Determining Aircraft Sizing Parameters through Machine Learning

J. Michael Vegh, Tim MacDonald, Brian Munguía

I. Introduction

Aircraft conceptual design is an inherently iterative process as many of the methods employed do not have an analytical solution.[1] Additional design requirements increase the computational cost of these iterations, especially in the case of unusual aircraft configurations; traditional tube-and-wing designs rely heavily on the use of empirical correlations for initial design performance, while more unusual designs with less data available require the direct use of expensive physics-based tools in order to properly estimate aircraft performance.[2] This leads to a desire to minimize the number of iterations when evaluating a new concept. In some cases, it is advantageous to split the optimization process into an optimization loop and a sizing loop, where the sizing loop contains design parameters that do not work well in the traditional optimization structure. This means that iteration occurs at two levels. The generalized problem to be solved in this paper is as follows:

$$\min f(x) \tag{1}$$

$$s.t. \quad g(x) > 0 \tag{2}$$

$$-\epsilon < h(y) < \epsilon \tag{3}$$

$$h(y) = (y_i - y_{i+1})/y_i \tag{4}$$

x refers to the set of design variables that are used in the outer optimization loop (such as wing area, taper ratio, span, cruise altitude, etc.). f is the objective function (such as fuel burn or operating cost). y refers to the design variables that are modified in the sizing loop, and ϵ is the tolerance used to converge these values. At iteration i, y_{i+1} is output by the mission solver requirements based on $[x, y_i]$. In other words, the subproblem is solved using successive substitution. The sizing loop is structured in a way that forces feasibility with respect to the sizing variables, although they are not explicitly constrained.

Two different aircraft classes are investigated in this project. The first is a conventional turbofan-powered aircraft with a single sizing variable where y is the gross takeoff weight. The second is an all-electric aircraft powered by a combination of aluminum-air and lithium-ion batteries, which requires convergence on each battery system energy, overall power, and aircraft mass. Additional details on this design can be seen in Reference 3. A side-by-side view of the two configurations can be seen Figure 1, where the conventional case is on the left, and the aluminum-air aircraft is on the right. A reference convergence plot for the base sizing loop for the aluminum-air aircraft is shown the Figure 2. As the sizing process proceeds (for fixed x), there is significant oscillation and error measures are out of phase, which significantly slowed convergence. Though the number of iterations may seem small, each requires a full mission evaluation, which is a significant computation especially if higher-fidelity methods are used. At this point we are exploring sizing variable estimation, and higher-fidelity analysis is not used. We are not aware of any similar work for comparison.

Figure 1: Aircraft Comparison

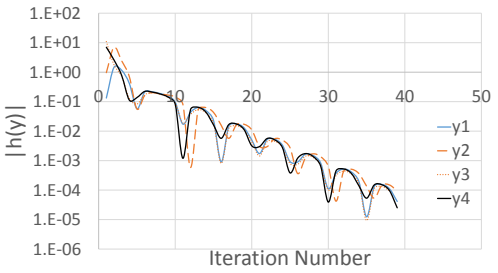

Figure 2: Sizing Convergence Example

II. Methodology

All of the aircraft analysis and optimization studies utilize SUAVE, an open-source design tool.[4] SUAVE was built to ensure compatibility with any number of exotic energy systems, and thus, is well-suited for studies such as this. A more recent paper highlighted SUAVE's flexibility in formulating optimization problems which this paper will heavily leverage.[5] Optimization was handled by SUAVE's pyOpt optimizer wrapper, which called SNOPT (a Sequential Quadratic Programming method).[6,7] The formulation of the aircraft optimization structure for this project is shown in Figure 3

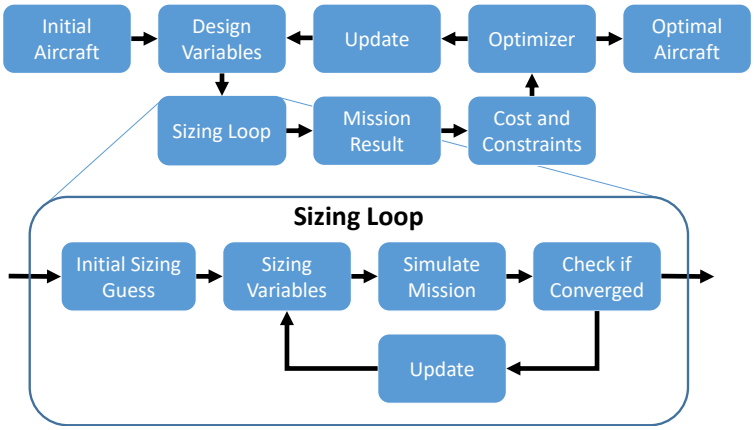

Figure 3: Optimization Loop

One potential way to increase code speed is to tabulate the converged solutions, and, when the input variables x are "close" to a tabulated value use the nearest converged y(x) as an initial guess. However, farther away from the tabulated data, one should use other methods to determine a "good" initial guess for the y values. This process falls between the Design Variables and Initial Sizing Guess in Figure 3. Design variables include wing loading, thrust loading, fan pressure ratio, wing thickness-to-chord ratio, as well as a set of aircraft speeds and altitudes to define the climb and descent trajectories.

A variety of regression algorithms from scikit-learn were evaluated for usefulness in determining a good initial guess for y in the sizing loop based on the design variables x, in an attempt to speed up the optimization process.[8] This is done by training the algorithms on converged sizing parameters as the optimization progresses and comparing the results to the output of the next iteration. This dataset comes from a reference optimization case, which has ≈ 600 converged sizing loops.

Once a comparison is made using previous data, the best performing algorithms are selected and tested in the loop. As the optimization process proceeds the algorithm is trained on iteration 1 through i, and tested on iteration i+1, where the norm of the error is output. This testing begins after a specified number of major optimizer iterations (in this case 3). Only data from the current optimization is used in this training process.

The regression methods were split into linear regression algorithms, which include methods such as Gaussian Process Regression, Support Vector Regression, and K Nearest Neighbors, and ensemble methods.[9–11] The ensemble methods, rather than using a specific algorithm to fit the variables, stochastically search in function space to determine what combinations of functions work well for the given dataset. Thus, the ensemble methods are nondeterministic, but may be more accurate and robust depending on the dataset.

Within the sizing loop itself, a recurrent neural network is tested for two purposes. First, to determine its suitability for estimating successive sequence values, and second, to determine its suitability as a classifier for whether or not the sequence will convergence to a feasible value. For regression, this works by using a given sequential set of steps as the input and the next step as the output. For classification, the first several steps are the input and the result (converged or diverged) is the output.

RNNs work by finding a recurrence relation between the sequence of input data and the resulting output. This recurrence relation allows us to use the same weight and bias parameters across all steps, significantly reducing the number of parameters required. A diagram of a typical RNN is shown in Figure 4. As the figure shows, the output of the hidden state s_t is used along with the input x to determine the hidden state at the next iteration step. The hidden state can therefore be thought of as the memory of the RNN, and

the output y_t is determined from the memory at step t.[12]

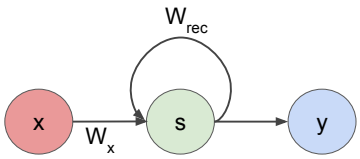

Figure 4: Recurrent Neural Network

Training data for this part is collected in two ways. The first uses Latin Hypercube sampling based on the chosen input parameters for the aircraft. In the case of the aluminum-air aircraft, this is a list of 12 parameters each with hand chosen bounds. They are scaled and used to initialize many sizing loops, and the sequence of sizing variables is collected as the loops are run. Although the input values (x) are constant throughout the sizing loop, they are added to the sequence for each step since the outcome of the loop is dependent on both the sizing variables (y) and the input variables (x). This gives a vector at each step with 16 values. LHS is an effective way to sample a variety of data, but the physical constraints of the aircraft system mean that many of the input combinations cannot form a feasible aircraft. In this situation, which accounts for about 95% of the samples, the sizing loop diverges, providing no data suitable for use in predicting a sized aircraft. To deal with this, data is also collected from the sizing done within the full aircraft optimization loop, which has a much higher proportion of converged values.

TensorFlow[13] is used to construct the neural network for this effort, with initial parameters and methods drawn from available examples.[14] We have attempted to tune the model to achieve acceptable performance by varying the learning rate, batch size, iteration count, number of sequence steps used in training, and more.

III. Results and Discussion

Statistics were calculated to compare the effectiveness of each algorithm over a test dataset. Table 1 compares some representative linear-regression algorithms (many others were tested, but these proved to be the most effective on this set of data). Note that the mean, median, and maximum errors were the most significant metrics for algorithm performance. The mean and median both represent the performance the algorithm is most likely to have when evaluating a particular optimization point, while the maximum error, if too high (meaning a poor initial guess), could cause the sizing loop to diverge, which could result in a breakdown of the optimization process when run in-the-loop.

Table 1: Aluminum-Air Aircraft Outer Loop Performance (Linear Regression Algorithms)

Algorithm	Mean Error	Median Error	Std Error	Min Error	Max Error
SVR (RBF Kernel)	0.830	0.432	1.576	0.070	6.935
Gaussian Process	0.462	0.300	0.433	0.025	2.297
Table	0.784	0.062	2.183	0.001	11.166
Kneighbors (5), distance weighted	0.454	0.131	1.037	0.006	6.505

In addition, several ensemble methods were tested on this dataset. Note that the ensemble methods are stochastic, so the results will vary slightly each time the method is run.[15–18] However, after running each regressor several times on the same dataset, it was found that the results in Table 2 were representative of the overall performance of each method. Extremely Randomized Trees (Extra trees) was found to slightly outperform the other methods on this dataset. This is not necessarily the case every time these algorithms are run, as they are stochastic.

For these cases, the simple table-lookup model possessed the lowest median error, but a higher maximum error (and with it, a higher standard deviation) than the other methods. This error occurs when the optimizer takes large steps, which tends to happen early on in the optimization process. This is why there is a large difference between the mean and median error in some cases; some regressors were unable to capture some of these large steps, but most data points were taken using smaller steps in x: hence, the small median.

Table 2: Aluminum-Air Aircraft Outer Loop Performance (Ensemble Algorithms)

Algorithm	Mean Error	Median Error	Std Error	Min Error	Max Error
Random Forest	0.403	0.293	0.402	0.003	2.110
Gradient Boosting	0.324	0.147	0.395	0.002	1.898
Bagging	0.400	0.361	0.366	0.008	1.721
Extra Trees	0.316	0.137	0.380	0.009	1.901

Compared to the table method, K Nearest Neighbor (KNN) has lower maximum, mean, and median errors. Gaussian Process Regression (GPR) reduced the maximum error by a factor of three, but with lower average accuracy. This is indicative of a more robust, but less accurate algorithm (or in other words, higher bias, but lower variance). The ensemble methods, on the other hand, all appear to maintain the robustness of GPR, with the addition of increased accuracy. Extra Trees slightly edges the other algorithms out in performance. Once the algorithm performance was estimated for the aluminum-air aircraft, the Table-Lookup, K Nearest Neighbors, Gaussian Process Regression, and Extra Trees were evaluated in-the-loop for the regional jet as well as the aluminum-air aircraft. These particular algorithms were chosen because they appeared to show slightly improved performance over comparable methods. Nonetheless the difference is not strong enough to declare a clear "winner." Sizing Loop results for the regional jet case are shown in Table 3. Note that, because of numerical noise, each of these algorithms may have resulted in a different number of outer loop iterations (sizing loops were converged to ϵ = 1E-4). Thus, $\frac{n_{size}}{n_{opt}}$ is considered the best metric for evaluating algorithm performance, as it is the average number of sizing calls per optimization iteration.

Table 3: Regional Jet Optimization Results

Initial Step	n_{size}	n_{opt}	$\frac{n_{size}}{n_{opt}}$	landing weight (lbs.)
Same Point	406	57	8.8	83,282
Table	149	46	3.2	83,454
KNN(5)	150	46	3.3	83,454
GPR	150	46	3.3	83,454
Extra Trees	149	46	3.2	83,457

The regional jet case was interesting in that the problem was simple enough that the optimization case converged after only $\approx$ 5 major iterations, meaning that the regression algorithms only ran for one or two iterations (three major iterations was chosen as the threshold before these regression were run, to prevent extrapolation to extremely unphysical results). As a result, the optimization path was essentially identical for all cases. Thus, the extra complication in setting up the more elaborate algorithms may not necessarily be cost effective in a more applied setting. Results for the aluminum-air aircraft can be seen in Table 4.

Table 4: Aluminum-Air Optimization Results

Initial Step	n_{size}	n_{opt}	$\frac{n_{size}}{n_{opt}}$	landing weight (lbs.)
Same Point	95,849	4,396	21.8	44,976
Table	7,227	1,343	5.4	45,040
KNN(5)	11,838	2,409	4.9	45,134
GPR	3,415	524	6.5	48,454
Extra Trees	10,555	2,045	5.2	44,979

The aluminum-air aircraft case had an order of magnitude reduction in computational cost when using these more informed initial guesses. Additionally, solving the sub-problem proved more troublesome for the inner-loop solver, due to the increased complexity (as seen in Figure 2). Furthermore, numerical noise

became a more significant issue here, which substantially increased optimization time. Note that here, for each of these algorithms, the optimizer design variables moved very close to the solution output (generally at $\approx$ 1,000 optimization calls (which included finite difference steps)), but was unable to satisfy the optimality conditions, so it spent a significant amount of time searching for better points; this is because the numerical noise made the gradient calculations less consistent than the conventional case.

Additionally, within the sizing loop, attempts have been made to speed the process by training a recurrent neural network to make use of a sequence of sizing inputs to determine subsequent values. This was trained with two to four steps at various points in the sequence being the input, and the next step being the output. We were able to train a network to roughly fit this, as shown in Table 5. Unfortunately this accuracy is likely not high enough to be useful in later stages of the sizing loop, where very high precision and accuracy are required. Attempts were made to improve this by changing various network parameters, but without substantial gains. For these results, the learning rate was 0.001, batch size of 100, 100,000 iterations and 150 nodes. Error refers to the mean squared error. Error decreases with number of the steps, but the usefulness of the network does as well because more steps would have to be computed in the traditional manner. The substantial difference in training and test error suggests the possibility of overfitting.

Table 5: Aluminum-Air Aircraft Sizing Loop RNN Regression Performance

Steps Used	# Training Examples	# Test Examples	Train Error	Test Error
3	134831	2289	0.237	1.13
4	128071	2232	0.0870	0.284

A second neural network was trained to classify whether an input sequence would result in a converged or diverged solution. Specifically, a bidirectional RNN with 2 hidden layers per direction was used for the classification problem to increase the amount of information available to the classifier. The results here are from a network with learning rate of 0.04, batch size of 100, 100,000 iterations and 56 nodes. As noted earlier we have two data collection methods, one that uses the full optimization to find samples, and one that use LHS. When the optimization data was included only in the training data, a sample run with 12400 training examples and 5800 test examples gave a correct converged classification of 87.3% and a correct diverged classification of 98.8%. However, when all of the data is shuffled together, the correct converged classifications are much higher. This suggests the optimization sourced values behave in a particular way that is driving the training more than originally anticipated. K-fold cross-validation was used to explore these results, with 10 folds and a total of 18000 examples. 200 samples from the original 18200 are randomly removed for ease of computation. Results are shown in Table 6. Although the cause is unclear, the out-sized effect of full shuffling versus only training shuffling is perhaps unsurprising since the optimization examples make up a large percentage of the converged examples.

Table 6: Aluminum-Air Aircraft Sizing Loop RNN Classification Performance

Steps Used	# Training Examples	# Testing Examples	Converged Correct	Diverged Correct
4	16200	1800	97.8%	99.1%

IV. Conclusions

Results here show than an order-of-magnitude reduction in sizing evaluations can be achieved when these regression methods are run in-the-loop, at least for gradient-based methods. Due to the numerical noise being a significant factor in the overall number of iterations, it was difficult to declare which algorithms performed "best" when run in-the-loop. However, each of these methods reduced the number of sizing evaluations by a factor of 10 over the baseline case. Future work will explore some of these methods using global optimizers (such as trust region model management or other surrogate-based methods); this is expected to further reduce computational cost while also allowing for a more complete search of the design space. For the RNN, future work could explore the best way to incorporate examples from different sampling methods. Another useful result could be the determination of likelihood that a classification is correct, since this would be useful in exploring the design space of futuristic aircraft.

Note: Professor Juan Alonso is the adviser for a related paper.

References

[1]Raymer, D., *Aircraft Design: A Conceptual Approach*, AIAA, Playa del Ray, California, 4th ed., 2006.

[2]Fredericks, W., Moore, M., and Busan, R., "Benefits of Hybrid-Electric Propulsion to Achieve 4x Increase in Cruise Efficiency for a VTOL Aircraft," AIAA Aviation Technology, Integration, and Operations (ATIO) Conference.

[3]Vegh, J. and Alonso, J., "Design and Optimization of Short-Range Aluminum-Air Powered Aircraft," *54th AIAA Aerospace Sciences Meeting*, AIAA Scitech, San Diego, CA, 2016.

[4]Lukaczyk, T., Wendorff, A., Botero, E., MacDonald, T., Momose, T., Varyar, A., Vegh, J., Colonno, M., Orra, T., Illaria da Silva, C., and Alonso, J., "SUAVE: An Open-Source Environment for Multi-Fidelity Conceptual Vehicle Design," AIAA Aviation Forum 2015, Dallas, TX, 2015.

[5]Botero, E., Wendorff, A. D., MacDonald, T., Variyar, A., Vegh, J. M., Alonso, J. J., Orra, T. H., and Ilario da Silva, C., ""SUAVE: An Open-Source Environment for Conceptual Vehicle Design and Optimization," *AIAA Scitech*, San Diego, CA, January 2016.

[6]Perez, R. E., Jansen, P. W., and Martins, J. R. R. A., "pyOpt: A Python-Based Object-Oriented Framework for Nonlinear Constrained Optimization," *Structures and Multidisciplinary Optimization*, Vol. 45, No. 1, 2012, pp. 101–118.

[7]Gill, P. E., Murray, W., and Saunders, M. A., "SNOPT: An SQP algorithm for large-scale constrained optimization," *SIAM Journal on Optimization*, Vol. 47, No. 1, 2002, pp. 99–131.

[8]Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, Vol. 12, 2011, pp. 2825–2830.

[9]Rasmussen, C. E. and Williams, C. K. I., *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*, The MIT Press, 2005.

[10]Smola, A. J. and Schölkopf, B., "A Tutorial on Support Vector Regression," *Statistics and Computing*, Vol. 14, No. 3, Aug. 2004, pp. 199–222.

[11]Cover, T. and Hart, P., "Nearest Neighbor Pattern Classification," *IEEE Trans. Inf. Theor.*, Vol. 13, No. 1, Sept. 2006, pp. 21–27.

[12]http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/.

[13]Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2015, Software available from tensorflow.org.

[14]https://github.com/aymericdamien/TensorFlow-Examples.

[15]Breiman, L., "Bagging Predictors," *Mach. Learn.*, Vol. 24, No. 2, Aug. 1996, pp. 123–140.

[16]Liaw, A. and Wiener, M., "Classification and Regression by randomForest," *R News*, Vol. 2, No. 3, 2002, pp. 18–22.

[17]Friedman, J. H., "Greedy Function Approximation: A Gradient Boosting Machine," *Annals of Statistics*, Vol. 29, 2000, pp. 1189–1232.

[18]Geurts, P., Ernst, D., and Wehenkel, L., "Extremely randomized trees," *Machine Learning*, Vol. 63, No. 1, 2006, pp. 3–42.