Lecture Notes in Computational Science and Engineering, Volume 64, 2008, p271-280

Robust Aircraft Conceptual Design using Automatic Differentiation in Matlab

Mattia Padulo[1], Shaun A. Forth[2], and Marin D. Guenov[1]

Abstract The need for robust optimisation in aircraft conceptual design, for which the design parameters are assumed stochastic, is introduced. We highlight two approaches, first-order method of moments and Sigma-Point reduced quadrature, to estimate the mean and variance of the design's outputs. The method of moments requires the design model's differentiation and here, since the model is implemented in Matlab, is performed using the AD tool MAD. Gradient-based constrained optimisation of the stochastic model is shown to be more efficient using AD-obtained gradients than finite-differencing. A post-optimality analysis, performed using AD-enabled third-order method of moments and Monte-Carlo analysis, confirms the attractiveness of the Sigma-Point technique for uncertainty propagation.

Key words: Aircraft conceptual design, uncertainty estimation, forward mode, higher derivatives, MAD

1 Introduction

In the *conceptual phase* of aeronautical design *low fidelity models* are used to predict an aircraft's performance from a moderate number of *design parameters*. For example, given thrust-to-weight ratio and wing loading, critical design requirements such as approach speed, rate of climb and take-off distance may be quantified. Such considerations also hold for major aircraft subsystems, e.g., estimation of engine performance from parameters such as turbine entry temperature, overall pressure ratio and bypass ratio [17].

[1] Engineering Design Group, Aerospace Engineering Department, School of Engineering, Cranfield University Bedfordshire MK43 0AL, UK, {M.Padulo,M.D.Guenov}@cranfield.ac.uk .[2] Applied Mathematics & Scientific Computing Group, Engineering Systems Department, Defence College of Management and Technology, Cranfield University, Shrivenham, Swindon SN6 8LA, UK, S.A.Forth@cranfield.ac.uk

The conceptual model is then optimised, typically a constrained multi-objective optimisation, giving an aircraft configuration used to initiate detailed design processes such as geometric design via a CAD model leading to computationally (or experimentally) expensive analyses of structural, aerodynamic, propulsive and control aspects. These analyses may indicate inadequacies in the conceptual design leading to its change and repeated design iterations until a suitable design is found.

To minimize nugatory cycling between conceptual and detailed design analyses there has been much interest in seeking designs whose performance is relatively insensitive to downstream changes. This reduces the likelihood of requiring any large scale design changes when the detailed analysis is performed [3]. Such *robust design* involves modelling the design parameters as taken from statistical distributions. Thus, accounting for robustness increases the complexity of the original deterministic problem. In the general context of nonlinear optimisation, if we assume that the design task is to minimise some deterministic objective (e.g., fuel consumption) subject to multiple deterministic constraints (e.g., maximum wing span, range, ...) then one assumes the design parameters are taken from known independent statistical distributions and then classically performs the estimation of the means and variances of the objective and constraints. Then a robust objective and constraints are formed which favour designs with: low objective value, small objective standard deviation, and a high likelihood of satisfying the deterministic constraints. This robust optimisation problem is then solved numerically [12].

Others have stressed the improvements Automatic Differentiation (AD) confers on the accuracy and the efficiency of Engineering Design and its robust extensions, but focused on Fortran- and C/C++ coded applications [1, 2, 15]. Here we demonstrate some of the benefits AD gives to robust optimisation for aircraft conceptual design of an industrially relevant aircraft sizing test case implemented in Matlab. Section 2 presents two strategies to uncertainty estimation considered for our sizing problem. Section 3 presents extensions to the MAD package [5] to facilitate these strategies. The test case and results of the robust optimisation are presented, together with a post-optimality analysis, in Sect. 4. Section 5 concludes.

2 Robust Design Optimization

We assume that the conceptual design analyses are performed by functions $f(\mathbf{x})$ and $g_i(\mathbf{x})$, $i = 1,2,\ldots,r$, where $\mathbf{x} \in \mathbb{R}^n$ is the vector of the design variables. The *deterministic design optimization* problem is hence formulated as follows:

$$\min_{\mathbf{x}} f(\mathbf{x}) \text{ such that: } g_i(\mathbf{x}) \leqslant 0, i = 1,2,\ldots,r,\ \mathbf{x}_L \leqslant \mathbf{x} \leqslant \mathbf{x}_U. \tag{1}$$

To ensure design robustness, the components of $\mathbf{x}$ are assumed to be stochastic and taken from known independent probability distributions with mean $\mathbf{E}_x$ and variance $\mathbf{V}_x$, so rendering the objective and constraints stochastic. The robust attribute of the objective function is achieved by simultaneously minimizing its variance V_f

and its expectation E_f, aggregated in a suitable robust objective $F(E_f(\mathbf{x}),V_f(\mathbf{x}))$, with respect to the mean of the input variables. Robustness of the constraints, each distributed with mean E_{g_i} and variance V_{g_i}, is sought by ensuring their probabilistic satisfaction via robust constraint functions $G_i(E_{g_i}(\mathbf{x}),V_{g_i}(\mathbf{x}))$. The range of the independent variables is also defined probabilistically. Hence (1) becomes:

$$\begin{aligned}&\min_{\mathbf{E}_x} F(E_f(\mathbf{x}),V_f(\mathbf{x})), \text{ such that:}\\ &G_i(E_{g_i}(\mathbf{x}),V_{g_i}(\mathbf{x})) \leqslant 0\,, i = 1,2,\ldots,r,\\ &P(\mathbf{x}_L \leqslant \mathbf{E}_x \leqslant \mathbf{x}_U) \geq \mathbf{p}_{bounds},\end{aligned}$$

where $\mathbf{p}_{bounds}$ is the prescribed probability with which the mean of the design variables belongs to the original deterministic range. If all variables are continuous, the mean and variance of $f(\mathbf{x})$ are given by:

$$E_f(\mathbf{x}) = \int_{-\infty}^{+\infty} f(\mathbf{t}) p_{\mathbf{x}}(\mathbf{t}) d\mathbf{t} \text{ and } V_f(\mathbf{x}) = \int_{-\infty}^{+\infty} [f(\mathbf{t}) - E_f(\mathbf{x})]^2 p_{\mathbf{x}}(\mathbf{t}) d\mathbf{t},$$

in which $p_{\mathbf{x}}$ is the joint probability function corresponding to the input variables' distributions. Unfortunately, a closed-form expression for these integrals exists for few cases of practical interest. Their numerical approximation involves a fundamental trade-off between computational cost and accuracy of the estimated statistical moments. Existing approaches to perform such a task, also termed *uncertainty propagation*, include Monte Carlo methods [7, 16], Taylor-based method of moments [13, 9, 4, 14], polynomial chaos expansions [18] and quadrature-based techniques [8, 11]. Two of those methods are considered to be adequate for the purpose of the present study, namely the first order method of moments (IMM) and the Sigma-Point method (SP).

2.1 Considered Uncertainty Propagation Methods

In the Taylor-based method of moments, the statistical moments of the system response are estimated from the moments of a truncated Taylor series expansion of $f(\mathbf{x})$ about the mean of the input variables. In particular, by using a third order approximation (IIIMM hereafter), mean $E_{f_{\text{IIIMM}}}$ and variance $V_{f_{\text{IIIMM}}}$ are given by, in the case of symmetric independent variables (with similar expressions for E_{g_i} and V_{g_i}):

$$E_{f_{\text{IIIMM}}} = \overbrace{f(\mathbf{E}_x)}^{m_1} + \overbrace{\frac{1}{2}\sum_{p=1}^{n}\left(\frac{\partial^2 f}{\partial x_p^2}\right) V_{x_p}}^{m_2} + O(\mathbf{V}_x^2), \tag{2}$$

$$V_{f_{\text{IIIMM}}} = \overbrace{\sum_{p=1}^{n}\left(\frac{\partial f}{\partial x_p}\right)^2 V_{x_p}}^{v_1} + \overbrace{\sum_{p=1}^{n}\left[\left(\frac{\partial^3 f}{\partial x_p^3}\right)\left(\frac{\partial f}{\partial x_p}\right)\frac{K_{x_p}}{3} + \left(\frac{\partial^2 f}{\partial x_p^2}\right)^2 \frac{K_{x_p}-1}{4}\right] V_{x_p}^2}^{v_2} + \\ + \overbrace{\sum_{p=1}^{n}\sum_{\substack{q=1 \\ q\neq p}}^{n}\left[\left(\frac{\partial^3 f}{\partial x_p^2 \partial x_q}\right)\left(\frac{\partial f}{\partial x_q}\right) + \frac{1}{2}\left(\frac{\partial^2 f}{\partial x_p \partial x_q}\right)^2\right] V_{x_p} V_{x_q}}^{v_3} + O(\mathbf{V}_x^3), \tag{3}$$

where K_{x_p} is the kurtosis of the input variable x_p. The first order method of moments (IMM) approximations $E_{f_{\text{IMM}}}$ and $V_{f_{\text{IMM}}}$ to the mean and variance respectively are obtained by retaining only terms m_1 and v_1. Note that optimisation involving an objective based on IMM requires calculation of the derivatives of $f(\mathbf{x})$ and, if gradient-based methods are to be employed, f's second derivatives.

The Sigma-Point technique [11], relies on a specific kind of reduced numerical quadrature, and gives approximations to the mean and variance respectively by,

$$E_{f_{\text{SP}}} = W_0 f(\mathbf{x}_0) + \sum_{p=1}^{n} W_p [f(\mathbf{x}_{p^+}) + f(\mathbf{x}_{p^-})], \tag{4}$$

$$V_{f_{\text{SP}}} = \frac{1}{2}\sum_{p=1}^{n}\Big\{W_p\left[f(\mathbf{x}_{p^+}) - f(\mathbf{x}_{p^-})\right]^2 \\ + (W_p - 2W_p^2)\left[f(\mathbf{x}_{p^+}) + f(\mathbf{x}_{p^-}) - 2f(\mathbf{x}_0)\right]^2\Big\}. \tag{5}$$

In (4) and (5), the sampling points are:

$$\mathbf{x}_0 = \mathbf{E}_x, \text{ and } \mathbf{x}_{p^\pm} = \mathbf{E}_x \pm \sqrt{V_{x_p} K_{x_p}}\,\mathbf{e}_p; \tag{6}$$

$\mathbf{e}_p$ is the p^{th} column of the $n \times n$ identity matrix. The weights in (4) and (5) are:

$$W_0 = 1 - \sum_{p=1}^{n} \frac{1}{K_{x_p}} \text{ and } W_p = \frac{1}{2K_{x_p}}.$$

The SP technique has a higher accuracy for the mean than IMM [11], and requires $2n+1$ function evaluations for each analysis. In contrast to the IMM method, objectives based on the SP approximations to the mean and variance do not require the derivatives of f. For gradient-based optimisation a combination of the gradients of f for the sampling points (6) yields the gradients of mean and variance.

3 Automatic Differentiation of the Conceptual Design Package

Unlike the Fortran or C/C++ coded design optimisation problems previously treated using AD [2, 1, 15], the aircraft conceptual design package considered was written in Matlab. The MAD package was therefore adopted due to its robust and efficient implementation [5]. MAD's `fmad` and `derivvec` classes facilitate forward mode AD for first derivatives geared toward Matlab's array-based arithmetic and both classes use only high-level array operations leading to good efficiency.

Objects of `fmad` class posses `value` and `deriv` components which store an object's value and derivative respectively. Then, for example, `fmad` objects `x` and `y` have element-wise product `z = x.*y` with `z`'s components determined by,

```
z.value = x.value.*y.value;
z.deriv = x.value.*y.deriv + y.value.*x.deriv;
```

Other arithmetic operations are evaluated in a similar manner. If we are determining a single directional derivative then all derivative components (`x.deriv`, `y.deriv`, etc.) are of Matlab's intrinsic class `double` with the same array dimensions as their corresponding value components (`x.value`, `y.value`, etc.). For multiple directional derivatives the `derivvec` class is utilised.

The `derivvec` class stores multiple directional derivatives as a single object manipulated, via overloaded arithmetic, as one would a single directional derivative. An `fmad` object whose `value` component is of dimensions $n \times m$ and which has d directional derivatives has its derivative component stored within its now `derivvec` class `deriv` component as an $nm \times d$ matrix. Element-wise multiplication and addition operators, amongst others, are defined for the `derivvec` class to allow the `fmad` class's operations to evaluate without modification.

There were two requirements for this conceptual design problem that necessitated extensions to the MAD package. Firstly, we required second derivatives to evaluate the IMM method's objective and constraint gradients and third derivatives for the associated post-optimality analysis. Secondly, the AD should differentiate the Newton-based `fsolve` function of Matlab's Optimisation Toolbox [10] used for the solution of nonlinear equation.

3.1 Calculating Higher Derivatives

We adopted the expedient strategy of rendering MAD's `fmad` and `derivvec` classes self-differentiable to allow a forward-over-forward(-over-forward...) differentiation strategy for higher derivatives. For the relatively low number of independent variables and low derivative orders required, other strategies such as forward-over-reverse and Taylor-series [6, Chaps. 4 and 10] were not considered worthwhile.

Under the self-differentiation approach, objects of the `derivvec` class contain derivative components that themselves must be differentiated to enable higher derivatives to be calculated. To enable this the source code line,

```
superiorto('fmad')
```

was added to the `derivvec` class constructor function to specify, via the intrinsic `superiorto`, that the `derivvec` class is *superior* to the `fmad` class. Consequently any operation (e.g., `x.value.*y.deriv`) involving an `fmad` object whose `value` component is itself an `fmad` object (e.g., `x.value`) and a `derivvec` class object (e.g., `y.deriv`) is dealt with by the appropriate overloaded `derivvec` class operation, and not an `fmad` class operation. Of course, within the resulting `derivvec` operations `fmad` operations may be required.

It was then possible to calculate all first and second derivatives of say the objective `f(x)`, with `x` is vector, by the following sequence of operations:

```
xfmad = fmad(x,eye(length(x)));            % step 1
xfmad2 = fmad(xfmad,eye(length(x)));       % step 2
yfmad2 = f(xfmad2);                        % step 3
y = getvalue(getvalue(yfmad2));            % step 4
Dy = getinternalderivs(getvalue(yfmad2)); % step 5
D2y=getinternalderivs(...
              getinternalderivs(yfmad2)); % step 6
```

with the steps labelled above with via the trailing comment and now detailed.

1. Define an `fmad` object, `xfmad`, from `x` whose derivatives are the identity matrix.
2. Define a second `fmad` object `xfmad2` whose value component is that defined in step 1, again with derivatives given by the identity matrix.
3. Evaluate the function.
4. Extract the value of `yfmad2`'s value component to obtain `y`'s value.
5. Extract `yfmad2`'s value's derivative to obtain first derivatives of `y`. Equivalently one could extract `yfmad2`'s derivative's value.
6. Extract `yfmad2`'s derivative's derivative to obtain second derivatives.

3.2 Differentiating the Nonlinear Solve of `fsolve`

Within the Matlab implementation of the conceptual design model, some intermediate variables, let us denote them $\mathbf{w} \in \mathbb{R}^p$, are found in terms of some predecessor variables, say $\mathbf{v} \in \mathbb{R}^p$, as the solution of a nonlinear equation of the form,

$$\mathbf{h}(\mathbf{w}, \mathbf{v}) = 0, \tag{7}$$

with function $\mathbf{h} \in \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}^p$ for some $p > 1$. Matlab's `fsolve` function solves such equations using a trust-region Newton algorithm [10] making use of $\mathbf{h}$'s Jacobian. Different strategies for differentiating such nonlinear solves are reviewed in [6, Sect. 11.4]. In the case of $\mathbf{w}$ being of `fmad` class, perhaps containing multiple orders of derivatives, we first solved (7) for the value component of $\mathbf{w}$ using a call to `fsolve` making use of only $\mathbf{v}$'s value, i.e., we solved the undifferentiated problem

to the required numerical accuracy. Then to calculate derivatives up to and including order d we performed d iterations of the Newton iteration,

$$\mathbf{w} \leftarrow \mathbf{w} - \left(\frac{\partial \mathbf{h}}{\partial \mathbf{w}}(\text{value}(\mathbf{w}), \text{value}(\mathbf{v})) \right)^{-1} \mathbf{h}(\mathbf{w}, \mathbf{v}), \tag{8}$$

with $\mathbf{w}$ and $\mathbf{v}$ manipulated as nested `fmad` objects storing derivatives up to and including those of order d, and value($\mathbf{w}$), value($\mathbf{w}$) denoting use of only their value component. This approach only requires the value of the Jacobian matrix $\partial \mathbf{h}/\partial \mathbf{w}$ and not its higher derivatives. It was implemented in a function `fsolveMAD` which itself calls `fsolve` to solve the undifferentiated problem and may make use of the `fmad` class to calculate the Jacobian for its trust-region Newton solve. The potentially more efficient strategy of evaluating only the updated i^{th} order derivatives of $\mathbf{w}$ at iteration i of (8) was not investigated because of the small size $\mathbf{w} \in \mathbb{R}^2$ in our test case of Section 4.

4 Aircraft Sizing Test Case

We demonstrate the benefits of using AD in robust optimization of a Matlab-implemented, industrially relevant, conceptual design test case. This conceptual design model determines performance and sizing of a short-to-medium range commercial passenger aircraft and makes use of 96 sub-models and 126 variables.

The original deterministic optimization problem is the following:
Objective: Minimize Maximum Take-Off Weight $MTOW$ with respect to the design variables $\mathbf{x}$ (described in Tab. 1 together with their permitted ranges).
Constraints:

1. Approach speed: $v_{app} < 120$ Kts $\Rightarrow\ g_1 = v_{app} - 120$;
2. Take-off field length: $TOFL < 2000$ m $\Rightarrow\ g_2 = TOFL - 2000$;
3. Percentage of total fuel stored in wing tanks: $K_F > 0.75 \Rightarrow\ g_3 = 0.75 - K_F$;
4. Percentage of sea-level thrust available during cruise: $K_T < 1 \Rightarrow\ g_4 = K_T - 1$;
5. Climb speed: $v_{zclimb} > 500$ ft/min $\Rightarrow\ g_5 = 500 - v_{zclimb}$;
6. Range: $R > 5800$ Km $\Rightarrow\ g_6 = 5800 - R$.

The problem's fixed parameters are given in Tab. 2. The corresponding robust problem is obtained by assuming that the input variables are independent Gaussian variables with $\mathbf{V}_x^{1/2} = 0.07\mathbf{E}_x$. The robust objective is calculated then as $MTOW_{\text{rob}} = E_{MTOW} + V_{MTOW}^{1/2}$, while the constraints take the form $G_i(\mathbf{x}) = E_{g_i}(\mathbf{x}) + kV_{g_i}^{1/2}(\mathbf{x})$ and $\mathbf{x}_L + k\mathbf{V}_x^{1/2} \leqslant \mathbf{E}_x \leqslant \mathbf{x}_U - k\mathbf{V}_x^{1/2}$, with coefficient $k = 1$ chosen to enforce constraint satisfaction with probability of about 84.1%.

We performed two robust optimisations with the first making use of IMM and the second the SP method to estimate mean and variance. Both optimization problems were solved using Matlab's gradient-based constrained optimizer `fmincon`.

Table 1 Considered design variables for the deterministic problem.

Design Variable	Definition [units]	Bounds [min,max]
S	Wing area [m^2]	[140, 180]
BPR	Engine bypass ratio [ ]	[5, 9]
b	Wing span [m]	[30, 40]
Λ	Wing sweep [deg]	[20, 30]
t/c	Wing thickness to chord ratio [ ]	[0.07, 0.12]
$T_{e_{SL}}$	Engine sea level thrust [kN]	[100, 150]
FW	Fuel weight [Kg]	[12000, 20000]

Table 2 Fixed parameters.

Parameter	Value	Parameter	Value
Number of passengers	150	Number of engines	2
Cruise Mach number	0.75	altitude [ft]	31000

In the IMM optimization, MAD was used to calculate first derivatives of the deterministic objective function and constraints required for (3), together with their second derivatives to form the gradient of the robust objective and constraints. In the SP-based optimizations, MAD is used to calculate the gradient of objectives and constraints. Table 3 presents the results of the two optimizations.

Table 3 Results of the robust optimisations.

Input variable	I MM	SP	Obj./Constr.	I MM	SP
E_S [m^2]	160.843	162.558	$MTOW_{\text{rob}}$ [Kg]	86023.272	86207.016
E_{BPR} [ ]	8.580	8.580	G_1 [Kts]	0.000	0.000
E_b [m]	37.753	37.753	G_2 [m]	-161.568	-151.806
E_Λ [deg]	21.531	21.531	G_3 [ ]	0.000	0.000
$E_{t/c}$ [ ]	0.095	0.094	G_4 [ ]	-0.114	-0.107
$E_{T_{e_{SL}}}$ [kN]	122.553	123.224	G_5 [ft/min]	0.000	0.000
E_{FW} [Kg]	18084.940	18171.282	G_6 [Km]	0.000	0.000

A post-optimal analysis was carried out by using MCS (Latin Hypercube with 10^4 samples) and IIIMM of (2) and (3) to partially validate Tab. 3's results. In these analyses, $\mathbf{x}$ was represented as a Gaussian random variable centred at the values of the input variables resulting from the two optimisations, while its variance was that used for the optimizations. The derivatives needed by the IIIMM estimation methods were obtained using MAD. The deviation of the IMM and SP results from those obtained by MCS are shown in Tab. 4. It appears that the SP method can attain an increased accuracy in the mean estimate, and thus benefit design optimisation. Results from IIIMM analysis at the optimal design points are comparable to those obtained by the SP method and consequently not shown. However, this case is one for which cross-derivatives, not modelled by the SP method, are negligible [11]. In fact, the accuracy of IIIMM estimates is higher than the other considered propagation methods, and hence IIIMM may be advantageously adopted to reduce the computational

Table 4 Post-optimality analysis: percentage error on mean and variance estimation of objective and constraint with respect to MCS.

	Mean estimation		Variance estimation	
Obj./Constr.	**IMM Opt. Solution**	**SP Opt. Solution**	**IMM Opt. Solution**	**SP Opt. Solution**
$MTOW_{rob}$ [Kg]	-0.83×10^{-4}	0.27×10^{-6}	0.41	0.39
G_1 [Kts]	-0.12	0.25×10^{-3}	-0.71	0.24
G_2 [m]	-0.88	0.17×10^{-2}	-1.89	-0.31
G_3 [ ]	-0.86	0.17×10^{-1}	-1.27	-0.48
G_4 [ ]	-0.87	0.12×10^{-2}	-2.18	0.17
G_5 [ft/min]	1.15	0.10×10^{-3}	0.23	0.73
G_6 [Km]	0.23	-0.11×10^{-2}	0.56	0.01

cost of the post-optimality analysis. For the case at hand, the c.p.u. time required to obtain the derivatives up to third order was about 8 seconds using MAD, which is about the 0.06% of the time required for the full MCS analysis.

To highlight the performance improvements made possible by AD with respect to finite differencing, the deterministic optimisation and the two robust optimisation problems were re-solved using finite differences (FD) for gradient evaluation. In these cases, the standard `fsolve` Matlab code was used. Results in terms of number of optimiser iterations and required c.p.u. time, shown in Tab. 5, support the conclusion that AD can significantly decrease the computational effort required by robust optimisation. We note that both robust techniques, IMM and SP, benefit from AD's fast and accurate derivatives, with IMM particularly advantaged since use of FD results in significantly more optimisation steps being required.

Table 5 Performance improvements yielded by AD to the optimisation problems.

	SP		IMM	
	Iterations	**c.p.u. time [s]**	**Iterations**	**c.p.u. time [s]**
AD	10	351	10	45
FD	10	708	24	1212

5 Conclusions

This paper's results demonstrate the benefits AD may give to robust optimisation for aircraft conceptual design. In particular, we performed optimisations of an industrially relevant, Matlab-implemented aircraft sizing problem using the AD tool MAD to facilitate two robust design strategies. The first strategy exploits AD-obtained first order sensitivities of the original function to approximate the robust objective and constraints using the method of moments, and second order sensitivities to calculate their gradients. The second uses reduced quadrature to approximate the robust objective and constraints and AD for their gradients. In the particular test case con-

sidered, a Monte-Carlo analysis indicated that the reduced quadrature approach was more accurate for estimation of the mean. In both cases use of numerically exact, AD gradients significantly reduced the c.p.u. time to perform the optimisations compared to those approximated by finite-differencing.

Acknowledgements The research reported in this paper has been carried out within the VIVACE Integrated Project (AIP3 CT-2003-502917) which is partly sponsored by the Sixth Framework Programme of the European Community (2002-2006) under priority 4 "Aeronautics and Space".

References

1. Barthelemy, J.F., Hall, L.E.: Automatic Differentiation as a Tool in Engineering Desing. TM 107661, NASA (1992)
2. Bischof, C.H., Griewank, A.: Computational Differentiation and Multidisciplinary Design. In: H. Engl, J. McLaughlin (eds.) Inverse Problems and Optimal Design in Industry, pp. 187–211. Teubner Verlag, Stuttgart (1994)
3. Chen, W., Allen, J.: A procedure for robust design: Minimizing variations caused by noise factors and control factors. Journal of Mechanical Design **118**(4), 478–493 (1996)
4. Du, X., Chen, W.: Efficient Uncertainty Analysis Methods for Multidisciplinary Robust Design. AIAA Journal **40**(3), 545–552 (2002)
5. Forth, S.A.: An efficient overloaded implementation of forward mode automatic differentiation in MATLAB. ACM Trans. Math. Softw. **32**(2), 195–222 (2006). DOI: `http://doi.acm.org/10.1145/1141885.1141888`
6. Griewank, A.: Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation. No. 19 in Frontiers in Applied Mathematics. SIAM, Philadelphia, PA (2000)
7. Helton, J.C., Davis, F.J.: Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems. Reliability Eng. System Safety **81**(1), 23–69 (2003/7)
8. Huang, B., Du, X.: Uncertainty analysis by dimension reduction integration and saddlepoint approximations. Transactions of the ASME **18**, 26–33 (2006)
9. Lewis, L., Parkinson, A.: Robust optimal design using a second order tolerance model. Research in Engineering Design **6**(1), 25–37 (1994)
10. The MathWorks, Inc., 3 Apple Hill Drive, Natick, MA 01760-2098: MATLAB Optimization Toolbox 3 - User's guide (2007)
11. Padulo, M., Campobasso, M.S., Guenov, M.D.: Comparative Analysis of Uncertainty Propagation methods for Robust Engineering Design. In: International Conference on Engineering Design ICED07. Paris (2007)
12. Park, G.J., Lee, T.H., Lee, K.H., Hwang, K.H.: Robust Design: An Overview. AIAA Journal **44**(1), 181–191 (2006)
13. Parkinson, A., Sorensen, C., Pourhassan: A General Approach for Robust Optimal Design. J. mech. Des **115**(1), 74–80 (1993)
14. Putko, M., Newman, P., Taylor, A., Green, L.: Approach for Uncertainty Propagation and Robust Design in CFD Using Sensitivity Derivatives. In: Proceedings of the 15^{th} AIAA Computational Fluid Dynamics Conference. Anaheim CA (2001). AIAA 2001–2528
15. Su, J., Renaud, J.E.: Automatic Differentiation in Robust Optimization. AIAA Journal **5**(6), 1072–1079 (1996)
16. Tari, M., Dahmani, A.: Refined descriptive sampling: A better approach to monte carlo simulation. Simulation Modelling Practice and Theory **14**(2), 143–160 (2006)
17. Torenbeek, E.: Synthesis of Subsonic Airplane Design. Delft University Press, Delft, Holland (1982)
18. Xiu, D., Karniadakis, E.M.: Modeling uncertainty in flow simulations via generalized polynomial chaos. Journal of Computational Physics **187**, 137–167 (2003)

Robust aircraft conceptual design using automatic differentiation in Matlab

Padulo, Mattia

2008-08-17T00:00:00Z

Mattia Padulo, Shaun A. Forth & Marin D. Guenov; Robust aircraft conceptual design using automatic differentiation in Matlab. Lecture Notes in Computational Science and Engineering, Volume 64, 2008, p271-280 eds. Christian H. Bishof, H. Martin Bücker, Paul Hovland, Uwe Naumann and Jean Utke. 2008.

http://dx.doi.org/10.1007/978-3-540-68942-3_24