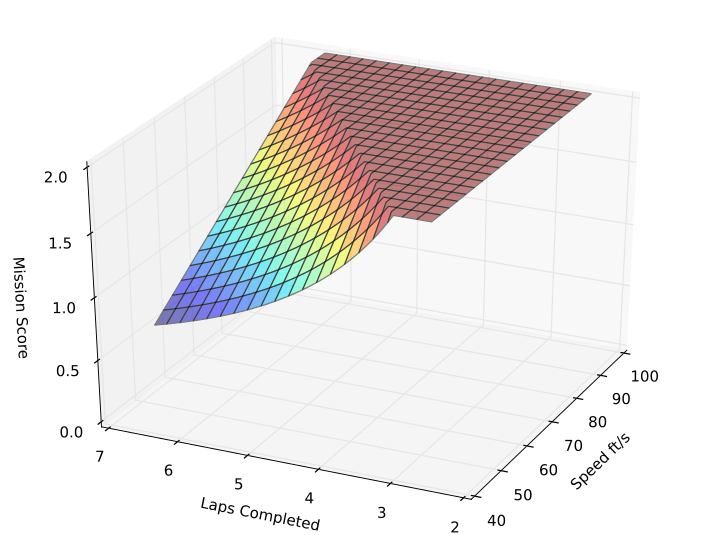
# Appendix A

## Score Optimization

### Mission One

Mission Objective

For the first mission the task is piloting the aircraft for the largest number of laps in a 4 minute span.   
Scoring



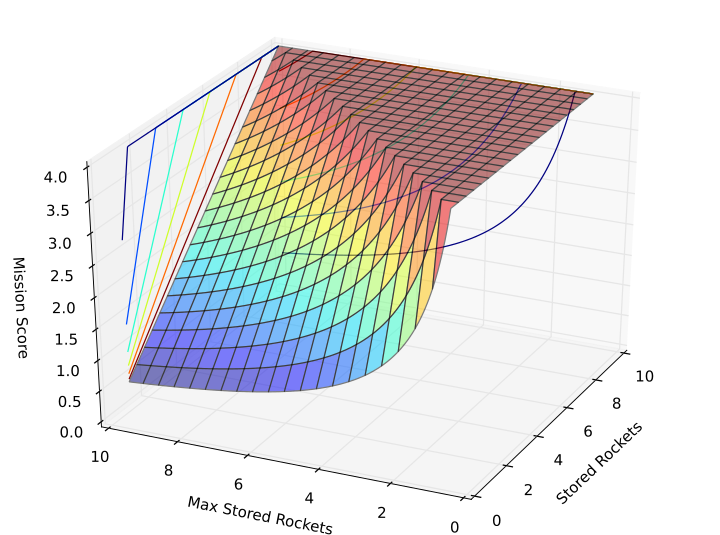
The DBF rules clearly show the length of each lap to be 3800ft, to get a score at least one lap must be completed. This leads to the lower limit of speed we must achieve, 3800/(4\*60) = 15.8 ft/s. Afterwards choosing an upper bound of 100 ft/s as a good range to explore. Using this formula plotting a graph with all the factor is trivial and then using it do pick a optimal speed for our project.  
  
The plot clearly shows three regions a slope, a ridge, and a plateau each of which is important to be interpreted. The slope highlights the region that the design would be suboptimal; this of course is where the project expects to be because in the plot the assumption of an ideal scenario is used. The plateau region is the area where there is overdesign. Overdesign is bad, due to waste of resources, space, and time. The last region is the ridge; this region is where the slope meets the plateau. This line is very important to any design; this ridge is where the best choice should be when picking a motor for our purposes.

Mission Two

Mission Objective

In mission two the goal is to carry as many mini-max rockets as possible and . These rockets each weight 1.3 ounces, this is a test of lift vs payload.

Scoring

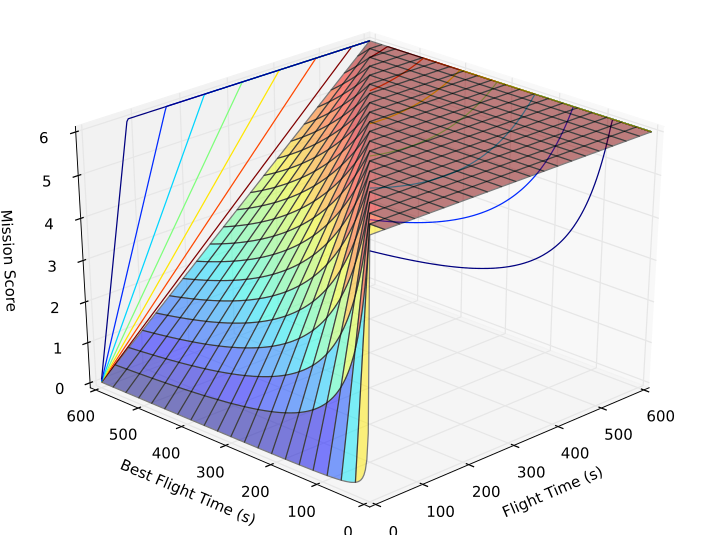


From the plot, it clearly demonstrates that aiming for a range of 2 to 10 rockets is preferable. The primary factor is not weight of the rockets, since the rockets weight 1.3 ounces each. The rockets shape are the issue, each rocket has a shaft of 1 in and 1.5 in fins on 4 sides of the thrust exit. Thus each rocket takes a span of 4 inches, for a drone this is a lot of space.

### Mission Three

Mission Objective

This Mission is to carry a random payload of rockets in a configuration decided on the day of the competition.  
Score

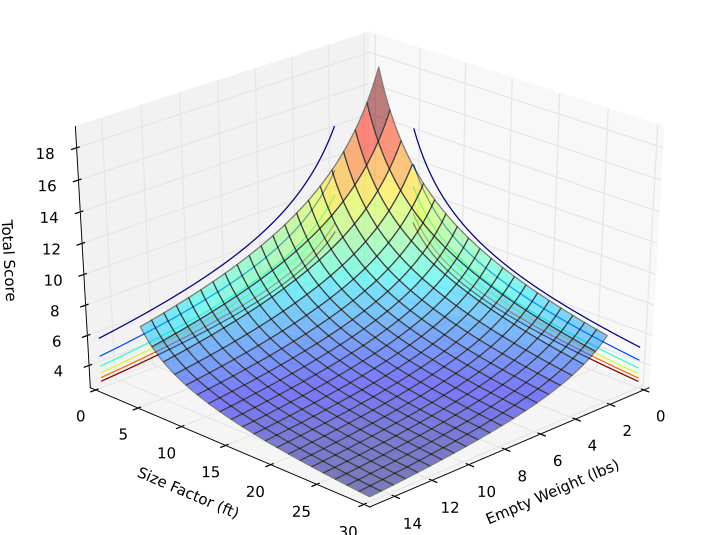


Also notice that the fixed value in the scoring is in the numerator this time, thus the fastest time flown is dependant on the score of all teams. This mission is very random, and thus hard to optimize for. This mission has each team carry a random load of rockets, the load will be decided on the day of the competition through a roll of a die. Each number has a specific configuration that we must be capable of carrying. Thus choosing a design parameter to use for mission three is hard. However plotting this score is still very useful because it illustrates how the times will affect our final score.

### Total Score Plot

Scoring:

Where :



The last plot that is included is the how the the size factor and the empty weight will affect the total final score. The final score is computed using Score = Total Mission Scores / Rated Aircraft Cost (RAC); where the RAC = sqrt(Empty Weight, Size Factor) / 10. For this plot it was necessary to assume a few values for certain parameters to achieve a plot. The values assumed are a total mission score of 6 out of a possible 12 points.

The plot illustrates how the sizing of the airplane affects the overall score. Notice that the plot does not plateau like the previous score, this means that there is always room to improve on our score by reducing the weight and reducing the size of the aircraft.

# Python Code For Score Optimization

"""

KEY

-----

WS Written Report Score

TFS Total Flight Score

RAC Rated Aircraft Cost

SF Size Factor

EWn Empty Weight n-th Flight

XMAX Longest possible dimension in direction of flight

YMAX Longest possible dimension in Normal to direction of flight

"""

import math #importing the base math library

import numpy as np #importing numpy

from mpl\_toolkits.mplot3d import Axes3D #importing 3D plotting

import matplotlib.pyplot as plt #importing plt definitions

from matplotlib import cm #importing colour gradients

def RAC(EW,SF):

return math.sqrt(EW \* SF) / 10

def EW(EWn):

Max=EWn[0]

for val in EWn:

Max = val if (Max < val) else Max

return Max

def SizeFactor(XMAX,YMAX):

return XMAX + 2 \* YMAX

"""

Mission Scoreing

Mn Mission-n Score

NOTE: MUST HAVE SUCCESSFUL LANDING TO GET SCORE

"""

"""

Mission 1 Short Take-off

"""

def M1(Laps,MaxLaps):

return 2 \* ( Laps / MaxLaps )

"""

Mission 2 Stealth Mission

"""

def M2(Store,MaxStore):

return 4 \* ( Store / MaxStore )

"""

Mission 3 Strike Mission

"""

def M3( FastestTime,TeamTime ):

return 6 \* ( FastestTime / TeamTime )

"""

Value Definations

"""

fig = plt.figure()

ax = Axes3D(fig)

"""

Mission One

"""

t = np.r\_[40:100:200j] #Speed rangeing from 40 to 90 ft/s in 200 divisions

u=[] #Empty array to hold the new values

v = np.r\_[2:6:200j] #Number of laps aimed

for i in t:

u.append(i\*240/3800.) #loop to calculate the speed

a = np.zeros((len(u),v.size)) #empty loop to hold the scores

for i in np.arange(len(u)):

for j in np.arange(v.size):

val = M1(u[i],v[j])

a[j,i] = val if val <=2 else 2

t,u = np.meshgrid(t,u) #Create a Meshgrid for the x,y plane

#Plot the surface

ax.plot\_surface(t,u,a,alpha=0.5,cmap=cm.jet)

#Plot the contours if desired

ax.contour(t,u,a,zdir='x',offset=100) #plot the X contour

ax.contour(t,u,a,zdir='y',offset=2) #plot the y contour

#Set the x,y,z limits

ax.set\_xlim(40,100)

ax.set\_ylim(2,7)

ax.set\_zlim(0,2)

#Create the axis labels

ax.set\_xlabel("Speed ft/s")

ax.set\_ylabel("Laps Completed")

ax.set\_zlabel("Mission Score")

"""

Mission Two

"""

#Weight of Mini-Max is 1.3oz or 37grams

WE = np.r\_[1:10:200j] #Empty weight from 1 to 10 rockets

WP = np.r\_[1:10:200j] #Team weight of 1 to 10 rockets

a = np.zeros((WE.size,WP.size))

for i in np.arange(WE.size):

for j in np.arange(WP.size):

val = M2(WE[i],WP[j])

a[j,i] = val if val <=4 else 4

WE,WP = np.meshgrid(WE,WP)

ax.plot\_surface(WE,WP,a,alpha=.5,cmap=cm.jet)

ax.contour(WE,WP,a,zdir='x',offset=10)

ax.contour(WE,WP,a,zdir='y',offset=10)

#ax.contour(WE,WP,a,zdir='z',offset=0)

ax.set\_xlim(0,10)

ax.set\_ylim(0,10)

ax.set\_zlim(0,4)

ax.set\_xlabel("Stored Rockets")

ax.set\_ylabel("Max Stored Rockets")

ax.set\_zlabel("Mission Score")

"""

Mission Three

"""

mTime=np.r\_[1:600:200j] #team time from 1 to 600 seconds

aTime=np.r\_[1:600:200j] #fastest time from 1 to 600 seconds

a = np.zeros((mTime.size,aTime.size))

for i in np.arange(mTime.size):

for j in np.arange(aTime.size):

val = M3(mTime[i],aTime[j])

a[j,i] = val if val <=6 else 6

mTime,aTime = np.meshgrid(mTime,aTime)

ax.plot\_surface(mTime,aTime,a,alpha=.5,cmap=cm.jet)

ax.contour(mTime,aTime,a,zdir='x',offset=600)

ax.contour(mTime,aTime,a,zdir='y',offset=600)

ax.set\_xlabel("Flight Time (s)")

ax.set\_ylabel("Best Flight Time (s)")

ax.set\_zlabel("Mission Score")

ax.set\_xlim(0,600)

ax.set\_ylim(0,600)

ax.set\_zlim(0,6)

"""

Total score Plots

"""

EW = np.r\_[2:15:200j]

SF = np.r\_[5:30:200j]

Z = np.zeros((EW.size,SF.size))

for i in np.arange(EW.size):

for j in np.arange(SF.size):

val = 6 / RAC(EW[i],SF[j])

Z[i,j] = val

EW,SF = np.meshgrid(EW,SF)

ax.plot\_surface(EW,SF,Z,cmap=cm.jet,alpha=0.5)

ax.contour(EW,SF,Z,zdir='x',offset=0)

ax.contour(EW,SF,Z,zdir='y',offset=0)

ax.set\_xlabel("Empty Weight (lbs)")

ax.set\_ylabel("Size Factor (ft)")

ax.set\_zlabel("Total Score")

ax.set\_xlim(0,15)

ax.set\_ylim(0,30)

"""

Display the plot

"""

ax.view\_init(30,-155) # set the viewing angle for the 3D plot

plt.show() # Display the plot