Table of Contents

[Introduction 3](#_Toc445926343)

[Workflow 3](#_Toc445926344)

[Important Structures 4](#_Toc445926345)

[How to create an aircraft object (main.m) 5](#_Toc445926346)

[How to do Monte Carlo Iteration (MCmain.m) 6](#_Toc445926347)

[Trim Aircraft (elevatorToTrim) 7](#_Toc445926348)

[Let Aircraft Cruise (cruise) 7](#_Toc445926349)

[Aircraft and Components 8](#_Toc445926350)

[Aircraft (Aircraft) 8](#_Toc445926351)

[Fuselage (Fuselage) 9](#_Toc445926352)

[Wing (Wing) 10](#_Toc445926353)

[Horizontal Tail (HTail) 11](#_Toc445926354)

[Vertical Tail (VTail) 12](#_Toc445926355)

[Engine (Engine) 13](#_Toc445926356)

[Fuel System (FuelSys) 14](#_Toc445926357)

[Helper Classes, Functions, and Methods 15](#_Toc445926358)

[Graph Important Graphs (GraphTool) 15](#_Toc445926359)

[Get drag coefficient (getCd) 17](#_Toc445926360)

[Pickup Moment Test (pickupTest) 18](#_Toc445926361)

[Test Aircraft and Get Performance (windTunnel) 18](#_Toc445926362)

[Draw Aircraft Diagram (WorkShop) 19](#_Toc445926363)

[Sort the sample collection (sortByFied) 20](#_Toc445926364)

[Show statics of result (showStatsOf) 20](#_Toc445926365)

[Show Trend of result (ShowTrendOf) 21](#_Toc445926366)

[Get Stability Derivatives of Aircraft (getDev) 22](#_Toc445926367)

# Introduction

## Workflow

The following diagram gives the high level description of the workflow of the whole program.

Define Design Specs

Define World Constants

Build each parts of the aircraft and build aircraft using these components and layout parameters

Output result

Put the aircraft into “wind tunnel” and combine the test result with aircraft object to construct a “sample”

Sort samples into two groups based on whether the result meet the specs

Monte Carlo iteration if applicable

## Important Structures

* WorldConstants

WorldConstants is a structure contains all physical properties from the real world. It is defined in main file and widely used throughout the whole program. It specifies:

AirDensity [slug/ft^3]

kinematicViscosity [ft^2/s]

airSpecificHeatRatio [Null]

airSpecificGasConstant [ft-lb/slug R]

airTemperature [R]

* Spec

Spec specifies all design requirement. It is defined in main file and used in graphing and checking if the random generated result meet the design specs. It specifies:

StallSpeed [ft/s]  
 Range [ft]  
 Endurance [s]  
 Payload [lbm]  
 ClimbRate [ft/s]  
 CruisingSpeed [ft/s]

* Sample

Sample is a structure contains an aircraft object and result from wind tunnel of that aircraft. The output format of the Monte Carlo iteration are two collections of samples. And samples are used in drawing aircraft diagram and showing statics of Monte Carlo results.

This is sample code to construct a sample structure

>> sample = struct('airCraft',testUAV,'result',windTunnel(testUAV,WorldConstants,Spec));

## How to create an aircraft object (main.m)

Main.m is the file that generate an aircraft. It consists of 5 sections of code. Creating aircraft object is just simply setting all parameters in each section and run main.m. The generated aircraft object will be named as ”testUAV”. Here is the list and brief description of each section of the code.

1. Design spec: specify the design requirements.
2. World Constants: specify the testing condition (air temperature, air density, etc.)

=========   
part 1 and 2 are not necessary for creating an aircraft object but are needed for testing the aircraft  
=========

1. Building components: the specification of each aircraft components (fuselage length, wing span, etc.) are set here.
2. Set aircraft state: the initial angle of attack, altitude, air speed, and payload are set in this section.
3. Aircraft layout: four important ratios are set in this section. They are :

FuselageNosetoWingFrontRatio  
TailtoFuselageFrontRatio  
FueltoFuselageFrontRatio   
PayloadtoFuselageFrontRatio

After setting all parameters, run main.m. “testUAV” is the aircraft object created based on the given parameters.

## Get Our Best Aircraft

Mainbest.m is composed based on the best result from our Monte Carlo iteration. Simply run it. The “testUAV” object is the best aircraft we get. It was also the aircraft we used in our further analysis.

## How to do Monte Carlo Iteration (MCmain.m)

MCmain.m is the file that can repeatedly generate aircraft with different initial parameters. It also tests each of the aircrafts and sorts them by whether the aircraft meet the mission specs. All parameters can be randomized in each iteration by using function MCrandom(var, [range]).

MCrandom(var, [range]) is a function that evenly returns a value in the range of var \* (1±range). If the range parameter is missing, the default value of range is assigned to be 0.3.

For example, the code randomize fuselage length for each iteration around 20 ft in the range of 30% will be:

L = MCrandom(21); or L = MCrandom(21, 0.3)

After setting all the parameters that need to be randomized, the parameter “iterationN”, defined at the beginning of the code, need to be set. It is the number of iteration.

Finally, run the code. All the randomized created aircrafts will be sorted into two lists – good and failed. As suggested by name, aircrafts in “good” list are the samples (aircraft + result) that meet all design specs. “Failed” list contains samples that don’t meet at least one spec. “failed” list also records the on which spec the aircraft fails.

## Trim Aircraft (elevatorToTrim)

The method “elevatorToTrim” is used to get the angle of attack and the elevator angle for a trimmed aircraft at a certain speed. It is defined under “Aircraft” class

Prototype

function [AoA, eleAngle] = elevatorToTrim(ac,WorldConstants)

Sample code

Adjust the angle of attach and elevator deflection of “testUAV” to the trim values.

[testUAV.AoA, testUAV.hTail.de] = testUAV.elevatorToTrim(WorldConstants);

## Let Aircraft Cruise (cruise)

Defined in “Aircraft” class, cruise function takes cruise speed specified in “Spec” and returns velocity, angle of attack, elevator angle, and throttle of cruising.

Prototype

function [v, AoA, eleAngle, throttle] = cruise(ac, WorldConstants, Spec)

Sample Code

Adjust aircraft velocity, angle of attach, elevator deflection, and throttle of “testUAV” to the cruising values.

>> [testUAV.v, testUAV.AoA, testUAV.hTail.de, testUAV.engine.throttle] = testUAV.cruise(…

WorldConstants, …

Spec);

# Aircraft and Components

## Aircraft (Aircraft)

Aircraft is an object contains all details of one aircraft. It contains component objects that make up an aircraft as well as the layout, flight state (velocity, altitude, angle of attack) of an aircraft.

% Components

W\_to; % this property is solely for weight estimation

fuselage;

wing;

fuelSys;

engine;

hTail;

vTail;

% state

AoA ; % Angle of attack [degree]

alt; % current altitude

v; % current airspeed

payloadWeight; % current payload weight

% Layout

% choose the leading edge of wing as the reference point

% every coefficient is the distance between the reference point and

% the CG or AC of the components.

h\_acw; % AC&CG of wing

h\_act; % AC&CG of horizontal tail

h\_acvt; % AC&CG of vertical tail

h\_fuselage; % CG of fuselage

h\_engine; % CG of engine

h\_fuelSys; % CG of fuelSys

h\_payload; % CG of payload

% Methods

getWeight; % get overall aircraft Weight

getLw; % wing AC to CG

getLt; % tail AC to CG

get\_hcg; % get cg of the aircraft

get\_hn % get neutral point

get\_hcgEmpty; % get cg of the aircraft when 0 payload

get\_hnEmpty; % get neutral point when 0 payload

get\_margin; % get static margin

get\_hcgToNose; % get the normalized distance from h\_cg to nose

get\_hcgToTail; % get the normalized distance from h\_cg to tail

getHv; % tail volume parameter

getVv; % tail volume ratio

getCl % get lift coefficient

getBaldeAngle; % get blade angle of propeller

## Fuselage (Fuselage)

Fuselage is a component class. It contains all properties of the fuselage.

properties

W\_to; % take-off weight [lbm]

N = 6.6; % ultimate load factor

L; % length (ft)

D; % max diameter (ft)

Ve = 200; % equivalent max airspeed [mph]

getWeight; % get fuselage weight [lbm]

getLength; % get fuselage length [ft]

getRadius; % get fuselage radius [ft]

getSwet; % get wetted area [ft^2]

## Wing (Wing)

Wing is a component class. It contains all wing properties and provide some wing-related values.

% inputs

b; % span length

cla; % 2D cl alpha

c\_mac; % 2D C\_mac

s; % wing area

e; % Oswald efficiency number

W\_to; % take-off weight

Cdminw; % minimum drag coefficient of the airfoil, see drag paper[p18-top] for detail

N = 6.6; % ultimate load factor

stallAngle; % Stall Angle

QuaChordSweep = 0 % Quarter Chord Sweep [rad]

Taper = 0.4 % Taper Ratio

Ve = 141; % equivalent max airspeed

tcRatio; % maximum thickness of wing over chord (t/c)m

xcRatio; % chordwise location of the airfoil maximum thickness point [Raymer 283]

% Computed Properties

getChordRoot; % root chord length [ft]

getChord; % get average chord length [ft]

getAR; % get aspect ratio

get2DCla; % get 2D lift slope (same value as cla)

get3DCla; % get 3D lift slope

getWeight; % get wing weight [lbm]

getSwet; % get wetted area [ft^2]

get3DCmac; % get 3D Cmac (same as 2D value)[ft-lbf]

getDownwashRate; % get the downwash effect of the wing to the tail

## Horizontal Tail (HTail)

This is a component class. It contains all horizontal tail properties as well as some horizontal-tail-related properties.

% inputs

b; % tail span

cla; % 2D cl alpha

s; % wing area

e; % Oswald efficiency number

it; % incidence angle

tau; % elevator area persentage

W\_to; % take-off weight

N = 6.6; % ultimate load factor

lt = 21; % Distance from wing 1/4 MAC to tail 1/4 MAC

QuaChordSweep = 0; % Quarter Chord Sweep

Taper = 1; % Taper Ratio

tcRatio; % maximum thickness of wing over chord (t/c)m

xcRatio; % chord wise location of the airfoil maximum thickness point [Raymer 283]

Ve = 200; % equivalent max airspeed [ft/s]

max\_de; % max abs of elevator angle [rad]

% Computed Properties

getAR; % get aspect ratio

get2DCla; % get 2D lift coefficient slope

get3DCla; % get 3D lift coefficient slope

getWeight; % get horizontal tail weight [lbm]

getSwet; % get horizontal tail wetted area [ft^2]

getChordRoot; % get root chord [ft]

getChord; % get average chord [ft]

## Vertical Tail (VTail)

This is a component class. It contains all vertical tail properties as well as some vertical-tail-related properties.

s; % area in ft^2

b; % span in ft

e = 1; % Oswald efficiency number

cla = 6.5; % 2D lift coefficient slop

tcRatio; % maximum thickness of wing over chord (t/c)m

W\_to; % take-off weight

N = 6.6; % ultimate load factor

Taper = 1; % taper ratio

xcRatio; % chordwise location of the airfoil maximum thickness point [Raymer 283]

tau = 0.3 % rudder area persentage

% Computed Properties

getWeight; % get weight [lbm]

getSwet; % get wetted area [ft^2]

getChordRoot; % get root chord length [ft]

getChord; % get average chord length [ft]

get3DCla; % get 3D lift coefficient slop

getAR; % get aspect ratio

getSidewashRate; % similar to downwash, estimated based on NACA TN 3356 pg29

getZ; % the distance of the vertical tail aerodynamic center above the vehicle center of mass [ft]

## Engine (Engine)

This is a component class. It contains all engine properties as well as some engine-dependent properties. Propeller properties and propeller-dependent properties are also specified in this class.

EngineWeight = 140.8; % engine Weight

susPower = 55000; % sustained Power

maxPower = 64250; % max power

maxRPM = 5800; % max RPM

cp; % specific fuel consumption [lb/(lb\*ft/s \* s) = 1/ft]

propD; % propeller Diameter [ft]

N = 2; % number of engine

throttle = 0.5 % default throttle

% computed properties

getWeight; % get engine weight

% digitized graphs

powerVsRPM; % power Vs RPM data. [hp];[RPM]

FVRVsRPM; % fuel Volume Rate Vs RMP data. [US gal/h];[RPM]

propEffVsPropAoA; % propeller efficiency vs propeller aoa

bladeAngleVsPropAoA; % blade angle vs propeller aoa

% Methods

function getRPMbyPower(En,power) % power [lbf.ft/s]

function getPowerByRPM(En,RPM) % power [lbf.ft/s]

function getThrottleByRPM(En, rpm)

function getRPMByThrottle(En,throttle)

function getFuelVolumeRateByRPM(En,rpm)

% Property getter

function get.cp(En)

* There are some useful methods in this class which can help to get current engine state. The name of each method are self-explained. The reason for having these method is to find any engine running parameters for given throttle. In other words, once throttle is fix, all other engine running parameters (rpm, power, fuel volume flow rate)are therefore fixed.
* Cp is the only computed property that does not start with “get” due to historical reason. The implementation is shown here to 1) show our mechanism of finding specific fuel consumption 2) demonstrate how to use the methods mentioned above

function cp = get.cp(En)

gasolineDensity = 6.073; % [lb/US Gal]

fuelMassRate = En.getFuelVolumeRateByRPM(En.getRPMByThrottle(En.throttle))\*gasolineDensity/3600; % [lb/s]

cp = fuelMassRate/En.getPowerByRPM(En.getRPMByThrottle(En.throttle));

end

## Fuel System (FuelSys)

This is a component class. It contains all fuel system properties as well as some fuel-system-dependent properties.

fuelWeight; % current fuel weight

fuelCapacity; % fuel carrying capacity

int = 1; % percent of fuel tanks that are integral

% computed properties

getFuelVolume; % get volume needed for fuel tank

getWeight; % get overall weigth of fuel system

# Helper Classes, Functions, and Methods

## Graph Important Graphs (GraphTool)

GraphTool is a drawing class. This class take aircraft as a parameter and draw different graphs of the aircraft. The graphs it draws are:

* Trim Angle of Attack / Elevator deflation Vs. speed
* Required power Vs. speed
* Cl/Cd Vs. speed & Cl/Cd Vs. AoA
* Drag force Vs. speed

Example Code and Graphs

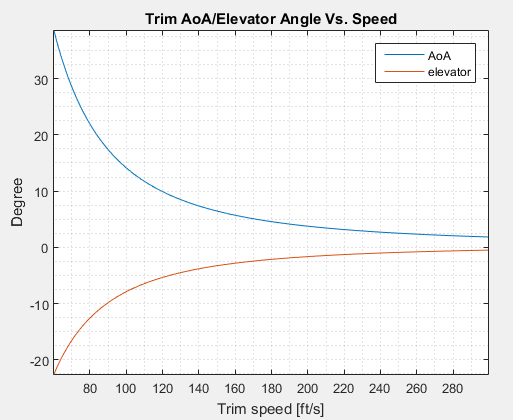
1. Construct a graphing class

gt = GraphTool(testUAV,WorldConstants,Spec,60.2:300);

The first parameter is aircraft object. The second and the third are world constants and spec, respectively. The fourth parameter is the speed range of the aircraft. Usually we use the stall speed as the bottom speed and a speed larger than cruising speed as the top speed.

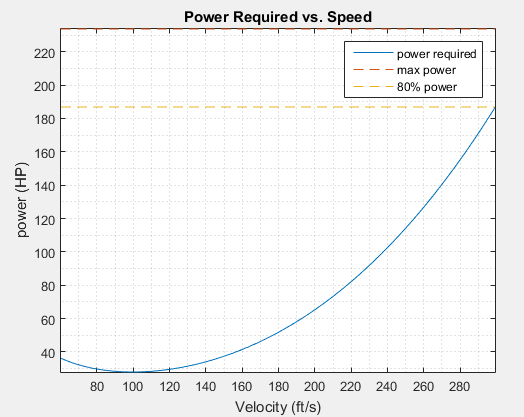
1. Draw Trim AoA/Elevator deflection graph

>> gt.drawTrimVsVelocity

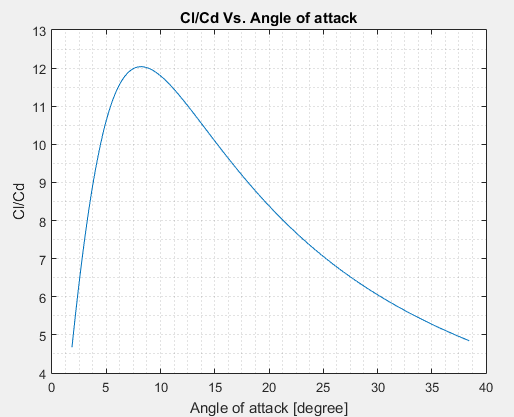


1. Draw required power graph

>> gt.drawPowerVsVelocity

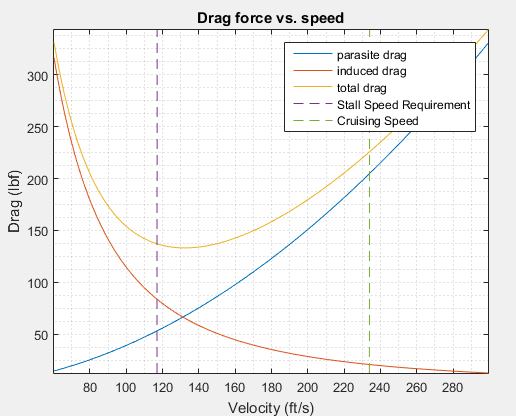


1. Draw Cd/Cl graphs (this will draw two graphs)

1. Draw Drag force graph

>> gt.drawDragVsVelocity



## Get drag coefficient (getCd)

This function can be used in two ways.

1. Get parasite drag coefficient (CD\_p), induced drag coefficient (CD\_i), and overall drag (CD) coefficient.
2. Get drag-related stability derivatives (CD0, CD\_a, CD\_de)

The prototype of the function is shown:

1. function [getCdp,getCdi,getCd] = getCd( AirCraft,WorldConstants)
2. function [CD0,CD\_a,CD\_de] = getCd( AirCraft,WorldConstants, [getDerivative])

* inputs: AirCraft: the aircraft object that need to be tested.

WorldConstants: structure contains all testing condition

getDerivative: indicates whether the function is used to get stability derivatives.

* Notice, getDerivative is an optional parameter, the input value of getDerivative does not affect the final result. When the function detects it has 2 parameters in inputs, it will switch to mode 1 and calculate drag coefficients. If the function detects it has 3 parameters in inputs, it will switch to mode 2 and generate stability derivatives.
* Example code
* 1. Get drag polar: [cdp,cdi,cd]=getCd(AirCraft,WorldConstants);

.2. Get stability derivatives: [CD0,CD\_a,CD\_de] = getCd(ac,WorldConstants,1);

## Pickup Moment Test (pickupTest)

Due to the specialty of the mission, the aircraft should be able to pick up the payload while flying. Picking up stationary payload will apply a large pitching-down moment onto the aircraft and therefore significantly affect the stability of the aircraft. This function returns the difference between the maximum pitching-up moment generated by the aircraft and the instantaneous moment applied on the aircraft. If the result is a positive number, then the aircraft has that amount of moment excess, which means it can recover from the disturbance. If the result is a negative number, it means the aircraft cannot recover from the disturbance.

* Example code:

pickupTest(testUAV, WorldConstants);

## Test Aircraft and Get Performance (windTunnel)

This function test UAV’s performance and returns the test result. The result is a structure contains the following fields:

* StallSpeed [ft/s]
* Range [ft]
* Endurance [s]
* Payload [lbm]
* ClimbRate [ft/s]
* PickupMoment [ft-lbf]
* Weight [lbm]

Note: “Payload” is the payload weight applied during the test, “Weight” is the sum of aircraft weight and payload weight.

Example Code:

1. Test UAV

result = windTunnel( testUAV, WorldConstants, Spec)

1. Access result

result.StallSpeed (get stall speed)

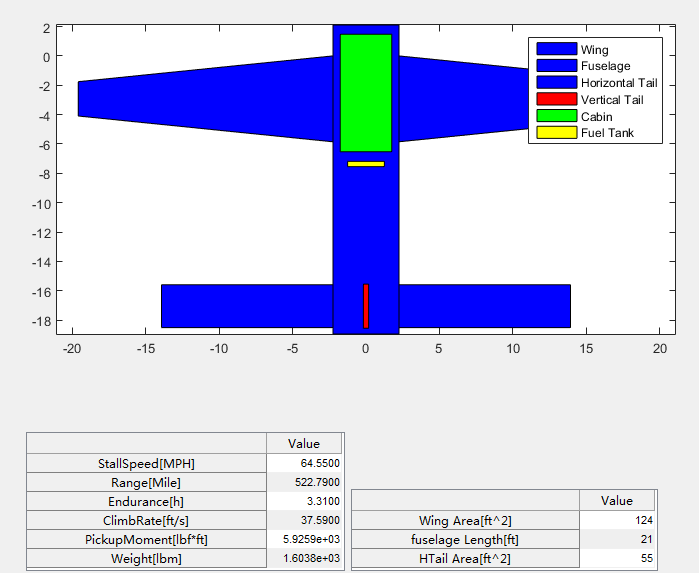
result.Range (get range)

## Draw Aircraft Diagram (WorkShop)

This is a function which take a “sample” as a parameter and draw a diagram of the layout of aircraft.

Example code and Graph

>> WorkShop(sampleAircraft);



## Sort the sample collection (sortByFied)

After Monte Carlo iteration, the program returns two collections of sample. In order to further analysis the result, each collection need to be sorted by a given field in the result. SortByField is design to fulfill that propose. It takes a sample collection and returns the sorted sample collection.

Example code

>> goodAircrafts = sortByField(goodAircrafts,'Weight')

The collection can be sorted by any field as long as the field name is in the result of the sample.

## Show statics of result (showStatsOf)

This function takes a sample collection and a property name as parameters and draws a histogram of the sample collection of a given property.

showStatsOf( collection, field, isInReport )

the first parameter is sample collection, the second parameter is property name, and the last parameter is a Boolean value indicating whether this property is in result (1) or in aircraft (other than 1).

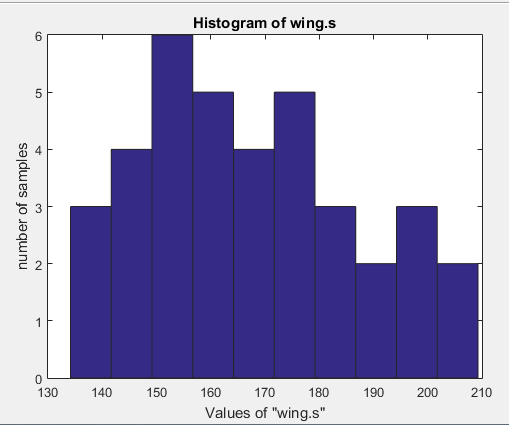
Example code and graph

1. Show histogram of ‘StallSpeed” in an aircraft collection (in result)

>> showStatsOf(goodAircrafts,'StallSpeed',1);

1. Show histogram of wing area in an aircraft collection (not in result)

>> showStatsOf(goodAircrafts,'wing.s',0);



## Show Trend of result (ShowTrendOf)

This function takes a sample collection, property, and percentage (optional) as parameters. It draws a graph of the trend of the given property in the sample collection.

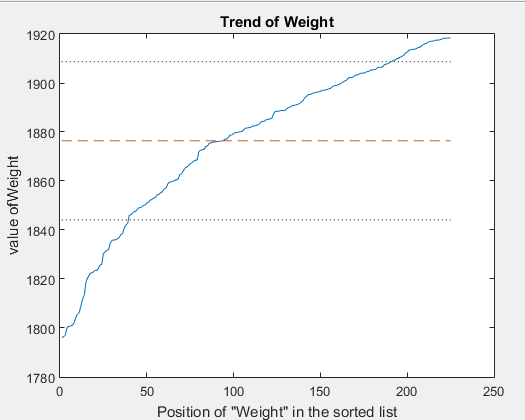
showTrendOf( collection, field, isInReport, topPercentage )

the first parameter is sample collection, the second parameter is property name, and the third parameter is a Boolean value indicating whether this property is in result (1) or in aircraft (other than 1). The last parameter is optional. It defines the range of which the sample collection that the function will draw. For example, if topPercentage = 0.5, the function will only consider the first 50% samples in the sample collection and draw the trend. If leave blank, the default value is 0.3.

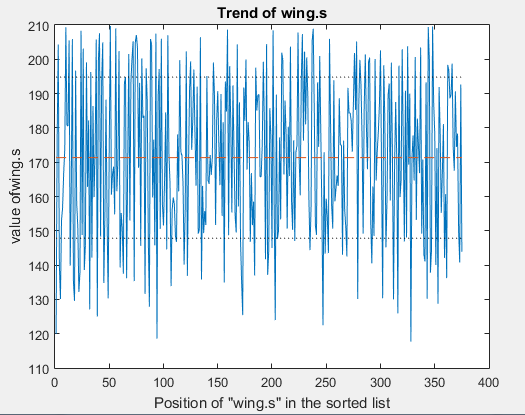
Sample code and graph

>> showTrendOf(good,'Weight',1)

The solid red line in the graph is the average value of the result. The two dash lines are the values of average plus/minus standard deviation.



>> showTrendOf(good,'wing.s',0,0.5);



## Get Stability Derivatives of Aircraft (getDev)

This function takes an aircraft object as a parameter and returns a structure of stability derivatives.

Sample code and result

>> getDev(testUAV,WorldConstants)

ans =

CL0: 0.0369

CL\_a: 6.1005

CL\_q: 13.3094

CL\_adot: 5.4337

CL\_de: 0.7396

CD0: 0.0319

CD\_a: 0.1593

CD\_de: 0.0193

Cm0: 0.0161

Cm\_a: -1.4007

Cm\_adot: -17.1108

Cm\_q: -41.9118

Cm\_de: -2.3292

CY\_beta: -0.4867

CY\_dr: 0.1267

CY\_r: 0.2776

CY\_p: -0.0297

Cl\_beta: -0.0672

Cl\_r: 0.0291

Cl\_p: -0.6929

Cl\_dr: 0

Cl\_da: 0.2100

Cn\_beta: 0.0127

Cn\_r: -0.1338

Cn\_p: 0.0195

Cn\_da: -0.0105

Cn\_dr: -0.0574