Data Fitting with Signomial Programming
Compatible Difference of Convex Functions

Cody Karcher

Received: date / Accepted: date

Abstract Signomial Programming (SP) has proven to be a powerful tool for engineering design optimization, striking a balance between the computational efficiency of Geometric Programming (GP) and the extensibility of more general optimization methods like Sequential Quadratic Programming (SQP). But when an existing engineering analysis tool is incompatible with the mathematics of the SP formulation, options are limited. Previous literature has suggested schemes for fitting GP compatible models to pre-computed data, but no methods have yet been proposed that take advantage of the increased modeling flexibility available in SP. This paper describes a new Soft Difference of Max Affine (SDMA) function class that is constructed from existing methods of GP compatible fitting and the theory of Difference of Convex (DC) functions. When a SDMA function is fit to data in log-log transformed space, it becomes either a signomial or a set of signomials upon inverse transformation. Three examples of fitting are presented here, including simple test cases in 2D and 3D, and a fit to the performance data of the NACA 24xx family of airfoils. In each case, RMS error is driven to less than 1%.

Keywords Data Fitting · Signomial Programming · Log Convexity · Difference of Convex Functions

1 Motivation

Geometric Programming (GP) and Signomial Programming (SP) are two related classes of non-linear optimization formulations that have recently been applied, with great success, to the field of aircraft design.

C. Karcher
Department of Aeronautics and Astronautics, Massachusetts Institute of Technology
77 Massachusetts Avenue, Cambridge, MA 02139, USA
E-mail: ckarcher@mit.edu
Despite these successes, the GP and SP formulations do not allow the use of black box analysis tools that are common in practical design problems [16]. In aircraft design, these tools include high fidelity analyses such as Computational Fluid Dynamics (CFD) and Finite Element Analysis (FEA), and low fidelity analyses like XFOIL [5] and AVL [6]. Previous work by Hoburg et. al. [11] addressed this issue for Geometric Programming by proposing three methods for fitting a GP compatible function to a pre-computed dataset, but the additional modeling flexibility available in Signomial Programming goes untreated. This work fills this gap by proposing a method for fitting a SP compatible function to data in cases where the Hoburg methods [11] lack sufficient accuracy.

2 Geometric Programming and Signomial Programming

2.1 Geometric Programming

Geometric Programs are built upon two fundamental building blocks: monomial and posynomial functions. A monomial function is defined as the product of a leading constant with each variable raised to a real power [2]:

\[ m(x) = cx_1^{a_1}x_2^{a_2}...x_n^{a_n} = c\prod_{i=1}^{N}x_i^{a_i} \] (1)

A posynomial is simply the sum of monomials [2], which can be defined in notation as:

\[ p(x) = m_1(x) + m_2(x) + ... + m_n(x) = \sum_{k=1}^{K}c_k\prod_{i=1}^{N}x_i^{a_{ik}} \] (2)

From these two building blocks, it is possible to construct the definition of a GP in standard form [2]:

\[
\begin{align*}
\text{minimize} & \quad p_0(x) \\
\text{subject to} & \quad m_i(x) = 1, \: i = 1, \ldots, N \\
& \quad p_j(x) \leq 1, \: j = 1, \ldots, M
\end{align*}
\] (3)

When constraints and objectives can be written in the form specified in Equation 3 it is said that the problem is GP compatible.

2.2 Signomial Programming

Signomial Programs (SPs) are a logical extension of Geometric Programs that allow the inclusion of negative leading constants and a broader set of equality
constraints. The key building blocks of the Signomial Programing are signomials, which are the difference between two posynomials \( p(x) \) and \( n(x) \):

\[ s(x) = p(x) - n(x) = \sum_{k=1}^{K} c_k \prod_{i=1}^{N} x_i^{a_{ik}} - \sum_{p=1}^{P} d_p \prod_{i=1}^{N} x_i^{g_{ip}} \] (4)

The posynomial \( n(x) \) is often referred to as a ‘neginomial’ because it is made up of all of the terms with negative leading coefficients. From this definition, the standard form for a Signomial Program can be constructed [13]:

\[
\text{minimize} \quad \frac{p_0(x)}{n_0(x)} \\
\text{subject to} \quad s_i(x) = 0, \ i = 1, \ldots, N \\
\quad s_j(x) \leq 0, \ j = 1, \ldots, M \] (5)

however, another useful form is [13]:

\[
\text{minimize} \quad \frac{p_0(x)}{n_0(x)} \\
\text{subject to} \quad \frac{p_i(x)}{n_i(x)} = 1, \ i = 1, \ldots, N \\
\quad \frac{p_j(x)}{n_j(x)} \leq 1, \ j = 1, \ldots, M \] (6)

In this alternative form, the neginomial is added to both sides, and then used as a divisor to construct an expression either equal to or constrained by a value of one.

SPs are not convex upon log-log transformation unlike their GP counterparts and therefore must be solved using general non-linear solution methods. However many signomial programs of interest still benefit from an underlying structure which is well approximated by a log-log convex formulation, and as a result can be efficiently solved by a series of GP approximations via the Difference of Convex Algorithm (DCA). In this process the various neginomials \( n(x) \) are replaced with local monomial approximations, yielding substantial benefits over other non-linear solution methods (see [14] and [21] for discussion).

3 Difference of Convex Functions for Data Fitting

3.1 Difference of Convex (DC) Functions

Signomials are extremely versatile functions, far more versatile than posynomials or monomials. But the construction of signomials is rather unique since they are the difference between two log-log convex functions.
Modern study of Difference of Convex (DC) functions is widely attributed to have started with a paper by Hartman et. al. [8], which states that a continuous function of bounded variation \( f(x) \) can be written as the difference of two convex functions \( g(x) \) and \( h(x) \) [8]:

\[
f(x) = g(x) - h(x)
\]  

(7)

which clearly mirrors signomial construction. Further extensions have opened up a wide variety of well known properties that DC functions possess [1,17].

The implication here is that a significant subset of continuous functions can be decomposed into the difference of two convex functions, including non-differentiable continuous functions. By extension, it should be possible to fit a DC function to nearly any dataset that can be well approximated with a continuous function.

Hoburg et. al. [11] provided three ways of fitting functions to datasets that were convex under the log-log transformation, but these methods entirely ignore the additional modeling flexibility available in Signomial Programming. So consider the use of a DC function under the log-log transformation, specifically the difference between two of the functions proposed by Hoburg et. al. [11].

3.2 Function Definitions

3.2.1 Notation

Consider a data set sampled from a black box mapping from \( \mathbb{R}^N \rightarrow \mathbb{R} \). Consistent with the notation in Hoburg [11] let the vector \( u_j \) represent the independent variables in \( \mathbb{R}^N \) for data point \( j \) and the scalar \( w_j \) represent the output in \( \mathbb{R} \) for data point \( j \). The logspace variables are then represented as \( x_j = \log u_j \) and \( y_j = \log w_j \).

3.2.2 Difference of Max Affine (DMA) Functions

The first function proposed by Hoburg et. al. [11] is the Max Affine (MA) function:

\[
f_{\text{MA}}(x) = \max_{k=1,\ldots,K} \left[ b_k + a_k^T x \right]
\]  

(8)

This function class is known to create a convex epigraph. In fact, any convex function can be reasonably approximated as a max affine function given a sufficient number of affine functions, \( K \).

Now consider the difference between two of these max affine functions (Equation 8), which henceforth will be called the Difference of Max Affine (DMA) function:

\[
f_{\text{DMA}}(x) = \max_{k=1,\ldots,K} \left[ b_k + a_k^T x \right] - \max_{m=1,\ldots,M} \left[ h_m + g_m^T x \right]
\]  

(9)
The subtracted term is represented by an entirely separable Max Affine function which is defined by fitting parameters $M$, $h$, and $g$.

While the Max Affine function has a realizable transformation back from logspace, the Difference of Max Affine function has no such transformation due to the inability to readily construct a meaningful epigraph or subgraph. Despite this, the DMA function is quite rapid to fit, and could be of application in other areas where a cheap surrogate is desired for non-convex fitting. Here, the DMA function is a useful as an intellectual building block, and as a seed for the function class that follows.

### 3.2.3 Soft Difference of Max Affine (SDMA) Functions

The second function proposed by Hoburg et. al. [11] is the Soft-Max Affine (SMA) function:

$$f_{SMA}(x) = \frac{1}{\alpha} \log \sum_{k=1}^{K} \exp \left( \alpha (b_k + a_k^T x) \right)$$  \hspace{1cm} (10)

The SMA function uses a global softening parameter ($\alpha$) to ‘smooth’ the sharp corners of the Max Affine function and has the benefit of requiring far fewer affine terms $K$ to capture smooth convex functions with reasonable accuracy. However, the global softening parameter results in a poor representation in regions where the curvature deviates substantially from the global average.

Consider the following function which is the difference between two Soft-Max Affine functions (Equation 10):

$$f_{SDMA}(x) = \frac{1}{\alpha} \log \sum_{k=1}^{K} \exp \left( \alpha (b_k + a_k^T x) \right) - \frac{1}{\beta} \log \sum_{m=1}^{M} \exp \left( \beta (h_m + g_m^T x) \right)$$

which is henceforth referred to as the Soft Difference of Max Affine (SDMA) function. As with SMA functions when compared to MA functions, the SDMA function compared to the DMA function significantly decreases the number of terms $K + M$ required to approximate smooth functions to a reasonable degree of accuracy by introducing scalar softening parameters $\alpha$ and $\beta$. 
Transforming the SDMA function back to the optimization variables $u_j$ and $w_j$ proceeds as follows:

\[
y = \frac{1}{\alpha} \log \sum_{k=1}^{K} \exp \left( \alpha (b_k + a_k^T x) \right) - \frac{1}{\beta} \log \sum_{m=1}^{M} \exp \left( \beta (h_m + g_m^T x) \right)
\]

\[
\exp(y) = \exp \left( \frac{1}{\alpha} \log \sum_{k=1}^{K} \exp \left( \alpha (b_k + a_k^T x) \right) \right) - \frac{1}{\beta} \log \sum_{m=1}^{M} \exp \left( \beta (h_m + g_m^T x) \right)
\]

\[
w = \exp \left( \frac{1}{\alpha} \log \sum_{k=1}^{K} \exp \left( \alpha (b_k + a_k^T x) \right) \right)
\]

\[
w = \exp \left( \frac{1}{\beta} \log \sum_{m=1}^{M} \exp \left( \beta (h_m + g_m^T x) \right) \right)
\]

(12)

At this point it is obvious from the definition of a posynomial function that the form will reduce to:

\[
w = \left( \sum_{k=1}^{K} e^{\alpha b_k} \prod_{i=1}^{N} u_i^{\alpha a_{ik}} \right)^{1/\alpha}
\]

\[
w = \left( \sum_{m=1}^{M} e^{\beta h_m} \prod_{i=1}^{N} u_i^{\beta g_{im}} \right)^{1/\beta}
\]

(13)

This is not compatible with the SP formulation, but consider the following substitutions:

\[
p_{\text{convex}} = \sum_{k=1}^{K} e^{\alpha b_k} \prod_{i=1}^{N} u_i^{\alpha a_{ik}}
\]

\[
p_{\text{concave}} = \sum_{m=1}^{M} e^{\beta h_m} \prod_{i=1}^{N} u_i^{\beta g_{im}}
\]

(14)

(15)

Which then reduces Equation 13 to:

\[
w = \left( p_{\text{convex}} \right)^{1/\alpha}
\]

\[
w = \left( p_{\text{concave}} \right)^{1/\beta}
\]

(16)

Thus, taking the three constraints (Equations 14, 15, 16) as a set does result in an SP compatible scheme. This method of substitution is consistent with other approaches to constructing GP and SP compatible constraints [2]. The table below outlines the operators that should be used in the new constraint set given the original mapping:
Table 1 Operators For a Given Initial Constraint

| Constraint Operator | ≤ | ≥ |
|---------------------|---|---|
| Original            | = |   |
| \(w\)               | ≤ | ≥ |
| \(w_{\text{convex}}\) | ≥ | ≤ |
| \(w_{\text{concave}}\) | ≤ | ≥ |

3.2.4 Consideration of Implicit Soft Difference of Max Affine (ISDMA) Functions

Hoburg et. al. [11] proposes a third function class, Implicit Soft Max Affine (ISMA). Unlike MA and SMA, the ISMA function class is an implicit function of \(y\):

\[
\tilde{f}_{\text{ISMA}}(x, y) = \log \sum_{k=1}^{K} \alpha_k \exp \left( \alpha_k (b_k + a_k^T x - y) \right) \quad (17)
\]

Since the ISMA function class provided a fit with lower RMS error than SMA, it is tempting to write an implicit soft difference of max affine function as:

\[
\tilde{f}_{\text{ISDMA}}(x, y) = \log \sum_{k=1}^{K} \alpha_k \exp \left( \alpha_k (b_k + a_k^T x - y_{\text{convex}}) \right) - \log \sum_{m=1}^{M} \beta_m \exp \left( \beta_m (h_m + g_m^T x - y_{\text{concave}}) \right) \quad (18)
\]

Fitting one of these functions to data would require that the data first be decomposed into convex \((y_{\text{convex}})\) and concave \((y_{\text{concave}})\), which has no obvious solution a priori. Fortunately, in practice the SDMA function is highly versatile and captures regions with varying curvature for even low fit orders due to the DC construction, somewhat negating the desire for ISDMA functions in the first place. However, non-differentiable corners and cusps would benefit from this type of function if the issues could be successfully resolved.

4 How Functions are Fit

The following process is used for fitting functions to the data points \((u_j, w_j)\):

1. Apply the log transformation \((x_j, y_j) = (\log u_j, \log w_j)\)
2. Fit a function \(f\) to the transformed data such that \(y_j \approx f(x_j)\)
3. Relax the equality \(y_j = f(x_j)\) as appropriate to construct the desired constraint
This process is nearly identical to that proposed by Hoburg et. al. [11], but two differences exist. First, the work here does not require convex functions in Step 2. Second, while Hoburg et. al. [11] could readily relax each equality constraint to construct a convex epigraph in Step 3, there is no similar approach when considering functions that are generally not convex and so selecting the appropriate constraint operator (=, ≤, or ≥) will depend on the constraint being considered.

For a set of \( m \) data points, the fitting problem can be written as an unconstrained least squares optimization problem:

\[
\min_{\boldsymbol{\gamma}} \sum_{i=1}^{m} (f(x_i; \boldsymbol{\gamma}) - y_i)^2
\]  

(19)

where the fitting parameters are stacked in the vector \( \boldsymbol{\gamma} \).

Based on the work of Hoburg [11], all of the functions presented here were fit using a Levenberg-Marquardt (LM) algorithm. The LM algorithm computes a step size at each iteration, but requires a Jacobian to be computed at each step. Derivatives are therefore presented for each function class with respect to the fitting parameters in the vector \( \boldsymbol{\gamma} \) in the two sections that follow.

For each SDMA function fit, a DMA function is first fit in order to provide an initial guess for the SDMA fitting algorithm. The ability to quickly achieve a DMA initial guess is critical to the success of SDMA functions, as starting from random positions rarely yields a good result.

In addition, the gradient based nature of the LM algorithm meant that the fitting process required a number of random restarts to be performed from varying initial guesses. In general 30-100 random restarts were performed for the work presented here, with the specific number varying with the individual fitting problem being considered.

Non-gradient based methods such as a Genetic Algorithm or Particle Swarm Optimization were considered for fitting, but there was no evidence that this approach would be superior to the LM algorithm established in the literature. A Genetic Algorithm was actually tested, but had worse performance than the LM method even in simple 2D cases.

4.1 Derivatives for the DMA Function

\[
\frac{\partial f_{DMA}}{\partial b_i} = \begin{cases} 
1, & \text{if } i = \arg\max_k b_k + a_k^T x \\
0, & \text{otherwise}
\end{cases}
\]  

(20)

\[
\frac{\partial f_{DMA}}{\partial a_i} = \begin{cases} 
x^T, & \text{if } i = \arg\max_k b_k + a_k^T x \\
0, & \text{otherwise}
\end{cases}
\]  

(21)

\[
\frac{\partial f_{DMA}}{\partial h_i} = \begin{cases} 
-1, & \text{if } i = \arg\max_k b_k + a_k^T x \\
0, & \text{otherwise}
\end{cases}
\]  

(22)

\[
\frac{\partial f_{DMA}}{\partial g_i} = \begin{cases} 
-x^T, & \text{if } i = \arg\max_k b_k + a_k^T x \\
0, & \text{otherwise}
\end{cases}
\]  

(23)
4.2 Derivatives for the SDMA Function

\[
\frac{\partial f_{\text{SDMA}}}{\partial b_i} = \frac{\exp(\alpha (b_i + a_i^T x))}{\sum_{k=1}^{K} \exp(\alpha (b_k + a_k^T x))}
\]

\[
\frac{\partial f_{\text{SDMA}}}{\partial a_i} = \frac{x^T \cdot \exp(\alpha (b_i + a_i^T x))}{\sum_{k=1}^{K} \exp(\alpha (b_k + a_k^T x))}
\]

\[
\frac{\partial f_{\text{SDMA}}}{\partial \alpha} = \frac{1}{\alpha} \frac{\sum_{k=1}^{K} (b_k + a_k^T x) \exp(\alpha (b_k + a_k^T x))}{\sum_{k=1}^{K} \exp(\alpha (b_k + a_k^T x))} - \frac{1}{\alpha^2} \log \sum_{k=1}^{K} \exp(\alpha (b_k + a_k^T x))
\]

\[
\frac{\partial f_{\text{SDMA}}}{\partial (\log \alpha)} = \frac{\sum_{k=1}^{K} (b_k + a_k^T x) \exp(\alpha (b_k + a_k^T x))}{\sum_{k=1}^{K} \exp(\alpha (b_k + a_k^T x))} - \frac{1}{\alpha} \log \sum_{k=1}^{K} \exp(\alpha (b_k + a_k^T x))
\]

\[
\frac{\partial f_{\text{SDMA}}}{\partial g_i} = -\frac{x^T \cdot \exp(\beta (h_i + g_i^T x))}{\sum_{m=1}^{M} \exp(\beta (h_m + g_m^T x))}
\]

\[
\frac{\partial f_{\text{SDMA}}}{\partial h_i} = -\frac{\exp(\beta (h_i + g_i^T x))}{\sum_{m=1}^{M} \exp(\beta (h_m + g_m^T x))}
\]

\[
\frac{\partial f_{\text{SDMA}}}{\partial \beta} = \frac{-1}{\beta} \frac{\sum_{m=1}^{M} (h_m + g_m^T x) \exp(\beta (h_m + g_m^T x))}{\sum_{m=1}^{M} \exp(\beta (h_m + g_m^T x))} + \frac{1}{\beta^2} \log \sum_{m=1}^{M} \exp(\beta (h_m + g_m^T x))
\]

\[
\frac{\partial f_{\text{SDMA}}}{\partial (\log \beta)} = \frac{-\sum_{m=1}^{M} (h_m + g_m^T x) \exp(\beta (h_m + g_m^T x))}{\sum_{m=1}^{M} \exp(\beta (h_m + g_m^T x))} + \frac{1}{\beta} \log \sum_{m=1}^{M} \exp(\beta (h_m + g_m^T x))
\]

5 Demonstrations of the Fitting Models

5.1 A 2D Fitting Problem

Consider the 2D function, which uses the log transformed variables \( x \) and \( y \):

\[
y = \max \left[ -6x - 6, x^4 - 3x^2 \right] \quad (25)
\]
This function was selected to test the SDMA function at a non-differentiable point and over variations in curvature. A set of 101 data points were sampled on even intervals over the function from $x = [-2, 2]$, and 5th order MA, SMA, ISMA, and SDMA functions were fit to this data:

Fig. 1 Fifth order fits to the simple 2D function. RMS Error is as follows: MA: 44.6%, SMA: 44.6%, ISMA: 44.8%, SDMA: 0.149%

The convex fitting methods all converge to nearly identical representations, none of which capture the significant non-convexity to any degree. In contrast, the SDMA function captures nearly all of the non-convexity in the function, to an RMS error of $1.5 \times 10^{-3}$. The key takeaway here is that unlike SMA functions, which can only capture a single curvature due to the single parameter $\alpha$, the SDMA function can capture complex, multi-radius curvature as a direct result of the DC construction.

Increasing fit order also improves the RMS error as expected:

Fig. 2 Variation of RMS error with increasing fit order
5.2 A 3D Fitting Problem

After proving on a 2D case, the next logical step was to consider a 3D dataset. Similar to the 2D test case, it was desirable to have a non-differentiable region and complex 3D curvature. Consider this test function, the eigenfunction of the wave equation:

![Eigenfunction of the wave equation](image1)

This function has become well known as the Matlab logo.

An SDMA fit yields a reasonably accurate surrogate (consider here 9th order):

![Fitted 9th order function and error](image2)

As might be expected, the fitting scheme struggles along the non-differentiable L-shaped curve, and at the point of the L specifically. Error near the point
(0,1) is due to an inflection in the data where curvature changes from concave to convex in a very small region, and this is difficult to capture in the fit.

And as before, plotting RMS error as a function of increasing fit order shows significant improvement for higher order fits:

![Variation of RMS error with increasing fit order](image)

**Fig. 5** Variation of RMS error with increasing fit order

5.3 Fitting XFOIL Performance Data of the NACA 24xx Family of Airfoils

Given the success of the two test cases, the final step was to validate on a more representative engineering example. The Hoburg et. al. [11] work was able to fit performance data for NACA 24xx airfoils generated from XFOIL [5], but only by considering curves of lift coefficient vs. drag coefficient. In many cases, it is more useful to have two separate curves of lift coefficient vs. angle of attack and drag coefficient vs. angle of attack, but the CL vs. α curve is not compatible with log-log convex fitting techniques.

Consider the problem of fitting the following function:

\[ C_L = f(\alpha, Re, \tau) \]  

(26)

where \( C_L \) is the airfoil lift coefficient, \( \alpha \) is the airfoil angle of attack, \( Re \) is the Reynolds number, and \( \tau \) is the airfoil thickness (ex, a \( \tau = 0.12 \) would be a NACA 2412 airfoil).

XFOIL was used to generate a grid of data from \( \alpha = [1, 23], Re = [10^6, 10^7], \) and \( \tau = [.09, .21] \). This data was then fit with an SDMA function of fit orders varying from 3 to 10. Plotting below are the results of a 5th order fit:
Fig. 6 Curves of Lift Coefficient vs. Angle of Attack for the NACA 24xx family of airfoils

As before, plotting the RMS error vs. increasing fit order shows improvement with an increased number of fit terms:
Fig. 7 Variation of RMS error with increasing fit order

6 Conclusions

This work serves as a successful development and validation of a Signomial Programming compatible method for fitting black box data. The Soft Difference of Max Affine (SDMA) function is a crucial link in the chain from Geometric Programming, where low fidelity analysis methods must be used, to Sequential Quadratic Programming or the more recently proposed Logspace Sequential Quadratic Programming [12], which typically impose no bounds on the type of analysis that can be utilized.

Acknowledgements

This material is based on research sponsored by the U.S. Air Force under agreement number FA8650-20-2-2002. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Air Force or the U.S. Government.

The author would like to thank Bob Haimes, Woody Hoburg, and Berk Ozturk for their input into the technical matter, along with the EnCAPS Technical Monitor Ryan Durscher.

References

1. Bačák, M., Borwein, J.M.: On difference convexity of locally lipschitz functions. Optimization 60(8-9), 961–978 (2011)
2. Boyd, S., Kim, S.J., Vandenberghe, L., Hassibi, A.: A tutorial on geometric programming. Optimization and Engineering 8(1), 67–127 (2007)
3. Brown, A., Harris, W.: A vehicle design and optimization model for on-demand aviation. In: 2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, p. 0105 (2018)
4. Burton, M., Hoburg, W.: Solar and gas powered long-endurance unmanned aircraft sizing via geometric programming. Journal of Aircraft 55(1), 212–225 (2018)
5. Drela, M.: Xfoil: An analysis and design system for low reynolds number airfoils. In: Low Reynolds Number Aerodynamics, pp. 1–12. Springer (1989)
6. Drela, M., Youngren, H.: Avl 3.30 user primer. Tech. rep., Massachusetts Institute of Technology (2010)
7. Hall, D.K., Dowdle, A., Gonzalez, J., Trollinger, L., Thalheimer, W.: Assessment of a boundary layer ingesting turboelectric aircraft configuration using signomial programming. In: 2018 Aviation Technology, Integration, and Operations Conference, p. 3973 (2018)
8. Hartman, P., et al.: On functions representable as a difference of convex functions. Pacific Journal of Mathematics 9(3), 707–713 (1959)
9. Hoburg, W., Abbeel, P.: Fast wind turbine design via geometric programming. In: 54th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference (2013)
10. Hoburg, W., Abbeel, P.: Geometric programming for aircraft design optimization. AIAA Journal 52(11), 2414–2426 (2014)
11. Hoburg, W., Kirschen, P., Abbeel, P.: Data fitting with geometric-programming-compatible softmax functions. Optimization and Engineering pp. 1–22 (2016)
12. Karcher, C.: Logspace sequential quadratic programming for design optimization. arXiv preprint arXiv:2105.14441 (2021)
13. Kirschen, P.G., Burnell, E.E., Hoburg, W.W.: Signomial programming models for aircraft design. In: 54th AIAA Aerospace Sciences Meeting (2016)
14. Kirschen, P.G., York, M.A., Ozturk, B., Hoburg, W.W.: Application of signomial programming to aircraft design. Journal of Aircraft 55(3), 965–987 (2018)
15. Lin, B., Carpenter, M., de Weck, O.: Simultaneous vehicle and trajectory design using convex optimization. In: AIAA Scitech 2020 Forum, p. 0160 (2020)
16. Martins, J.R., Lambe, A.B.: Multidisciplinary design optimization: A survey of architectures. AIAA Journal 51(9), 2049–2075 (2013)
17. Rudin, C.: 15.097 prediction: Machine learning and statistics. Massachusetts Institute of Technology: MIT OpenCourseWare, https://ocw.mit.edu/. License: Creative Commons BY-NC-SA. https://ocw.mit.edu/courses/sloan-school-of-management/15-097-prediction-machine-learning-and-statistics-spring-2012/projects/MIT15_097S12_proj5.pdf (2012)
18. Saab, A., Burnell, E., Hoburg, W.W.: Robust designs via geometric programming. arXiv (2018)
19. Torenbeek, E.: Advanced Aircraft Design: Conceptual Design, Analysis and Optimization of Subsonic Civil Airplanes, 2nd edn. John Wiley & Sons, Ltd., Chichester, United Kingdom (2013)
20. York, M.A., Hoburg, W.W., Drela, M.: Turbofan engine sizing and tradeoff analysis via signomial programming. Journal of Aircraft 55(3), 988–1003 (2018)
21. York, M.A., Ozturk, B., Burnell, E., Hoburg, W.W.: Efficient aircraft multidisciplinary design optimization and sensitivity analysis via signomial programming. AIAA Journal 56(11), 4546–4561 (2018)