**\*\* On the following pages we have information about each of the scripts and functions written, but many of them are sub-functions. Here, we explain just which scripts to run to reproduce the results and plots displayed in our project presentation and report** (excluding the Monotonicity Analysis and Parametric study)**. \*\***

*The multi-objective optimization scripts using the weighted method, epsilon-constrained method, and Global Criterion Method include single-objective optimization when the weights or epsilon are 0/1 and for finding the ideal point for calculation of the Lq norms.*

run **GridEval.m** – Evaluates a 3D cubic grid of point spanning the entire range of the variables. Saves just those points which satisfy all the constraints, i.e. the feasible region.

run **PlotGridEval.m** – Plots of the feasible region by color according to the two objectives and Lq norms (1,2, and Inf) in both the Design Space and Criterion Space. This will take a long time.

run **weighted\_fmincon\_Script** – Custom Weighted Method for Multi-Objective Optimization, based on a built in MATLAB function fmincon for Single-Objective Constrained Optimization.

run **weighted\_PenaltyMethod** – Custom Weighted Method for Multi-Objective Optimization, based on a custom Penalty Method for Single-Objective Constrained Optimization, , using built in MATLAB function fminunc for Single-Objective Unconstrained Optimization.

run **weighted\_PenaltyMethod2** – Same as previous but using a different starting point.

run **weighted\_AugmentedLagranianMethod** – Custom Weighted Method for Multi-Objective Optimization, based on a custom ALM Method for Single-Objective Constrained Optimization, using built in MATLAB function fminunc for Unconstrained Optimization.

run **EpsilonConstrainedMethod\_OptDrag** – Custom Epsilon Constrained Method for Multi-Objective Optimization, based on a built in MATLAB function fmincon for Single-Objective Constrained Optimization. Optimizing Drag, while Constraining Volume.

run **EpsilonConstrainedMethod\_OptVolume** – Same as previous but Optimizing Volume, while Constraining Drag.

run **fminimax\_script** –Multi-Objective Optimization using the built in MATLAB function fminmax.

run **GlobalCriterionMethod** with q = 1 – Custom Global Criterion Method for Multi-Objective Optimization, based on a built in MATLAB function fmincon for Single-Objective Constrained Optimization.

run **GlobalCriterionMethod** with q = 2 – Same as previous but with q = 2

run **GlobalCriterionMethod** with q = Inf – Same as previous but with q = Inf

run **Compile\_All\_Opt** –save all the optima from the different methods

run **Plot\_All\_Opt\_wColors** – to create the plot of all the optima with various colors/markers

run **isolate\_Pareto** – to create the plot just showing just the Pareto Front in the criterion space

run **Plot\_Opt\_on\_Feas** – to create the plot showing the Pareto Front points in the design space

**function [g, rho, mu, rho\_load, rho\_fins, rho\_hull, Sy\_hull, v, depth, T, theta, alpha, tfins, l, w] = set\_Params()**

Description:

User defines Parameter values using this function like a script.

Inputs:

NONE

Outputs:

g - acceleration due to gravity (m/s^2)

rho - density of water at 25 deg-C (kg/m^3)

rho\_load - density of payload material (kg/m^3)

rho\_hull - density of hull material (kg/m^3)

rho\_fins - density of fins material (kg/m^3)

Sy\_hull - tensile yield strength of hull material (N/m^2)

V - speed of the AUV (m/s)

depth - distance below surface of the water (m)

theta - angle of conic tail section of AUV (deg)

alpha - angle of attack of fins (deg)

tfins - thickness of hollow fins (m)

l - length of fins (m)

w - width of fins (m)

d - inner diameter of the hull (m)

t - thickness of the hull (m)

L - length of the cylindrical section of the hull (m)

**function [d\_L, d\_U, t\_L, t\_U, L\_L, L\_U, W\_lim, FS, F\_D\_lim, V\_i\_lim] = set\_Lims()**

Description:

User defines Limit values using this function like a script.

Inputs:

NONE

Outputs:

d\_L - lower bound for AUV inner diameter (m)

d\_U - upper bound for AUV inner diameter (m)

t\_L - lower bound for AUV hull thickness (m)

t\_U - upper bound for AUV hull thickness (m)

L\_L - lower bound for AUV length (m)

L\_U - upper bound for AUV length (m)

W\_lim - upper limit for AUV weight (N)

FS - factor of safety (n/a - unitless)

**function [L\_T] = calc\_L\_T(theta, D, L)**

Description:

Calculates the total length of the AUV (L\_T) given the variables D, L, and theta

Inputs:

theta - angle of the conic tail section of the hull (deg)

D - outer diameter of the hull (m)

L - length of the cylindrical section of the hull (m)

Outputs:

L\_T - total length of the hull (m)

**function [S] = calc\_S(theta, D, L)**

Description:

Calculates the surface area of the AUV (S) given the variables D, L, and theta

Inputs:

theta - angle of the conic tail section of the hull (deg)

D - outer diameter of the hull (m)

L - length of the cylindrical section of the hull (m)

Outputs:

S - surface area of the hull (m^2)

**function [V] = calc\_V(theta, D, L)**

Description:

Calculates the volume of water displaced by the AUV (V) given the variables D, L, and theta

Inputs:

theta - angle of the conic tail section of the hull (deg)

D - outer diameter of the hull (m)

L - length of the cylindrical section of the hull (m)

Outputs:

V - volume of water displaced by the hull (m^3)

**function [Rn] = calc\_Rn(rho, mu, v, L\_T)**

Description:

Calculates the Reynolds number (Rn) given the AUV's total length and velocity

Inputs:

rho - density of water at 25 deg-C (kg/m^3)

mu - dynamic viscosity of water at 25 deg-C (N\*s/m^2)

v - speed of the AUV (m/s)

L\_T - total length of the hull (m)

Outputs:

Rn - Reynold's Number (n/a - unitless)

**function [C\_f] = calc\_C\_f(Rn)**

Description:

Calculates the coefficient of drag of the hull of the AUV due to skin-friction (C\_f) given the Reynold's Number

Inputs:

Rn - Reynold's Number (n/a - unitless)

Outputs:

C\_f - coefficient of drag of the hull due to skin-friction (n/a - unitless)

**function [C\_p] = calc\_C\_p(D, L\_T, V)**

Description:

Calculates the coefficient of drag of the hull of the AUV due to pressure loss (C\_p) given some geometric variables

Inputs:

D - outer diameter of the hull (m)

L\_T - total length of the hull (m)

V - volume of water displaced by the hull (m^3)

Outputs:

C\_p - coefficient of drag of the hull due to skin-friction (n/a - unitless)

**function [C\_v] = calc\_C\_v(D, L\_T, C\_f, C\_p)**

Description:

Calculates the total coefficient of drag of the hull of the AUV (C\_v) using the MIT method given some geometric variables, and the component coefficients of drag due to skin friction and pressure losses (C\_f, and C\_v)

Inputs:

D - outer diameter of the hull (m)

L\_T - total length of the hull (m)

C\_f - coefficient of drag of the hull due to skin-friction (n/a - unitless)

C\_p - coefficient of drag of the hull due to skin-friction (n/a - unitless)

Outputs:

C\_v - total coefficient of drag of the hull (n/a - unitless)

**function [C\_v] = calc\_HydroCoeff(D, L\_T, V, Rn)**

Description:

Calculates the total coefficient of drag of the hull of the AUV (C\_v) based on geometric quantities and Reynolds Number (Rn)… a wrapper for calc\_C\_f, calc\_C\_p, and calc\_C\_v.

Inputs:

D - outer diameter of the hull (m)

L\_T - total length of the hull (m)

V - volume of water displaced by the hull (m^3)

Rn - Reynold's Number (n/a - unitless)

Outputs:

C\_v - total coefficient of drag of the hull (n/a - unitless)

**function [V\_hull] = calc\_V\_hull(theta, D, d, L)**

Description:

Calculates the mass-volume of the AUV (V\_hull) given d, D, L, and theta

Inputs:

theta - angle of the conic tail section of the hull (deg)

D - outer diameter of the hull (m)

d - inner diameter of the hull (m)

L - length of the cylindrical section of the hull (m)

Outputs:

V\_hull - volume of the hull material (m^3)

**function [V\_fins] = calc\_V\_fins(l, w)**

Description:

Calculates the volume of water displaced by the fins

Inputs:

l - length of fins (m)

w - width of fins (m)

Outputs:

V\_fins - volume of water displaced by the fins (m^3)

**function [V\_fins2] = calc\_V\_fins2(tfins, l, w)**

Description:

Calculates the mass-volume of the hollow fins

Inputs:

tfins - thickness of fins material (m)

l - length of fins (m)

w - width of fins (m)

Outputs:

V\_fins2 - volume of water displaced by the fins (m^3)

**function [s\_h] = calc\_s\_h(D, d, P)**

Description:

Calculates the hoop stress in the cylindrical middle section of the AUV using the thick-wall full pressure vessel equation

Inputs:

D - outer diameter of the hull (m)

d - inner diameter of the hull (m)

P - hydrostatic pressure (N)

Outputs:

s\_h - max hoop-stress in a cylinder (N)

**function [s\_a] = calc\_s\_a(D, d, P)**

Description:

Calculates the axial stress in the cylindrical middle section of the AUV using the thick-wall full pressure vessel equation

Inputs:

D - outer diameter of the hull (m)

d - inner diameter of the hull (m)

P - hydrostatic pressure (N)

Outputs:

s\_a - max axial stress in a cylinder (N)

**function [s\_cylinder] = calc\_s\_cylinder(s\_h, s\_a)**

Description:

Calculates the Von-Mises combined stresses in the cylindrical middle section of the AUV hull

based on the component stresses

Inputs:

s\_h - max hoop-stress in a cylinder (N)

s\_a - max axial stress in a cylinder (N)

Outputs:

s\_cylinder - Von-Mises Stress in the cylindrical section of the hull (N)

**function [s\_t] = calc\_s\_t(D, d, P)**

Description:

Calculates the tangential stress in the hemispherical front section of the AUV using the thick-wall full pressure vessel equation

Inputs:

D - outer diameter of the hull (m)

d - inner diameter of the hull (m)

P - hydrostatic pressure (N)

Outputs:

s\_t - max tangential stresses in a sphere (N)

**function [s\_sphere] = calc\_s\_sphere(s\_t)**

Description:

Calculates the Von-Mises combined stresses in the spherical front section of the hull based on the component stresses

Inputs:

s\_t - max tangential stresses in a sphere (N)

Outputs:

s\_sphere - Von-Mises Stress in the spherical front end of the hull (N)

**function [s\_H] = calc\_s\_HH(theta, D, d, P)**

Description:

Calculates the hoop stress in the conical tail section of the AUV using the thin-wall approximate pressure vessel equation

Inputs:

theta - angle of the conic tail section of the hull (deg)

D - outer diameter of the hull (m)

d - inner diameter of the hull (m)

P - hydrostatic pressure (N)

Outputs:

s\_H - max hoop stress in a cone (N)

**function [s\_AA] = calc\_s\_AA(D, d, theta, P)**

Description:

Calculates the axial stress in the conical tail section of the AUV

\*Using the thin-wall approximate pressure vessel equation\*

Inputs:

D - outer diameter of the hull (m)

d - inner diameter of the hull (m)

theta - angle of the conic tail section of the hull (deg)

P - hydrostatic pressure (N)

Outputs:

s\_AA - max axial stress in a cone (N)

**function [s\_cone] = calc\_s\_cone(s\_HH, s\_AA)**

Description:

Calculates the Von-Mises combined stresses in the conical tail section of the AUV hull

based on the component stresses

Inputs:

s\_HH - max hoop stress in a cone (N)

s\_AA - max axial stress in a cone (N)

Outputs:

s\_cone - Von-Mises Stress in the conical tail section of the hull (N)

**function [s\_t, s\_h, s\_a, s\_HH, s\_AA] = calc\_PV\_Stresses(theta, d, D, P)**

Description:

Calculates the pressure-vessel component stresses in the AUV hull Using the thick-wall full pressure vessel equation, given the basic geometric properties, and hydrostatic pressure

Inputs:

theta - angle of the conic tail section of the hull (deg)

d - inner diameter of the hull (m)

D - outer diameter of the hull (m)

P - hydrostatic pressure (N)

Outputs:

s\_t - max tangential stresses in a sphere (N)

s\_h - max hoop-stress in a cylinder (N)

s\_a - max axial stress in a cylinder (N)

s\_HH - max hoop stress in a cone (N)

s\_AA - max axial stress in a cone (N)

**function [s\_vonMises] = calc\_vonMises(s\_1, s\_2, s\_3)**

Description:

Calculates the Von-Mises combined stresses based on the principle stresses

Inputs:

s\_1 - 1st principle stress (N/m^2)

s\_2 - 2nd principle stress (N/m^2)

s\_3 - 3rd principle stress (N/m^2)

Outputs:

s\_vonMises - Von-Mises Max Stress (N/m^2)

**function [s\_max] = calc\_s\_max(s\_t, s\_h, s\_a, s\_HH, s\_AA)**

Description:

Calculates the Von-Mises combined stresses in the AUV sections, and the maximum throughout the entire AUV hull, based on the component stresses

Inputs:

s\_t - max tangential stresses in a sphere (N)

s\_h - max hoop-stress in a cylinder (N)

s\_a - max axial stress in a cylinder (N)

s\_HH - max hoop stress in a cone (N)

s\_AA - max axial stress in a cone (N)

Outputs:

s\_sphere - Von-Mises Stress in the spherical front end of the hull (N)

s\_cylinder - Von-Mises Stress in the cylindrical section of the hull (N)

s\_cone - Von-Mises Stress in the conical tail section of the hull (N)

s\_max - maximum Von-Mises stress in the entire AUV hull (N)

**function [f1] = eval\_f1(rho, mu, v, theta, alpha, l, w, d, t, L)**

Description:

Evaluates the objective function f1 - Force of Drag on the AUV

Inputs:

rho - density of water at 25 deg-C (kg/m^3)

mu - dynamic viscosity of water at 25 deg-C (N\*s/m^2)

v - speed of the AUV (m/s)

theta - angle of conic tail section of AUV (deg)

alpha - angle of attack of fins (deg)

l - length of fins (m)

w - width of fins (m)

d - inner diameter of the hull (m)

t - thickness of the hull (m)

L - length of the cylindrical section of the hull (m)

Outputs:

f1 - Force of Drag on the AUV (N)

**function [f2] = eval\_f2(theta, d, L)**

Description:

Evaluates the objective function f2 - % Internal Volume of the hull

Inputs:

d - inner diameter of the hull (m)

L - length of the cylindrical section of the hull (m)

Outputs:

f2 - Internal Volume of the hull (m^3)

**function [g7, g8, g9, g10] = eval\_g710( g, rho, rho\_load, rho\_fins, rho\_hull, Sy\_hull, v, depth, theta, alpha, tfins, l, w, d, t, L, W\_lim, FS)**

Description:

Evaluates the inequality constraints g7 , g8, g9, and g10 which are not simple upper and lower bounds on the variables no need to evaluate those in here for fmincon or fminimax which has separate inputs for variable bounds (LB, UB)

Inputs:

g - acceleration due to gravity (m/s^2)

rho - density of water at 25 deg-C (kg/m^3)

rho\_load - density of payload material (kg/m^3)

rho\_hull - density of hull material (kg/m^3)

rho\_fins - density of fins material (kg/m^3)

Sy\_hull - tensile yield strength of hull material (N/m^2)

V - speed of the AUV (m/s)

depth - distance below surface of the water (m)

theta - angle of conic tail section of AUV (deg)

alpha - angle of attack of fins (deg)

tfins - thickness of hollow fins (m)

l - length of fins (m)

w - width of fins (m)

d - inner diameter of the hull (m)

t - thickness of the hull (m)

L - length of the cylindrical section of the hull (m)

W\_lim - upper limit for AUV weight (N)

FS - factor of safety (n/a - unitless)

Outputs:

g7 - upper bound weight constraint

g8 - lower bound buoyancy constraint

g9 - upper bound buoyancy constraint

g10 - upper bound stress constraint

**function [g1, g2, g3, g4, g5, g6, g7, g8, g9, g10] = eval\_gALL(g, rho, rho\_load, rho\_fins, rho\_hull, Sy\_hull, v, depth, theta, alpha, tfins, l, w, d, t, L, d\_L, d\_U, t\_L, t\_U, L\_L, L\_U, W\_lim, FS)**

Description:

Evaluates all the inequality constraints

Inputs:

g - acceleration due to gravity (m/s^2)

rho - density of water at 25 deg-C (kg/m^3)

rho\_load - density of payload material (kg/m^3)

rho\_hull - density of hull material (kg/m^3)

rho\_fins - density of fins material (kg/m^3)

Sy\_hull - tensile yield strength of hull material (N/m^2)

V - speed of the AUV (m/s)

depth - distance below surface of the water (m)

theta - angle of conic tail section of AUV (deg)

alpha - angle of attack of fins (deg)

tfins - thickness of hollow fins (m)

l - length of fins (m)

w - width of fins (m)

d - inner diameter of the hull (m)

t - thickness of the hull (m)

L - length of the cylindrical section of the hull (m)

W\_lim - upper limit for AUV weight (N)

FS - factor of safety (n/a - unitless)

Outputs:

g1 - inner-diameter lower bound constraint

g2 - inner-diameter upper bound constraint

g3 - thickness lower bound constraint

g4 - thickness upper bound constraint

g5 - length lower bound constraint

g6 - length upper bound constraint

g7 - upper bound weight constraint

g8 - lower bound buoyancy constraint

g9 - upper bound buoyancy constraint

g10 - upper bound stress constraint

**function [g11] = eval\_g11(d, L, f2\_g, f2\_b, epsilon)**

Description:

evaluates the inequality constraint value g11- lower limit internal volume constraint   
(epsilon constrained / minimizing drag)

Inputs:

d - inner diameter of the hull (m)

L - length of the cylindrical section of the hull (m)

f1\_g - "good value" for volume (m^3)

f1\_b - "bad value" for volume (m^3)

epsilon - constraint level for volume, as a value between 0 and 1 for

the normalized volume

Outputs:

g11 - lower bound internal volume constraint (<= 0)

**function [g12] = eval\_g12(rho, mu, v, theta, alpha, l, w, d, t, L, f1\_g, f1\_b, epsilon)**

Description:

evaluates the inequality constraint value g12- upper limit drag constraint   
(epsilon constrained / maximizing internal volume)

Inputs:

rho - density of water at 25 deg-C (kg/m^3)

mu - dynamic viscosity of water at 25 deg-C (N\*s/m^2)

v - speed of the AUV (m/s)

theta - angle of conic tail section of AUV (deg)

alpha - angle of attack of fins (deg)

tfins - thickness of hollow fins (m)

d - inner diameter of the hull (m)

t - thickness of the hull (m)

L - length of the cylindrical section of the hull (m)

l - length of fins (m)

w - width of fins (m)

f1\_g - "good value" for drag (N)

f1\_b - "bad value" for drag (N)

epsilon - constraint level for drag, as a value between 0 and 1 for

the normalized drag

Outputs:

g12 - upper bound drag constraint (<= 0)

**function [W, F\_B, s\_max, F\_D, V\_i, W\_load, F\_L] = calc\_Everything(d, t, L)**

Description:

Calculates the objective functions and the final values related to the constraints other than the variable bounds. It is used for the visualization of the feasible region.

Inputs:

d - inner diameter of the hull (m)

t - thickness of the hull (m)

L - length of the cylindrical section of the hull (m)

Outputs:

W - weight of the empty AUV (N)

F\_B - buoyant force on the AUV (N)

s\_max - stress at max depth (N/m^2)

F\_D - force of drag on the AUV (N)

V\_i - internal volume of the cylindrical middle section of the AUV (m^3)

W\_load - weight of payload (N)

F\_L - force of lift generated by fins (N)

**function [X0] = makePaternGrid(d\_L, d\_U, t\_L, t\_U, L\_L, L\_U, Nd, Nt, NL)**

Description:

makes a 3D cubic grid of points covering the full range of variable values   
with resolution of points dictated by Nd, Nt, and NL

Inputs:

d\_L - lower bound for AUV inner diameter (m)

d\_U - upper bound for AUV inner diameter (m)

t\_L - lower bound for AUV hull thickness (m)

t\_U - upper bound for AUV hull thickness (m)

L\_L - lower bound for AUV length (m)

L\_U - upper bound for AUV length (m)

Nd - number of points in d (for grid only)

Nt - number of points in t (for grid only)

NL - number of points in L (for grid only)

Outputs:

X0 - set of start points

**function [X0] = makePaternRND(d\_L, d\_U, t\_L, t\_U, L\_L, L\_U, N)**

Description:

Samples "N" random points from a gaussian distribution   
covering the full range of variable values

Inputs:

d\_L - lower bound for AUV inner diameter (m)

d\_U - upper bound for AUV inner diameter (m)

t\_L - lower bound for AUV hull thickness (m)

t\_U - upper bound for AUV hull thickness (m)

L\_L - lower bound for AUV length (m)

L\_U - upper bound for AUV length (m)

N - number of points to generate

Outputs:

X0 - set of start points

**function [X0] = makePaternRND\_Halton(d\_L, d\_U, t\_L, t\_U, L\_L, L\_U, N)**

Description:

Samples "N" pseudo-random points from a Halton set   
covering the full range of variable values

Inputs:

d\_L - lower bound for AUV inner diameter (m)

d\_U - upper bound for AUV inner diameter (m)

t\_L - lower bound for AUV hull thickness (m)

t\_U - upper bound for AUV hull thickness (m)

L\_L - lower bound for AUV length (m)

L\_U - upper bound for AUV length (m)

N - number of points to generate

Outputs:

X0 - set of start points

**function [X0] = genX0(d\_L, d\_U, t\_L, t\_U, L\_L, L\_U, N, Nd, Nt, NL, m)**

Description:

Generates a set of start points for multi-start

Inputs:

d\_L - lower bound for AUV inner diameter (m)

d\_U - upper bound for AUV inner diameter (m)

t\_L - lower bound for AUV hull thickness (m)

t\_U - upper bound for AUV hull thickness (m)

L\_L - lower bound for AUV length (m)

L\_U - upper bound for AUV length (m)

N - number of points to generate

Nd - number of points in d (for grid only)

Nt - number of points in t (for grid only)

NL - number of points in L (for grid only)

M - method of point generation

1 - gaussian random

2 - Halton pseudo-random

3 - grid

Outputs:

X0 set of start points

**script GridEval**

Description:

Evaluates the inequality constraints for a full 3D cubic grid of points spanning the variable bounds. For those points which satisfy all the constraints, the objective functions are also evaluated.

Instructions:

Just hit "run". It will save the results to GRID\_results.mat. Plots will be generated showing the feasible domain in the design space, criterion space, and normalized criterion space.

**script PlotGridEval**

Description:

Generates plots of the feasible domain in the design space and normalized criterion space where color represents how good the point is according to a certain metric: f1, f2, L1, L2, Linf

Instructions:

Check that 'GRID\_results.mat' has been created and is in the current folder, then just hit "Run". It will generate the plots. Currently, saving of the plots and rotating gifs is commented out.

**script "fminimax\_script"**

Description

Uses the built-in MATLAB function "fminimax" for multi-objective optimization.

Instructions:

Just hit "Run". It will display the results in the command window and it will save the results in 'fminimax\_results.mat'

**script "weighted\_fimincon\_script"**

Description:

Uses the built-in MATLAB function "fmincon" in conjuntion with the Weighted Method to acheive multi-objective optimization. The weight for the first objective is swept from 0 to 1.

Instructions:

Just hit "Run". A plot of the optima achieved in the criterion space is generated. A plot of the objective functions vs the weight assigned to the first objective is generated. And the results are saved in 'fmincon\_results.mat'Currently, saving of the plots are commented out.

**script "weighted\_PenaltyMethod"**

Description:

Uses a custom-made Penalty Method (utilizing fminunc for unconstrained optimization) in conjunction with the Weighted Method to achieve multi-objective optimization. The weight for the first objective is swept from 0 to 1.

Instructions:

Set the initial guess (X0), using the following lines:

d0 = d\_L + (d\_U - d\_L)\*(?1);

t0 = t\_L + (t\_U - t\_L)\*(?2);

L0 = L\_L + (L\_U - L\_L)\*(?3);

Where the “?#” are replaced by some value between 0 and 1 to linearly interpolate between the variable bounds.

Then hit "Run". A plot of the optima acheived in the criterion space is generated. A plot of the objective functions vs the weight assigned to the first objective is generated. And the results are saved in 'Penalty\_results.mat'. Currently, saving of the plots are commented out.

NOTE:

We used two different start points. This function is currently set up with one of the two start points (lower bound of variables). For your convenience in reproducing our results we have created a copy of this function (weighted\_PenaltyMethod2) set up with the other start point (half way between variable bounds).

**script "MultiStart\_weighted\_PenaltyMethod"**

Description:

Uses a custom-made Penalty Method (utilizing fminunc for unconstrained optimization) in conjunction with the Weighted Method to achieve multi-objective optimization. Multi-Start is also used to overcome the discrepancy between solutions depending on the start point. Many different starting points are tried, and the best solution is selected. The weight for the first objective is swept from 0 to 1. For each weight, multi-start is used.

Instructions:

Select the parameters for the Multi-Start generator, then, hit "Run". A plot of the optima achieved in the criterion space is generated. A plot of the objective functions vs the weight assigned to the first objective is generated. And the results are saved in 'mutlistart\_results.mat'. Currently, saving of the plots are commented out.

NOTE:

Does not work because depending on the starting location, the search diverges to a crazy infeasible solution and the following error is thrown by MATLAB:

Error using lineSearch

Search direction is not a descent direction;

roundoff errors may be affecting convergence.

Error in fminusub (line 188)

lineSearch(funfcn,x,dir,f,dirDerivative,

...

Error in fminunc (line 446)

[x,FVAL,GRAD,HESSIAN,EXITFLAG,OUTPUT] =

fminusub(funfcn,x, ...

Error in MultiStart\_weighted\_PenaltyMethod (line

129)

[X\_min, f\_min] = fminunc(@(X)fun(X,

rp, w1, w2), X0(i,:));

**script "weighted\_AugmentedLagrangianMethod"**

Description:

Uses a custom-made Augmented Lagrangian Method (utilizing fminunc for unconstrained optimization) in conjunction with the Weighted Method to achieve multi-objective optimization.The weight for the first objective is swept from 0 to 1.

Instructions:

Set the initial guess (X0), and then hit "Run". A plot of the optima acheived in the criterion space is generated. A plot of the objective functions vs the weight assigned to the first objective is generated. And the results are saved in 'ALM\_results.mat'. Currently, saving of the plots are commented out.

**script "GlobalCriterionMethod"**

Description:

A custom Lq method multi-objective optimizer (utilizing fmincon for constrained single-objective optimization).

Instructions:

Choose a value for q, and then hit "Run". It will print out the chosen q as well as the multi-objective solution.

**script "EpsilonConstrainedMethod\_OptDrag"**

Description:

A custom-made epsilon constrained method for multi-objective optimization by converting one of the two objectives into an inequality constraint (utilizing fmincon for constrained single-objective optimization). There are two separate files for this method; this one is for optimizing drag while constraining volume.

Instructions:

Just hit "Run". A plot of the objective functions vs epsilon is generated.

And the results are saved in 'EC\_results.mat'.

Currently, saving of the plots are commented out.

**script "EpsilonConstrainedMethod\_OptVolume"**

Description:

A custom-made epsilon constrained method for multi-objective optimization by converting one of the two objectives into an inequality constraint (utilizing fmincon for constrained single-objective optimization). There are two separate files for this method; this one is for optimizing volume while constraining drag.

Instructions:

Just hit "Run". A plot of the objective functions vs epsilon is generated.

And the results are saved in 'EC2\_results.mat'.

Currently, saving of the plots are commented out.

**script "Plot\_Opt\_on\_Feas"**

Description:

Evaluates the inequality constraints for a full 3D cubic grid of points spanning the variable bounds. Clusters the evaluated points according to the inequality constraints violated. Generates a plot showing the various regions. The solid green region is the feasible domain. The transparent grey region violates the stress constraint. The transparent blue region violates the buoyancy constraints. The transparent red region violates the weight constraint. Note that these infeasible regions overlap. Also plots the optima saved in 'fmincon\_results.mat' on top.

Instructions:

(Check that 'fmincon\_results.mat' has been created and is in the current folder...Note: fmincon could be easily replaced to plot other optima), then just hit "Run". It will generate a 3D plot of the deign space showing the feasible domain and infeasible regions. Hit any button to continue when ready. It will add the optima points to the plot.

**script CheckFeas**

Description:

Checks points for feasibility.

Instructions:

Fill in the variable x with the optima point, then hit "Run". It will display the values for each of the inequality constraints, and “Y” or “N” depending on if the point is feasible or not, respectively.

**script "Compile\_All\_Opt"**

Description:

Combines all of the optima generated by the various methods into a single set which is saved as 'ALL\_results.mat'.

Instructions:

Verify that all the following files exist before hitting 'Run':

'fmincon\_results.mat'

'penalty\_results.mat'

'penalty2\_results.mat'

('ALM\_results.mat')

'fminimax\_results.mat'

'Lq1\_results.mat'

'Lq2\_results.mat'

'LqInf\_results.mat'

'EC\_results.mat'

'EC2\_results.mat'

**script "Plot\_All\_Opt\_wColors"**

Description:

Plots all of the optima generated by the various methods as different colors/markers in the normalized criterion space.

Instructions:

Verify that all the following files exist before hitting 'Run':

'GRID\_results.mat'

'ALL\_results.mat'

'fmincon\_results.mat'

'penalty\_results.mat'

'penalty2\_results.mat'

('ALM\_results.mat')

'fminimax\_results.mat'

'Lq1\_results.mat'

'Lq2\_results.mat'

'LqInf\_results.mat'

'EC\_results.mat'

'EC2\_results.mat'

**script isolate\_Pareto**

Description:

Isolates the Pareto Frontier from a set of points

Instructions:

Check that 'GRID\_results.mat' and 'ALL\_results.mat' have been created and were saved in the current folder, then just hit "Run". It will plot all the points in the feasible domain in grey, and the Pareto Frontier in green. Then it will save the plot as 'Pareto.jpg'

**script "Monotonicity\_Analysis\_d"**

Description:

Plots the objective functions and inequality constraints as functions

of a single variable: d

Instructions:

Just hit "Run". A plot will be generated.

**script "Monotonicity\_Analysis\_t"**

Description:

Plots the objective functions and inequality constraints as functions

of a single variable: t

Instructions:

Just hit "Run". A plot will be generated.

**script "Monotonicity\_Analysis\_L"**

Description:

Plots the objective functions and inequality constraints as functions

of a single variable: L

Instructions:

Just hit "Run". A plot will be generated.

**function make\_animated\_gif(varargin)**

Description:

function to build an animated gif

Instructions:

There a few different inputs needed at various points within a function to successfully create a gif. First, use make\_animated\_gif('clear') to start a new gif. Then for each frame of the gif, use make\_animated\_gif('snap', [handle]), and when done with all the frames, use make\_animated\_gif('write', filename, [delayTime], [loopCount]) to generate the gif file.

Inputs:

filename - name of the file you want to make

delayTime - Delay time between images (s, default is 0)

loopCount - Number of times to loop gif (default is infinite)

Outputs:

[filename].gif - the animated gif file

**function [v] = calc\_vmax(g, rho, mu, D, L\_T, V, S, A, T, C\_D\_fins, W)**

Description:

Calculates the Terminal Velocity of AUV. This function was created just to check that our calculations were correct and confirmed them. It showed that the AUV could reach or exceed the set parameter value for velocity (v) when the drag calculated at v was less than the available thrust T. This function could potentially be used as another objective in a future iteration of this project.

Inputs:

g - acceleration due to gravity (m/s^2)

rho - density of water at 25 deg-C (kg/m^3)

mu - dynamic viscosity of water at 25 deg-C (N\*s/m^2)

D - outer diameter of the hull (m)

L\_T - total length of the hull (m)

V - volume of water displaced by the hull (m^3)

S - surface area of the hull (m^2)

A - cross-section area of fins (m^2)

T - thrust from the AUV's propeller (N)

C\_D\_fins - Drag Coefficient for fins (n/a - unitless)

W - weight of the empty AUV (N)

Outputs:

v - terminal velocity (m/s)