Basic Analysis on Flight Model of Arma 3 and comparison to AWESome Aerodynamics

Abstract

Since Arma 3 provides a large warfare environment and aircrafts, fixed-wing aircrafts serve an important role in Arma 3. While there are numerous fixed-wings developed by addon developers, how exactly the fixed-wing aircrafts behave are mostly not known. This paper introduces discoveries related to fixed-wing aircraft flight model during the development of AWESome addon and compare vanilla and AWESome aerodynamics to find out how differently they behave.

The equation and coefficients for drag and lift has been found, and an abnormal behavior of lift has also been discovered, which warns aircraft addon developers to increase maxSpeed config to prevent such abnormal behaviors.

Author: Mingi Park (Orbis)

Multiplay Game Management

I. Introduction

Fixed-wing aircrafts serve an important role in Arma 3, both for anti-air and anti-ground and therefore there are various aircrafts developed by Bohemia Interactive and private developers. While aerodynamic model is a large factor on designing aircrafts, most of the aerodynamic model is not known to personal developers and acted a big obstacle. There are several webpages explaining fact related to aerodynamic model [1][2], but most of the equations and coefficients are not known.

During recent development of AWESome (Aerial Warfare Enhanced Somehow) addon [3], there has been significant improvement in understandings of flight model in Arma 3 on both equations and coefficients. Here, this paper will explain what has been discovered and compare vanilla flight model with the aerodynamics in AWESome addon.

II. Theatrical Background

1. Aerodynamics in real world

A. Drag

Drag is defined as *'the aerodynamic force that opposes an aircraft's motion through the air'* [4]. Aerodynamic drag in real world is a composition of many causes. The three constituents in aerodynamic drag are: parasitic drag; lift-induced drag; and wave drag [5]. Parasitic drag has two reasons, the friction between aircraft and atmosphere, and pressure created in front of the aircraft. Lift-induced drag also has two reason, trailing vortexes and lift-induced viscous drag. The Wave drag is created when the fluid flow is near-sonic.

The parasitic drag is proportional to the square of speed, and therefore the drag coefficient is given as Eq. 1.

$$\mathrm{Cd} = \frac{2\mathrm{F_d}}{\rho u^2 A} \qquad \text{(Eq. 1)}$$

Where

$\mathrm{C_d}$: drag coefficient

$\mathrm{F_d}$: drag force

ρ: air density

u: fluid velocity

A: reference area

While this equation works not only for parasitic drag but for all other drag constituents, but they aren't proportional to speed squared, the drag coefficient shows a trend giving a peak in Mach 1 as given in Fig. 1.

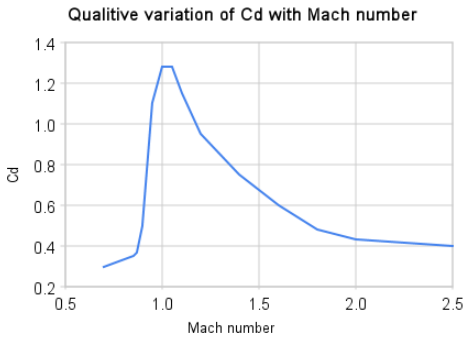

Fig. 1 Wikimedia Commons. (2009, September 12). Qualitive variation of cd with mach number. Retrieved January 02, 2019, from https://commons.wikimedia.org/wiki/File:Qualitive_variation_of_cd_with_mach_number.png

B. Lift

Lift is a force that is perpendicular to oncoming flow that opposes the weight of the aircraft [6][7]. The force has various factors, but generally increases as the aircraft velocity increases, lift coefficient increases, and wing area increases.

III. Experiment Setup

The experiment is held with sqf codes, executed in Eden Editor. Test vehicles selected were FIR F-16, FIR F-15 and JSJC J/A-18, since addon aircrafts are much easier to browse config values. There were three experiments held. Experiment 1 is testing accelerations for 3 axes with given y-axis model velocity. Experiment 2 is calculating lift with expected coefficient and confirm those with comparing to measured z-axis acceleration. Experiment 3 is calculating vanilla drag and each drag components in AWESome aerodynamics for comparison.

1. Experiment 1

Experiment 1 was held with F-15 and F/A-18 each with newest version uploaded on Steam Workshop at Jan 01, 2019. The code used in this experiment works by placing the aircraft at a given point with given speed every iteration, then measuring acceleration for 0.2 seconds. Each iteration was executed 5 times and calculated the mean value, increasing the accuracy.

The code is as follows:

```
[] spawn {
	break = false;
	timeOld = time;
	velsum = 0;
	accsum = [0, 0, 0];
	testrun = 5;
	veltest = 100;
	testalt = 1000;
	targetval = 500;
	stepval = 1;
	(vehicle player) setPos [0, 0, testalt];
	(vehicle player) setVectorDirAndUp [[0,1,0],[0,0,1]];
	(vehicle player) setVelocity [0, 0, 0];
	sleep 1;
	while {(veltest < targetval) && !break} do {
		for "_i" from 1 to testrun do {
			(vehicle player) setPos [0, 0, testalt];
			(vehicle player) setVectorDirAndUp [[0,1,0],[0,0,1]];
			(vehicle player) setVelocity [veltest, 0, 0];
			timeOld = time;
			velOld = velocityModelSpace (vehicle player);
			sleep 0.2;
			velnow = velocityModelSpace (vehicle player);
			acc = (velnow vectorDiff velOld) vectorMultiply (1 / (time - timeOld));
			accsum = accsum vectorAdd acc;
			veldis = vectorMagnitude ((velnow vectorAdd velOld) vectorMultiply 0.5);
			velsum = velsum + veldis;
			systemChat str [_i, veldis, acc];
		};
		acc = accsum vectorMultiply (1 / testrun);
		accsum = [0, 0, 0];
		veldis = velsum / testrun;
		velsum = 0;
		systemChat format ["loopdone: [%1, %2, %3, %4, %5]", testalt, veldis, acc select 0,
acc select 1, acc select 2];
		diag_log str [testalt, veldis, acc select 0, acc select 1, acc select 2];
		veltest = veltest + stepval;
	};
};
```

2. Experiment 2

Experiment 2 was held with FIR F-15 and JSJC F/A-18. Measuring lift was done by measuring z-axis acceleration and compensating gravity, while calculating expected lift was done by using function in AWESome aerodynamics. The function can be viewed at:

https://github.com/mgkid3310/AWESome/blob/dc71ca9ebca6e48e53dffab4c328f7665530e9e6/addons/orbis_aerodynamics/scripts/fnc_extractCoefArray.sqf

And the code executed is as below:

```
[] spawn {
	break = false;
	timeOld = time;
	velsum = 0;
	accsum = [0, 0, 0];
	testrun = 5;
	veltest = 0;
	testalt = 1000;
	targetval = 500;
	stepval = 1;
	(vehicle player) setPos [0, 0, testalt];
	(vehicle player) setVectorDirAndUp [[0,1,0],[0,0,1]];
	(vehicle player) setVelocity [0, 0, 0];
	liftArray = getArray (configFile >> "CfgVehicles" >> (typeOf vehicle player) >> "envelope");
	maxSpeed = getNumber (configFile >> "CfgVehicles" >> (typeOf vehicle player) >> "maxSpeed");
	sleep 1;
	while {(veltest < targetval) && !break} do {
		for "_i" from 1 to testrun do {
			(vehicle player) setPos [0, 0, testalt];
			(vehicle player) setVectorDirAndUp [[0,1,0],[0,0,1]];
			(vehicle player) setVelocity [0, veltest, 0];
			timeOld = time;
			velOld = velocityModelSpace (vehicle player);
			sleep 0.2;
			velnow = velocityModelSpace (vehicle player);
			acc = (velnow vectorDiff velOld) vectorMultiply (1 / (time - timeOld));
			accsum = accsum vectorAdd acc;
```

```
                    veldis = vectorMagnitude ((velnow vectorAdd velOld) vectorMultiply 0.5);
                    velsum = velsum + veldis;
                };
                acc = accsum vectorMultiply (1 / testrun);
                accsum = [0, 0, 0];
                veldis = velsum / testrun;
                lift = acc select 2;
                predict = [liftArray, maxSpeed, 1.25 / (count liftArray - 1), veldis * 3.6] call
orbis_aerodynamics_fnc_extractCoefArray;
                velsum = 0;
                systemChat format ["loopdone: [%1, %2, %3, %4, %5]", testalt, veldis, lift, predict,
predict / lift];
                diag_log format ["loopdone: [%1, %2, %3, %4, %5]", testalt, veldis, lift, predict,
predict / lift];
                veltest = veltest + stepval;
        };
};
```

3. Experiment 3

Experiment 3 was held with FIR F-16 and JSJC F/A-18 by using log feature in AWESome addon. The code works similar to Experiment 1 but does not measure accelerations and gives AWESome addon a signal to log calculated values. Note that the vanilla drag is not measured but calculated with equations from Experiment 1. Refer the following link for calculating and logging of values:

https://github.com/mgkid3310/AWESome/blob/22e73105c6536b38cca9b2374d8fd7e8d3ce5ed3/addons/orbis_aerodynamics/scripts/fnc_getDragEnhanced.sqf

For the code used to set position/velocity and trigger AWESome to log, refer below:

```
[] spawn {
        break = false;
        testrun = 1;
        veltest = 100 / 3.6;
        testalt = 1000;
        targetval = 2500 / 3.6;
        stepval = 10 / 3.6;
        (vehicle player) setPos [0, 0, testalt];
```

```
		(vehicle player) setVectorDirAndUp [[0,1,0],[0,0,1]];
		(vehicle player) setVelocity [0, 0, 0];
		sleep 1;
		while {(veltest < targetval) && !break} do {
			for "_i" from 1 to testrun do {
				(vehicle player) setPos [0, 0, testalt];
				(vehicle player) setVectorDirAndUp [[0,1,0],[0,0,1]];
				(vehicle player) setVelocity [0, veltest, 0];
				AWESOME_DEVMODE_LOG = true;
				sleep 0.1;
				AWESOME_DEVMODE_LOG = false;
				systemChat format ["%1/%2", _i, testrun];
			};
			systemChat format ["loopdone: [%1, %2]", testalt, veltest * 3.6];
			veltest = veltest + stepval;
		};
};
```

IV. Results

1. Experiment 1

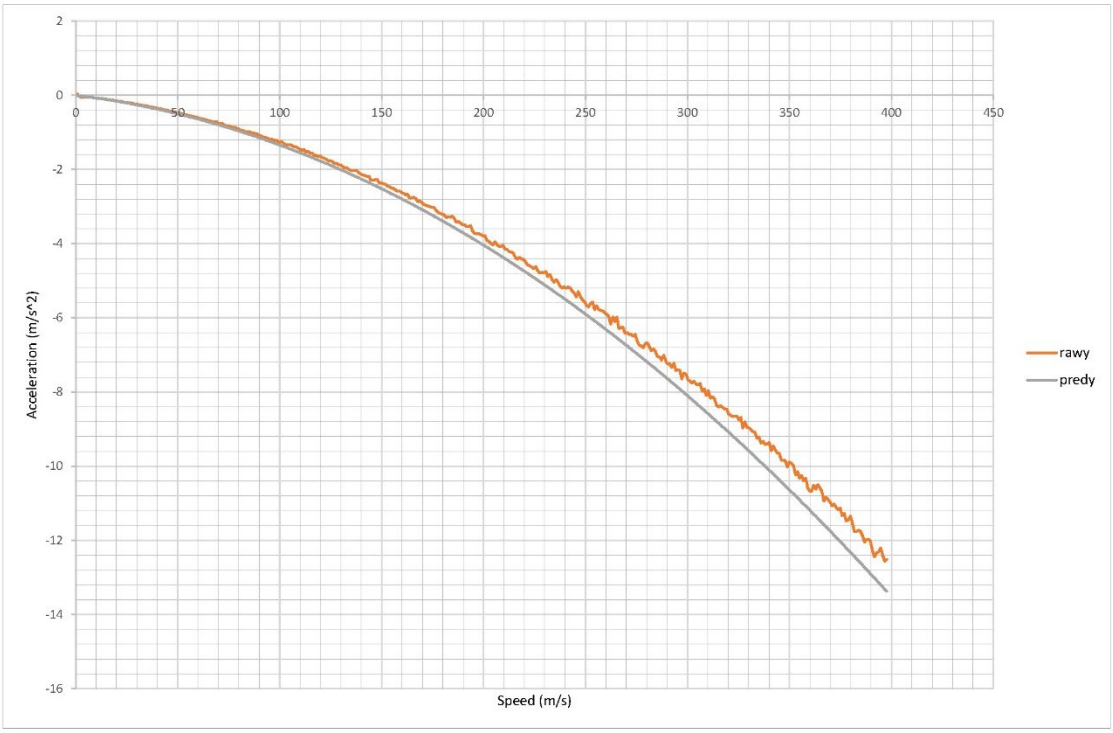

Fig. 2 FIR F-15 y-axis acceleration measured & predicted vs y-axis speed

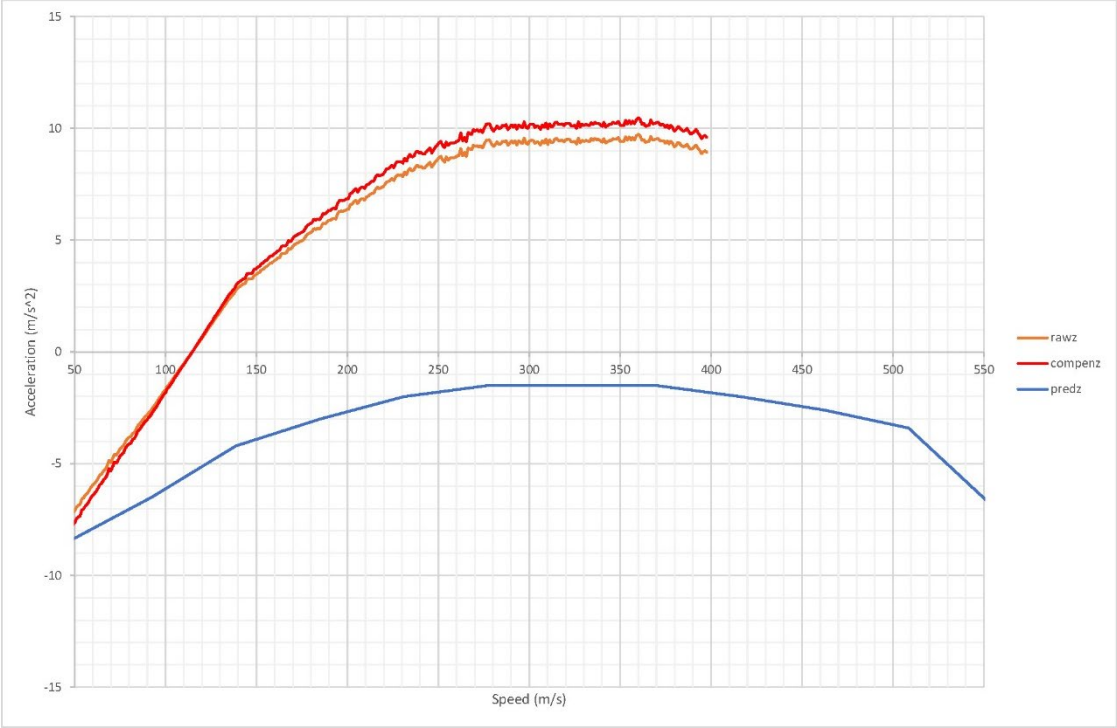

Fig. 3 FIR F-15 z-axis acceleration measured & predicted vs y-axis speed

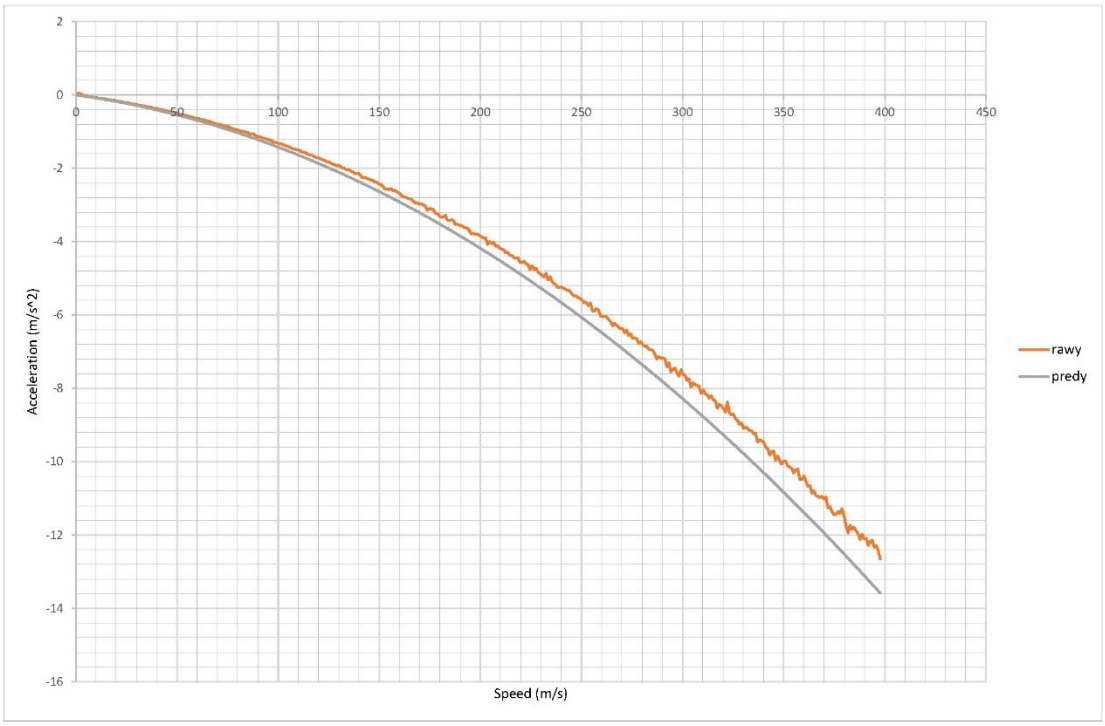

Fig. 4 JSJC F/A-18 y-axis acceleration measured & predicted vs y-axis speed

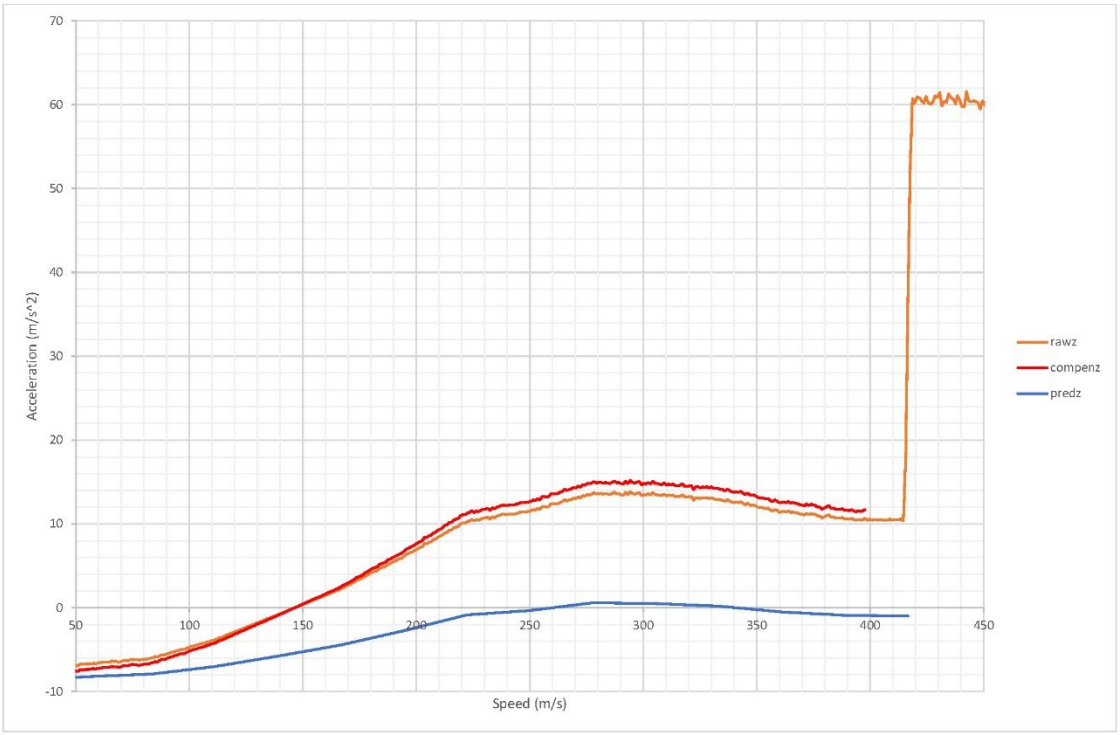

Fig. 5 JSJC F/A-18 z-axis acceleration measured & predicted vs y-axis speed

2. Experiment 2

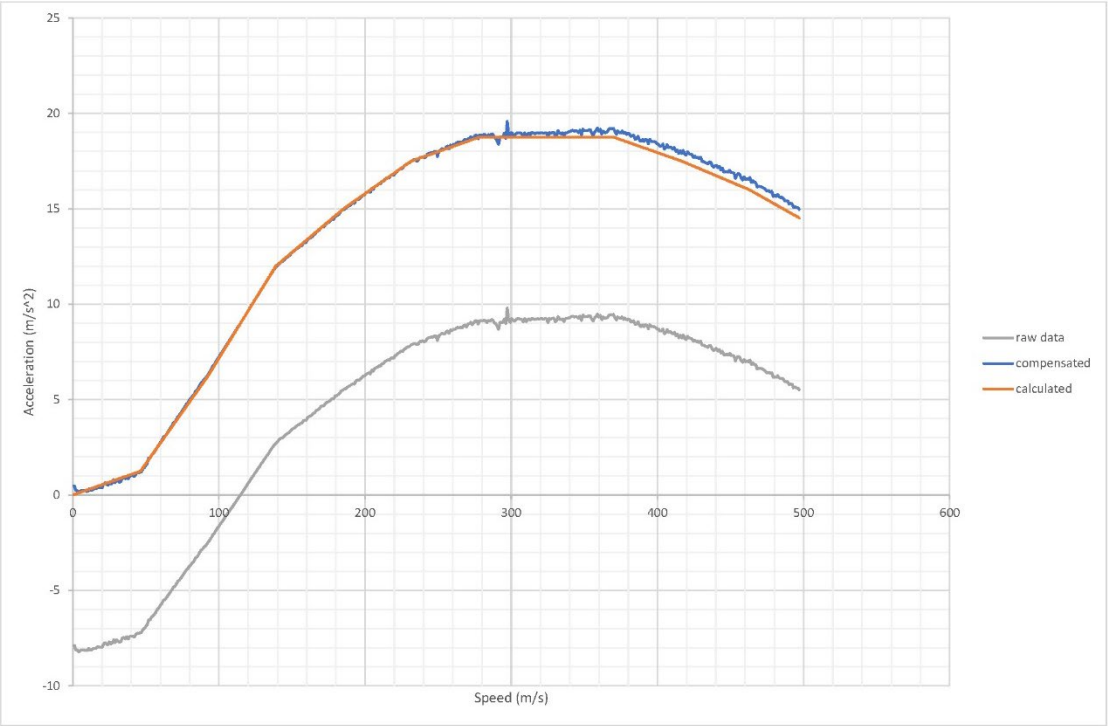

Fig. 6 FIR F-15 z-axis acceleration measured & calculated vs y-axis speed

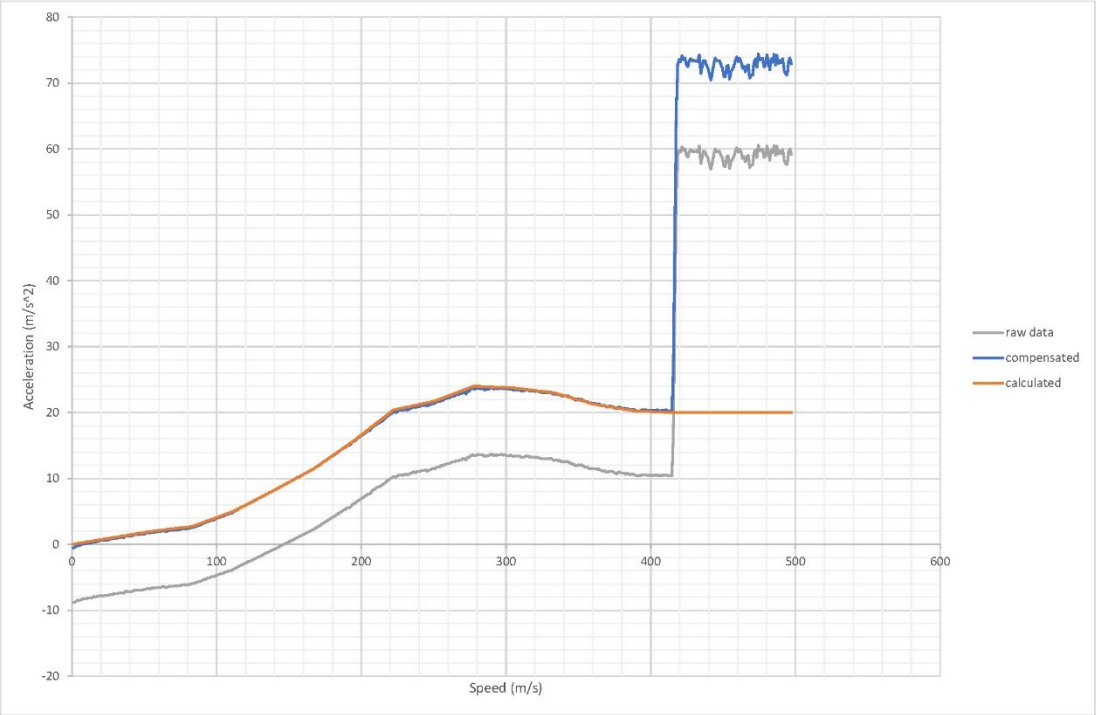

Fig. 7 JSJC F/A-18 z-axis acceleration measured & calculated vs y-axis speed

3. Experiment 3

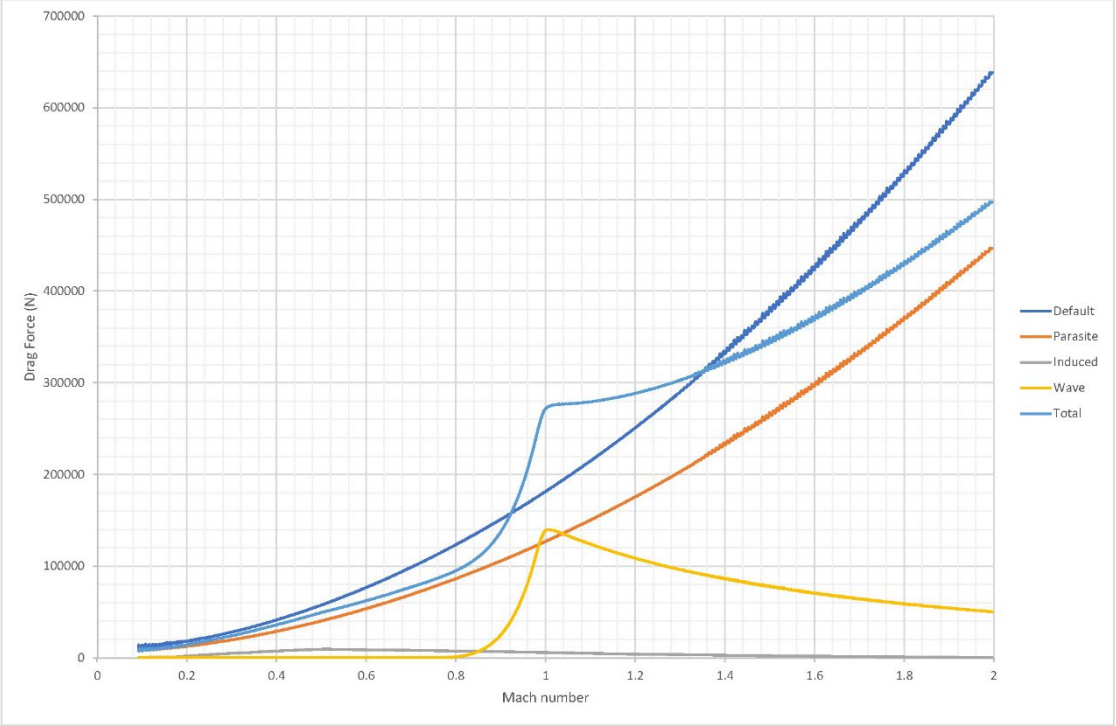

Fig. 8 FIR F-16 drag force vs Mach number

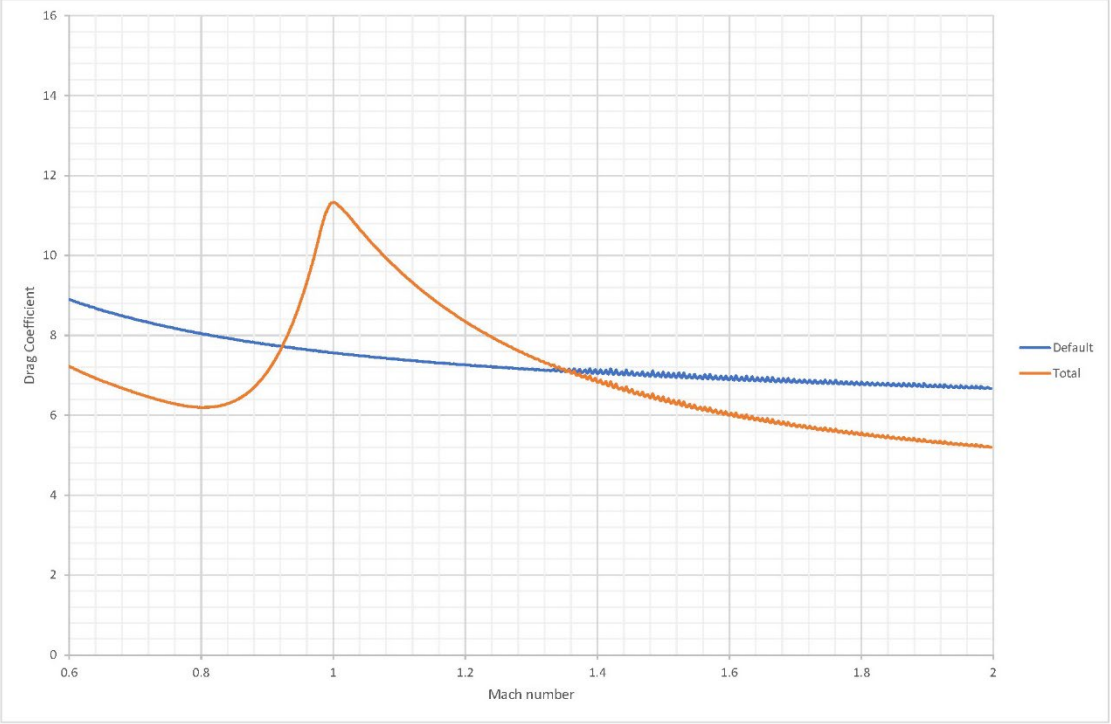

Fig. 9 FIR F-16 drag coefficient vs Mach number

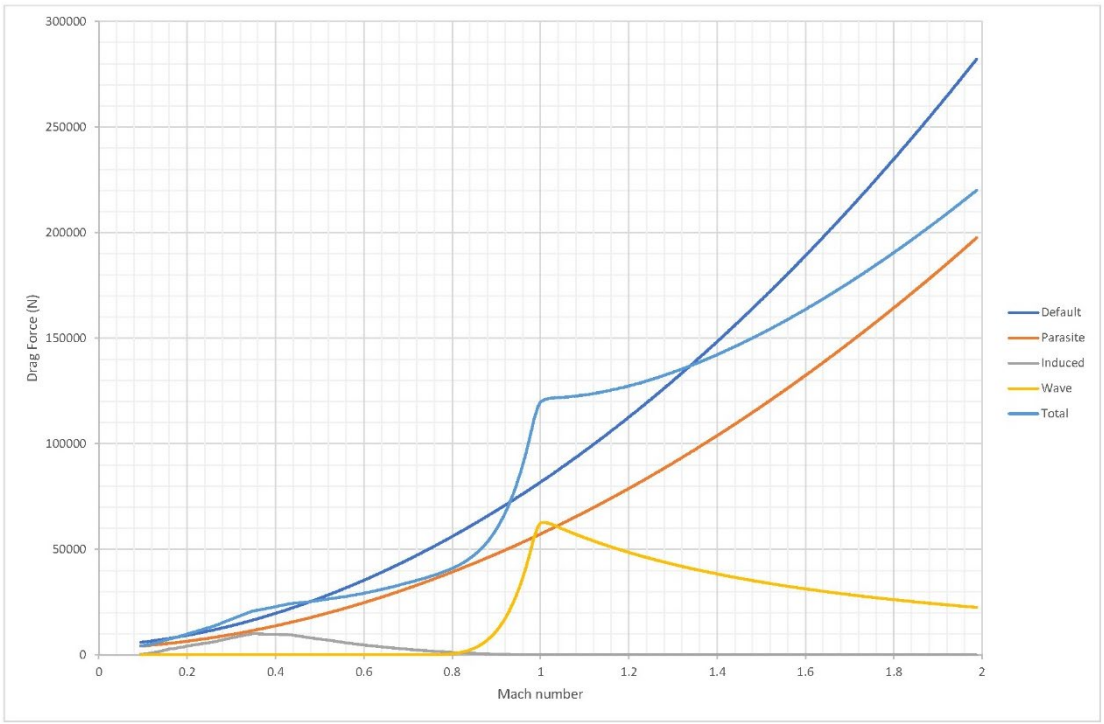

Fig. 10 JSJC F/A-18 drag force vs Mach number

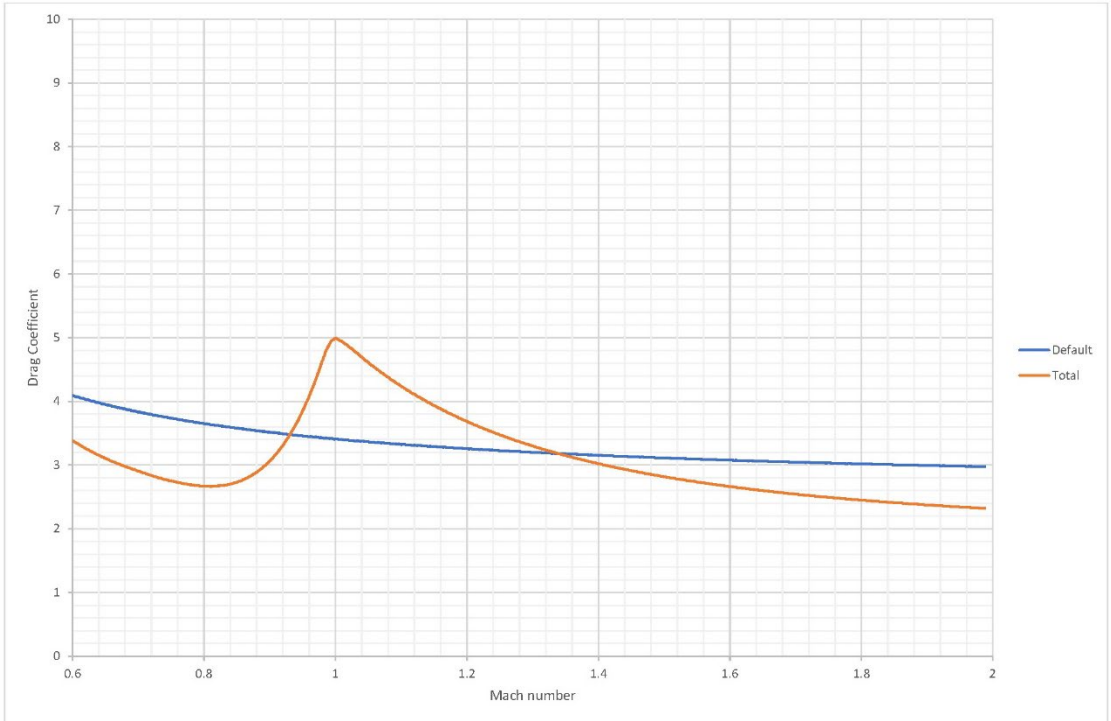

Fig. 11 JSJC F/A-18 drag coefficient vs Mach number

V. Conclusion and Discussions

1. Experiment 1

The predicted y-axis acceleration was calculated via following equation:

$$a_y = c_2 v_y|v| + c_1 v_y + c_0 \qquad \text{(Eq. 2)}$$

Where:

a_y: y-axis acceleration

v_y: y-axis component of speed vector

$|v|$: size of speed vector

c_2: y-axis component of airFrictionCoefs2 (3rd item), 0.000068 in this case

c_1: y-axis component of airFrictionCoefs1 (3rd item), 0.0066 in this case

c_0: y-axis component of airFrictionCoefs0 (3rd item), 0 in this case

From the Fig. 2, Fig. 4 and provided information [1], we can safely assume that the equation fits well and the ≈8% of difference between raw data and predicted value is caused by delay between code lines.

Fig. 3 and Fig. 5 shows z-axis acceleration, which is related to lift. The lift is calculated from current y-axis speed, angle of incidence, and envelope array, but since the angle of incidence value is small and can be ignored in the test vehicles, it was not included in the prediction for simplicity.

Lift calculation has conflicting information in two Bohemia Interactive pages [1][2]. To test which is right, the blue line was drawn. It represents lift calculated assuming that the first item represents lift at speed 0, and last item at 125% of maxSpeed and each step has same speed intervals. For easier comparison the gravity is assumed to be 9 and was subtracted from the prediction, matching the y intercept with measured data. While the overall trend fits, the predicted value is much smaller, meaning that there is an unknown multiplier for lift.

2. Experiment 2

From the Experiment 1, we know the presence of a multiplier for lift calculation. To find the right value, some works were done to raw data. 1.08 was multiplied to compensate the measurement error as found in Experiment 1, then added 9 to compensate gravity. The gravitational acceleration value is an approximation that matches measurement and

calculated value's y intercept at 0, implying that the gravitational acceleration in Arma 3 is 9, not 9.81.

After matching the y intercept at 0, the multiplier has been tested and is found to be 2.5. The calculated lift value assuming the multiplier to be 2.5 is shown at Fig. 6 and Fig. 7. By comparing the values with compensated values shown in blue, we can say that the multiplier is 2.5. The resulting mechanism for lift calculation can be found at the flowing link:

https://github.com/mgkid3310/AWESome/blob/dc71ca9ebca6e48e53dffab4c328f7665530e9e6/addons/orbis_aerodynamics/scripts/fnc_getLiftDefault.sqf

One strange behavior with the test result can be found at Fig. 7. Around the speed of 420$\mathrm{m/s}$, the lift increases abnormally, reaching over 70$\mathrm{m/s^2}$. Assuming from the fact that JSJC F/A-18 has maxSpeed config value of 1200$(\mathrm{km/h})$ and 125% of the value is equivalent to 415.7$\mathrm{m/s}$, exceeding 125% of the maxSpeed value is thought to be the reason.

3. Experiment 3

Note that this experiment was held before recent updates for FIR and JSJC aircrafts, so specific aerodynamic performance may be different from the newest version.

Fig. 8 and Fig. 10 shows comparisons between vanilla and AWESome aerodynamic drag. The dark blue lines represent vanilla drag and light blue lines represent AWESome aerodynamics drag. While both shows a trend of increasing value proportional to speed squared in a long term, AWESome aerodynamics show a bump near Mach 1, showing a similar effect to the sound barrier. The induced drag is mainly shown in low velocity area especially in Fig. 10, which is because of the large lift force for F/A-18 in low velocity. Comparing Fig. 1, Fig. 9 and Fig. 11, the noticeable hill near Mach 1 can be found on both real world's data and AWESome aerodynamics but not in vanilla.

Acknowledgments

I would like to express my deepest thank to Firewill and Nimitz Team for creating such great addons that were used as testbed. I also thank to all the developers at Bohemia Interactive for making a great game and a nice development environment, and all the addon developers for developing wonderful addons and creating a great society.

VI. References

[1] Bohemia Interactive. (n.d.). CfgVehicles Config Reference. Retrieved January 02, 2019, from https://community.bistudio.com/wiki/CfgVehicles_Config_Reference

[2] Bohemia Interactive. (n.d.). Arma 3 CfgVehicles Plane class config reference. Retrieved January 02, 2019, from https://community.bistudio.com/wiki/Arma_3_CfgVehicles_Plane_class_config_reference

[3] Park, M. (2018, February 15). AWESome : Aerial Warfare Enhanced Somehow. Retrieved January 02, 2019, from https://steamcommunity.com/sharedfiles/filedetails/?id=1301399507

[4] National Aeronautics and Space Administration. (2015, May 5). What is Drag?. Retrieved Jan 02, 2019, from https://www.grc.nasa.gov/www/k-12/airplane/drag1.html

[5] Wikipedia. (2006, May 30). Drag (physics). Retrieved January 02, 2019, from https://en.wikipedia.org/wiki/Drag_(physics)

[6] National Aeronautics and Space Administration. (2015, May 5). What is Lift?. Retrieved Jan 02, 2019, from https://www.grc.nasa.gov/WWW/K-12/airplane/lift1.html

[7] Wikipedia. (2004, August 01). Lift (force). Retrieved January 02, 2019, from https://en.wikipedia.org/wiki/Lift_(force)