JOURNAL OF SPACECRAFT AND ROCKETS
Vol. 56, No. 5, September–October 2019

Smart Projectile Parameter Estimation Using Meta-Optimization

Matthew Gross* and Mark Costello†
Georgia Institute of Technology, Atlanta, Georgia 30332

DOI: 10.2514/1.A34059

Parameter estimation and system identification of smart projectiles is an important and commonly used industrial tool. Existing methods rely on good initial estimates of the parameters to avoid local minima and ensure proper convergence. New projectile configurations may include highly nonlinear dynamics or unknown control parameters that cannot be known a priori. A new method for projectile parameter estimation is proposed that combines an output error parameter estimation algorithm with meta-optimization. Meta-optimization uses a suite of optimizers in an intelligent manner to reliably minimize a cost function. This new method is applied to the identification of a smart projectile system equipped with microspoilers using simulated spark range data. The method is able to reliably estimate the aerodynamic coefficients of the projectile body as well as the properties of the control mechanism based on a fit of multiple trajectories.

Nomenclature

C_{N_α}	=	normal force derivative coefficient
C_{X_0}	=	zero-yaw axial force coefficient
C_{X_2}	=	yaw-squared axial force coefficient
$C_{Y_{p\alpha}}$	=	Magnus force coefficient
C_{Y_0}, C_{Z_0}	=	zero-yaw normal force coefficients
C_{l_p}	=	roll damping coefficient
C_{l_0}	=	zero-yaw rolling moment coefficient
C_{m_q}	=	pitch damping coefficient
C_{m_0}, C_{n_0}	=	zero-yaw pitching/yawing moment coefficients
C_{m_α}	=	pitching moment derivative coefficient
$C_{n_{p\alpha}}$	=	Magnus moment coefficient
d	=	reference diameter of projectile
G_i	=	measurement scaling weighting matrix
g	=	gravitational constant
I_B	=	mass moment of inertia matrix
J	=	output error method cost function
J_{ref}	=	reference cost reduction
L, M, N	=	moment measure numbers in body frame
m	=	mass of projectile
N	=	number of measurements
N_{opt}	=	number of optimizers
N_p	=	population size
n	=	number of unknown parameters
p, q, r	=	components of rotational velocity of projectile body with respect to an inertial observer written in the projectile frame
p_i	=	optimizer selection probabilities
Q	=	dynamic pressure of projectile
r_i	=	penalty coefficient
t	=	time
t_{ref}	=	reference computation time
u, v, w	=	velocity vector scalar numbers in body reference frame
V	=	total velocity of projectile
W_0, W_1	=	optimizer weighting function parameters
w_i	=	optimizer performance weights
X, Y, Z	=	force measure numbers in body frame
x, y, z	=	position vector measure numbers in inertial reference frame
$\boldsymbol{x}$	=	unknown parameter vector
$\boldsymbol{y_i}$	=	measurement vector at point i
$\boldsymbol{z_i}$	=	estimated measurement vector at point i
$\bar{\alpha}$	=	total angle of attack
δ_A	=	microspoiler axial force
δ_N	=	microspoiler normal force
δ_m	=	microspoiler pitching moment
η	=	optimizer efficiency metric
λ	=	microspoiler scaling function
ϕ, θ, ψ	=	Euler roll, pitch, and yaw of the body
ρ	=	atmospheric density
σ_i	=	standard deviation of measurement noise on ith state
σ_d	=	optimizer population diversity
τ_{ms}	=	microspoiler time constant
χ^2	=	goodness of fit metric
Ω_0	=	microspoiler nominal spin rate
ω_0	=	microspoiler initial phase

Subscripts

A	=	aerodynamic force or moment
C	=	control force or moment
G	=	gravity force
0	=	initial value

Superscript

$*$	=	normalized

I. Introduction

CURRENTLY, there are a number of methods employed for parameter estimation of projectile flight dynamic models. Historically, parameter estimation has been performed on data from test firing projectiles under a range of conditions, either in a spark range [1–3] or using on board sensors [4]. More recently, computational fluid dynamics (CFD) simulations have been employed to determine the aerodynamics of new projectiles [5,6]. In addition, CFD techniques have been combined with flight dynamics simulations to create virtual flyout data that is used in place of flight testing [7–9].

Most existing projectile parameter estimation methods fall under the category of maximum likelihood estimators (MLE). These methods pose the parameter estimation problem in terms of an optimization problem seeking the parameters that maximize the likelihood of matching a set of experimental data. General versions of these methods employ a numerical optimizer such as a Newton style algorithm to solve for unknown parameters. One common algorithm is the Aeroballistics Research Facility Data Analysis System

Presented as Paper 2016-0538 at the AIAA Atmospheric Flight Mechanics Conference, San Diego, CA, 4–8 January 2016; received 1 January 2018; revision received 4 March 2019; accepted for publication 10 March 2019; published online 29 April 2019.

*Graduate Research Assistant, Guggenheim School of Aerospace Engineering. Student Member AIAA.

†William R.T. Oakes School Chair and Professor, Guggenheim School of Aerospace Engineering. Associate Fellow AIAA.

(ARFDAS) [10–12]. ARFDAS uses projectile linear theory to produce parameter estimates used as the starting point for the MLE. An iterative approach is used to match simulated six-degree-of-freedom (6DOF) trajectories with experimental data [1].

A similar approach was developed by Montalvo and Costello using the output error method (OEM). Like ARFDAS, linear theory is employed to determine initial values for the unknown parameters before the Levenberg-Marquardt process is used to estimate the parameters based on coupled CFD/rigid body dynamics (RBD) virtual flyouts [9]. Burchett used a gradient-based approach based on a linear model of the projectile dynamics. Simulated yaw card data with no measurement noise were used to demonstrate the algorithm. The gradient-based method was also compared with a genetic-algorithm-based approach [13,14]. Condaminet et al. investigated four different problem configurations for estimating the parameters of a reduced order ballistic model using partial flight data. In all cases, a Newton–Raphson technique is used to solve the optimization problems [15].

Although existing projectile parameter estimations are powerful and have been successfully deployed to date, they work best when given initial parameter values close to the solution. Newton-style optimizers are local search techniques and are prone to converging on local optima that are common in projectile parameter estimation problems. Projectile linear theory can provide reasonable estimates for some parameters, but does not capture the fully nonlinear behavior of the projectile and may not be able to provide reasonable estimates for some parameters.

Like all optimization problems, there are numerous optimizers that can be used to solve the projectile parameter estimation problem, each with various strengths and weaknesses. While Newton-based optimizers are prone to converging on local minima, they are extremely efficient near a local minima and can converge in few iterations [16]. Alternatively, global optimizers such as particle swarm [17,18] and differential evolution [19] are able to converge on the global minimum without getting caught in local minima. Although these methods work well for systems with many local minima, they may require many iterations to converge. Thus, there is a need for optimizers for projectile parameter estimation that are easy to use and highly reliable at solving a wide range of problems [20–22].

This work proposes a new method for reliable parameter estimation of smart projectile systems. To achieve this goal, a new robust numerical optimization strategy–dubbed meta-optimization is developed based on the framework of algorithm portfolios and hybrid optimizers. The algorithm iteratively deploys a diverse set of optimizers in an intelligent manner, improving accuracy and reliability across a wide set of problems. The meta-optimizer must be able to select appropriate optimizers, ensure smooth transitions between different optimizers, prevent premature convergence in local minima, and improve optimizer parameters that are poorly tuned. This paper begins with a description of the smart projectile parameter estimation problem and the example projectile system considered. Next, an analysis is performed on the parameter estimation problem to understand the complex topology of the typical projectile parameter estimation problem. Next, the algorithm is described, highlighting the different components of the framework. Finally, the proposed parameter estimation method is applied to the example smart projectile system using simulated flight test data that include measurement noise.

II. Smart Projectile Parameter Estimation

Parameter estimation data are typically collected from flight testing in a spark range that provides measurements of the position and orientation of the projectile at discrete points along the flight. The U.S. Army Research Laboratory Transonic Experimental Facility spark range consists of 25 orthogonal spark shadowgraph stations arranged in groups of 5 along an approximately 200 m range. From each set of images gathered from the spark stations, position and angular measurements are taken in addition to the time of the measurements. Trajectory data can also be generated using CFD/RBD virtual flyouts that use a CFD model to compute the aerodynamic forces and moment on the projectile, creating realistic trajectories without the need for expensive flight testing [8,9]. The system identification problem is formulated using the output error method (OEM), which defines a cost function as a function of the difference between given trajectory data and a dynamic model prediction of the same trajectory. Trajectory data come in the form of measured data from flight testing or CFD/RBD virtual flyouts. Predictions of the projectile trajectory are obtained from a 6DOF projectile flight dynamics representation that simulates the flight of the projectile given estimates of unknown parameters.

A. Output Error Method

The projectile system identification problem is cast in the output error format. Under this formulation, a cost function is defined based on comparing estimated measurements to known data points along a trajectory. In this case, the estimated measurements are generated from a simulated trajectory generated using estimates for any unknown parameters. Figure 1 shows a schematic of how OEM computes the cost. Here, there are three measurements of θ at specific range locations. A trajectory is simulated using an estimate of the parameters of the system. The error between the predicted trajectory and the measurements is found by taking the difference between the measurement and the predicted trajectory at the same range value. These errors are computed for every measurement of every state and combined into a single cost function. The typical cost function for the OEM is the sum squared error as defined in Eq. (1) [23,24].

$$J(\boldsymbol{x}) = \frac{1}{2}\sum_{i=1}^{N}[\boldsymbol{z}_i - \boldsymbol{y}_i]^T G_i[\boldsymbol{z}_i - \boldsymbol{y}_i] + \sum_{i=1}^{n} r_i |\max(0, g(\boldsymbol{x}))|^3 \quad (1)$$

Here, $\boldsymbol{x}$ is the vector of unknown parameters, $\boldsymbol{y}$ is the measured state vector, $\boldsymbol{z}$ is the estimated measurement, N is the number of measurements, n is the number of parameters, G_i is the weighting and scaling matrix, r_i is the penalty coefficient, and $g(\boldsymbol{x})$ is the inequality constraint function that is positive when the inequality is not satisfied. The weighting and scaling matrix G_i is generated from three components. First, the state errors are scaled by the standard deviation of the measurement noise, ensuring that all states contribute to the cost equally. Second, the states can be weighed against each other to limit the impact of certain states or to highlight others. Finally, weighting can be applied to each individual measurement along the trajectory in order to highlight certain behavior and improve the accuracy of the parameter estimates.

Each parameter is constrained to its search range by an exterior penalty function. In Eq. (1), this is represented by the function $g(\boldsymbol{x})$, which treats each boundary condition as an inequality constraint. This penalty function adds to the cost when a parameter exceeds its bounds. A cubic penalty function is chosen because the second derivative is zero on the boundary, providing sufficient smoothness for hill-climbing-based optimization algorithms. The magnitude of the penalty function is set to allow some exploration beyond the

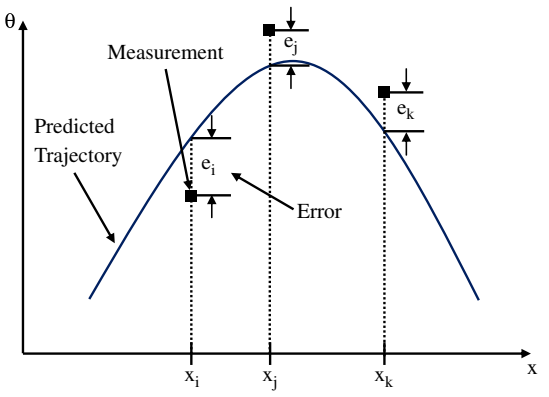

Fig. 1 Example schematic of OEM cost function trajectory errors.

boundary while the overall cost remains high. As the cost is reduced, the boundary grows steeper relative to the current cost.

In addition to the cost function, the quality of fit for a given set of parameters is evaluated using the metric χ^2. For the jth state, χ^2 is computed using Eq. (2) [25]:

$$(\chi^2)^j = \sum_{i=1}^{N} \frac{(z_i^j - y_i^j)^2}{\sigma_j^2} \tag{2}$$

where σ_j is the standard deviation of the measurement noise for this state. Lower values indicate a better fit with the expected value of χ^2 for an optimal trajectory on the order of the number of measurements.

B. Projectile Flight Dynamics Model

The 6DOF rigid-body flight dynamics simulation is the heart of the cost function calculation. This model predicts the position, orientation, velocity, and angular velocity of the projectile as a function of time over the trajectory. It includes gravity (G), aerodynamic (A), and control (C) forces and moments. For brevity, the kinematic and dynamic equations of motion are not presented here and can be found in the literature [26]. The aerodynamic forces on the projectile are modeled using existing ballistic expansions with known coefficients given in Eq. (3).

$$\begin{Bmatrix} X_A \\ Y_A \\ Z_A \end{Bmatrix} = -Qd \begin{Bmatrix} C_{X_0} + C_{X_2}\frac{v^2+w^2}{V^2} \\ C_{Y_0} + C_{N_\alpha}\frac{v}{V} - C_{Y_{p\alpha}}\frac{w}{V}\frac{pd}{2V} \\ C_{Z_0} + C_{N_\alpha}\frac{W}{V} + C_{Y_{p\alpha}}\frac{v}{V}\frac{pd}{2V} \end{Bmatrix} \tag{3}$$

Here, C_{X0} is the zero yaw drag coefficient, C_{X2} is the dynamic drag coefficient, C_{Y0} and C_{Z0} are the zero yaw normal force coefficients, $C_{N\alpha}$ is the normal force coefficient, $C_{Yp\alpha}$ is the Magnus force coefficient, d is the aerodynamic diameter, V is the total velocity, and Q is the dynamic pressure given by $Q = (1/8)\pi\rho d^2V^2$. The weight of the projectile expressed in the projectile body frame is given by:

$$\begin{Bmatrix} X_G \\ Y_G \\ Z_G \end{Bmatrix} = \begin{Bmatrix} -s_\theta \\ s_\phi c_\theta \\ c_\phi c_\theta \end{Bmatrix} mg \tag{4}$$

The total aerodynamic moments are given in Eq. (5):

$$\begin{Bmatrix} L_A \\ M_A \\ N_A \end{Bmatrix} = Qd^2 \begin{Bmatrix} C_{l_0} + C_{l_p}\frac{pd}{2V} \\ C_{m_0} + C_{m_\alpha}\frac{w}{V} + C_{n_{p\alpha}}\frac{v}{V}\frac{pd}{2V} + C_{m_q}\frac{qd}{2V} \\ C_{n_0} - C_{m_\alpha}\frac{v}{V} + C_{n_{p\alpha}}\frac{w}{V}\frac{pd}{2V} + C_{m_q}\frac{rd}{2V} \end{Bmatrix} \tag{5}$$

where C_{l0} is the zero-yaw rolling moment coefficient, C_{lp} is the roll damping moment coefficient, C_{m0} and C_{n0} are the zero yaw pitch and yaw moment coefficients, $C_{m\alpha}$ is the pitching moment coefficient, $C_{np\alpha}$ is the Magnus moment coefficient, and C_{mq} is the pitch damping coefficient. These aerodynamic coefficients are typically functions of Mach number. The control forces X_C, Y_C, Z_C and control moments L_C, M_C, N_C are application dependent, with each control method having unique effects on the projectile dynamics. With all of the applied forces and moments computed, the equations of motion are numerically integrated forward in time using a fourth-order Runge–Kutta method to generate a trajectory for the projectile configuration.

For most projectile parameter estimation problems, only the aerodynamic coefficients are estimated, but any parameter used in the model could be estimated as well. This could include parameters such as the projectile mass m, the moments of inertia $[I_B]$, and the projectile diameter d. The projectile aerodynamic model presented here has 12 coefficients that need to be estimated. However, for symmetric projectiles, C_{Y_0}, C_{Z_0}, C_{m_0}, and C_{n_0} are zero. In addition, for finned projectiles, it is usually assumed that the Magnus coefficients, $C_{Yp\alpha}$ and $C_{Np\alpha}$, are zero as roll rates tend to be small, resulting in negligible Magnus effects. This reduces the number of aerodynamic coefficients for finned projectiles to seven: C_{X0}, C_{X2}, $C_{N\alpha}$, C_{l0}, C_{lp}, $C_{m\alpha}$, and C_{mq}.

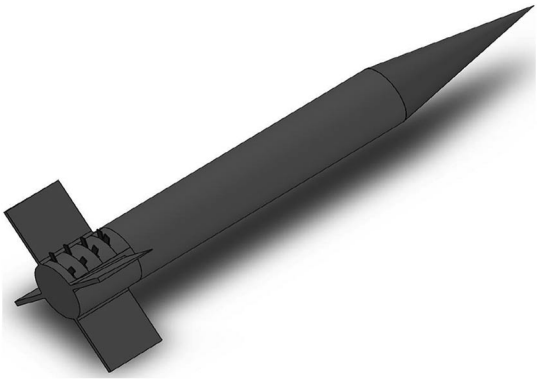

Fig. 2 The 30 mm Army-Navy Finner with microspoilers.

C. Example Smart Projectile System

The smart projectile configuration considered in this work is a finned projectile equipped with a single set of microspoilers. The base projectile is a 30 mm Army-Navy Finner (ANF). This round is a popular test bed for new control mechanisms as it has been studied extensively by the community with well-documented aerodynamics. This projectile configuration is shown in Fig. 2. The round is axisymmetric with four fins at the rear of the body. The mass properties of the standard 30 mm ANF are given in Table 1.

The microspoiler mechanism consists of four sets of small protrusions that extend and retract from the projectile body with a prescribed motion and a set frequency. As seen in Fig. 2, the microspoilers are placed between the rear fins of the projectile and are oriented such that they are on the top of the projectile body. Microspoilers add additional forces and moments acting on the projectile and are incorporated into the equations of motion of the projectile through the control forces and moments. The magnitude of the forces and moments at a given time is a function of how much the microspoilers are exposed.

The mechanism is designed to spin at a set rate with a spin up period at launch. A first-order model based on bench testing of the mechanism is used to approximate the spin rate as it reaches steady state. The expansion for the microspoiler forces and moments is given by:

$$\begin{Bmatrix} X_C \\ Y_C \\ Z_C \end{Bmatrix} = \lambda(\omega_0 + \Omega_0(t + \tau_{ms}e^{-(t/\tau_{ms})} - \tau_{ms})) \begin{Bmatrix} \delta_A \\ 0 \\ \delta_N \end{Bmatrix} \tag{6}$$

$$\begin{Bmatrix} L_C \\ M_C \\ N_C \end{Bmatrix} = \lambda(\omega_0 + \Omega_0(t + \tau_{ms}e^{-(t/\tau_{ms})} - \tau_{ms})) \begin{Bmatrix} 0 \\ \delta_m \\ 0 \end{Bmatrix} \tag{7}$$

Table 1 Physical properties of 30 mm Army-Navy Finner

Physical property	Value
Mass, kg	1.5887
Diameter, m	0.03
Length, m	0.3
Center of gravity—I_P, m	0.135
Center of gravity—J_P, m	0.0
I_{XX}, kg · m^2	0.000192388
I_{YY}, kg · m^2	0.00987337
I_{ZZ}, kg · m^2	0.00987337
$I_{XY} = I_{YX}$, kg · m^2	0.0
$I_{XZ} = I_{ZX}$, kg · m^2	0.0
$I_{YZ} = I_{ZY}$, kg · m^2	0.0

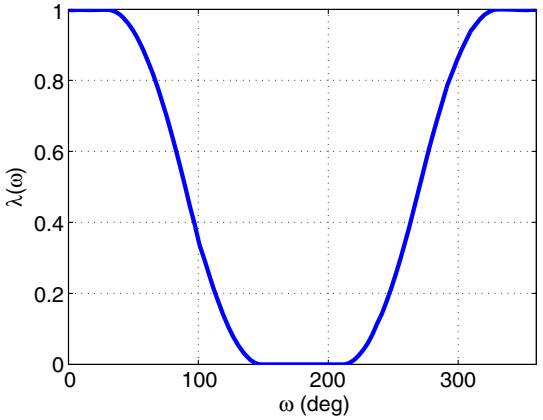

Fig. 3 Microspoiler actuation profile.

The effect of the microspoilers is parameterized by six parameters: the axial force coefficient δ_a, the normal force coefficient δ_N, the pitching moment coefficient δ_m, the initial microspoiler phase ω_0, the microspoiler spin rate Ω_0, and the microspoiler time constant τ_{ms}. The magnitude function λ is determined by the design of the microspoiler mechanism and is given in Fig. 3.

III. Topology Analysis of Smart Projectile Parameter Estimation Problem

The structure and topology of an optimization problem has a significant impact on which optimizers will succeed and which will struggle to solve the problem. Unlike benchmark functions that have a defined mathematical form that can be easily visualized and analyzed, more practical optimization problems provide very little intuition on which optimizers are most appropriate. In the case of the smart projectile parameter estimation problem, the high dimensionality and complexity of the objective function makes it a difficult problem to analyze. However, valuable insight can be obtained by observing the nature of this problem in a range of situations to understand the underlying topology of the associated optimization problem.

As an example, consider an analysis of the projectile roll dynamics. The parameters associated with roll are the zero-yaw rolling moment coefficient (C_{l0}), roll damping coefficient (C_{lp}), and initial roll rate (p_0). In general, the roll dynamics are decoupled from the other states as the impact of other states on the roll rate is minimal. This allows for easy investigation of the roll dynamics. Roll data are typically wrapped, meaning all angles are restricted to −180° to 180°. Sometimes roll data are unwrapped to an absolute angle. With unwrapped roll measurements, fitting the roll parameters is very simple and the associated optimization problem is unimodal. However, when the roll measurements are wrapped, the topology becomes multimodal with complex character and a number of local minima.

To observe this phenomenon, a case is constructed with only the roll measurements included in the cost function, restricting the optimization problem to only the roll dynamics. The cross section is taken about C_{l0} and p_0 with the remaining parameters held fixed. The search range for C_{l0} is from 0.03 to 0.06 and the search range for p_0 is from 50 to 150 rad/s. Figure 4 shows the contours of the cost function over these two parameters. In the figure, the box represents the typical search bounds on these parameters. The boundary is color coded to indicate if the gradient is pointing into or out of the search space and the arrows represent the direction of the gradient on the boundary. The black lines denote the nominal values of the parameters with the global minimum at their intersection. Overall, this landscape is highly multimodal with a few local minima within the search bounds as well as some outside of the search space that would attract optimizers out of the search space. The local minima occur when the predicted trajectories come into phase with the roll measurements for brief periods of time. This yields a low cost value for those few measurements, with any change in the parameters causing an increase in cost. A small, deep oval occurs around the global minimum at $C_{l0} = 0.04375$ and $p_0 = 104.2$ rad/s, with most of the search space outside of this basin of attraction.

Another important component of the projectile parameter estimation problem is the microspoiler dynamics. The model for the microspoilers is given in Sec. II.C. Of particular interest is the effect of the microspoiler spin rate (Ω_0) on the cost function. For this case, the spin rate is assumed to be constant for the entire flight, isolating the effects of the spin rate from the time constant τ_{ms}. ϕ is again excluded from the cost function as the microspoilers do not affect the roll dynamics. A landscape is constructed over the axial force δ_A and Ω_0. Typically, δ_A varies from −45 to −15 N, whereas Ω_0 varies from 350 to 500 rad/s. The cost contours in Fig. 5 show multiple local minima in terms of Ω_0. As with the roll dynamics, these local minima form when the predicted trajectories align with a few measurements in each state, achieving a low cost value. While all of the gradients point into the search space, there are two local minima that occur in this space. Given the large basins of attraction for these local minima, only a narrow band of about 25–30 rad/s around the global minimum stays within its basin. These basins are also relatively deep with costs not much higher than the optimal solution. Additional local minima appear as Ω_0 increases outside of the search space, which may be reached if the boundaries are not enforced.

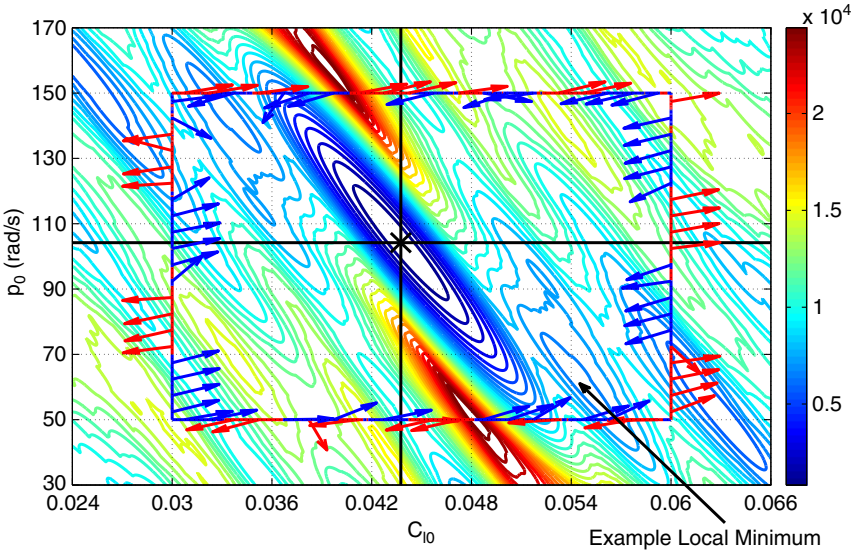

Fig. 4 Contour of cost function over zero-yaw rolling moment coefficient and initial roll rate, nominal parameters $C_{l0} = 0.04375$ and $p_0 = 104.2$ rad/s.

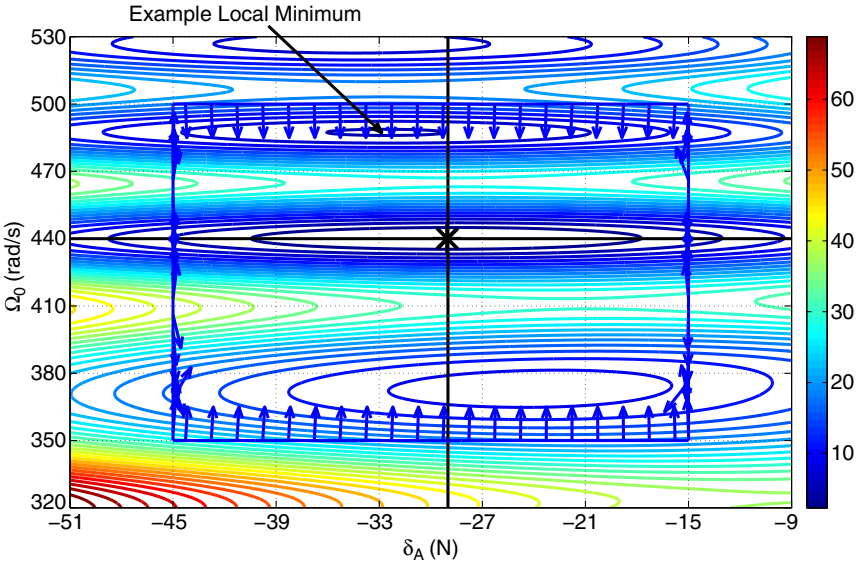

Fig. 5 Contour of cost function over microspoiler axial force and spin rate, nominal parameters $\delta_A = -29$ N and $\Omega_0 = 440$ rad/s.

These two cases provide a snapshot of the complex landscapes observed in the projectile parameter estimation problem. The presence of numerous local minima ensures difficulties for hill climbers, which are likely to get stuck in a local minimum. For both of the cases considered, the basin of attraction of the global minimum covers only a fraction of the search space, necessitating reasonable guesses of the parameters to guarantee convergence for these algorithms. These topologies can also present issues for other methods that may be very slow to converge or end up wandering the search space, moving from one local minimum to the next.

IV. Meta-Optimization

The proposed framework consists of five main parts: the bank of optimizers, the performance metric, the optimizer selection routine, the optimizer manager, and the auto-tuning algorithm. An overview of this framework is given by Fig. 6, which shows the general flow of the algorithm. The basic flow of the algorithm proceeds as follows: an optimizer is chosen, resources are allocated to the optimizer, the optimizer runs for a period of time, and performance of the optimizer is evaluated. This process is then repeated until a solution to the optimization problem of sufficient quality is found or it has exhausted its resources.

A. Bank of Optimizers

A key part of meta-optimization is the resident bank of individual numerical optimizers that can be used. Many different optimizers, each with numerous variants, could be selected for inclusion in the bank of optimizers. Each optimizer is individually capable of obtaining a solution to the given optimization problem. A mixture of local and global search methods is used to provide maximum diversity to the meta-optimizer. The primary category of local search methods is considered hill climbers, where the algorithm searches for a better solution by incrementally varying the current best solution, gradually moving toward a local optimum. Included hill climbers are steepest descent (SD) [27], conjugate gradient (CG) [28], and the

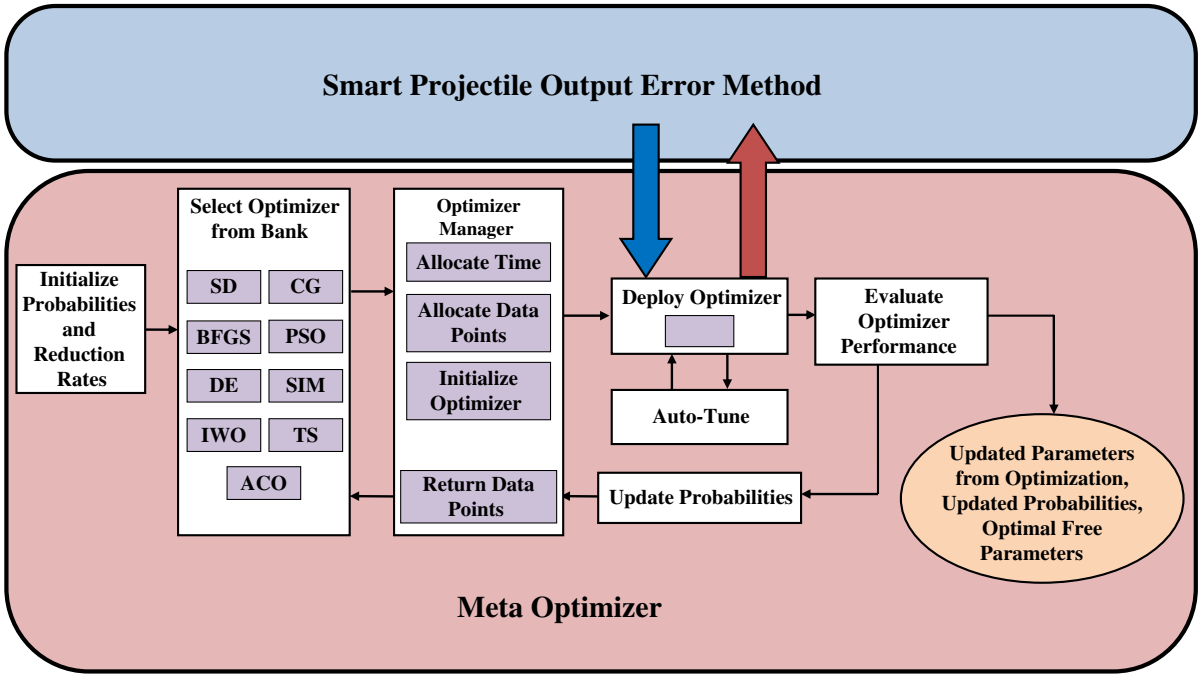

Fig. 6 Meta-optimization flow chart.

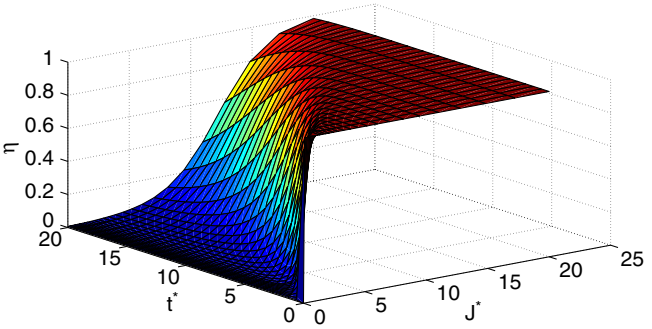

Fig. 7 Optimizer performance metric based on objective function reduction and computation time.

Broyden–Fletcher–Goldfarb–Shanno (BFGS) method [16]. It is known that the primary drawback with hill climbers is that they are prone to converging to local minima. To balance these methods, global optimizers are included in the bank of optimizers. These methods search the entire parameter space by operating on a large set of points, continually seeking the global minimum. The global optimizers employed here are particle swarm optimization (PSO) [17,18], differential evolution (DE) [19], invasive weed optimization (IWO) [29–32], and ant colony optimization (ACO) [33]. The Nelder–Mead (Simplex) method [34,35] and tabu search [36,37] are other metaheuristic methods used. These methods were selected as a representative set of common optimizers with a roughly even mixture of local and global search methods.

B. Optimizer Performance Evaluation

A key aspect of the algorithm is an assessment of an optimizer's performance when employed on a particular problem. The effectiveness of each optimizer is used to determine when it is appropriate to change optimizers and which optimizer should be deployed next. For the purposes of meta-optimization, a good optimizer is one that has low computation time and high cost reduction. Computation time is measured as the number of objective function calls each optimizer makes during its run. A metric combining these two measures provides a characterization of the efficiency of the optimizer, evaluating the cost reduction per unit of computation time. Slightly greater weight is placed on the magnitude of the cost reductions than computation time as optimizers that achieve greater cost reductions should be rated well. Finally, the metric must provide reasonable evaluation of optimizer performance throughout the optimization process and across all problems.

Based on these considerations, the function shown in Fig. 7 was chosen as a performance metric. The overall shape of the function was modeled on a two-dimensional sigmoid function with the output η ranging from 0 for poor performance to 1 for good performance. The inputs to the function are the normalized percent cost reduction J^* and the normalized computation time t^*. To obtain these measures, the percent objective function reduction J is divided by a reference cost reduction J_{ref}, whereas the computation time t is divided by a reference computation time t_{ref}. These reference values are set to represent satisfactory performance for each optimizer. The function is designed such that the point $J^* = 1$ and $t^* = 1$ has a value of 0.5, indicating neutral performance. Also, the function places a larger weight on higher cost reductions, benefiting functions that run longer while achieving good cost reductions.

While running, every optimizer is monitored for adequate performance to determine if the optimizer is progressing well or has slowed or stalled and should be stopped. Efficiency is evaluated over a moving window of function calls. The first efficiency check is performed after a minimum number of 125% of the window, with a delay of half of the window before efficiency is evaluated again. This ensures that the optimizer is given a reasonable chance to reduce the cost and allows the optimizer to continue running for some time as long as it is performing well. If the efficiency is above a threshold of 0.95, the optimizer will continue to run; otherwise it will continue with probability equal to the efficiency value.

C. Optimizer Selection

The role of the optimizer selection process is to iteratively deploy a single optimizer with preference toward methods that perform well on the current problem. As the algorithm progresses toward the solution, different optimizers will be more effective on the problem than others. The optimizer selector learns from the performance of each optimizer to ensure that appropriate optimizers are selected. The optimizer selection takes the form of a variable structure learning automaton where an optimizer is selected based on a probability distribution. Probabilities are allocated according to the relative performance of each optimizer. Each optimizer is assigned a weight based on the efficiency of that optimizer from the last time it was used. The probabilities for each optimizer are distributed based on the weights according to Eq. (8):

$$p_i = \frac{w_i}{\sum_{i=1}^{N_{\text{opt}}} w_i} \tag{8}$$

The weights w_i are assigned using a continuous function defined to relate efficiency to weight given by $w_i = W_0 + W_1\sqrt{\eta_i}$. The efficiency values for each optimizer are also scaled using an exponential decay based on the time since that optimizer was last called.

D. Manager

The manager is the interface between different optimizers and ensures that each optimizer is provided the information it needs to operate on the problem. Given the diverse range of optimizers employed by the meta-optimizer, the exchange between optimizers is critical to the smooth operation of the algorithm. Each optimizer has various inputs and outputs that may not necessarily match with the next optimizer that will be employed. The responsibility of the optimizer manager is to provide a centralized system for starting each optimizer when it is selected, including providing a set of initial points and any information needed to initialize the optimizer. This also includes reseeding portions of the population and restarting the optimization process when necessary.

Initialization: At the beginning of the process, the global population is seeded with a large number of points generated based on a uniform random distribution over the search ranges for each parameter. This population is shared between all of the optimizers, allowing for the exchange of information between optimizers. The population must also be sufficiently large to provide enough points for any optimizer.

Transition to Single-Point Optimizer: For SD, CG, BFGS, and TS, only a single point is operated on at a time. When deployed, the best solution from the global population is used as the starting point for these optimizers. When the optimizer has completed, the previous best solution is replaced with the new value obtained from the optimizer.

Transition to Population-Based Optimizer: PSO, DE, SIM, IWO, and ACO all require a number of points drawn from the common population. Based on the population sizes for each optimizer, the best points are taken from the population, guaranteeing preservation of the global best solution found so far. Before these points are provided to the optimizer, the diversity in the set is checked using Eq. (9).

Diversity measures how spread out or clustered the points are in the population and is given by:

$$\sigma_d = \sqrt{\frac{\sum_{i=1}^{N_p} (x_{r,i} - \bar{x}_r)^2}{N_p - 1}} \tag{9}$$

where N_p is the population size, $x_{r,i} = \sqrt{\sum_{j=1}^{n} (x_i^j)^2}$, and $\bar{x}_r = (1/N) \sum_{i=1}^{N_p} x_{r,i}$ [38]. For these optimizers, the diversity plays a critical role in the ability of the optimizer to explore the parameter space. If the diversity falls below a threshold $\sigma_{d,\min}$, a portion of the set is reseeded. After running, the set of points is returned to the global population to be used by the next optimizer.

Reseeding of the Current Population: When reseeding is triggered, the manager reseeds a percentage of the population. Two methods are employed to generate new points, balancing exploitation of a region of interest with exploration of the parameter space. The first process is exploitation, which is performed with a given probability. New points are seeded based on a kernel density function (KDF) built from all previous solutions evaluated by the optimizers. As the meta-optimizer runs, optimizers will tend toward certain regions of the parameter space with low cost. Exploitation seeks to place more points within these regions, aiding the optimizers in refining the cost. The remaining points are seeded through exploration by randomly sampling from the full search range. This approach provides the optimizers with a highly diverse population, greatly increasing their exploration capacity. Most important, exploration places points far from any local minima, giving the optimizers an opportunity to search a new region of the parameter space and potentially escape a local minimum.

Restarting the Process: In the event that the meta-optimizer remains locked in a local minimum for a long period of time, the entire meta-optimization process is restarted. The frequency of restarts is governed by a limit on the number of function calls meta-optimization can expend while stalled in a local minimum. When the new population is generated, an exclusion zone is placed around all previously detected local minima to prevent any optimizer from starting too close to the local minima and quickly returning. The exclusion zone is given as a range about the local minimum for each parameter. After a certain number of restarts, the meta-optimizer stops, returning the best local minima found as the solution.

E. Auto-Tuning

The performance of any optimizer is dependent on the various parameters that control the optimizer's behavior. For example, PSO has four tuning parameters: population size, inertial weight, and two learning factor coefficients. Tuning parameters are unique to every optimizer and can vary greatly between applications. Typically, the selection of algorithm tuning parameters is determined by the user, whether from good values that have been used previously or through manual tuning of the parameters. For new optimization problems, the best parameters may not be known initially and require a significant amount of user effort to properly tune. Instead, auto-tuning is performed where the tuning procedure is conducted online within the meta-optimization procedure with a small chance of being called at any given iteration. This improves reliability by adapting the optimizers to the current problem, allowing for hands off operation of the individual optimizers.

Auto-tuning is performed using a wandering search that gradually explores the parameter space in a random manner. When auto-tuning is activated, the optimizer is run twice for 1000 function calls, once with the current parameters and once with new parameters. This provides a side-by-side comparison of the optimizer with the two parameter sets over a reasonable number of function calls. Only one parameter is tuned at a time to limit interactions between tuning parameters, with each parameter having an equal probability of being selected. New parameters are selected if it achieves a lower average cost and higher diversity.

V. Simulated Trajectory Results

To evaluate the performance of this new projectile parameter estimation method, a test case was considered using synthetic spark range data of the ANF with microspoilers described in Sec. II.C. These data are designed to mimic typical data obtained from a spark range, including measurement noise and Mach varying aerodynamic coefficients. During typical projectile flight testing, a buildup procedure is employed to accurately estimate the projectile aerodynamic coefficients and control parameters. First, the round is fired uncontrolled to estimate the body aerodynamic coefficients. Next, the microspoiler parameters are estimated from controlled shots with the body aerodynamic coefficients fixed to their estimated values. This helps simplify the estimation problem and isolates the effects of the microspoiler from the body aerodynamic behavior, allowing for more accurate estimates. The results presented here show only the controlled estimation case with estimates of the body aerodynamic coefficients previously determined by the parameter estimation method. For this test, the aerodynamic coefficients and microspoiler parameters are assumed to vary linearly with Mach number. This assumption generally holds in practice over small ranges in Mach number.

The initial conditions for each simulated trajectory were randomly generated based on a nominal value of $h_0 = 5$ m and $u_0 = 1023$ m/s and standard deviations given in Table 2. The standard deviation on u_0 was selected to produce a sufficient distribution of Mach numbers to properly estimate the linear aerodynamic coefficients. ϕ_0 was initialized to a random value between $-180°$ and $180°$, and ω_0 was set to a random value between $0°$ and $360°$.

Simulated measurements are recorded at the range locations of the spark stations at the U.S. Army Research Laboratory Transonic Experimental Facility spark range. Measurement noise is added with standard deviation of 3 mm and $0.1°$ for position and angle measurements, respectively. It is also assumed that there is a 10% chance that a measurement cannot be made, reducing the total number of measurements for each shot. The ϕ measurements are wrapped to $-180°$ and $180°$. Equation (10) is used to characterize the aerodynamic coefficients within the cost function with a Mach range from 2.75 to 3.25.

$$C(M) = C_{lo} + (C_{hi} - C_{lo}) \frac{M - M_{lo}}{M_{hi} - M_{lo}} \tag{10}$$

where C_{lo} and C_{hi} are estimates of the coefficient at the Mach number limits. This range covers the potential distribution of Mach numbers encountered in the simulated trajectories.

A. Microspoiler Parameter Estimation

The parameters associated with the microspoilers include the axial force coefficient δ_a, the normal force coefficient δ_N, and the pitching moment coefficient δ_m as well as the microspoiler mechanism initial phase ω_0, spin rate Ω_0, and time constant τ_{ms}. The initial phase, spin rate, and time constant are estimated for every trajectory. Unlike the previous cases, the trajectory prediction simulations in the cost function were started from the first measurement. This was done to limit the interactions between the initial pitch rate at launch and the microspoiler pitching moment as the microspoilers produce a similar effect at launch as a large angular velocity perturbation. Under some conditions, changes in the estimates of the microspoiler parameters may be matched by changes in the estimated pitch rate, yielding minimal changes in the overall trajectory. To handle fitting the data in this manner, a modification is needed to the microspoiler model. When starting the simulation at the first measurement, the microspoiler mechanism will already be spinning at some rate based

Table 2 Standard deviations of initial conditions used to generate synthetic data

θ, rad	ψ, rad	u, m/s	p, rad/s	q, rad/s	r, rad/s
0.001	0.001	35.0	2.0	1.0	1.0

Table 3 Simulated microspoiler coefficient parameter estimation results

	$M = 2.75$			$M = 3.25$		
Parameter	Actual	Estimate	% Error	Actual	Estimate	% Error
δ_A (N)	−26.325	−27.046	2.544	−31.625	−33.495	5.914
δ_N (N)	67.225	84.99	26.44	80.175	73.78	7.979
δ_m (N · m)	8.4175	8.341	0.9096	10.0825	10.249	1.648

Table 4 Simulated microspoiler mechanism parameter estimation results

Parameter	Actual	Mean estimate	STD estimate	% Error
Ω_0, rad/s	440.0	440.44	0.5569	0.101
τ_{ms}, 1/s	0.025	0.026	0.0023	4.051

Table 5 χ^2 values for simulated active microspoiler trajectories

	x	y	z	ϕ	θ	ψ
Average	23.87	18.12	22.88	21.87	24.91	23.43
Standard deviation	11.19	3.575	8.074	6.973	2.546	9.551

on the time since launch. The time of launch can therefore be added to the model such that the spin rate is given by:

$$\Omega(t) = \Omega_0(1 - e^{-((t-t_{\text{initial}})/\tau_{ms})}) \tag{11}$$

and the phase is given by:

$$\omega(t) = \omega_0 + \Omega_0((t - t_{\text{initial}}) + \tau_{ms}e^{-((t-t_{\text{initial}})/\tau_{ms})} - \tau_{ms}) \tag{12}$$

with the initial time estimated based on a least squares fit of the x and t data.

Five trajectories were used to estimate the microspoiler parameters. All six states were included in the cost function with equal weighting. By beginning the trajectory prediction simulations at the first measurement, estimates are needed for the initial conditions of every state, resulting in 81 parameters to estimate. The test results for this case are given in Tables 3 and 4 and the final χ^2 values in Table 5.

Overall, the parameter estimation algorithm is able to accurately estimate the microspoiler parameters with only small errors in δ_A and larger errors δ_N. These errors may be due to the small errors in C_{X2} and $C_{N\alpha}$ previously estimated. These parameters are also difficult to estimate as the cost function is not sensitive to errors in these parameters. There may also be some coupling between the initial conditions and the microspoiler parameters, which induce additional errors in the estimates. The estimation algorithm is also successful in estimating the spin rate and time constant for each trajectory, which have more indirect effects on the projectile dynamics. The χ^2 values indicate excellent fits of all states with errors on the order of the number of measurements for every trajectory.

The trajectory results for one of the data sets used by the parameter estimation algorithm are given in Figs. 8–13. Here, the final estimated trajectory is shown alongside the measurements and the initial trajectory obtained during the initialization phase of the parameter estimation process. The initial trajectory provides an indication of how poor the initial parameter estimates were and how much of an improvement this method makes, with the initial trajectories quickly diverging from the measurements. Looking at the trajectories for each state, the estimation algorithm does an excellent job at fitting a trajectory to the available data as indicated by χ^2. Especially for y and h, there is little disagreement between the fit trajectory and the measurements. Considering how well the estimation algorithm fit these states, the large errors in δ_N could not have played a major role in obtaining this fit. Figures 11 and 12 show excellent fitting of θ and ψ, which allows for accurate estimation of δ_m.

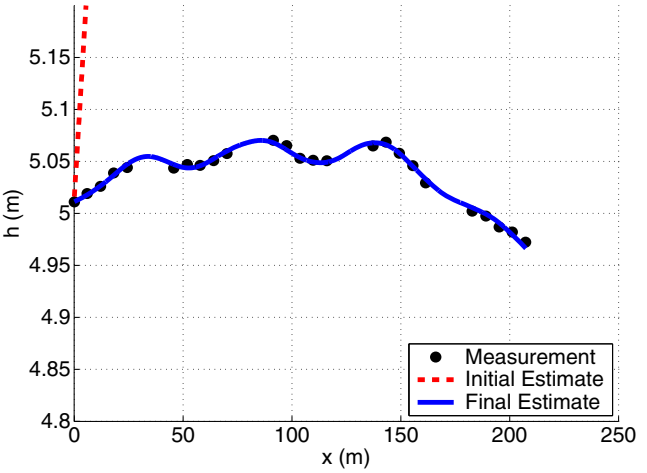

Fig. 9 Simulated active microspoilers altitude vs range.

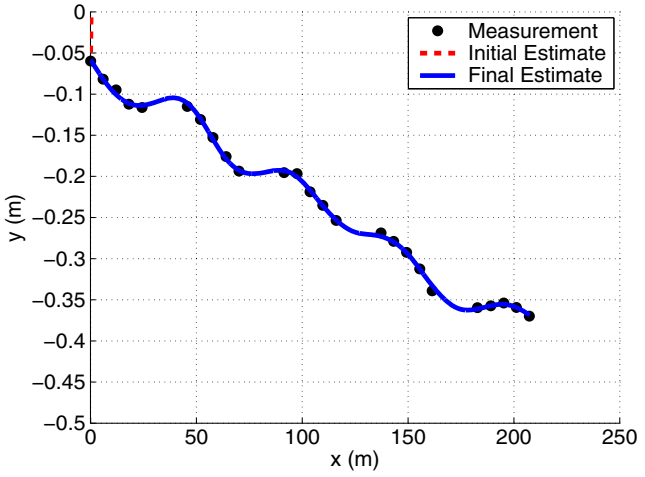

Fig. 8 Simulated active microspoilers inertial-Y position vs range.

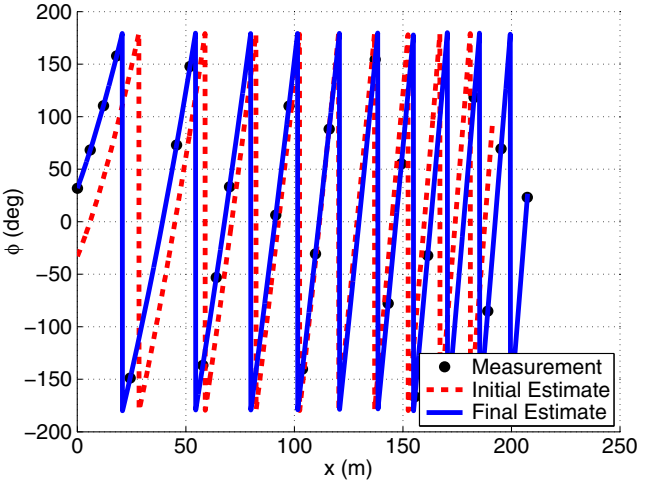

Fig. 10 Simulated active microspoilers roll angle vs range.

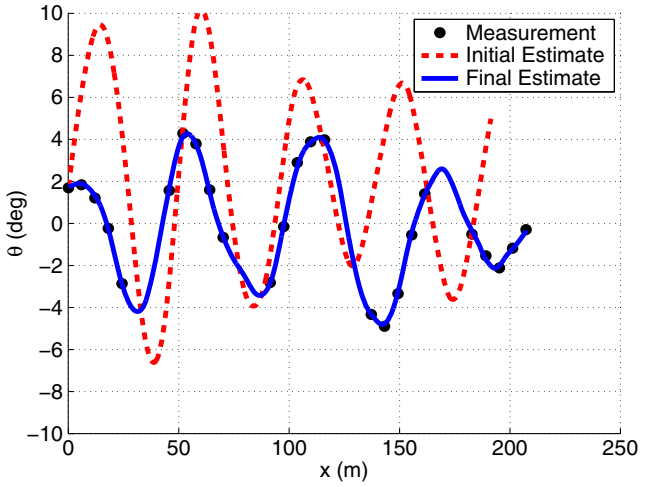

Fig. 11 Simulated active microspoilers pitch angle vs range.

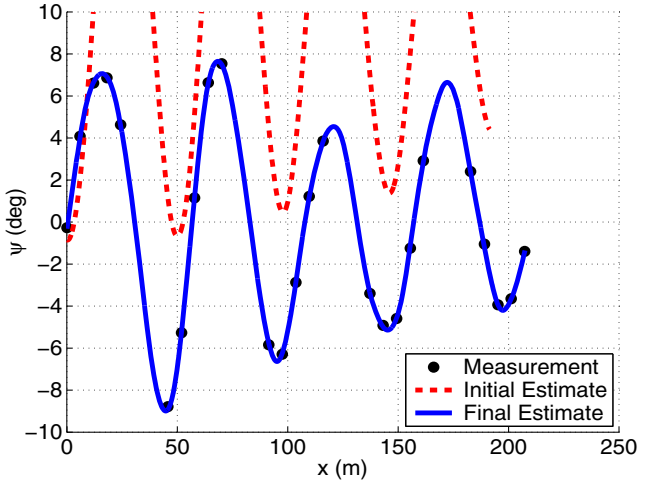

Fig. 12 Simulated active microspoilers yaw angle vs range.

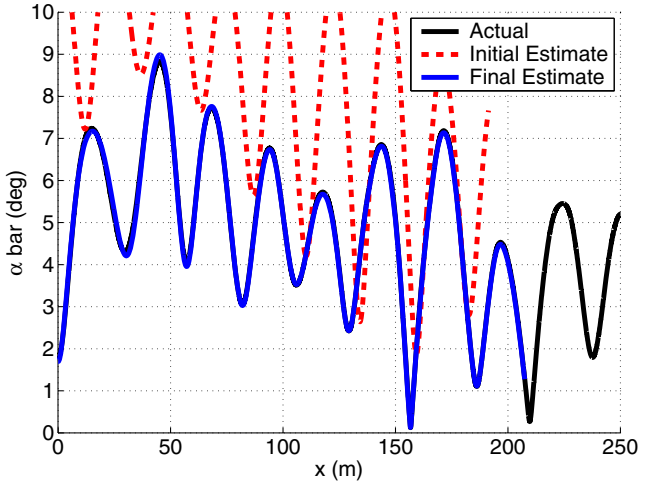

Fig. 13 Simulated active microspoilers total angle of attack vs range.

Looking at the cost profile in Fig. 14, the meta-optimizer requires a large number of function calls to solve this problem. A cost threshold is defined based on an acceptable χ^2 value for each trajectory. The meta-optimizer stops when there has been no cost reduction for a period of time after crossing the threshold. The meta-optimizer begins with a large initial reduction over the first 200,000 function calls, followed by some smaller reductions over the next 500,000 function calls before progress stalls almost completely after about

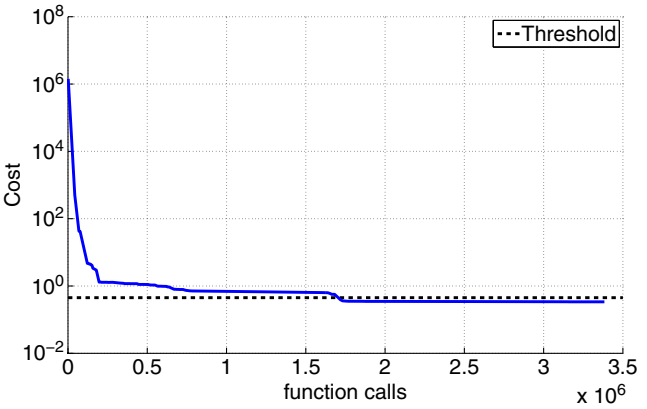

Fig. 14 Simulated active microspoilers cost profile.

800,000 function calls. It takes the meta-optimizer over 800,000 more function calls to get going again when SIM is able to make significant progress. After crossing the threshold at this time, the meta-optimizer continues making small improvements over another 1.5 million function calls.

Figure 15 shows the total percent cost reduction for each optimizer over this run. This is a cumulative metric that evaluates the total contributions of each optimizer toward reducing the cost. On this run, CG, BFGS, and SIM dominate the cost reductions with small amounts from SD, DE, and TS. Because of the stochastic nature of the optimizer selection in meta-optimization, some optimizers will not be given good opportunities to reduce the cost in an individual run. Trends between the individual optimizers could be observed over a large number of trials. The total number of times each optimizer was deployed is shown in Fig. 16. Overall, the optimizers were given

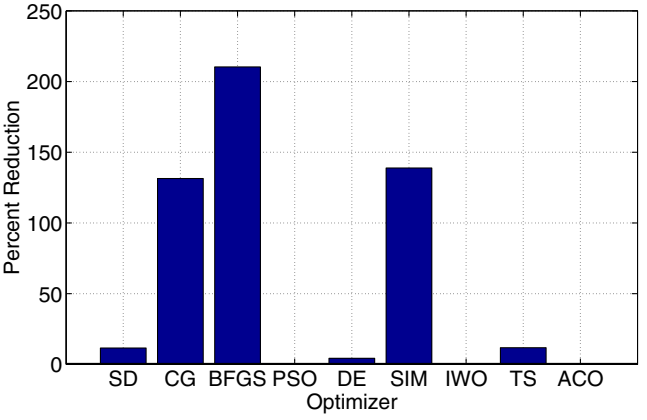

Fig. 15 Simulated active microspoilers total percent cost reduction.

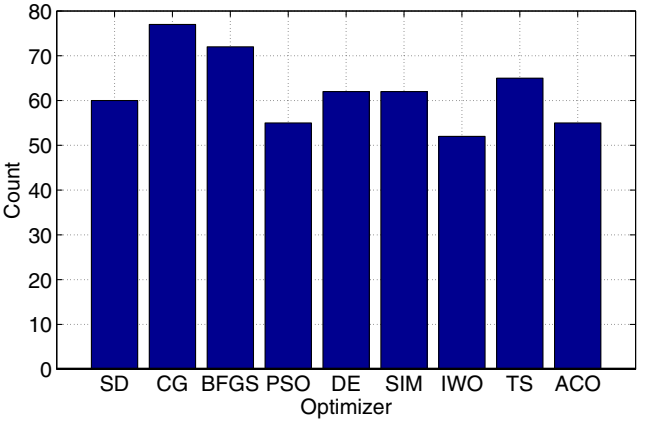

Fig. 16 Simulated active microspoilers total number of calls of each optimizer.

roughly equal opportunities with some preference toward CG and BFGS, which performed well. The total number of function calls used by each optimizer in Fig. 17 provides additional insight into the distribution of resources between the optimizers. Here, SIM used the most function calls as it was very effective at reducing the cost. CG and BFGS also used a large portion of the function calls, which also corresponds to these optimizers being deployed the most. The total function calls used can be combined with the cost reduction to get insight into the efficiency of each optimizer. In particular, CG and BFGS were very efficient compared with SIM, achieving larger cost reductions for less computation time. TS, on the other hand, was very inefficient as it used the second most function calls, but only achieved a small cost reduction.

Profiles for each of the microspoiler parameters are given in Figs. 18–22. The first parameters to converge are the spin rates for

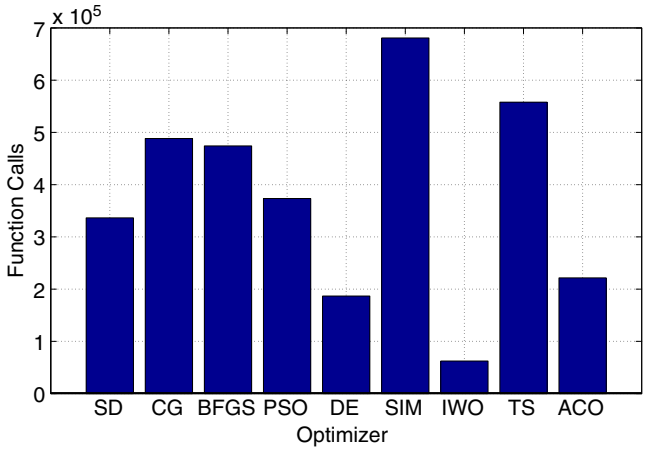

Fig. 17 Simulated active microspoilers total function calls used by each optimizer.

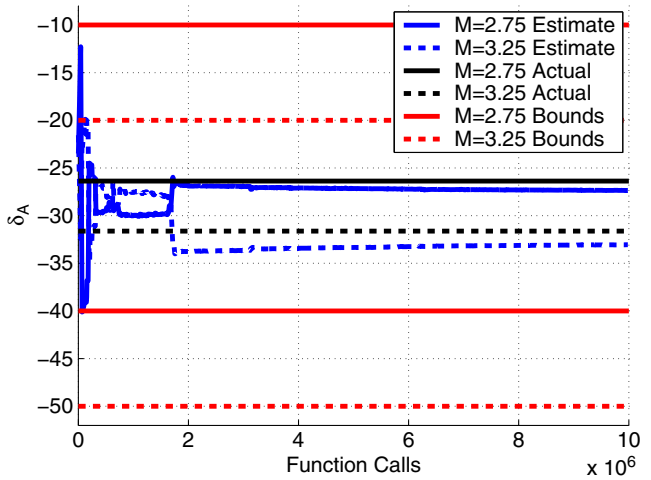

Fig. 18 Simulated active microspoilers axial force profile.

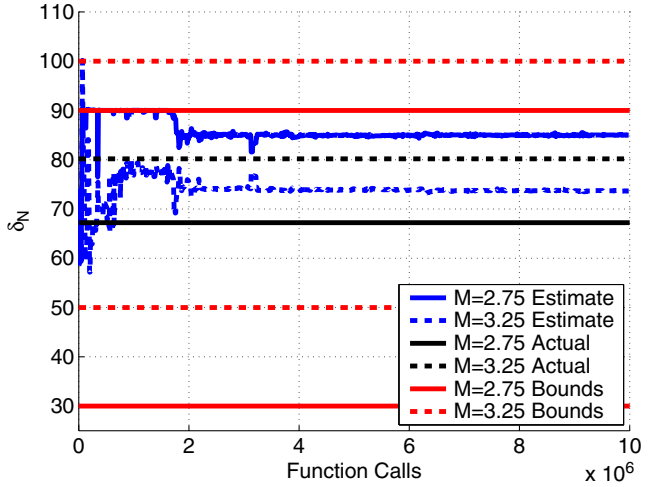

Fig. 19 Simulated active microspoilers normal force profile.

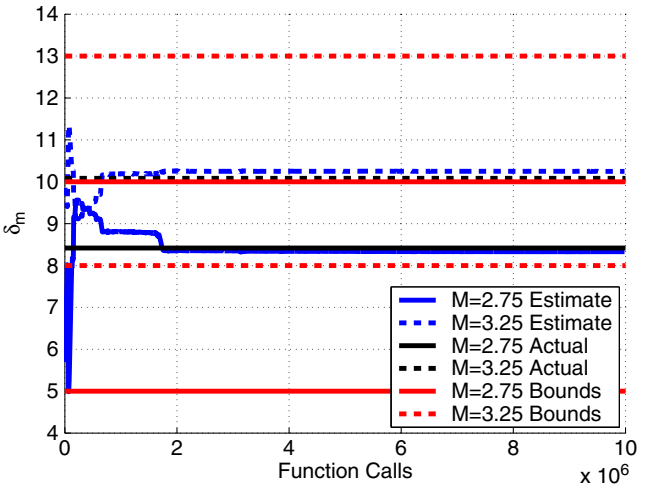

Fig. 20 Simulated active microspoilers pitching moment profile.

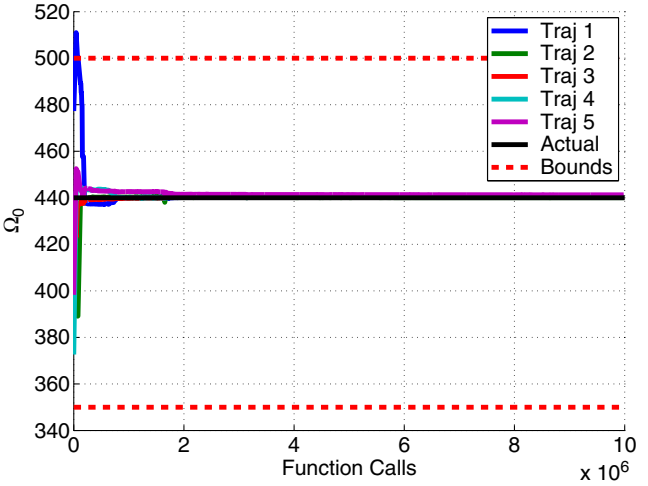

Fig. 21 Simulated active microspoilers nominal spin rate profile.

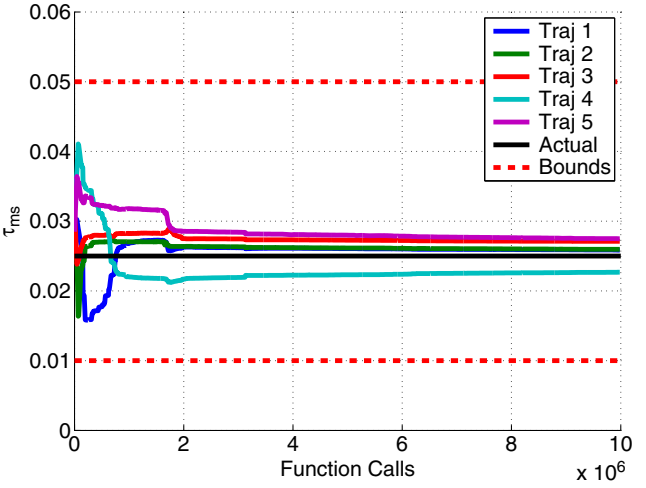

Fig. 22 Simulated active microspoilers time constant profile.

Table 6 Final cost for different optimizers

SD	CG	BFGS	PSO	DE	SIM	IWO	TS	ACO	MO
135.66	24.021	119.43	2538.4	0.3987	970.21	236.39	83.839	0.4281	0.2975

each trajectory with trajectories 2 and 3 the first to settle, followed by 1 and 4 after 800,000 function calls, and finally, trajectory 5 at around 1.6 million function calls, corresponding to the final jump in cost in Fig. 14. All of the parameters make large changes at this time, with δ_A and δ_m moving very close to their actual values. The $M = 2.75$ estimate of δ_N remains far from its actual value for the entire process, remaining on the search boundary of 90 for a long time. Ω_0 for trajectory 1 exceeds the boundary early on, but quickly returns to the search space. At least one parameter for each of the coefficients also briefly reach the boundary. Finally, τ_{ms} varies for each trajectory with all of the final values within 15% of the actual values. When the simulation is started from the first measurement, the microspoilers are already spinning at about 75% of the max speed by the time the simulation starts, making the impact of τ_{ms} on the trajectory less than if the simulation began at launch.

B. Individual Optimizer Comparison

It is also useful to examine a comparison between the performance of the parameter estimation algorithm using both the standard individual optimizers and the new method meta-optimization. The simulated high-angle-of-attack case with measurement noise is used for this test. Here, each optimizer was started from the same initial point and with the same set of points. The optimizers then run for 1,000,000 function calls or satisfy their stopping criteria. The final cost for each of the optimizers is given in Table 6. The cost threshold for this problem was 0.45, which DE, ACO, and meta-optimization reach. PSO performs very poorly, indicating poor tuning of its parameters for this problem. All of the hill climbers reach a local minima far from the solution with SIM stalling as well.

The cost reduction profile for each optimizer is shown in Fig. 23. In this case, meta-optimization is the first optimizer to cross the cost threshold and the only optimizer to reach the solution in the allotted time. However, almost all of the other optimizers are faster at initially reducing the cost with SIM particularly efficient. However, it quickly plateaus after only 9000 function calls. The hill climbers also quickly reach local minima and stop there. PSO, IWO, and TS all gradually reduce the cost, but do not come close to the solution. Of the two remaining optimizers, ACO is very efficient over the first 50,000 function calls, but then slows considerably as it approaches the solution. DE takes a more gradual path, also slowing down as the cost decreases. The meta-optimizer, on the other hand, is slightly less efficient over the initial phase, but rapidly reaches the solution in under 200,000 function calls.

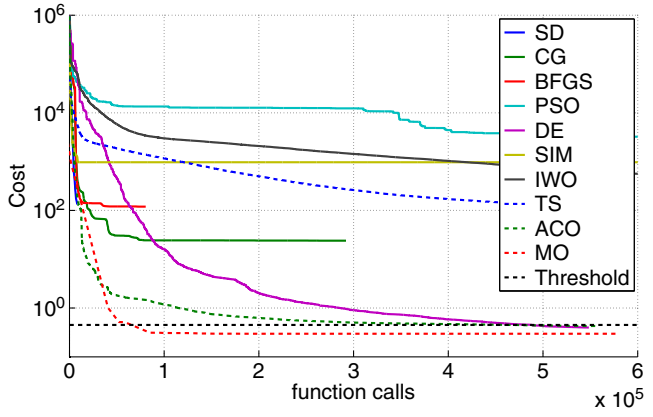

Fig. 23 Comparison of optimizer performance on simulated high-angle-of-attack problem.

VI. Conclusions

The process of performing parameter estimation for new projectile configurations is a key component of the design process. Typically, flight test data are used in addition to CFD and wind tunnel data when available. Although many techniques currently exist for estimating the aerodynamic coefficients for projectiles based on flight test data, these methods rely on accurate initial estimates for the coefficients to ensure convergence. For new projectile systems with complicated control mechanisms and generally unknown parameters, a new method is needed to perform parameter estimation. The proposed method is based on the concept of meta-optimization where the most efficient optimizer is deployed to solve a given problem. The meta-optimizer uses reinforcement learning to select which optimizer to use at a given point in the solution process. This new parameter estimation method was used to estimate the parameters for a new projectile equipped with microspoilers using simulated test data. The proposed method was able to estimate the microspoiler parameters with good accuracy when using noisy data. Residual errors in the estimates can be attributed to the noise in the data and a lack of observability of some parameters. This new parameter estimation algorithm using meta-optimization has proven to be an effective tool for analyzing new smart projectile systems.

Acknowledgment

This work has been supported by the U.S. Army Research Laboratory under contract W911QX-12-C-0119, with Frank Fresconi as the contract monitor.

References

[1] Fresconi, F., Celmins, I., and Howell, B., "Obtaining the Aerodynamic and Flight Dynamic Characteristics of an Asymmetric Projectile Through Experimental Spark Range Firings," *AIAA Atmospheric Flight Mechanics Conference*, AIAA Paper 2011-6334, 2011. doi:10.2514/6.2011-6334

[2] Massey, K. C., and Silton, S., "Testing the Maneuvering Performance of a Mach 4 Projectile," *AIAA Applied Aerodynamics Conference*, AIAA Paper 2006-3649, 2006. doi:10.2514/6.2006-3649

[3] Scheuermann, E., Costello, M., Silton, S., and Sahu, J., "Aerodynamic Characterization of a Microspoiler System for Supersonic Finned Projectiles," *Journal of Spacecraft and Rockets*, Vol. 52, No. 1, 2015, pp. 253–263. doi:10.2514/1.A33005

[4] Fresconi, F., and Harkins, T., "Experimental Flight Characterization of Asymmetric and Maneuvering Projectiles from Elevated Gun Firings," *Journal of Spacecraft and Rockets*, Vol. 49, No. 6, 2012, pp. 1120–1130. doi:10.2514/1.A32200

[5] Silton, S., "Numerical Experiments on Finned Bodies," *AIAA Applied Aerodynamics Conference*, AIAA Paper 2014-3021, 2014. doi:10.2514/6.2014-3021

[6] Silton, S., and Fresconi, F., "The Effect of Canard Interactions on Aerodynamic Performance of a Fin-Stabilized Projectile," *AIAA Aerospace Sciences Meeting*, AIAA Paper 2015-1924, 2015. doi:10.2514/6.2015-1924

[7] Dykes, J., Montalvo, C., Costello, M., and Sahu, J., "Use of Microspoilers for Control of Finned Projectiles," *Journal of Spacecraft and Rockets*, Vol. 49, No. 6, 2012, pp. 1131–1140. doi:10.2514/1.A32274

[8] Stahl, J., Costello, M., and Sahu, J., "Projectile Aerodynamic Coefficient Estimation Using Integrated CFD/RBD and Flight Control System Modeling," *AIAA Atmospheric Flight Mechanics Conference*, AIAA Paper 2009-5715, 2009. doi:10.2514/6.2009-5715

[9] Montalvo, C., and Costello, M., "Estimation of Projectile Aerodynamic Coefficients Using Coupled CFD/RBD Simulation Results," *AIAA Atmospheric Flight Mechanics Conference*, AIAA Paper

2010-8249, 2010.
doi:10.2514/6.2010-8249

[10] Hathaway, W., Steinhoff, M., Whyte, R., Brown, D., Choate, J., and Aldergren, R., "Expert Systems and Ballistic Range Data Analysis," *AIAA Aerospace Ground Testing Conference*, AIAA Paper 1992-3999, 1992.
doi:10.2514/6.1992-3999

[11] Fischer, M., and Hathaway, W., "Aeroballistic Research Facility Data Analysis System (ARFDAS)," U.S. Air Force Armament Laboratory AFATL-TR-88-48, Eglin Air Force Base, FL, Sept. 1988.

[12] Sahu, J., "Time-Accurate Numerical Prediction of Free Flight Aerodynamics of a Finned Projectile," *AIAA Atmospheric Flight Mechanics Conference and Exhibit*, AIAA Paper 2005-5817, 2005.
doi:10.2514/6.2005-5817

[13] Burchett, B. T., "Aerodynamic Parameter Identification for Symmetric Projectiles: Comparing Gradient Based and Evolutionary Algorithms," *AIAA Atmospheric Flight Mechanics Conference*, AIAA, Reston, VA, 2011.
doi:10.1016/j.ast.2013.07.010

[14] Burchett, B. T., "Aerodynamic Parameter Identification for Symmetric Projectiles: An Improved Gradient Based Method," *Aerospace Science and Technology*, Vol. 30, No. 1, 2013, pp. 119–127.
doi:10.1016/j.ast.2013.07.010

[15] Condaminet, V., Delvare, F., Choi, D., Demailly, H., Grignon, C., and Heddadj, S., "Identification of Aerodynamic Coefficients of a Projectile and Reconstruction of Its Trajectory from Partial Flight Data," *Computer Assisted Method in Engineering and Science*, Vol. 21, Jan. 2014, pp. 177–186.

[16] Griva, I., Nash, S. G., and Sofer, A., *Linear and Nonlinear Optimization*, 2nd ed., Soc. for Industrial and Applied Mathematics, 2009.

[17] Kennedy, J., and Eberhart, R., "Particle Swarm Optimization," *IEEE Conference on Neural Networks*, IEEE Publ., Piscataway, NJ, 1995.
doi:10.1109/ICNN.1995.488968

[18] Shi, Y., and Eberhart, R., "A Modified Particle Swarm Optimizer," *IEEE World Congress on Computational Intelligence*, IEEE Publ., Piscataway, NJ, 1998.
doi:10.1109/ICEC.1998.699146

[19] Storn, R., and Price, K., "Differential Evolution—A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization*, Vol. 11, No. 4, 1997, pp. 341–359.
doi:10.1023/A:1008202821328

[20] Neri, F., Cotta, C., and Moscato, P., *Handbook of Memetic Algorithms*, Studies in Computational Intelligence, Springer, 2012.
doi:10.1007/978-3-642-23247-3

[21] Moscato, P., "On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Toward Memetic Algorithms," Caltech Concurrent Computation Program TR 826, California Institute of Technology, Pasadena, CA, 1989.
doi:10.1.1.27.9474

[22] Chen, X., Ong, Y.-S., Lim, M.-H., and Tan, K. C., "A Multi-Facet Survey on Memetic Computation," *IEEE Transactions on Evolutionary Computation*, Vol. 15, No. 5, 2011, pp. 591–607.
doi:10.1109/TEVC.2011.2132725

[23] Klein, V., and Morelli, E. A., *Aircraft System Identification: Theory and Practice*, AIAA, Reston, VA, 2006, Chap. 6.
doi:10.2514/4.861505

[24] Raol, J. R., Singh, J., and Girija, G., *Modelling and Parameter Estimation of Dynamic Systems*, The Institution of Engineering and Technology, 2004, Chap. 3.
doi:10.1049/PBCE065E

[25] Hughes, I., and Hase, T., *Measurements and Their Uncertainties: A Practical Guide to Modern Error Analysis*, Oxford Univ. Press, 2010.

[26] Etkin, B., *Dynamics of Atmospheric Flight*, Dover Publ., 2000.

[27] Vanderplaats, G. N., *Multidiscipline Design Optimization*, Vanderplaats Research and Development, 2007.

[28] Fletcher, R., and Reeves, C. M., "Function Minimization by Conjugate Gradients," *The Computer Journal*, Vol. 7, No. 2, 1964, pp. 149–154.
doi:10.1093/comjnl/7.2.149

[29] Mehrabian, A. R., and Lucas, C., "A Novel Numerical Optimization Algorithm Inspired from Weed Colonization," *Ecological Informatics*, Vol. 1, No. 4, 2006, pp. 355–366.
doi:10.1016/j.ecoinf.2006.07.003

[30] Bolandi, H., Ashtari Larki, M. H., Sedighy, S. H., Zeighami, M. S., and Esmailzadeh, M., "Estimation of Simplified General Perturbations Model for Orbital Elements from Global Positioning System Data by Invasive Weed Optimization Algorithm," *Journal of Aerospace Engineering*, Vol. 229, No. 8, 2014, pp. 1384–1394.
doi:10.1177/0954410014550323

[31] Karimkashi, S., and Kishk, A. A., "Invasive Weed Optimization and Its Features in Electromagnetics," *IEEE Transactions on Antennas and Propagation*, Vol. 58, No. 4, 2010, pp. 1269–1278.
doi:10.1109/TAP.2010.2041163

[32] Zaharis, Z. D., Skeberis, C., Xenos, T. D., Lazaridis, P. I., and Cosmas, J., "Design of a Novel Antenna Array Beamformer Using Neural Networks Trained by Modified Adaptive Dispersion Invasive Weed Optimization Based Data," *IEEE Transactions on Broadcasting*, Vol. 59, No. 3, 2013, pp. 455–460.
doi:10.1109/TBC.2013.2244793

[33] Socha, K., and Dorigo, M., "Ant Colony Optimization for Continuous Domains," *European Journal of Operational Research*, Vol. 185, No. 3, 2008, pp. 1155–1173.
doi:10.1016/j.ejor.2006.06.046

[34] Nelder, J. A., and Mead, R., "A Simplex Method for Function Minimization," *The Computer Journal*, Vol. 7, No. 4, 1965, pp. 308–313.
doi:10.1093/comjnl/7.4.308

[35] Lagarias, J. C., Reeds, J. A., Wright, M. H., and Wright, P. E., "Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions," *SIAM Journal of Optimization*, Vol. 9, No. 1, 1998, pp. 112–147.
doi:10.1137/S1052623496303470

[36] Chelouah, R., and Siarry, P., "Tabu Search Applied to Global Optimization," *European Journal of Operational Research*, Vol. 123, No. 2, 2000, pp. 256–270.
doi:10.1016/S0377-2217(99)00255-6

[37] Siarry, P., and Berthiau, G., "Fitting of Tabu Search to Optimize Functions of Continuous Variables," *International Journal for Numerical Methods in Engineering*, Vol. 40, No. 13, 1997, pp. 2449–2457.
doi:10.1002/(ISSN)1097-0207

[38] Zielinksi, K., Peters, D., and Laur, R., "Stopping Criteria for Single-Objective Optimization," *Proceedings of the Third International Conference on Computational Intelligence, Robotics and Autonomous Systems*, 2005.

A.V. Rao
Associate Editor