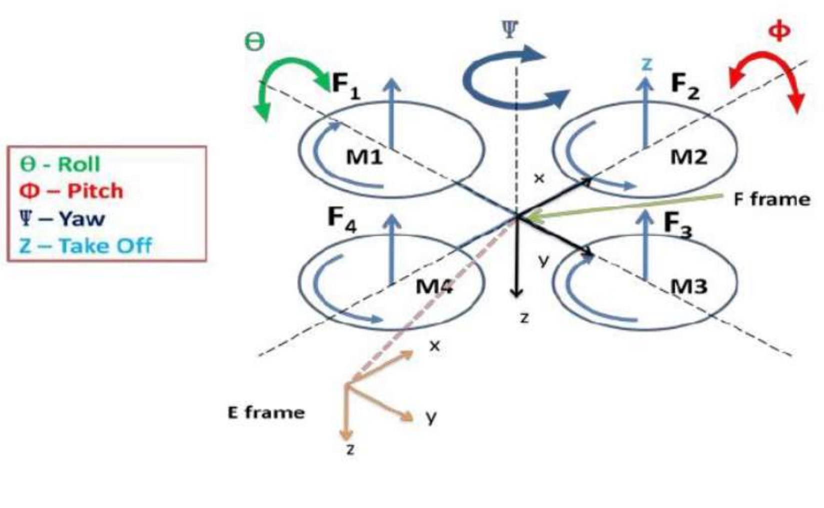


Figure 1. Quadcopter Control Frames

-see references [1]
-notice how F1,F3 spin CW (looking down) and F2,F4 spin CCW. This gives control to rotate about z
-note that z is pointing up for the drone reference.

$$R_v^b(\Phi,\Theta,\Psi) = R_X(\Phi)R_Y(\Theta)R_Z(\Psi)$$

$$R_v^b(\Phi,\Theta,\Psi) = \begin{bmatrix}1 & 0 & 0\\ 0 & \cos\Phi & \sin\Phi \\ 0 & -\sin\Phi & \cos\Phi\end{bmatrix}\begin{bmatrix}\cos\Theta & 0 & -\sin\Theta\\ 0 & 1 & 0\\ \sin\Theta & 0 & \cos\Theta\end{bmatrix}\begin{bmatrix}\cos\Psi & \sin\Psi & 0\\ -\sin\Psi & \cos\Psi & 0\\ 0 & 0 & 1\end{bmatrix}$$

-rotation matrix [1]

Let us define $\vec{T}$ to be trust force and $\vec{H}$ is hub torque for every (1, 2, 3 and 4) DC Motor and Propeller system:

$$\vec{T} = b_T(\vec{\omega}_{propeller})^2, \qquad \vec{H} = b_H(\vec{\omega}_{propeller})^2 \tag{2-4}$$

-see references [1]
(hub torque = motor torque [2],[3])

Properties of the Drone:
$thrust = 37N\ per\ motor$
$m = mass\ =\ 10kg. l = 0.2m.$
$b_t\ =\ 8.5 * 10^{-6}. b_h\ =\ 2.15 * 10^{-7}$
$b = \frac{b_h}{b_t}\ =\ .0253$
Each of the 4 thrusters were used as the controlling variables.

where b_T and b_H are torque and thurst coefficients which can be determined experimentally. Then the following equations hold for angle and position acceleration [23]:

$$\ddot{\Phi} = \frac{l(\overrightarrow{T_2} - \overrightarrow{T_4}) - (I_{z_b z_b} - I_{y_b y_b})\dot{\Theta}\dot{\Psi}}{I_{x_b x_b}} \quad (2\text{-}5)$$

$$\ddot{\Theta} = \frac{l(\overrightarrow{T_3} - \overrightarrow{T_1}) - (I_{x_b x_b} - I_{z_b z_b})\dot{\Phi}\dot{\Psi}}{I_{y_b y_b}} \quad (2\text{-}6)$$

$$\ddot{\Psi} = \frac{(\overrightarrow{H_1} + \overrightarrow{H_3}) - (\overrightarrow{H_2} + \overrightarrow{H_4}) - (I_{y_b y_b} - I_{x_b x_b})\dot{\Phi}\dot{\Theta}}{I_{z_b z_b}} \quad (2\text{-}7)$$

$$\ddot{\vec{r}_X} = \frac{(\cos\Phi \sin\Theta \cos\Psi + \sin\Phi \sin\Psi)T}{m} \quad (2\text{-}8)$$

$$\ddot{\vec{r}_Y} = \frac{(\cos\Phi \sin\Theta \sin\Psi - \sin\Phi \cos\Psi)T}{m} \quad (2\text{-}9)$$

$$\ddot{\vec{r}_Z} = \frac{(\cos\Phi \cos\Theta)T - mg}{m} \quad (2\text{-}10)$$

-see reference [1]

Note: variable l (lower case L) is the length from 0 to the propeller.
variable $T = T_1 + T_2 + T_3 + T_4$

The H_n variables will be replaced with their relationship to T_n:

$H_n = T_n * \frac{b_h}{b_t} = T_n * b,\ where\ b = \frac{b_h}{b_t}$

$x_1 = \Phi; x_3 = \Theta\,; x_5 = \Psi\,; x_7 = r_x; x_9 = r_y; x_{11} = r_z$

The state equations:

$\dot{x}_1 = \dot{\Phi} = x_2$

$\dot{x}_2 = \ddot{\Phi} = \frac{l(T_2 - T_4) - (I_{zz} - I_{yy})x_4 x_6}{I_{xx}}$

$\dot{x}_3 = \dot{\Theta} = x_4$

$\dot{x}_4 = \ddot{\Theta} = \frac{l(T_3 - T_1) - (I_{xx} - I_{zz})x_2 x_6}{I_{yy}}$

$\dot{x}_5 = \dot{\Psi} = x_6$

$\dot{x}_6 = \ddot{\Psi} = \frac{b(T_1 + T_3 - T_2 - T_4) - (I_{yy} - I_{xx})x_2 x_4}{I_{zz}}$

$\dot{x}_7 = \dot{r}_x = x_8$

$\dot{x}_8 = \ddot{r}_x = \frac{(\cos(x_1)\sin(x_3)\cos(x_5) + \sin(x_1)\sin(x_5))T}{m}$

$\dot{x}_9 = \dot{r}_y = x_{10}$

$\dot{x}_{10} = \ddot{r}_y = \frac{(\cos(x_1)\sin(x_3)\sin(x_5) - \sin(x_1)\cos(x_5))T}{m}$

$\dot{x}_{11} = \dot{r}_z = x_{12}$

$\dot{x}_{12} = \ddot{r}_z = \frac{(\cos(x_1)\cos(x_3))T - mg}{m}$

The next step is to set up these state equations and test the dynamics using RK4.

Note: the full code is in the appendix

```
%variables for properties of the drone
Ixx=8e-4;Iyy=8e-4;Izz=6e-4;%double check these
%w=1000 rad/s. thrust = 8.5N. Torque = .215Nm
b=.0253;g=9.81;l=.2;m=10;
```

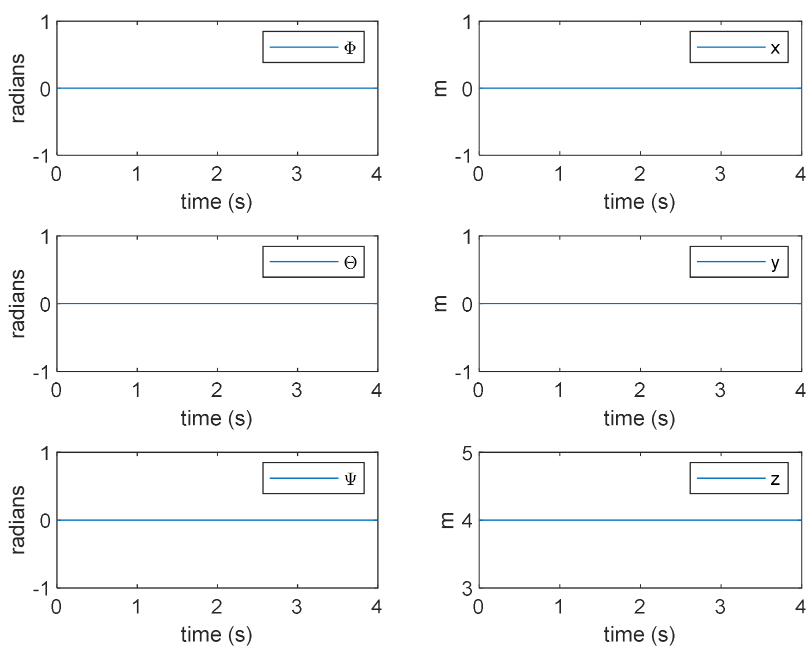
```
%initialize time vectors/parameters
dt=.01;
t0=0;
tt=t0:dt:t0+4;
L=length(tt);

%initial conditions
x=zeros(12,L);
x(11,1)=4;%initial height at 4m
```

The above properties are used for the different test cases for part 1A. In all tests, I've kept all 4 thrusters at some constant value and checked the resulting output (position/orientation).
I used the RK4 algorithm to solve this ODE (shown in the appendix)

Test 1:

```
TT=ones(4,1);%use constant thrust for part 1A
%TT(2)=1.0005;TT(4)=1.0005;
TT=TT.*((m*g)/sum(TT));%adjust so total thrust is equal to the weight
FF=@(t,x) F(t,x,TT);
```

All thrusters are set equal and the total thrust force is the weight of the drone. Result:

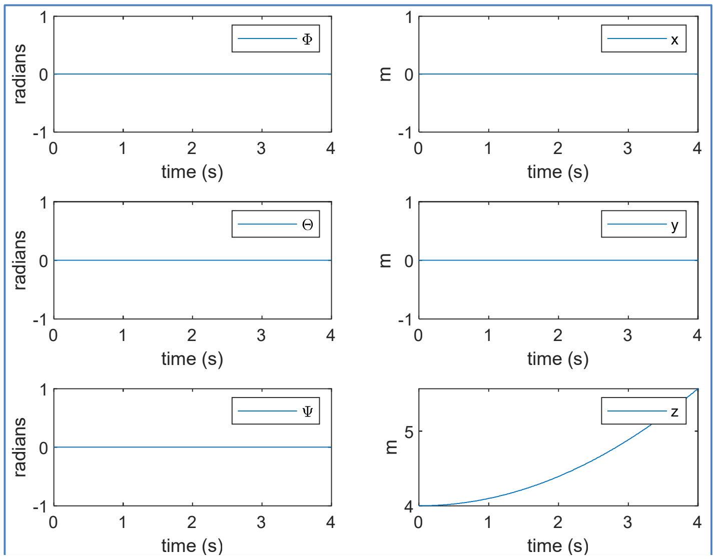


As expected, there's no rotation and no change in position.

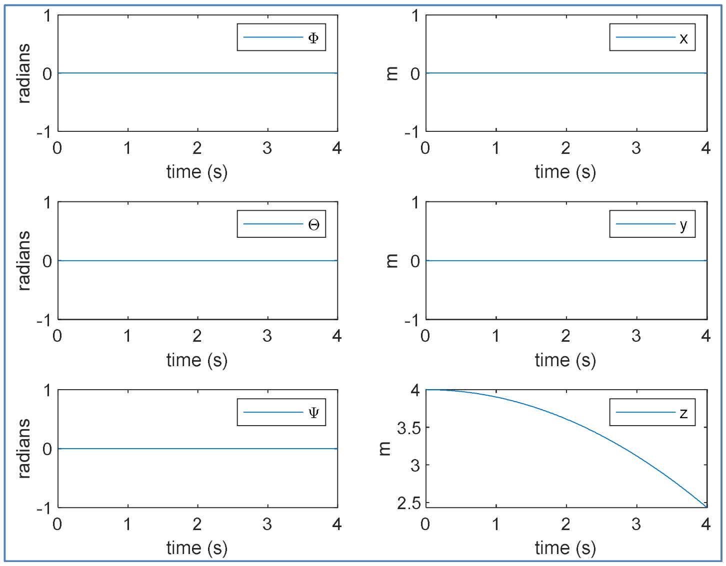
```
TT=ones(4,1);%use constant thrust for part 1A
%TT(2)=1.0005;TT(4)=1.0005;
TT=TT.*((m*g)/sum(TT))*1.02;%adjust so total thrust is equal to the weight
FF=@(t,x) F(t,x,TT);
```

```
TT=ones(4,1);%use constant thrust for part 1A
%TT(2)=1.0005;TT(4)=1.0005;
TT=TT.*((m*g)/sum(TT))*0.98;%adjust so total thrust is equal to the weight
FF=@(t,x) F(t,x,TT);
```

This time I increased the total thrust by 2% and then decreased the thrust by 2%. The results:



As expected, the drone increases height exponentially when there’s an increase in thrust. Note that my assumptions did not consider the change in thrust vs altitude or the effects of drag. This will allow the drone to increase in height at a more rapid pace.

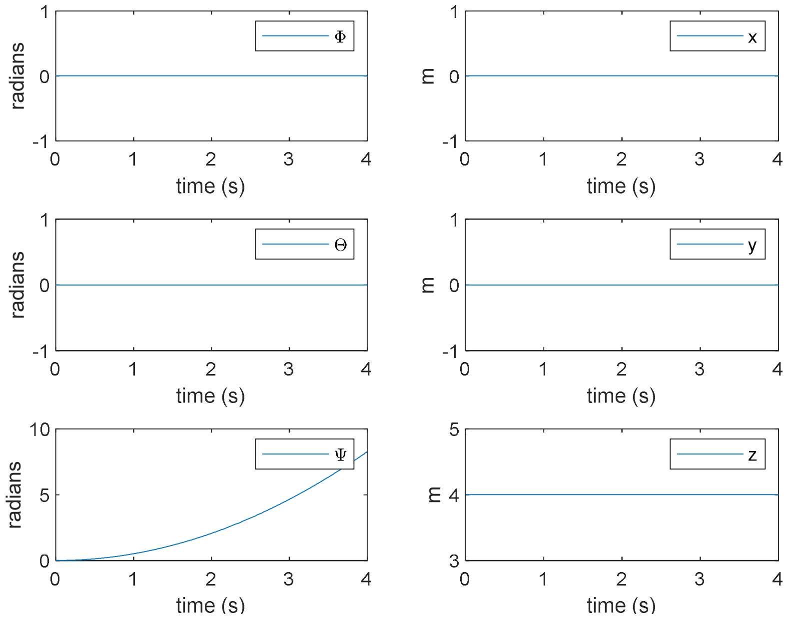


Similarly, the decrease in thrust causes the drone to decrease in altitude exponentially.

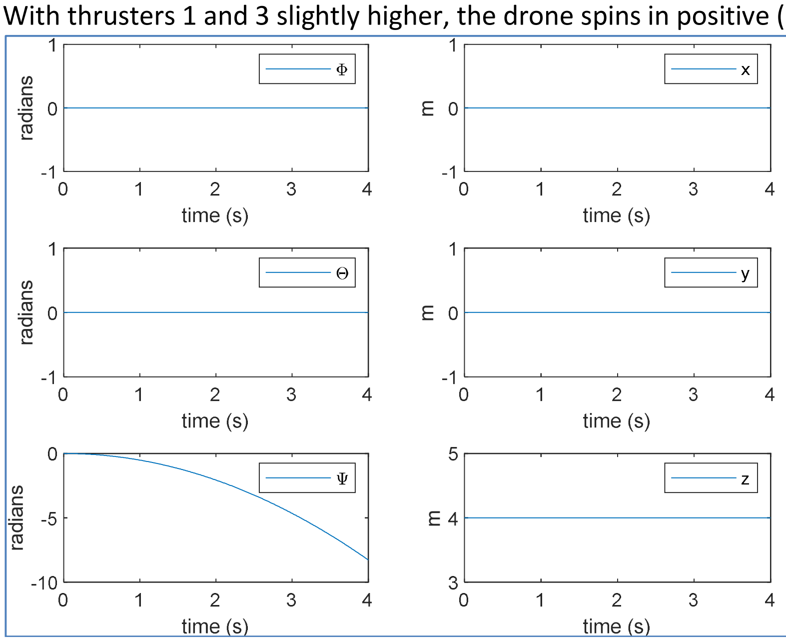
```
TT=ones(4,1);%use constant thrust for part 1A
TT(1)=1.0005;TT(3)=1.0005;
TT=TT.*((m*g)/sum(TT));%adjust so total thrust is equal to the weight
FF=@(t,x) F(t,x,TT);
```

```
TT=ones(4,1);%use constant thrust for part 1A
TT(2)=1.0005;TT(4)=1.0005;
TT=TT.*((m*g)/sum(TT));%adjust so total thrust is equal to the weight
FF=@(t,x) F(t,x,TT);
```

In these tests, I've slightly increased the thrust of the propeller's opposite of each other. First thrusters 1 and 3 and then another test on thrusters 2 and 4. This should cause the drone to spin. Results:



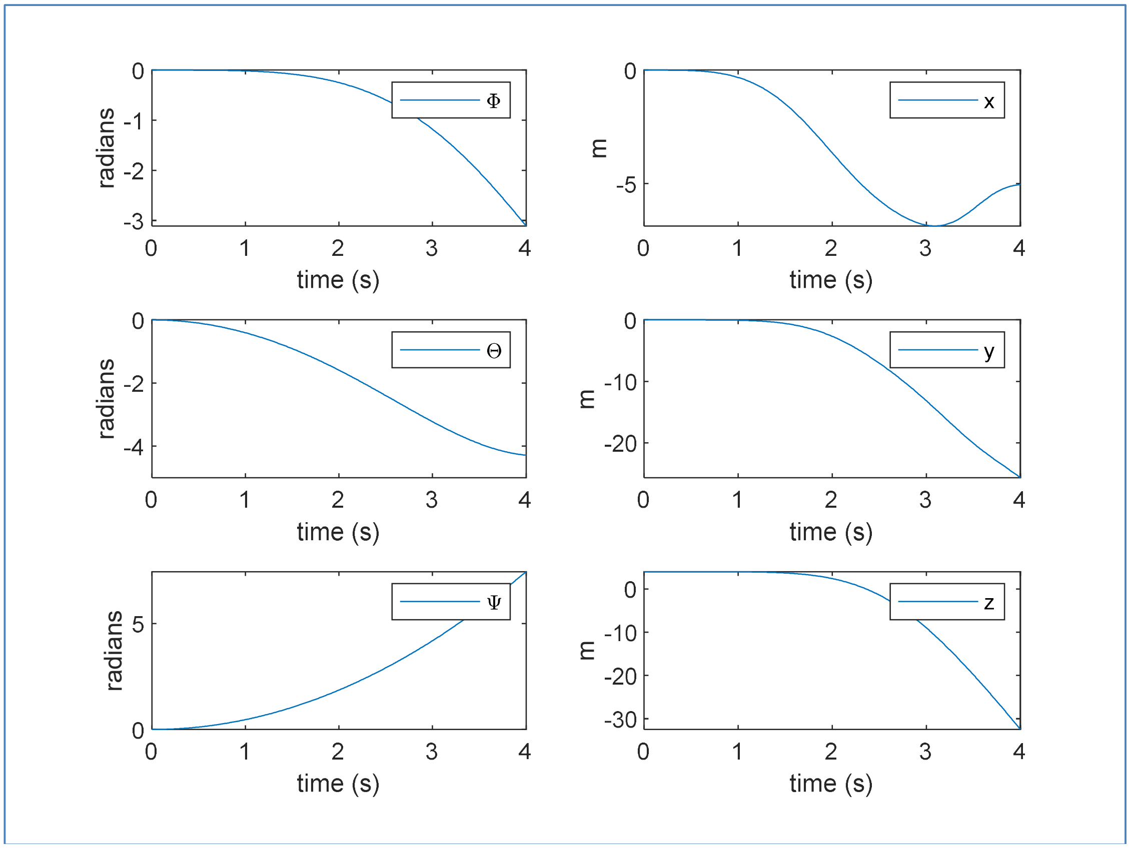
With thrusters 1 and 3 slightly higher, the drone spins in positive (CCW) direction about the z-axis.



Similarly, thrusters 2 and 4 being higher caused the drone to spin in negative (CW) direction about z-axis.

```
TT=ones(4,1);%use constant thrust for part 1A
TT(1)=1.0005;TT(3)=1.0004;
TT=TT.*((m*g)/sum(TT));%adjust so total thrust is equal to the weight
FF=@(t,x) F(t,x,TT);
```

In this test, I made thrusters 1 and 3 slightly higher but thruster 1 is slightly higher than thruster 3. This will cause the drone to tilt at an uneven orientation. Results:



The results show that the drone drops in altitude after about 1 second. The unbalance in the thrusters cause the drone to rotate about both the x and y axes.
The x-position goes negative and then slowly comes back up which indicates that the drone is spiraling out of control. (This can also be observed in the animation).

Part C: Numerical Optimal Control
Reiterating:
$H_n = T_n * b,\ where\ b = \frac{b_h}{b_t}\ ; T = \sum T_n$
$x_1 = \Phi; x_3 = \Theta\ ; x_5 = \Psi\ ; x_7 = r_x; x_9 = r_y; x_{11} = r_z$

The state equations:
$\dot{x}_1 = \dot{\Phi} = x_2$
$\dot{x}_2 = \ddot{\Phi} = \frac{l(T_2-T_4)-(I_{zz}-I_{yy})x_4x_6}{I_{xx}}$
$\dot{x}_3 = \dot{\Theta} = x_4$
$\dot{x}_4 = \ddot{\Theta} = \frac{l(T_3-T_1)-(I_{xx}-I_{zz})x_2x_6}{I_{yy}}$
$\dot{x}_5 = \dot{\Psi} = x_6$
$\dot{x}_6 = \ddot{\Psi} = \frac{b(T_1+T_3-T_2-T_4)-(I_{yy}-I_{xx})x_2x_4}{I_{zz}}$
$\dot{x}_7 = \dot{r}_x = x_8$
$\dot{x}_8 = \ddot{r}_x = \frac{(\cos(x_1)\sin(x_3)\cos(x_5)+\sin(x_1)\sin(x_5))T}{m}$
$\dot{x}_9 = \dot{r}_y = x_{10}$
$\dot{x}_{10} = \ddot{r}_y = \frac{(\cos(x_1)\sin(x_3)\sin(x_5)-\sin(x_1)\cos(x_5))T}{m}$
$\dot{x}_{11} = \dot{r}_z = x_{12}$
$\dot{x}_{12} = \ddot{r}_z = \frac{(\cos(x_1)\cos(x_3))T-mg}{m}$

Given, some initial condition, an optimal control will be used to bring the drone to a fixed final position while minimizing the control and minimizing time.
Constraints:
$0 \le T_n \le T_{max}$
$x_{11} \ge 0$ ←the drone cannot go below the ground at 0 altitude
t_f is unspecified. However, $t_0 = 0, t_f > 0$.

The cost function is calculated as:
$J = t_f + \frac{1}{2}\int_{t_0}^{t_f}(T_1^2 + T_2^2 + T_3^2 + T_4^2)dt = t_f + \sum_{i=1}^{N}\frac{1}{2}W_i\left(\sum_{j=1}^{4}T_j^2\right)$

Since LGL points will be used, the cost function will need to be modified to account for the fact that the integral is always assumed from -1 to 1 and then adjusted later:

$J = t_f + \frac{t_f-t_0}{2}\sum_{i=1}^{N}\frac{1}{2}W_i\left(\sum_{j=1}^{4}T_j^2\right)$

The gradient of the cost is such that:
$\frac{\delta J}{\delta x_i} = 0$
$\frac{\delta J}{\delta T_i} = \frac{t_f-t_0}{2}\sum_{i=1}^{N}W_i\,T_i$
$\frac{\delta J}{\delta t_f} = 1 + \frac{1}{2}\sum_{i=1}^{N}\frac{1}{2}W_i\left(\sum_{j=1}^{4}T_j^2\right)$

The decision vector:
$\{x_0, x_1, \dots x_{17}\}$
Where the last value is t_f and the 4 values before that are the 4 controls.

The constraint equation is of the form:

$$\begin{bmatrix} c_1 \\ \dots \\ c_{12} \end{bmatrix} = D \begin{bmatrix} x_1 \\ \dots \\ x_{12} \end{bmatrix} - \frac{t_f - t_0}{2} \begin{bmatrix} f_1(x,T) \\ \dots \\ f_{12}(x,T) \end{bmatrix} = 0$$

The gradient of the constraint equation is such that:
$\frac{\delta c_i}{\delta x_j} = D^T - \frac{t_f - t_0}{2} diag\left(\frac{\delta f_i}{\delta x_j}\right)$ ←diagonals only
$\frac{\delta c_i}{\delta x_j} = -\frac{t_f - t_0}{2} diag\left(\frac{\delta f_i}{\delta x_j}\right)$ ←non-diagonals
$\frac{\delta c_i}{\delta T_j} = -\frac{t_f - t_0}{2} diag\left(\frac{\delta f_i}{\delta T_j}\right)$
$\frac{\delta c_i}{\delta t_f} = 0 - \frac{1}{2} f_i(x,T)$

After a lot of testing, it was proving difficult to get a good simulation result and the simulations were taking a long time. For this reason, the next step was to make simplifications to the dynamics equations to improve the results.

Simplifications

$$U = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} T_1 + T_2 + T_3 + T_4 \\ T_2 - T_4 \\ T_3 - T_4 \\ (T_1 + T_3) - (T_2 + T_4) \end{bmatrix}$$

-these are the new controls: U1 (total thrust) ,U2 (pitch torque),U3 (roll torque),U4 (yaw torque)
-see reference [1]

$H_n = T_n * b,\ where\ b = \frac{b_h}{b_t}\ ; T = \sum T_n$
$x_1 = \Phi; x_3 = \Theta\ ; x_5 = \Psi\ ; x_7 = r_x; x_9 = r_y; x_{11} = r_z$

Substituting U into the state equations:

$\dot{x}_1 = \dot{\Phi} = x_2$
$\dot{x}_2 = \ddot{\Phi} = \frac{lU_2 - (I_{zz} - I_{yy})x_4 x_6}{I_{xx}}$
$\dot{x}_3 = \dot{\Theta} = x_4$
$\dot{x}_4 = \ddot{\Theta} = \frac{lU_3 - (I_{xx} - I_{zz})x_2 x_6}{I_{yy}}$
$\dot{x}_5 = \dot{\Psi} = x_6$
$\dot{x}_6 = \ddot{\Psi} = \frac{bU_4 - (I_{yy} - I_{xx})x_2 x_4}{I_{zz}}$
$\dot{x}_7 = \dot{r}_x = x_8$
$\dot{x}_8 = \ddot{r}_x = \frac{(\cos(x_1)\sin(x_3)\cos(x_5) + \sin(x_1)\sin(x_5))U_1}{m}$
$\dot{x}_9 = \dot{r}_y = x_{10}$
$\dot{x}_{10} = \ddot{r}_y = \frac{(\cos(x_1)\sin(x_3)\sin(x_5) - \sin(x_1)\cos(x_5))U_1}{m}$
$\dot{x}_{11} = \dot{r}_z = x_{12}$
$\dot{x}_{12} = \ddot{r}_z = \frac{(\cos(x_1)\cos(x_3))U_1 - mg}{m}$

The equations can still be simplified even further. It can be assumed that all inertia is the same. It can also be simplified with the small angles assumption, limiting the amount of roll/pitch. It is best to keep the full rotation available for yaw, in case it is desired to make the drone spin.

Assuming small angles on roll and pitch only and Ixx=Iyy=Izz=I:

$H_n = T_n * b, \; where \; b = \frac{b_h}{b_t}$;

$x_1 = \Phi; x_3 = \Theta \, ; x_5 = \Psi \, ; x_7 = r_x; x_9 = r_y; x_{11} = r_z$

The state equations:

$\dot{x}_1 = \dot{\Phi} = x_2$

$\dot{x}_2 = \ddot{\Phi} = \frac{lU_2}{I}$

$\dot{x}_3 = \dot{\Theta} = x_4$

$\dot{x}_4 = \ddot{\Theta} = \frac{lU_3}{I}$

$\dot{x}_5 = \dot{\Psi} = x_6$

$\dot{x}_6 = \ddot{\Psi} = \frac{bU_4}{I}$

$\dot{x}_7 = \dot{r}_x = x_8$

$\dot{x}_8 = \ddot{r}_x = \frac{(x_3\cos(x_5)+x_1\sin(x_5))U_1}{m}$

$\dot{x}_9 = \dot{r}_y = x_{10}$

$\dot{x}_{10} = \ddot{r}_y = \frac{(x_3\sin(x_5)-x_1\cos(x_5))U_1}{m}$

$\dot{x}_{11} = \dot{r}_z = x_{12}$

$\dot{x}_{12} = \ddot{r}_z = \frac{U_1 - mg}{m}$

$U_1 = x_{13}, U_2 = x_{14}, U_3 = x_{15}, U_4 = x_{16}$

Now the equations are much simpler and should hopefully help speed up calculation and improve the chances of finding a minimum within consraints.
Note that for the gradients, small angle approximation is only used after taking the partial derivatives.

Boundaries: boundaries were set per the I.C. and F.C. on the positions,orientations,velocities, and radial velocities. Boundaries were also set up to add the appropriate limits to the roll/pitch angles (+-5 degrees max) and all the controls (U).

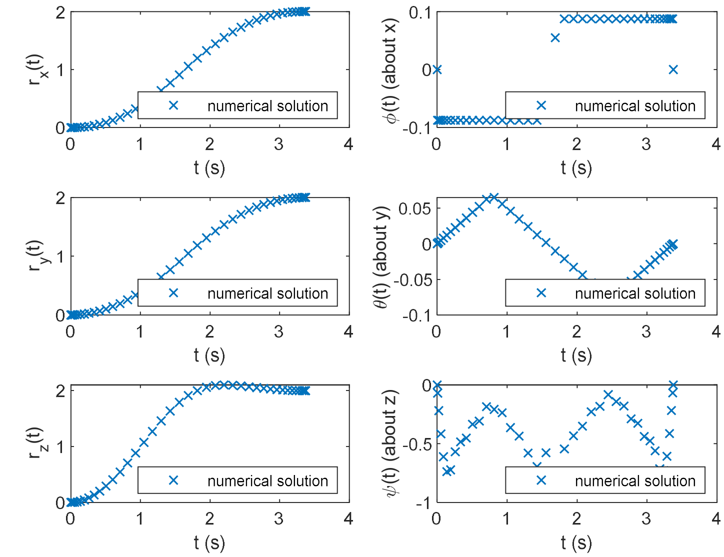
```
%initial guess
%*********toggle these values. the boundary conditions automatically match
%   up with what I have here for I.C. and F.C.
x0=zeros(Nv*N+1,1);
%x0(zrange)=linspace(0,2,N)';%initially guess a linear climb
x0(11*N)=2;%rz(tf)
x0(7*N)=2;%rx(tf)
x0(9*N)=2;%ry(tf)
x0(Nv*N+1)=10;%final time tf to start with (initial guess)
```

-for all simulations, the start and end positions were toggled to update the study. When running the code, just change the ending rx,ry,rz. There is one version of this code in the appendix which can easily be modified at this spot in the code for each of the cases shown in the next pages.
The cost function was updated to:

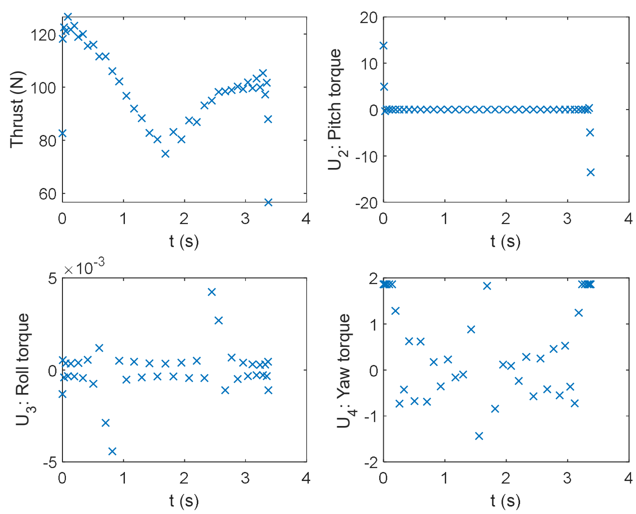
$J = t_f + \frac{1}{2}\int_{t_0}^{t_f} U_1^2 \, dt$

First, the drone was set to go from x,y,z=0 to x,y,z=2. N=41 LGL points were used until a local minimum was found:

```
Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the default value of the step size tolerance and constraints are
satisfied to within the default value of the constraint tolerance.
```



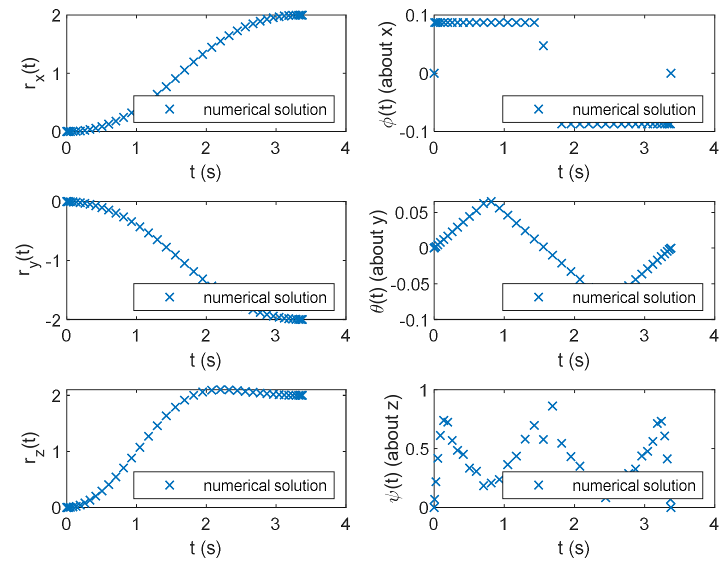
The control performed exactly as expected. There's an animation to go with this and it looks pretty realistic. The drone needed to rotate about z so that it could orient the right way such that it'll go up and tilt in the x and y direction and then it tilted back, which allowed itself to come to a stop at this position. At this point, it rotated the z-axis back.



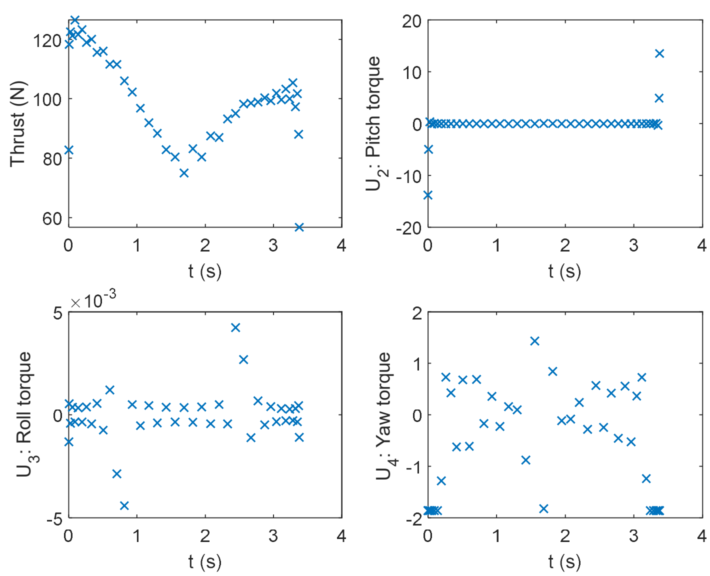
The total thrust acted appropriately, showing the max in the beginning, then as momentum picked up, it slowed down. Then as it needed to stabilize at the new position, it ramped up a little bit before throttling down. The pitch torque was much higher than the roll torque. Perhaps the rotation of the drone didn't require both a pitch and roll torque. The yaw torque was definitely throttled up and down to get the drone to behave this way.

This time the final conditions were updated so that y goes from 0 to -2. It gave similar results as expected:

```
Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the default value of the step size tolerance and constraints are
satisfied to within the default value of the constraint tolerance.
```



The graphs look about the same as before except that y goes from 0 to -2.

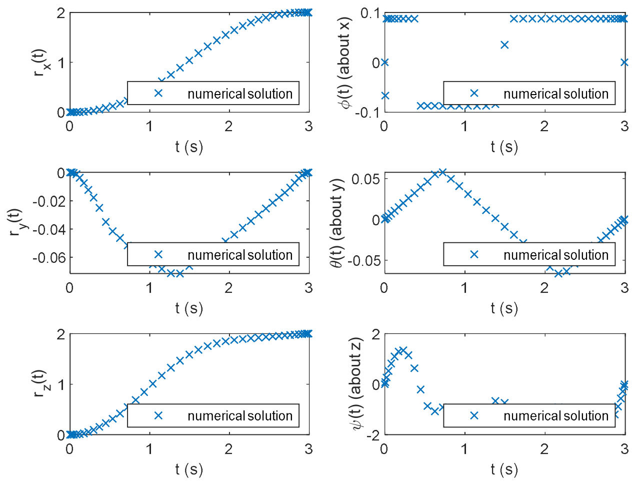


It took 3.3748 seconds to reach the goal position. Please see the appendix which includes the code for the simulation as well as the code for the animation.

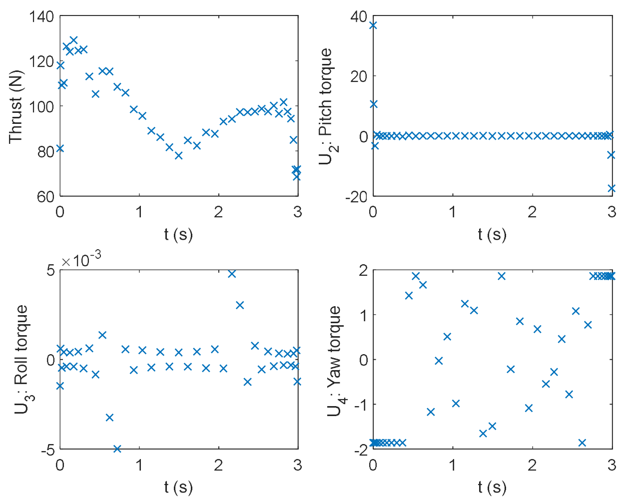
This simulation used 41 LGL points and it took about a minute to find the minimum.

This time x,z=0 to x,z=2. The y axis was set to stay at its current position.

```
Local minimum possible. Constraints satisfied.

fmincon stopped because the size of the current step is less than
the default value of the step size tolerance and constraints are
satisfied to within the default value of the constraint tolerance.
```

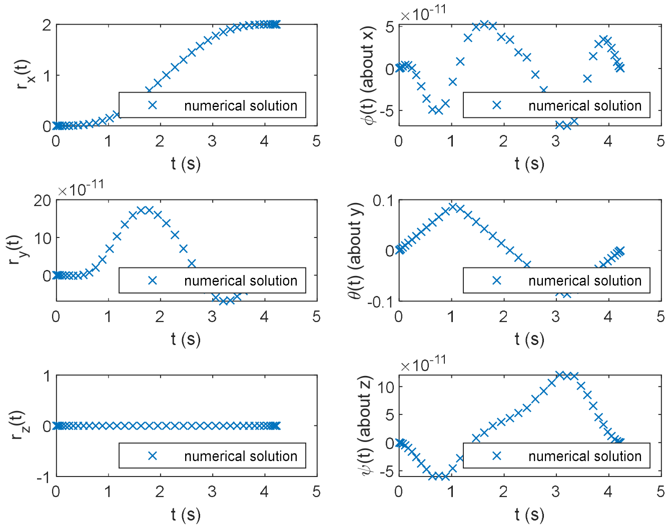


The drone dipped very slightly in the y-direction but found its way back to 0. Looking at the graphs, this was also a success.



The yaw torque seems to jump around quite a bit. When looking at the animation, it does appear like the spinning of the drone (about z-axis) is a little shaky and not perfectly smooth. The animation does still appear realistic. It is just interesting to see the result of this numerical control.

In this last study, the start and end height stayed at 0. The drone went from x=0 to x=2. There's a boundary that prevents the drone from ever going below 0. Here's the result:



The drone did not fly up and over as intuitively expected. However, this is only because of the simple constraint that prevents it from going below 0. To account for the ground, special constraints would need to be built into the equations to adjust the minimum height for different roll/pitch angles or the parameters (boundary conditions) would need to tell the drone to go up and then back down to prevent it from hitting the ground. In the end, the drone did exactly what it was told to do by the parameters. The motion in the animation for this study was very smooth.

Conclusion

Using LGL points to find a numerical solution of optimizing the control proved to be a challenging but effective approach. There were many attempts and it seems the approach that worked best was using the roll,pitch,yaw torque and the total thrust as the controls. Simplifying with the small angles assumption also helped improve how quickly a solution could be found.

Future Studies

On a future study, it wouldn't be too difficult to break this simulation up to have several way points for the drone (rather than just a start and end). To accomplish a more complicated path, the control optimization could be broken it up into multiple simulations – each with new initial/final conditions. The trajectory planning shown is evident that this should be a possibility.

References

[1] Control and Estimation of a Quadcopter Dynamic Model by Sevkuthan Kurak & Migdat Hodzic
[2] https://www.droneomega.com/drone-motor-essentials/
[3] https://www.micromo.com/technical-library/dc-motor-tutorials/motor-calculations