Collaborative Multidisciplinary Design Optimization with Neural Networks

Jean de Becdelièvre
Stanford University
jeandb@stanford.edu

Ilan Kroo
Stanford University
kroo@stanford.edu

Abstract

The design of complex engineering systems leads to solving very large optimization problems involving different disciplines. Strategies allowing disciplines to optimize in parallel by providing sub-objectives and splitting the problem into smaller parts, such as Collaborative Optimization, are promising solutions. However, most of them have slow convergence which reduces their practical use. Earlier efforts to fasten convergence by learning surrogate models have not yet succeeded at sufficiently improving the competitiveness of these strategies. This paper shows that, in the case of Collaborative Optimization, faster and more reliable convergence can be obtained by solving an interesting instance of binary classification: on top of the target label, the training data of one of the two classes contains the distance to the decision boundary and its derivative. Leveraging this information, we propose to train a neural network with an asymmetric loss function, a structure that guarantees Lipshitz continuity, and a regularization towards respecting basic distance function properties. The approach is demonstrated on a toy learning example, and then applied to a multidisciplinary aircraft design problem.

1   A Bilevel Architecture for Design Optimization

1.1   Introduction

Design optimization of complex engineering systems such as airplanes, robots or buildings often involves a large number of variables and multiple disciplines. For instance, an aircraft aerodynamics team must choose the wing geometry that will efficiently lift the weight of the airplane. This same weight is impacted by the structures team, who decides of the inner structure of the wing to sustain the lifting loads. While the disciplines operate mostly independently, some of their inputs and outputs are critically coupled.

An important body of research on *design architectures* [1] focuses on efficient approaches for solving design problems. Convergence speed and convenience of implementation are major priorities of the engineering companies adopting them. Solving design problems as a large single-level optimization problem has been shown to work well when all disciplinary calculations can be centralised [2]. However, it is often impractical due to the high number of variables and to the added complexity of combining problems of different nature (continuous and discrete variables for instance).

Bilevel architectures are specifically conceived to be convenient given the organization of companies and design groups. An upper *system* level makes decisions about the shared variables and assigns optimization sub-objectives to each disciplinary team at the *subspace* level. The value of local variables - such as the inner wing structure - is decided at the subspace level only, which allows the use of discipline specific optimizers and simulation tools.

Collaborative Optimization (CO)[3, 4] is the design architecture that is closest to current systems engineering practices and is the focus of our work. The system level chooses target values for the

shared variables that minimize the objective function. The sub-objective of each subspace is to match the targets, or minimize the discrepancy, while satisfying its disciplinary constraints. Collaborative optimization has been successfully applied to various large-scale design applications, including supersonic business jets [5], satellite constellation design [6], internal combustion engines [7], and bridge structural design under dynamic wing and seismic constraints [8].

However, the widespread adoption of CO is limited by its slow convergence speed, which often is an issue with bilevel architectures [2]. Mathematical issues when solving the CO system-level problem with conventional sequential quadratic programming optimizers have been documented in [9], [4] and [10], and various adapted versions have been attempted [11, 12].

A promising approach is to train surrogate models at predicting whether each discipline will be able to match the given target value. Using such representations of the feasible set of each discipline, the system-level problem can choose better informed target values for the shared variables. The use of sequentially refined quadratic surrogate models is demonstrated in [13], and several other studies use the mean of a Gaussian process (GP) regression [14]. While the earlier work focused on predicting the discrepancies of each discipline using regression, [15] recognized that the main goal is to classify targets as feasible or infeasible, and proposes an original mix between a neural network classifier and a GP regression. Our paper builds upon this work, and shows that the information obtained from the subspace solution can be used to improve classification performance.

The rest of the paper is organized as follows. Section 2 shows how CO reformulates design problems and presents an aircraft design example problem. Section 3 takes a deeper dive into the structure of the data available to train a classifier, and proposes to use Lipschitz networks [16]. It demonstrates that they generalize better on this task and provides intuition for it. In Section 4 the example problem of Section 2.2 is solved with our approach. Finally, Section 5 concludes, discusses the drawbacks of the current approach and suggests directions for improvements.

2 Collaborative Optimization

2.1 Mathematical Formulation

Mathematically, the original MDO problem can be written:

$$\underset{x_{1:n}, z}{\text{minimize}} \quad f(z) \tag{1}$$
$$\text{subject to} \quad c_i(x_i, z) \leq 0 \qquad \forall i \in 1 \ldots N_d$$

where N_d is the number of disciplines, z are the shared variables and x_i the local variables of discipline i. Typically, the objective f is easy to compute, but evaluating the constraints c_i requires time consuming disciplinary specific analysis. With CO, we form the *system*-level problem that only contains shared variables:

$$\underset{z}{\text{minimize}} \quad f(z) \tag{2}$$
$$\text{subject to} \quad J_i^*(z) \leq 0 \qquad \forall i \in 1 \ldots N_d$$

The $(J_i^*)_{i \in 1 \ldots n}$ are the values of the *subspace*-level optimizations:

$$J_i^*(z) = \min_{\bar{z}_i \in \{\bar{z}_i \mid \exists x_i \ c_i(x_i, \bar{z}_i) \leq 0\}} \|\bar{z}_i - z\|_2^2 \tag{3}$$

$\bar{z}_i$ is a local copy of the global variables. The subspace problem sets $\bar{z}_i$ to the feasible point closest to z. In practice, the constraint $J_i^*(z) \leq 0$ is often replaced by $J_i^*(z) \leq \epsilon$, with ϵ a positive number.

2.2 Simple Aircraft Marathon Design Example Problem

We consider the design of an electric powered RC airplane to fly a marathon as fast as possible. The fuselage, tail, motor and propeller are are given, and we must design the wing and pick a battery size. There are two disciplines:

1. Choosing a wing shape (span b and area S) and computing its drag D_{wing}, weight W_{wing} and lift L. Making sure L is greater than the total weight $W = (W_{bat} + W_{fixed} + W_{wing})$. Also, computing the total aircraft drag D.
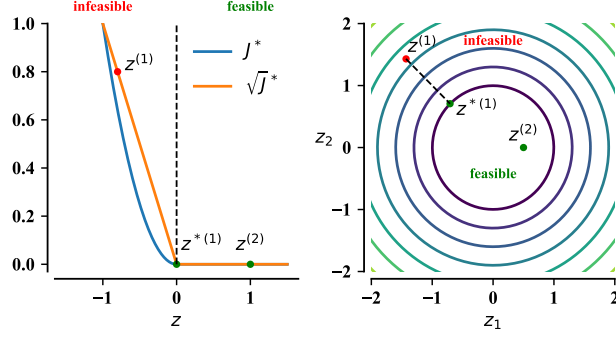
Figure 1: Plots of the square root subspace value function $\sqrt{J^*}$ for an imaginary 1-d problem where the feasible set is $\mathbf{R}^+$(left), and an imaginary 2-d problem where the feasible set is the unit ball (right). The 1-d case also shows the value function J^* itself, where the reader can note the null derivative at the boundary which makes classification ambiguous. In both cases, $z^{(1)}$ is shown as an example of an infeasible point (along with its feasible projection $z^*(1)$) while $z^{(2)}$ is shown as a example of a feasible point.

2. Choosing the propulsion system operating conditions (voltage U and RPM) such that the propeller torque Q_p matches the motor torque Q_m, and computing the thrust T. Making sure T is greater than the total drag $D = (D_{fixed} + D_{wing})$. Using V and the electric power consumed by the motor P_{in} to compute W_{bat}.

In CO form, the *system*-level problem only contains the drag D and speed V:

$$\min_{D,V,W_{bat}} \quad -V$$
$$\text{subject to} \quad J^*_{aerodynamics}(D, V, W_{bat}) = 0$$
$$J^*_{propulsion}(D, V, W_{bat}) = 0$$

where $J^*_{aerodynamics}$ and $J^*_{propulsion}$ are the optimal values of the following *subspace*-problems:

$$\min_{\bar{D},\bar{V},\bar{W}_{bat},b,S} \quad \begin{aligned} (\bar{D} - D)^2 + (\bar{V} - V)^2 \\ +(\bar{W}_{bat} - W_{bat})^2 \end{aligned}$$
$$\text{subject to} \quad W_{wing} = \text{Weight}(b, S, L)$$
$$L, \bar{D} = \text{Aero}(b, S, \bar{V})$$
$$L \geq W_{wing} + W_{fixed} + \bar{W}_{bat}$$

$$\min_{\bar{D},\bar{V},\bar{W}_{bat},\omega,U} \quad \begin{aligned} (\bar{D} - D)^2 + (\bar{V} - V)^2 \\ +(\bar{W}_{bat} - W_{bat})^2 \end{aligned}$$
$$\text{subject to} \quad Q_m, P_{in} = \text{Motor}(\omega, U)$$
$$Q_p, T = \text{Propeller}(\omega, \bar{V})$$
$$Q_p = Q_m, \quad T \geq \bar{D}$$
$$\bar{W}_{bat} = \text{Battery}(\bar{V}, P_{in})$$

Details about the various models (Weight, Aero, Motor, Battery and Propeller), as well as a nomenclature can be found in appendix A. This problem is useful to explain CO, but note that it can easily be solved using single-level optimization because there is a small number of variables and each discipline is easily evaluated.

3 From the Subspace Problem to Signed Distance Functions

3.1 A Mixed Classification and Regression approach

The subspace optimization problem (Eq. 3), is a projection operation onto the disciplinary feasible set. Hence, the optimal value of the i-th subspace $J^*_i(z)$ is the square of the distance between z and closest point to z inside the feasible set of discipline i: it is 0 in the feasible region and positive outside. Figure 1 shows the aspect of $\sqrt{J^*_i}$ for a 1-d and 2-d simple examples. The figure also shows hypothetical datapoints z_i, as well as the result z^*_i of the projection.

The main purpose of the $J^*_i(z) \leq 0$ constraint in the system-level problem (2) is to indicate the feasibility of z. As was noted in [1, 2], this constraint is numerically ambiguous. $J^*_i(z) \leq \epsilon$ is
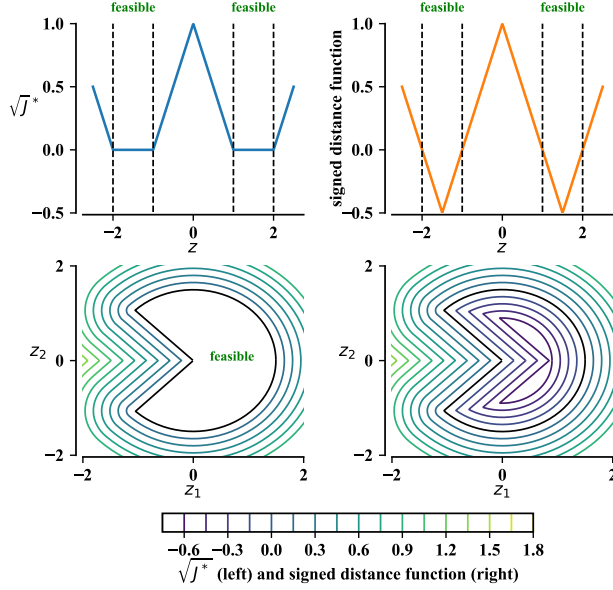
3

Figure 2: Plots of the square root of the subspace value function $\sqrt{J^*}$ as well as the signed distance function of the boundary for an two imaginary problems. *Top:* The feasible set is the union of the $[-2, -1]$ and $[1, 2]$. *Bottom:* The feasible set is 2-dimensional and resembles a Pac-Man character. In this project, we have data about $\sqrt{J^*}$, but find it more efficient for classification to aim at representing the signed distance function.

typically used instead, where the choice of the threshold ϵ can negatively impact the feasibility of the final design. The fact the norm of the gradient of $J_i^*(z)$ is 0 at the boundary reinforces the ambiguity. With surrogate models of J_i^*, the ambiguity becomes even more of a problem. Typical nonlinear regression approaches do not represent well the large flat areas of the feasible region and tend to oscillate around zero.

In this work we propose to train a simple feedforward neural network to classify whether a point z is feasible or not. While exactly regressing on J_i^* does not lead to satisfying results, the information about the distance between each infeasible point and the boundary of the feasible domain remains very precious. Let h_i be a neural network and $\mathcal{D} = (J_i^{*(j)}, z^{(j)})_{j \in 1 \ldots N}$ a set of evaluations of the i-th subspace problem (Eq. 3), we propose the following loss function:

$$l(z^{(j)}) = \begin{cases} |h_i(z^{(j)}) - \sqrt{J_i^{*(j)}}| & \text{if } z^{(j)} \text{ is infeasible} \\ \max(h_i(z^{(j)}), 0) & \text{otherwise} \end{cases} \tag{4}$$

The network is only equal to $\sqrt{J_i^*}$ in the infeasible region. In the feasible region, it is trained to be non-positive. The change of sign at the boundary creates a non-ambiguous classification criterion.

In practice, each subspace solution at $z^{(j)}$ yields a projected point $z^{*(j)}$ and the gradient of $J_i^*(z^{(j)})$. This information can easily be included in the loss function in equation 4, see appendix B for details.

3.2   Additional Useful Properties of $\sqrt{J_i^*}$

For any infeasible point z, $\sqrt{J^*(z)}$ is by definition the distance between z and the feasible set. Hence, there exist no feasible point in a ball of radius $\sqrt{J^*(z)}$ around z. For the trained neural network h that represents this distance, this property is immediately equivalent to being 1-Lipshitz:

$$\forall (z_1, z_2), \quad \|h(z_1) - h(z_2)\| \leq \|z_1 - z_2\|$$

4

Moreover, as can be seen on figure 2, $\sqrt{J^*(z)}$ is a continuous function. But its derivative can be non-continuous if the feasible set is not convex, which discourages the use of any continuously differentiable activation function in the neural network h.

These properties lead to constraining the type of neural network. In the remaining examples of this paper, Lipshitz networks designate feedforward neural networks with orthogonal weight matrices and GroupSort activation function, as described in [16]. This choice is compatible with fitting non-continuously differentiable functions and allows to automatically enforce 1-Lipshitz continuity.

Finally, any function h that represents a distance to a set respects the following basic properties. For almost every z outside of the set:

$$\|\nabla h(z)\| = 1 \tag{5}$$
$$h\left(z - h(z) \cdot \nabla h(z)\right) = 0 \tag{6}$$
$$\nabla h\left(x - h(z) \cdot \nabla h(z)\right) = \nabla h(x) \tag{7}$$

This property is enforced through regularization. At each training iteration, samples are randomly drawn within the domain, and a term proportional to the mean squared violation is added to the loss function. Importantly, we apply this regularization both inside and outside the feasible domain, which encourages h to represent a *Signed Distance Function* (SDF) [17] of the feasibility boundary, as shown on the right graphs of figure 2.

3.3 Performance comparison as dimension increases

While most fitting approaches do well in low dimension, higher dimensional problems typically reveal more weaknesses. Building onto the disk example of figure 1, we let the dimension increase while maintaining the same number of points. We compare the ability to classify unseen data (as feasible or unfeasible) of four neural network training approaches:

Name	Loss function	Network type
J^*-fit	Mean squared error regression on J	Tanh, feedforward
Classifier	Hinge loss	Tanh, feedforward
Hybrid	Mixed loss (see eqn. 4)	Tanh, feedforward
SDF	Mixed loss (see eqn. 4), with SDF regularization (see eqn. 5)	Lipshitz network

The results are shown on figure 3, and clearly show that much better generalization is obtained using the signed distance function approach. The details of the experiment can be found in appendix C.

4 Solution to the Aircraft Design Problem

The aircraft design problem of section 2.2 is solved using the neural network classification approach outlined in the previous section. We adopt a very simple design of experiment:

1. Sample points $\left(z^{(j)}\right)_{j \in 1 \ldots N_{ini}}$ randomly in the domain and evaluate $J_i^*(z^{(j)})$ for each discipline i.
2. Repeat:
   (a) For each discipline i, fit a SDF neural network h_i following the methods of section 3.2.
   (b) Find a new candidate point z and add it to the dataset:

$$z = \operatorname*{argmin}_{z \in \{z | \forall i \in 1 \ldots N_d \ h_i(z) \leq 0\}} f(z) \tag{8}$$

If there is no solution to equation 8, we use the point that is closest to being feasible. While our work focuses on surrogate modeling approaches, more principled exploration could improve the performance on more complex problems.
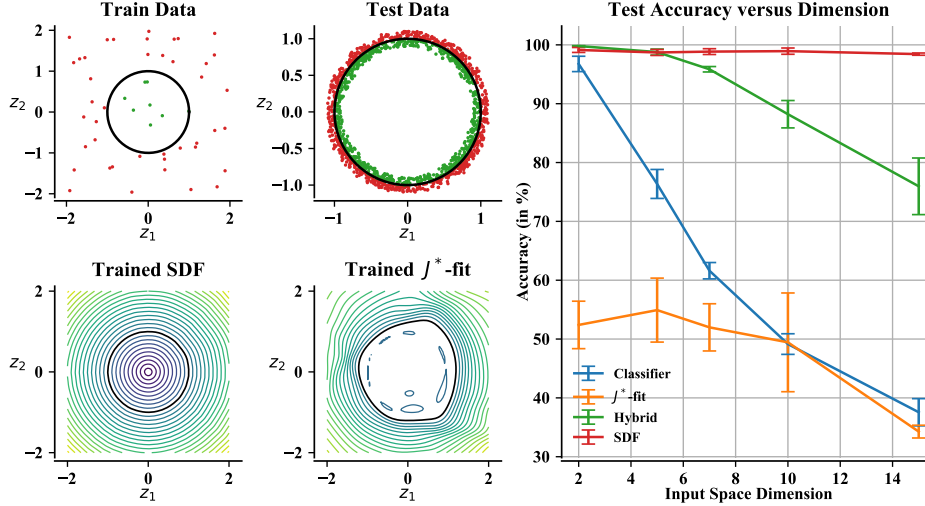
Figure 3: Demonstration of the benefit of fitting a signed distance function to classify disciplinary feasible points given the type of data available in this work. In 2-d, the top left plots show the train and test data while the bottom left ones show the contour lines of the learned network in this case. Fitting J^* directly is harder, and also requires choosing an arbitrary numerical threshold for classification, whereas the SDF only requires a sign check. On the right, the input dimension is increased while maintaining the same number of datapoints, demonstrating a much better generalization using the SDF approach. The experiment is repeated 5 times.

Table 4 compares the performance of our approach with a direct application of CO where the system level problem is solved with sequential quadratic programming [4], as well a with a conventional surrogate modeling approach using Gaussian process (GP) regression on J. Appendix D gives details about the algorithm and the baselines, and shows visualizations of the SDF network's build up.

	Conventional CO with SQP	Gaussian Process surrogate model of J^i	SDF approach (ours)
Number of system-level iterations to 5% of optimal value	74 (+/- 28)	18 (+/- 6)	6 (+/- 2)
Number of aerodynamic function evaluations to 5% of optimal value	2161 (+/- 796)	267 (+/- 63)	110 (+/- 57)
Number of propulsion function evaluations to 5% of optimal value	496 (+/- 209)	94 (+/- 23)	31 (+/- 8)

Table 1: Solution of the aircraft design problem of section 2.2 using previous CO approaches and with our approach. The rounded mean and standard deviation over 20 trials are indicated. The two last column clearly demonstrate the advantage of using a surrogate model over the original CO architecture. The performance of our approach comes from the resolution of the ambiguous decision rule described in section 3.1, as well from a family of function adequate to this particular problem.

5 Conclusion and Future Work

The convergence speed of CO, a distributed multidisciplinary design optimization architecture is improved by using neural networks to classify target shared variables as feasible or infeasible. For infeasible points, the training data contains the distance to the closest feasible point and its derivative.

Leveraging this information by training Lipshitz networks with a custom asymmetric loss function and proper regularization, we show that more reliable and faster convergence can be obtained.

An important drawback of the approach presented here is the lack of a principled exploration strategy. In typical Bayesian optimization [18, 19, 20, 21], the exploration-exploitation trade off is carefully taken care of by optimizing an acquisition function such as expected improvement. This limitation will be addressed in future work, for instance by building upon the work in [20].

Future work will also consist in applying our approach to more complex and larger scale problems. Further extensions could develop multi-fidelity and multi-objective versions [22, 23], for which surrogate modeling and Bayesian optimization are already often used [24, 25, 26].

References

[1] Joaquim R.R.A. Martins and Andrew B. Lambe. Multidisciplinary design optimization: A survey of architectures. *AIAA Journal*, 51(9):2049–2075, 2013.

[2] Nathan P. Tedford and Joaquim R.R.A. Martins. Benchmarking multidisciplinary design optimization algorithms. *Optimization and Engineering*, 11(1):159–183, 2010.

[3] Robert D Braun. Collaborative optimization: An architecture for large-scale distributed design. 1997.

[4] Robert Braun, Peter Gage, Ilan Kroo, and Ian Sobieski. Implementation and performance issues in collaborative optimization. *6th Symposium on Multidisciplinary Analysis and Optimization*, pages 295–305, 1996.

[5] Valerie Michelle Manning. Large-scale design of supersonic aircraft via collaborative optimization. *ProQuest Dissertations and Theses*, page 205, 1999.

[6] Irene A Budianto and John R Olds. Design and deployment of a satellite constellation using collaborative optimization. *Journal of spacecraft and rockets*, 41(6):956–963, 2004.

[7] Charles D McAllister and Timothy W Simpson. Multidisciplinary robust design optimization of an internal combustion engine. *J. Mech. Des.*, 125(1):124–130, 2003.

[8] R Balling and MR Rawlings. Collaborative optimization with disciplinary conceptual design. *Structural and multidisciplinary optimization*, 20(3):232–241, 2000.

[9] Angel Victor Demiguel and Walter Murray. An analysis of collaborative optimization methods. *8th Symposium on Multidisciplinary Analysis and Optimization*, (c), 2000.

[10] Natalia M. Alexandrov and Michael Lewis Robert. Analytical and computational aspects of collaborative optimization. *NASA Technical Memorandum*, (210104):1–26, 2000.

[11] Brian Douglas Roth. Aircraft family design using enhanced collaborative optimization. 3332916(August):169, 2008.

[12] Parviz M. Zadeh, Vassili V. Toropov, and Alastair S. Wood. Metamodel-based collaborative optimization framework. *Structural and Multidisciplinary Optimization*, 38(2):103–115, 2009.

[13] I. P. Sobieski and I. M. Kroo. Collaborative optimization using response surface estimation. *AIAA journal*, 38(10):1931–1938, 2000.

[14] Siyu Tao, Kohei Shintani, Ramin Bostanabad, Yu-Chin Chan, Guang Yang, Herb Meingast, and Wei Chen. Enhanced gaussian process metamodeling and collaborative optimization for vehicle suspension design optimization. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 58134, page V02BT03A039. American Society of Mechanical Engineers, 2017.

[15] B. S. Jang, Y. S. Yang, H. S. Jung, and Y. S. Yeun. Managing approximation models in collaborative optimization. *Structural and Multidisciplinary Optimization*, 30(1):11–26, 2005.

[16] Cem Anil, James Lucas, and Roger Grosse. Sorting out lipschitz function approximation. In *International Conference on Machine Learning*, pages 291–301, 2019.

[17] Stanley Osher and Ronald Fedkiw. Signed distance functions. In *Level set methods and dynamic implicit surfaces*, pages 17–22. Springer, 2003.

[18] Peter I. Frazier. A Tutorial on Bayesian Optimization. (Section 5):1–22, 2018.

[19] Jasper Snoek. Bayesian Optimization and Semiparametric Models with Applications to Assistive Technology. 2013.

[20] Jasper Snoek, Oren Ripped, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Md Mostofa Ali Patwary, Prabhat, and Ryan P. Adams. Scalable Bayesian optimization using deep neural networks. *32nd International Conference on Machine Learning, ICML 2015*, 3:2161–2170, 2015.

[21] José Miguel Hernández-Lobato, Michael A. Gelbart, Ryan P. Adams, Matthew W. Hoffman, and Zoubin Ghahramani. A general framework for constrained Bayesian optimization using information-based search. *Journal of Machine Learning Research*, 17:1–53, 2016.

[22] Haiyan Huang and Deyu Wang. A framework of multiobjective collaborative optimization. In Yong Yuan, Junzhi Cui, and Herbert A. Mang, editors, *Computational Structural Engineering*, pages 925–933, Dordrecht, 2009. Springer Netherlands.

[23] Parviz M. Zadeh and Vassili Toropov. Multi-fidelity multidisciplinary design optimization based on collaborative optimization framework. In *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, page 5504, 2002.

[24] Victor Picheny. Multiobjective optimization using gaussian process emulators via stepwise uncertainty reduction. *Statistics and Computing*, 25(6):1265–1280, 2015.

[25] Dev Rajnarayan, Alex Haas, and Ilan Kroo. A multifidelity gradient-free optimization method and application to aerodynamic design. *12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, MAO*, (September), 2008.

[26] Mostafa Meliani, Nathalie Bartoli, Thierry Lefebvre, Mohamed-Amine Bouhlel, Joaquim R. R. A. Martins, and Joseph Morlier. Multi-fidelity efficient global optimization: Methodology and application to airfoil shape design. 2019.

[27] Mark Drela. *First-Order DC Electric Motor Model*. downloaded from `http://web.mit.edu/drela/Public/web/qprop/motor1_theory.pdf` on 2020-09-12.

[28] J. B Brandt, R. W. Deters, G. K. Ananda, Dantsker O. D, and M. S. Selig. *UIUC Propeller Database, Vols 1-3*. downloaded from `https://m-selig.ae.illinois.edu/props/propDB.html` on 2020-07-12.

[29] Philip E Gill, Walter Murray, and Michael A Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM review*, 47(1):99–131, 2005.

[30] Jacob Gardner, Geoff Pleiss, Kilian Q Weinberger, David Bindel, and Andrew G Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, pages 7576–7586, 2018.

A Details of the Aircraft Design Example Problem

Nomenclature $\bar{X}$ is the subspace-level local copy of the system-level global variable X.

D_{wing}, D_{fixed}, D	Wing drag, fixed drag (drag of everything but the wing), total drag
η	Propulsion system efficiency
V	Cruise speed
L	Lift
b, S	Wing span and wing area
$W_{wing}, W_{bat}, W_{fixed}, W$	Wing weight, battery weight, fixed weight (everything but the wing), total weight
ω	propeller RPM
T	Thrust
Q_p, Q_m	Propeller and motor torque magnitude
U	Motor voltage
P_{in}	Electric power consumed by the motor
l_r	Desired Range

Propulsion System Model: The motor voltage U and propeller RPM ω are the design variables for this discipline. U is bounded between 0 and 9V, and the ω between 5000 and 10000.

The motor model is simply a 3 constants model (see for instance [27]). We use a Turnigy D2836/9 950KV Brushless Outrunner (Voltage 7.4V - 14.8V, Max current: 23.2A).

1. K_v = 950 RPM/V
2. R_M = 0.07 Ω
3. I_0 = 1.0 A

We use the following model to compute the motor torque and as a function of the voltage:

$$I = (U - \omega/K_v)/R$$
$$Q_m = (I - I_0)\frac{1}{K_q}$$

Here we choose K_q as equal to K_v (converted to units of A/Nm).

The propeller model uses measurements from the UIUC propeller database [28] for a Graupner 9x5 Slim CAM propeller data. The propeller radius is $R = 0.1143m$. The torque and thrust data for the propeller are fitted using polynomials:

$$Q_p = \rho 4/\pi^3 R^3 (0.0363\omega^2 R^2 + 0.0147V\omega R\pi - 0.0953V^2\pi^2)$$
$$C_T = 0.090 - 0.0735J - 0.1141J^2$$
$$T = C_T * \rho * n^2 * (2 * R)^4;$$
$$n = \omega/(2\pi)$$
$$J = V/n/R$$

Finally the battery weight is computed using a battery energy density of $\nu_e = 720e3$ J/kg. The power consumed by the propulsive system is $P_{in} = IU$, and the flighttime for a distance $l_r = 42000m$ is $t = l_r/V$, so we get:

$$W_{bat} = \frac{P_i n t}{\nu_e} = \frac{IUl_r}{V\nu_e}$$

Wing Model: The wing local variables are the span b, the wing area S, the lift L and the wing weight W_{wing}.

Geometrically the wing is assumed to have a taper ratio $t_r = 0.75$, a form factor $k = 2.04$, a thickness to chord ratio $t_c = 0.12$ and a tail area ratio $t_t = 1.3$. The area ratio of the airfoil is $k_{airfoil} = 0.44$. The fixed weight of the fuselage, motor and propeller is $W_fixed = 37.28N$. The fuselage has a wetted area $S_f = 0.18m^2$, a body form factor $k_f = 1.22$ and a length of $l_f = 0.6m$. We use $g = 9.81N/kg$ and $\rho = 1.225kg/m^3$.

We first use simple aerodynamic theory to compute the lift and the drag of the wing. We assume $C_{L,max} = 1.$ and a span efficiency factor $e = 0.8$. The dynamic viscosity of the air on that day is chosen to be $\nu = 1.46e - 5m^2/s$. The lift simply is equal to the weight:

$$W = W_{fixes} + W_{bat} + W_{wing}$$
$$L = W$$

Then, the induced drag coefficient can be computed:

$$C_L = 2L/(S\rho V^2)$$
$$C_{Di} = C_L^2 S/(\pi b^2 e)$$

The parasitic drag is computed assuming fully turbulent flow on the wing and the fuselage:

$$Re_{wing} = \frac{VS}{b\nu}$$
$$C_{f,wing} = 0.074/Re_{wing}^{0.2}$$
$$C_{Dp,wing} = (1 + 2t_c)C_{f,wing}kt_t$$
$$Re_{fuse} = \frac{Vl_f}{\nu}$$
$$C_{f,fuse} = 0.074/Re_{fuse}^{0.2}$$
$$C_{Dp,fuse} = C_{f,fuse}\frac{S_f}{S}k$$

Finally we add a penalty C_{Ds} for post-stall flight and complete the final drag:

$$C_{Ds} = 0.1\max(0, C_L - C_{L,max})^2$$
$$D = \frac{1}{2}\rho V^2 S \left(C_{Di} + C_{Dp,wing} + C_{Dp,fuse} + C_{Ds}\right)$$

The wing is assumed to be made of a main carbon spar along with styrofoam. The foam weight is simply related to the wing area. We use $\rho_{foam} = 40kg/m^3$

$$W_{foam} = g\rho_{foam}\frac{S^2}{b}t_c k_{airfoil}$$

The gauge of the carbon spar is computed based on stress and deflection, with a maximum thickness of $\tau_m = 1.14mm$. The maximum possible stress is $\sigma_{max} = 4.413e9N/m^2$, the Young modulus is $E = 2.344e11N/m^2$ and the density is $\rho_{carbon} = 1380kg/m^3$. The radius of the spar is computed using the wing dimensions: $r_s = \frac{S}{4b}t_c$. The minimum carbon thickness to withstand the stress is computed by:

$$M_{root} = Lb/8$$
$$I = \pi r_s^3 \tau_m$$
$$\tau_{stress} = \tau_m Lb/8r_s/I/\sigma_{max}/0.07$$

The minimum carbon thickness to avoid deflections is computed by:

$$\delta = Lb^4/(64EI)$$
$$\tau_{defl} = \tau_m \frac{2\delta}{b}/0.07$$

The actual gauge is the thickest one: $\tau = \max(\tau_{defl}, \tau_{stress}, \tau_m)$. The total mass can then be computed:

$$W_{spar} = 2\pi r_s \tau b\rho_{carbon}$$
$$W_{wing} = W_{spar} + W_{foam}$$

The optimal value of the problem happens for

$$
\begin{aligned}
D &= 2.15N \\
V &= 13.71\ m/s \\
b &= 2.7574\ m \\
S &= 0.397\ m^2 \\
RPM &= 8127 \\
U &= 9V
\end{aligned}
$$

B   Including the Subspace Problem Meta-Information in the Loss Function

As mentioned in the main paper, the subspace optimization problem is a projection onto the disciplinary feasible set.

$$
J_i^*(z) = \min_{\bar{z}_i \in \{\bar{z}_i \mid \exists x_i\ c_i(x_i, \bar{z}_i) \leq 0\}} \|\bar{z}_i - z\|_2^2 \tag{9}
$$

[4] proves that, for each evaluation $J_i^*(z)$, we can compute the gradient of the converged point:

$$
\nabla J_i(z) = (z - \bar{z}_i^*)
$$

where $\bar{z}_i^*$ is the optimal point of the subspace evaluation 9. We also know from 5 that $J^i(\bar{z}_i^*) = 0$, and $\nabla J^i(\bar{z}_i^*) = \nabla J^i(z)$, In cases like section 3 where we are interested in information about $\sqrt{J_i^*}$, the gradient information can be derived easily from this using the chain rule.

A slightly modified version of the loss function of equation 4 is then:

$$
l(z^{(j)}) =
\begin{cases}
|h_i(z^{(j)}) - \sqrt{J_i^{*(j)}}| & +\|\nabla h_i(z^{(j)}) - \nabla\sqrt{J_i^*}^{(j)}\|_1 + \\
\quad |h_i((\bar{z}_i^*)^{(j)})| & +\|\nabla h_i((\bar{z}_i^*)^{(j)}) - \nabla\sqrt{J_i^*}^{(j)}\|_1 & \text{if } z^{(j)} \text{ is infeasible} \\
\max(h_i(z^{(j)}), 0) & & \text{otherwise}
\end{cases} \tag{10}
$$

C   Details on the Fits of a Disk of Increasing Dimension

In this experiment, there are 5 different training sets each containing 50 points, while the test set remains the same and contains 500 points. The experiment is repeated on each train sets in order to obtain error bars on the reported performance.

All feedforward networks have 3 layers of 8 units each. They are trained with the Adam Optimizer with $\beta = (0.9, 0.999)$ and a 10^{-3} learning rate tuned with gridsearch.

The Lipshitz networks use linear layers orthonormalized after each update using 15 iterations of Bjorck's procedure of order 1 with $\beta = 0.5$. They use a full vector sort as activation function (GroupSort with groupsize 1).

D   Details on the Solution to the Aircraft Design Problem

D.1   Details and Parameter Choices

The system level problem is solved 20 times using each approach, using the same random seed for each solution approach. The wing and propulsion subspace problems (see equation 3) are solved using the sequential quadratic programming solver SNOPT [29].

The airspeed V is bound to be between 5 and 15 m/s, the battery mass between 0.1 and 1 kg, and the drag D between 1 and 6 N.

For both surrogate model approaches (Gaussian process and SDF), the initial number of samples N_{INI} is chosen to be simply 1, which allows to compare with the direct approach more easily. The choice of a new candidate point is then done by optimizing the surrogate model, as shown in 8. This optimization is also performed by SNOPT, with 15 random restarts.

Regular CO: For regular CO, the system level problem is solved directly using SNOPT. A feasibility threshold of 10^{-4} is used to help convergence.

Gaussian processes baseline: At each iteration, a Gaussian Process is fit to the currently available data for J_i^*, as well as the gradient information. The prior mean is simply the zero function, and the kernel is a scaled squared-exponential kernel. The hyper parameters are tuned using type II maximum likelihood minimization and the Adam Optimizer with $\beta = (0.9, 0.999)$ and a 10^{-3} learning rate for 200 iterations. The code uses the GpyTorch framework [30]. Again, a feasibility threshold of 10^{-4} is required to help convergence.

SDF neural networks: The SDF surrogate models use three layers of 12 units with GroupSort activation functions (groupsize of 1). The linear layers are orthonormalized after each update using 15 iterations of Bjorck's procedure with order 1 and $\beta = 0.5$. They are trained with the Adam optimizer with $\beta = (0.9, 0.999)$ and a learning rate that is tuned at each iteration (we use the best network after out of 10 randomly sampled learning rates between 10^{-5} and 10^{-3}).