ESE 4481 Midterm / Seif Elkhashab / 04-09-2024

Drone Control Mixer

Introduction:

This report details the development of a motor mixer for the crazyflie 2.1 drone. Instead of commanding motor speeds, the goal is to command moments and forces as our inputs. The mixer problem is converting commanded moments and forces into desired motor speed and then into the PWM values needed for each motor. These calculations will be directly executed in the firmware of the crazyflie drone.

Methods:

Equations for the moments:

The equations for the moments and vertical force are as follows:

$$Z = \rho C_t D^4 (v_{fl} + v_{fr} + v_{rl} + v_{rr})$$
$$L = \ell \rho C_t D^4 (v_{fl} - v_{fr} + v_{rl} - v_{rr})$$
$$M = \ell \rho C_t D^4 (v_{fl} + v_{fr} - v_{rl} - v_{rr})$$
$$N = \rho C_q D^5 (-v_{fl} + v_{fr} + v_{rl} - v_{rr})$$

The equations above describe Z representing the total vertical thrust force generated by the propellers, whereas L, M, and N represent the moments (torques) about the drone's longitudinal x, lateral y, and vertical z axes, respectively.

- $v_{fl}, v_{fr}, v_{rl}, v_{rr}$, denote the speeds of the front-left, front-right, rear-left, and rear-right propellers, respectively. These speeds are what is commanded to the drone in the form of PWM values, the mapping between motor speed and PWM will be derived later in this report.
- ρ stands for the air density, which influences the thrust and torque generated by the propellers.
- C_t and C_q are the thrust and torque coefficients, respectively,which were derived from a previous assignment.
- D is the diameter of the propellers, affecting how much thrust is generated.
- ℓ represents the distance from the center of the drone to each of the propellers, which is important for how the moments affect the drone as it acts as a lever.

For this drone the following parameter values are used:

$$\rho = 1.225\ kg/m^3 \quad D = 0.045\ m \quad \ell = 0.046\ m \quad C_t = 0.1732223315 \quad C_q = 0.0535322322$$

The moments can be written in matrix form:

$$\begin{pmatrix} Z \\ L \\ M \\ N \end{pmatrix} = \begin{pmatrix} C_t \, \mathrm{D}^4 \, \rho \, \left({n_1}^2 + {n_2}^2 + {n_3}^2 + {n_4}^2 \right) \\ C_t \, \mathrm{D}^4 \, l \, \rho \, \left({n_1}^2 - {n_2}^2 + {n_3}^2 - {n_4}^2 \right) \\ C_t \, \mathrm{D}^4 \, l \, \rho \, \left({n_1}^2 + {n_2}^2 - {n_3}^2 - {n_4}^2 \right) \\ -C_q \, \mathrm{D}^5 \, \rho \, \left({n_1}^2 - {n_2}^2 - {n_3}^2 + {n_4}^2 \right) \end{pmatrix}$$

The moments can be rewritten as :

$$\begin{pmatrix} Z \\ L \\ M \\ N \end{pmatrix} = \begin{pmatrix} C_t \, \mathrm{D}^4 \, \rho \\ C_t \, \mathrm{D}^4 \, l \, \rho \\ C_t \, \mathrm{D}^4 \, l \, \rho \\ C_q \, \mathrm{D}^5 \, \rho \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ -1 & 1 & 1 & -1 \end{pmatrix} \begin{pmatrix} {n_1}^2 \\ {n_2}^2 \\ {n_3}^2 \\ {n_4}^2 \end{pmatrix}$$

$$\begin{pmatrix} Z \\ L \\ M \\ N \end{pmatrix} = \begin{pmatrix} C_t \, \mathrm{D}^4 \, \rho & C_t \, \mathrm{D}^4 \, \rho & C_t \, \mathrm{D}^4 \, \rho & C_t \, \mathrm{D}^4 \, \rho \\ C_t \, \mathrm{D}^4 \, l \, \rho & -C_t \, \mathrm{D}^4 \, l \, \rho & C_t \, \mathrm{D}^4 \, l \, \rho & -C_t \, \mathrm{D}^4 \, l \, \rho \\ C_t \, \mathrm{D}^4 \, l \, \rho & C_t \, \mathrm{D}^4 \, l \, \rho & -C_t \, \mathrm{D}^4 \, l \, \rho & -C_t \, \mathrm{D}^4 \, l \, \rho \\ -C_q \, \mathrm{D}^5 \, \rho & C_q \, \mathrm{D}^5 \, \rho & C_q \, \mathrm{D}^5 \, \rho & -C_q \, \mathrm{D}^5 \, \rho \end{pmatrix} \begin{pmatrix} {n_1}^2 \\ {n_2}^2 \\ {n_3}^2 \\ {n_4}^2 \end{pmatrix}$$

Motor Speeds in terms of Forces & Moments:

Some simple linear algebra can be done to isolate the motor speeds and get them to be a funciton of the forces and moments commanded to the drone.

The inverse of the mixer matrix is taken to isolate the propoller speeds :

$$\begin{pmatrix} {n_1}^2 \\ {n_2}^2 \\ {n_3}^2 \\ {n_4}^2 \end{pmatrix} = \begin{pmatrix} \frac{1}{4 \, C_t \, \mathrm{D}^4 \, \rho} & \frac{1}{4 \, C_t \, \mathrm{D}^4 \, l \, \rho} & \frac{1}{4 \, C_t \, \mathrm{D}^4 \, l \, \rho} & -\frac{1}{4 \, C_q \, \mathrm{D}^5 \, \rho} \\ \frac{1}{4 \, C_t \, \mathrm{D}^4 \, \rho} & -\frac{1}{4 \, C_t \, \mathrm{D}^4 \, l \, \rho} & \frac{1}{4 \, C_t \, \mathrm{D}^4 \, l \, \rho} & \frac{1}{4 \, C_q \, \mathrm{D}^5 \, \rho} \\ \frac{1}{4 \, C_t \, \mathrm{D}^4 \, \rho} & \frac{1}{4 \, C_t \, \mathrm{D}^4 \, l \, \rho} & -\frac{1}{4 \, C_t \, \mathrm{D}^4 \, l \, \rho} & \frac{1}{4 \, C_q \, \mathrm{D}^5 \, \rho} \\ \frac{1}{4 \, C_t \, \mathrm{D}^4 \, \rho} & -\frac{1}{4 \, C_t \, \mathrm{D}^4 \, l \, \rho} & -\frac{1}{4 \, C_t \, \mathrm{D}^4 \, l \, \rho} & -\frac{1}{4 \, C_q \, \mathrm{D}^5 \, \rho} \end{pmatrix} \begin{pmatrix} Z \\ L \\ M \\ N \end{pmatrix}$$

The numerical values are then plugged in to get the computational equations:

$$\begin{pmatrix} {n_1}^2 \\ {n_2}^2 \\ {n_3}^2 \\ {n_4}^2 \end{pmatrix} = \begin{pmatrix} 287310.0 & 6.2459\mathrm{e}{+6} & 6.2459\mathrm{e}{+6} & -2.066\mathrm{e}{+7} \\ 287310.0 & -6.2459\mathrm{e}{+6} & 6.2459\mathrm{e}{+6} & 2.066\mathrm{e}{+7} \\ 287310.0 & 6.2459\mathrm{e}{+6} & -6.2459\mathrm{e}{+6} & 2.066\mathrm{e}{+7} \\ 287310.0 & -6.2459\mathrm{e}{+6} & -6.2459\mathrm{e}{+6} & -2.066\mathrm{e}{+7} \end{pmatrix} \begin{pmatrix} Z \\ L \\ M \\ N \end{pmatrix}$$

The right hand side can be multiplied to consolidate terms:

$$\begin{pmatrix} {n_1}^2 \\ {n_2}^2 \\ {n_3}^2 \\ {n_4}^2 \end{pmatrix} = \begin{pmatrix} 6.2459\mathrm{e}{+6} \, L + 6.2459\mathrm{e}{+6} \, M - 2.066\mathrm{e}{+7} \, N + 287310.0 \, Z \\ 6.2459\mathrm{e}{+6} \, M - 6.2459\mathrm{e}{+6} \, L + 2.066\mathrm{e}{+7} \, N + 287310.0 \, Z \\ 6.2459\mathrm{e}{+6} \, L - 6.2459\mathrm{e}{+6} \, M + 2.066\mathrm{e}{+7} \, N + 287310.0 \, Z \\ 287310.0 \, Z - 6.2459\mathrm{e}{+6} \, M - 2.066\mathrm{e}{+7} \, N - 6.2459\mathrm{e}{+6} \, L \end{pmatrix}$$

The square root of both sides is taken to get speed instead of speed squared:

$$\sqrt{\begin{pmatrix} {n_1}^2 \\ {n_2}^2 \\ {n_3}^2 \\ {n_4}^2 \end{pmatrix}} = \sqrt{\begin{pmatrix} 6.2459\text{e+}6\,L + 6.2459\text{e+}6\,M - 2.066\text{e+}7\,N + 287310.0\,Z \\ 6.2459\text{e+}6\,M - 6.2459\text{e+}6\,L + 2.066\text{e+}7\,N + 287310.0\,Z \\ 6.2459\text{e+}6\,L - 6.2459\text{e+}6\,M + 2.066\text{e+}7\,N + 287310.0\,Z \\ 287310.0\,Z - 6.2459\text{e+}6\,M - 2.066\text{e+}7\,N - 6.2459\text{e+}6\,L \end{pmatrix}}$$

$$\begin{pmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \end{pmatrix} = \sqrt{\begin{pmatrix} 6.2459\text{e+}6\,L + 6.2459\text{e+}6\,M - 2.066\text{e+}7\,N + 287310.0\,Z \\ 6.2459\text{e+}6\,M - 6.2459\text{e+}6\,L + 2.066\text{e+}7\,N + 287310.0\,Z \\ 6.2459\text{e+}6\,L - 6.2459\text{e+}6\,M + 2.066\text{e+}7\,N + 287310.0\,Z \\ 287310.0\,Z - 6.2459\text{e+}6\,M - 2.066\text{e+}7\,N - 6.2459\text{e+}6\,L \end{pmatrix}}$$

The final equations for the speeds of the 4 motors as a function of the forces & moments are:

$$n_1 = \sqrt{6.2459\text{e+}6\,L + 6.2459\text{e+}6\,M - 2.066\text{e+}7\,N + 287310.0\,Z}$$
$$n_2 = \sqrt{6.2459\text{e+}6\,M - 6.2459\text{e+}6\,L + 2.066\text{e+}7\,N + 287310.0\,Z}$$
$$n_3 = \sqrt{6.2459\text{e+}6\,L - 6.2459\text{e+}6\,M + 2.066\text{e+}7\,N + 287310.0\,Z}$$
$$n_4 = \sqrt{287310.0\,Z - 6.2459\text{e+}6\,M - 2.066\text{e+}7\,N - 6.2459\text{e+}6\,L}$$

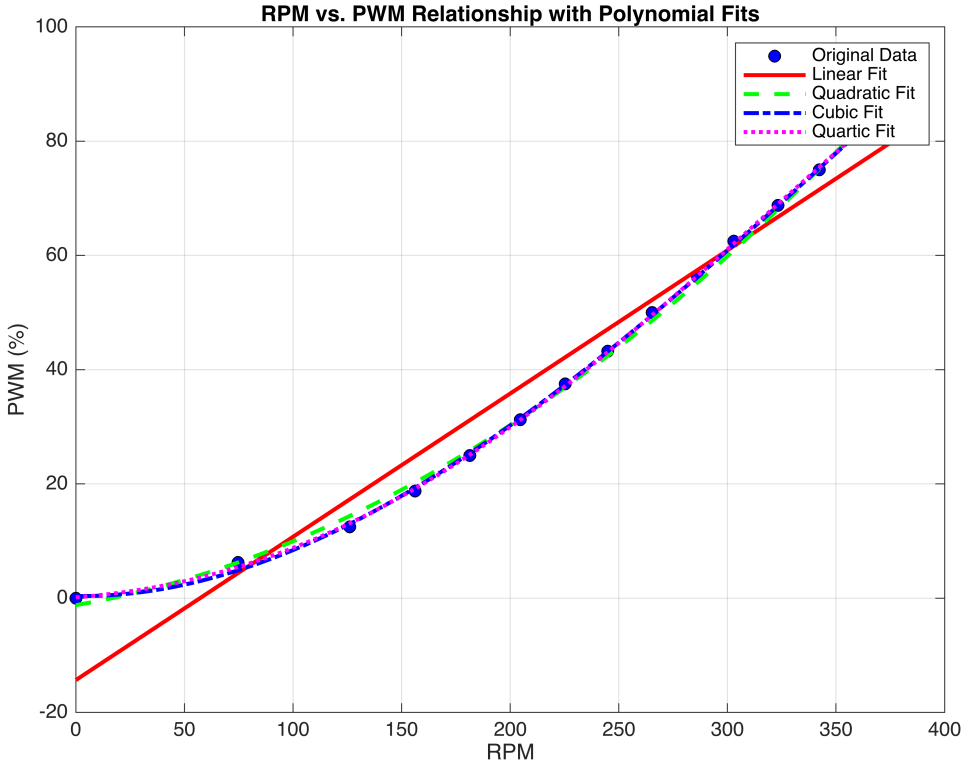
Relating Motors speed to PWM Values:

The purpose of this section is to map the motor speeds calculated in the previous section to a PWM value to command the motors of the drone. These equations will then be coded into the firmware of the drone, the code will be in the src/power_distribution_quadrotor.c file.

PWM (%)	RPM
0	0
6.25	4485
12.5	7570
18.75	9374
25	10885
31.25	12277
37.5	13522
43.25	14691
50	15924
56.25	17174
62.5	18179
68.75	19397
75	20539
81.25	21692
87.5	22598

The data above is collected from a crazyflie 2.1 drone, and using this data, a relationship can be established to map RPM to PWM or PWM to RPM. For more accurate results, the analysis will be done with RPS (revolutions

per second) so the RPM values will be divided by 60. This conversion to RPS is compatable with the SI units used for all the derivations.



As can be seen by the graph above, the data can be fitted with a line that correctly captures the relationship between PWM and RPM. Unsurprisingly, the higher order fits perform better, but the computational cost may not be worth it. I will be using the the Quadratic fit as it hugs the data well enough but if more preformance is needed the higher order fits can be used.

The equations resulting from the 4 diffrent fits:

$$\text{PWM} = 0.25079\ \text{RPS} - 14.328$$

$$\text{PWM} = 0.00045962\ \text{RPS}^2 + 0.065704\ \text{RPS} - 1.2081$$

$$\text{PWM} = -8.2801\text{e-}7\ \text{RPS}^3 + 0.00093662\ \text{RPS}^2 - 0.0049072\ \text{RPS} + 0.38155$$

$$\text{PWM} = -2.3545\text{e-}9\ \text{RPS}^4 + 9.8738\text{e-}7\ \text{RPS}^3 + 0.00049435\ \text{RPS}^2 + 0.029512\ \text{RPS} + 0.15577$$

As seen above by the equations the cubic term's magnitude is very small so there is diminshing returns past using a quadratic term. Now that the mapping between RPS and PWM is found, motor speeds as a function of forces from above can ve converted to PWMs as a function of forces and moments for the 4 motors as can be seen below.

The final equations for the PWM values of the 4 motors as a function of the forces & moments are:

$$\text{PWM}_1 = 2870.7\ L + 2870.7\ M - 9495.6\ N + 132.05\ Z + 0.065704\ \sqrt{6.2459\text{e+}6\ L + 6.2459\text{e+}6\ M - 2.0}$$

$$\text{PWM}_2 = 2870.7\ M - 2870.7\ L + 9495.6\ N + 132.05\ Z + 0.065704\ \sqrt{6.2459\text{e+}6\ M - 6.2459\text{e+}6\ L + 2.0}$$

$$\text{PWM}_3 = 2870.7\ L - 2870.7\ M + 9495.6\ N + 132.05\ Z + 0.065704\ \sqrt{6.2459\text{e+}6\ L - 6.2459\text{e+}6\ M + 2.0}$$

$$\text{PWM}_4 = 132.05\ Z - 2870.7\ M - 9495.6\ N - 2870.7\ L + 0.065704\ \sqrt{287310.0\ Z - 6.2459\text{e+}6\ M - 2.06}$$

Coding the Mixer into the Drone's Firmaware:

As can be seen in the picture below, the mathmatical formulation describe previously through out the report was synthesised into the powerDistributionForceTorque Function. The commanded speed of each motor is calculated based on the commanded lift and torques about the drone's axis. Those motor speeds are then converted to PWM values for each motor based on the quadratic fit found above, and each motor is then set to that value. These values are bounded between 0 and 65535 so saturation is taken into account in the design. This is the majority of the code that is required for the firmware.

```
// Adding RPS to PWM conversion coeffecients, quadratic fit
static float a = 0.00046; // Quadratic term
static float b = 0.0657;   // Linear term
static float c = -1.208;    // Constant term

static void powerDistributionForceTorque(const control_t *control, motors_thrust_uncapped_t* motorThrustUncapped) {

    // Defining the control variables as in legacy function
    const float L=control->roll/2.0f;
    const float M=control->pitch/2.0f;
    const float N=control->yaw;
    const float Z=control->thrust;

    // Speed equations for each motor
    float n_1_speed = sqrt(6245858.678*L + 6245858.678*M - 20659794.532*N + 287309.499*Z);
    float n_2_speed = sqrt(-6245858.678*L + 6245858.678*M  + 20659794.532*N + 287309.499*Z);
    float n_3_speed = sqrt(6245858.678*L - 6245858.678*M + 20659794.532*N + 287309.499*Z);
    float n_4_speed = sqrt(-6245858.678*L - 6245858.678*M - 20659794.532*N + 287309.499*Z);

    // Convert motor speeds to PWM values, ensuring they are within the range 0 to 65535
    uint16_t pwm_m1 = (uint16_t) fminf(fmaxf(a * pow(n_1_speed, 2) + b * n_1_speed + c, 0), 65535);
    uint16_t pwm_m2 = (uint16_t) fminf(fmaxf(a * pow(n_2_speed, 2) + b * n_2_speed + c, 0), 65535);
    uint16_t pwm_m3 = (uint16_t) fminf(fmaxf(a * pow(n_3_speed, 2) + b * n_3_speed + c, 0), 65535);
    uint16_t pwm_m4 = (uint16_t) fminf(fmaxf(a * pow(n_4_speed, 2) + b * n_4_speed + c, 0), 65535);

    // Assign calculated PWM values to the motorThrustUncapped structure
    motorThrustUncapped->motors.m1 = pwm_m1;
    motorThrustUncapped->motors.m2 = pwm_m2;
    motorThrustUncapped->motors.m3 = pwm_m3;
    motorThrustUncapped->motors.m4 = pwm_m4;
}
```

One other function was changed to make it so the powerDistributionForceTorque function is used as the default control scheme instead of the `powerDistributionLegacy` function which was used previously. The change can be seen in the image below.

```
void powerDistribution(const control_t *control, motors_thrust_uncapped_t* motorThrustUncapped)
{
  switch (control->controlMode) {
    case controlModeLegacy:
      powerDistributionLegacy(control, motorThrustUncapped);
      break;
    case controlModeForceTorque:
      powerDistributionForceTorque(control, motorThrustUncapped);
      break;
    case controlModeForce:
      powerDistributionForce(control, motorThrustUncapped);
      break;
    default:
      powerDistributionForceTorque(control, motorThrustUncapped);
      break;
  }
}
```

Conclusions:

A drone mixer was successfully developed to take in commanded forces and moments of the crazyflie 2.1 drone and transform them into commanded motor speeds. A relationship between motor speed and motor PWM was then found and implemented by using a quadratic fit for the PWM/RPS data.

There was a lot of technical issues faced with connecting and flashing the drone, as the drone entered bootloader mode indefinately. As such I was not able to connect to the drone and test my code during the majority of this assigmnent. I have now exchanged my drone for a working model and will test this code as soon as I figure out the connection issues I have been experiencing.

While there are many technical issues on the hardware side, I believe the mathmatical formulation is found and that by choosing to use the symbolic matlab through out the whole assignment, I have made development and debugging much easier. This is because when I change one variable or equation earlier in the code, the changes are carried through symbolically and mathmatically to the end of the script, making it very easy to make changes and test results.

I worked alone on this assignment, JC and I discussed the assignment once, but other than that we did all the work independently.