

Parachute-smart machine learning algorithm for inflation time assessment from test data.

Jean Potvin¹

Saint Louis University, St. Louis, MO, 63103, USA

John Rangel²

*National Aeronautics and Space Administration
Johnson Space Center, Houston, TX, 77058, USA*

A novel “parachute-smart” neural network algorithm is proposed as a tool for finding hard-to-get algebraic relationships tying performance and canopy design data collected during test drops, wind tunnel investigations or high-order computer simulations. Herein, inflation performance data of the USAF C-9 personnel recovery parachute and NASA’s Orion three-cluster landing ringsail parachutes are used to computationally establish, in each case, novel functional relationships between their inflation times (non-dimensional), inflated canopy volume, initial skirt area and overall canopy size scale. Interesting analysis byproducts are produced as well, thereby offering new insights on unmeasurable dynamical variables such as the average through-skirt flux and the role of the sails in ringsail designs acting as “air scoops” during the inflation process.

I. Nomenclature

CDT	= Cluster Development Test (series)
CPAS	= Capsule Parachute Assembly System
CQT	= Cluster Qualification Test (series)
D_0	= nominal diameter of a fully dis-reefed inflated canopy
D_{ref}	= reference diameter ($=D_0 (\epsilon_{final} - \epsilon_{initial})^{1/2}$)
dt	= integration time increment
EDU	= Engineering Development Unit (test series)
$F_D(t)$	= drag force
F_{peak}	= maximum drag sustained during inflation
g	= constant of gravitational acceleration
I	= parachute drag integral (Eq. 15)
m_{total}	= canopy mass, plus mass of the payload shared by each cluster member
MSE	= mean square error
n_{infl}	= non-dimensional inflation time
PINN	= physics-informed neural networks
S_0	= canopy surface area based on the nominal diameter
S_{skirt}	= area of the skirt
t	= time
t_{infl}	= inflation time
t_{init}	= time at the beginning of inflation
t_{final}	= time at the end of inflation
$Vol(t_{final})$	= canopy inflated volume at the end of inflation
$V(t)$	= parachute-payload speed

¹ Professor, Department of Electrical and Computer Engineering, and AIAA Associate Fellow.

² Analysis Engineer, CCP Parachutes Analysis Lead, AIAA Young Professionals member.

$u_{skirt}(t)$	= through-skirt flow velocity at time “t”
U_{init}	= parachute-payload speed at the beginning of (dis-reefing) inflation
U_{final}	= parachute-payload speed towards the end of (dis-reefing) inflation
w_i	= neural network weight associated with input variable “i”
X_i	= neural network input variable “i”
$\hat{Y}_i$	= sigmoid activation function (Eq. (3))
α	= learning parameter
ϕ	= “mouth-to-vent” porosity ratio; an indirect measurement of porosity
ρ	= air density during inflation
Σ	= Summation (Eq. (1))
$\Sigma^{(k)}$	= Summation for dataset “k”

II. Introduction

Artificial Intelligence (“AI”) emulates, via computer simulations akin to neural networks, a brain’s information handling via neuronal connections to solve a variety of problems in science, engineering and technology. Articulated into computer coding, AI has also increased its reach into the written, visual and musical arts, as well as in decision-making. (A simple introduction is presented in Ref. [1].) Often sold as a silver bullet capable of solving most of society’s problems, AI has nevertheless yielded outcomes that have turned out erroneous and/or misleading [2]. Generally, AI can be successful as long as the user is aware of the neural network underlying assumptions and input data limitations [2]. This isn’t to say that AI is unsuitable to engineering – quite to the contrary - as long as one is aware of the assumptions and data limitations - as illustrated by the analysis presented here.

A large number of AI algorithms have been devised over the past three decades. *All* attempt at mimicking a brain’s neuronal system in which each neuron outputs an electrical signal upon receiving one or many input signals (Fig. 1). In neural network simulations, the input and output are numbers, or combination of numbers, arranged/modified by each neuron through the use of mathematically conditional criteria, formulas, etc. (Fig. 1). A network of simulated neurons is allowed to communicate their output to other “neurons” i.e., in parallel to other computer processors, or sequentially via program loops, to mathematically carry out a workflow similar biological neuronal networks. Nowadays, neural network algorithms can treat all categories of data including, 1) the written word (e.g., Chat-GPT and others), 2) images (e.g., ResNet and many others

(https://en.wikipedia.org/wiki/Image_analysis)), 3) sound (e.g., https://en.wikipedia.org/wiki/Music_and_artificial_intelligence), and finally, 4) the numerical outputs of sensors used in science and engineering. This paper shall deal with this latter kind, as applied to data collected in parachute test drop, wind tunnel investigations and/or FSI simulations.

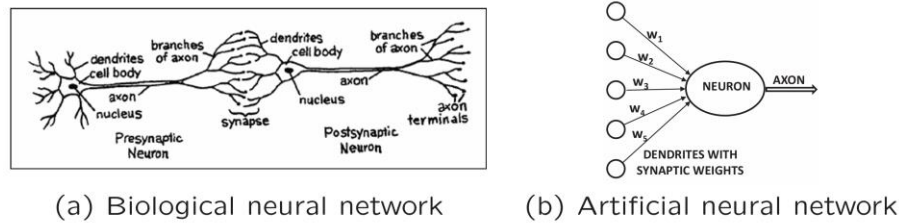


Figure 1. Schematic renditions of a biological neural network (a) and a single-neuron artificial neuron network (b). In both, inputs (“stimuli”) enter the neuron with varying intensity weighed by the factors w_i . The neuron carries out a computation which then determines its output. Figure extracted from [1].

AI has seen increasing use to extract system performance trends out of a large body of data, rather than from the explicit solutions of complex differential equations systems coupling kinematics and dynamics. One reason is that AI isn’t explicitly depending on a complete set of initial-value parameters – oftentimes lacking in many applications. Moreover, AI isn’t using aerodynamical approximations which may not be applicable to the problem at hand, as is often the case with parachute dynamics. In principle, and as demonstrated in the *Universal Approximation Theorem* [3], availability of *large-enough* datasets should “contain” all allowable motions of a system as emergent behaviors.

Unfortunately, limited availability – or even unavailability - of “large-enough” datasets will stand in the way of attaining such Ideal. One work-around has been to explicitly embed in the neuronal decision-making process the physical laws that govern the “behaviors” underlying the dataset via, for example, partial differential equations (e.g., Navier-Stokes), or as described below, algebraic expressions that are true – even exact - to the problem at hand. This approach, also known as *Physics-Informed Neural Networks* (PINNs) has been used widely and produced a voluminous body of work (e.g., Refs. [4, 5]). The paper by Yang et al. [6] which follows this presentation is another example but applied to parachute motion modeling; and so are the parachute analyses of Refs. [7] and [8] which have appeared in the recent past.

Nowadays, large data sets are made possible by the use of miniaturized sensor suites and data collection/transmission hardware that can be outfitted to all relevant parts of a system. (By-the-way, such large sets can also be constructed from the data shared by thousands of users utilizing similar sensor suite and/or AI platform.) Unfortunately, parachute datasets remain small, mostly due to the very nature of parachutes as flexible structures for which the structural forces, as well as the materials’ interaction with the air, are difficult to probe over their entire volume of relevance. Herein, we present a novel PINN or “parachute-smart” AI algorithm based on a simple “perceptron” neural network [1] (i.e., a single hidden layer neuron in AI parlance) and constructed for the analysis of parachute inflation dynamics. This algorithm aims at capturing the best of two worlds by using universally-valid but simple physical laws to guide the neural network iterative process towards the correct solution quickly and from limited amounts of data. Herein, parachute inflation data relevant to the USAF C-9 personnel recovery parachute [9-13], as well as those of the main parachute cluster of NASA’s Orion capsule recovery system [14, 15], are fed into a novel PINN to establish novel functional relationships between their (non-dimensional) inflation times, inflated canopy volume, initial skirt area and overall canopy size scale. Interesting analysis byproducts are produced as well, thereby offering new insights on unmeasurable dynamical variables that are also critical to the inflation process.

This paper is organized as follows. Section III first presents a simple introduction to neural network simulations before embarking onto the description of our parachute-smart neural network calculations. Section IV presents the results which are then further discussed in Section V. The paper concludes, in Section VI, with observations on the required dataset size.

III. A Parachute-Smart Neural Network Algorithm

A. Curve-fitting vs. AI

The overarching goal is to use a neural network-based approach to enable the reduction of experimental or computational (FSI) data into algebraic relationships that not only interpolate among the input datasets but ones that are capable of extrapolation as well. The process is similar to the familiar curve-fitting approach of old, in which a given functional form (e.g., linear, polynomial, power, logarithmic, etc.) is fitted through experimental data [16]. Here “fitting” alludes to a minimization process in which the coefficients defining the given functional form, say the coefficients A and B in the form $y = Ax + B$, have been “picked” so to minimize the error between a predictive straight line and each point within the dataset [16]. This curve-fitting approach is advantageous, obviously, where the data 1) depend on a handful of independent variables; 2) follows simple trend lines that are easily expressed as simple mathematical functions; and 3) the number of data points far exceeds the number of parameters needed to define the functional form/trend line. But the approach falls apart when the number of dependent variables is very large (e.g., parachute system construction parameters and test drop kinematic and atmospheric envelope, etc.), thereby pointing to a multidimensional relationship among these parameters and the dependent variable of interest (inflation times, drag force, etc.).

Extreme multidimensionality is bad as it generally prevents trend lines from following easily-identifiable functions that could be used in standard curve-fitting. PINN-based approaches, on the other hand, avoid the problem by firstly using as input a well-known and general functional dependence of the physical laws governing the “behavior” of an engineered system, rather than the anticipated forms/trends of the solutions (the output); secondly, by realizing this through an error minimization process similar to standard curve-fitting; and thirdly, and most importantly, by considering both dependent (say inflation time) and independent variables (canopy construction and drop test data) as parts of the datasets inputs into the PINN.

As a simple illustration of a PINN, consider a neuron receiving a set of “input” numbers, i.e., values of the measured variables labeled as “ X_i ”. In a 3-variable case, a neuronal computation would take the following form with the physical law reduced into a summation:

$$\Sigma = w_1X_1 + w_2X_2 + w_3X_3 \quad (1)$$

Here the neural network “inputs” include the independent variables X_1 and X_2 as well as the also-measured dependent variable represented by X_3 (the latter could be the inflation time). It is here that the so-called AI “weights” appear (“ w_i ”), of which end values are to be obtained from an error-minimizing process. (More on this process in Section B that follows.)

Not all physical laws are linear, obviously, but could be expressed so via the use of inverses, logarithms, etc. For example, in the case of the *thin lens* equation of optics, one would start with

$$\Sigma = w_1 \left(\frac{1}{d_{\text{object}}} \right) + w_2 \left(\frac{1}{d_{\text{image}}} \right) - w_3 \left(\frac{1}{f} \right) \quad (2)$$

with X_1 as the inverse of the distance of the object to the lens ($1/d_{\text{object}}$); X_2 as the inverse distance of the image to the lens ($1/d_{\text{image}}$); and X_3 as the inverse of the lens focal length ($1/f$; a.k.a., the dependent variable). All three X_i variables would be measured in several experiments and form the input datasets feeding the PINN process (one dataset per experiment). As with most curve-fitting schemes, an iterative error-minimization process is used to re-calculate the values of those weights (Sect. III.B). This process is defined to yield weight values at each iteration that progressively lead to the smallest Σ possible (via Eq. (1)), ideally one converging to a zero-value ($\Sigma \rightarrow 0$). When achieved, the converged weights yield the solution $w_3 X_3 = -(w_1 X_1 + w_2 X_2)$, i.e., the sought-after relationship between

the dependent and the independent variables, i.e., $X_3 = -X_1 \left(\frac{w_1}{w_3} \right) - X_2 \left(\frac{w_2}{w_3} \right)$. In the case of the *lens equation* (Eq.

(2)) the PINN-calculated weights are expected to converge to values such that $w_1/w_3 \approx w_2/w_3 \approx 1$. (Note that the weight ratio values may not equal unity in other cases, such as with the parachute-smart PINN.)

B. The Minimization Process

B.1 Which error to minimize?

The PINN process begins with trial values of the weights which are used to yield a first Σ . Such weights are then adjusted/updated in a second iteration, i.e., in a way to yield a smaller value of Σ . This process repeats over several iterations until the “true” Σ is reached (i.e., $\Sigma = 0$). (This is the “learning” stage in AI parlance.) At every iteration, and to yield a smaller Σ , the process (“neuron”) tries to reduce the error between the just-calculated Σ and the sought-after solution ($\Sigma = 0$) by using a so-called *activation* function which in our case is a sigmoid function:

$$\hat{Y}(\Sigma) = \frac{1}{1 + e^{-\Sigma}} \quad (3)$$

(There are many types of activation functions, most of which are designed to plateau in the limit $\Sigma \rightarrow \pm\infty$ [17].) Activations functions are the basic ingredient of the error-minimization scheme determining the “best” updates of the weights. Those are in turn used to construct the *Mean Square Error* (MSE)

$$MSE(\Sigma) = \left(\hat{Y}(0) - \hat{Y}(\Sigma) \right)^2 = \left(\frac{1}{2} - \hat{Y}(\Sigma) \right)^2 \quad (4)$$

In principle, and per Eq. (3), the MSE reaches the sought-after solution as $\hat{Y} \rightarrow 1/2$, which corresponds to $\Sigma \rightarrow 0$. Note that minimizing the error in $\hat{Y}$, instead of minimizing an error proportional to $(0 - \Sigma)^2$ iteratively yields a more stable and convergent sequence. This is particularly crucial where many of the dataset variables X_1 , X_2 and X_3 are very large, as in the parachute cases below.

B.2 Weight update for the next iteration

Updating the weights to minimize the MSE is carried out via partial derivatives in a well-known data optimization procedure [1]:

$$w_i|_{\text{new}} = w_i|_{\text{old}} - \alpha \frac{\partial}{\partial w_i} \text{MSE}(\Sigma) \quad (5)$$

This rule uses partial derivatives derived from Eqs. (1), (3) and (4) via the chain rule. Here parameter “ α ” corresponds to a user-adjustable “learning” parameter. This parameter, along with the very first (or “trial”) set of the weights are user-defined as further explained in section III.D. These “assessment” and “training” steps (in AI language) are to be repeated in the aforementioned iterative scheme until the MSE has converged to a small value.

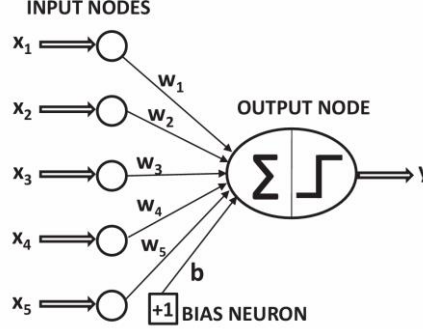


Figure 2. Schematic description of a 5-variable input of a single dataset, feeding into a neuron that calculates a summation function (Eq. (1)) (sigma) and activation function (Eq. (3)) (square-S). Herein the output (“y”) is a set of updated weights. Note that there is no bias neuron in the PINN discussed here. Figure extracted from [1].

Note that the best results are obtained with the use of a large number of *datasets* rather than just one dataset, i.e., collected by repeating an experiment (or drop test or FSI simulation) many times over, say “ N ” times. Such datasets would yield, at the same values of w_1, w_2 and w_3 , a total of N different values of $\hat{Y}^{(k)}(\Sigma^{(k)})$ and $\text{MSE}^{(k)}(\Sigma^{(k)})$ given the defining summation $\Sigma^{(k)} = w_1 X_1^{(k)} + w_2 X_2^{(k)} + w_3 X_3^{(k)}$. In this context, averaging the $\text{MSE}^{(k)}$ ’s would in turn yield what is known as the *cost function* C . The weight adjustment then proceeds by replacing Eq. (5) with

$$w_i|_{\text{new}} = w_i|_{\text{old}} - \alpha \frac{1}{N} \sum_{k=1}^N \frac{\partial}{\partial w_i} \text{MSE}^{(k)}(\Sigma^{(k)}) \equiv w_i|_{\text{old}} - \alpha \frac{\partial}{\partial w_i} C \quad (6)$$

(A full list of the partial derivatives is provided in the Appendix (Sect. A1).) After many iterations that try to minimize the cost C , one is to get values of the weights that would yield the actual algebraic relationship among the variables in Eq. (1) in the $\Sigma \rightarrow 0$ limit. In the *thin lens* equation example above, all three weights would be identical in value but only up to an overall constant. This is not a problem here, as the weights and constant all factor out at $\Sigma \sim 0$.

Experienced AI practitioners may notice the absence of the so-called “bias parameter (b)” [1] which often times enters in the calculation of Σ via $\Sigma = w_1 X_1 + w_2 X_2 + w_3 X_3 + b$. Such parameter was omitted here due to the lack of a well-defined and symmetric well near $\text{MSE}(0)$ in the parachute case (Sect. III.C). Absence of symmetry generally lead to “solutions” that can converge away from the “true” solution at $\Sigma \sim 0$.

C. Parachute-Smart Neural Network

The first step is the definition of a linear summation of the type embodied in Eq. (1), via the use of a suitable physical law. With most physics-based equations being non-linear (think of $F = Ma$ or $U(t) = at + U(0)$), linearization of *some* problems may be only possible after variable inversion, as with the thin lens equation above, or changes of variables involving powers. In our PINN, the first step is to first express inflation time (t_{infl}) in terms of its non-dimensional equivalent (n_{infl}) [18-20]:

$$n_{\text{infl}} = t_{\text{infl}} \frac{U_{\text{init}}}{D_{\text{ref}}} \quad (7)$$

with U_{init} as the parachute system descent speed at the onset of the inflation stage under consideration, and D_{ref} a reference canopy diameter compatible with the reefing constraints at hand [14, 15]. As shown elsewhere [20], this non-dimensional inflation time can be expressed exactly as

$$n_{infl} = \frac{U_{init}}{D_{ref}} \cdot \frac{Vol_{final}}{\langle u_{skirt} S_{skirt} \rangle} \cdot \left(\frac{1}{1 - \phi} \right) \quad (8)$$

with parameters Vol_{final} , $\langle u_{skirt} S_{skirt} \rangle$ and ϕ corresponding to the parachute's volume at the end of the inflation stage (see for example ref. [21]), the time-averaged air flux through the skirt, and a ‘‘mouth-to-vent’’ porosity ratio [20] (an indirect measurement of porosity), respectively. This expression is intuitively true given the inflation time expected to increase with final volume and decrease at greater skirt area. Here, the key dynamical input is the averaged through-skirt flux $\langle u_{skirt} S_{skirt} \rangle$ which is a hard-to-measure variable, but one that can nonetheless be meaningfully approximated for AI use. Equation (8) is the relevant physical law to linearize.

Linearization begins with using an approximated through-skirt flux for canopies of low-enough (total) porosity. With the final through-skirt flux (at the end of inflation) being small in comparison to the initial through-skirt flux, one has:

$$\langle u_{skirt} S_{skirt} \rangle \approx \frac{1}{2} (U_{init} S_{skirt}(t_{init}) + u_{final} S_{skirt}(t_{final})) \approx \frac{U_{init} S_{skirt}(t_{init})}{2} \quad (9)$$

Merging Eqs. (8) and (9) and the rearranging, yields the following result

$$\frac{1}{2} S_{skirt}(t_{init})(1 - \phi) \approx \frac{Vol_{final}}{D_{ref} n_{infl}} \quad (10a)$$

or more strictly, after inserting a ‘‘correction’’ factor G :

$$\frac{1}{2} S_{skirt}(t_{init})(1 - \phi) = G \frac{Vol_{final}}{D_{ref} n_{infl}} \quad (10b)$$

Here the value of G would embody the role of the unsteady flows entering the skirt.

A PINN algorithm is thus set up with defining a Σ -summation as follows:

$$\Sigma = w_1 \frac{1}{2} S_{skirt}(t_{init})(1 - w_2 \phi) - w_3 \frac{Vol_{final}}{D_{ref} n_{infl}} \quad (11)$$

where the weight w_3 takes up the role of ‘‘G’’, and the others, of porosity and other effects. Note that this is an explicitly speed-independent form which obviates the need for inclusion of descent rates values in the datasets (after calculating n_{infl} from Eq. (7)) – a fortunate feature. In AI notation, Eq. (11) yields the following partial-linear form

$$\Sigma = [w_1 X_1 (1 - w_2 X_2)] - w_3 X_3 \quad (12a)$$

i.e., one for which $X_1 = (1/2) S_{skirt}(t_{init})$, $X_2 = \phi$ and $X_3 = Vol_{final} / D_{ref} n_{infl}$. In the cases at hand where total canopy porosity is at 10% or less, the product $w_2 \phi$ is assumed as vanishingly small, thereby yielding

$$\Sigma = w_1 X_1 - w_3 X_3 \quad (12b)$$

Equations (12a) and (12b) make this neuron a *parachute-smart* neuron on account of using Eq. (8) as the input physical law of relevance. With Σ evaluated from (12b) over each dataset, one can calculate Eqs. (3) and (4) for each dataset and average the MSE for the entire set prior to updating the next set of weights via (6) (in order to reduce the cost C). In the $\Sigma \rightarrow 0$ limit, the PINN process would thus arrive at $\mathbf{G} = \mathbf{w}_3 / \mathbf{w}_1$, from which an algebraic relationship

between inflation time and the various canopy dimension arise. Most importantly, it isn't the absolute values of the weight that matters but only their ratio.

D. Values Criteria for the Starting Iteration (“trial” values)

The iterative sequence of the PINN starts with a set of user-defined trial values of the weights to be used in the very first iteration. Obviously, the closest the trial- w_i 's are to the “solution”, the fewer iterations needed. In our case, Eq. (12b), the limit $\Sigma \rightarrow 0$ implies $w_3^{\text{trial}} = w_1^{\text{trial}} \langle X_1 \rangle / \langle X_3 \rangle$ with $\langle X_i \rangle$ as the value of X_i averaged over the entire dataset. This leaves the user with only *one* trial value to manually enter (w_1 here). In the simulations described in the next section, 7-to-13 different values of w_1^{trial} were used (i.e., one per neuron) to yield 7-to-13 converged values of w_1 and w_3 (i.e., as computed by 7-to-13 neurons). Typical w_1^{trial} values were typically set smaller than unity, i.e., anywhere between 10^{-1} down to 10^{-4} , to keep Σ below unity during the iteration process. Those were crucial choices, given the summation Σ being the result of a *difference* between $w_1 X_1$ and $w_3 X_3$ (per Eq. (12b)) and the very large values of X_1 and X_3 that were encountered. Interestingly, changing w_1^{trial} by orders of magnitude also changed all the weight (and Σ) by the same factor throughout the iterations by each neuron. This turned out to be a non-problem as the ratio w_3/w_1 – i.e., the desired result – tended to remain unchanged (in order of magnitude). In the end, the converged weight ratios had the same value regardless of the trial w_1^{trial} values. Less than 10 iterations were needed to reach converged values of the weights.

Another user-input is the value of the learning parameter α . Given that Eq. (6) is only valid as a first order approximation of the error minimization process, the value of α is set as $\alpha = 0.001 \times w_1^{\text{trial}}$, a seeming small iterative step but one large enough if the trial weight are already close to the solution.

IV. Results

This parachute-smart neural network algorithm was tested with the databases collected in two drop test programs conducted by the US Air Force (USAF) with the C-9 personnel recovery parachute [9-13, 19]; and by NASA for its Orion capsule landing parachute cluster system (CPAS program) [14, 15, 22]. Examination of these two cases is illuminating since the latter highlights the inflation duration of each cluster member, as influenced by lead-lag, canopy interference and skirt reefing; and the former, on the testing of low-porosity and un-reefed parachute used singly.

A. USAF C-9 Personnel Recovery Parachute

The USAF C-9 personnel (i.e., “pilot”) recovery parachute is a 28ft-diameter (D_0) flat-circular canopy which has seen use ever since its introduction in the 1950's [19, 23, 24]. Extensive series of instrumented test drops were carried out by Berndt, DeWeese and colleagues in the 1960s to study inflation time and parachute drag at various altitudes (6000, 13,000 and 21,000ft AGL), line stretch speeds (170-400fps) and payload weights (220, 440 and 880lbs) [9-13]. Single parachutes were dropped un-reefed while trailing the forebody sketched in figure 3 [11]. The data used in Table A.1 below was published by Heinrich in Ref. [12], thereby generating between 20 and 32 datasets depending on payload weight. Each payload weight was processed separately by the PINN, given the inflation performance dependence on mass ratio. (See for example Ref. [15].) The corresponding non-dimensional time is plotted in figure 4.

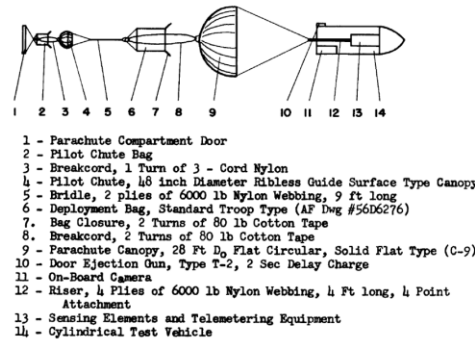


Figure 3. Setup used in the USAF drop test program described in references [9-13].

Figure extracted from [11].

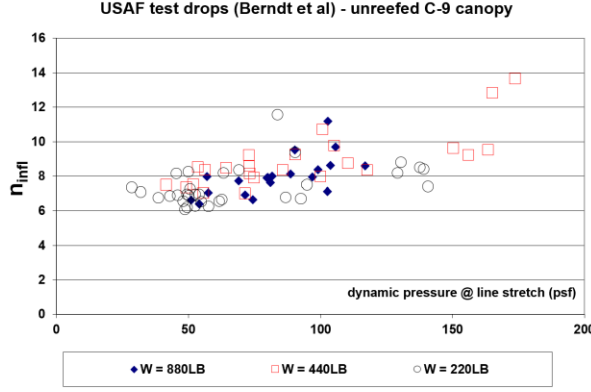


Figure 4. Non-dimensional inflation time versus dynamic pressure at line-stretch [12].
Figure created by the authors.

Per Eqs. (11) and (12b), the datasets included the non-dimensional inflation times (n_{infl}), the skirt area at the beginning of the inflation process ($S_{skirt}(t_{init})$), and final inflated volume (VOL_{final}). These three variables were reconstructed from the available data [12] (Table A.1) as none were directly measured at the time of the tests. More specifically, the values of n_{infl} were calculated from the measured (dimensional) inflation time (t_{infl}), nominal diameter (D_0) [19] and system's fall rate at line stretch (U_{init}). Lacking measurements of the inflated volume, on the other hand, lead to the estimate of this variable from the assumed hemispherical shape of this low-porosity canopy as $VOL_{final} = (4/48) (0.7 D_0)^3$ - a reasonable assumption. (The $0.7 D_0$ factor arises from the hemisphere featuring a radius equal to the projected diameter [19].) On the other hand, the initial skirt diameter $S_{skirt}(t_{init})$ came from a data analysis by Berndt [11] which suggested $S_{skirt}(t_{init}) = 0.0024 S_0$ (with $S_0 = \pi D_0^2/4$) [19].

Given the trial values discussed in Sect. III.D, typical PINN iterative marches converged in 10 iterations or less for each of the neuron using differing trial weight w_1^{trial} (in the range of 10^{-1} down to 10^{-4}). As many as 10 neurons were used in our single-hidden layer PINN. The results are graphed in Figs. 5.

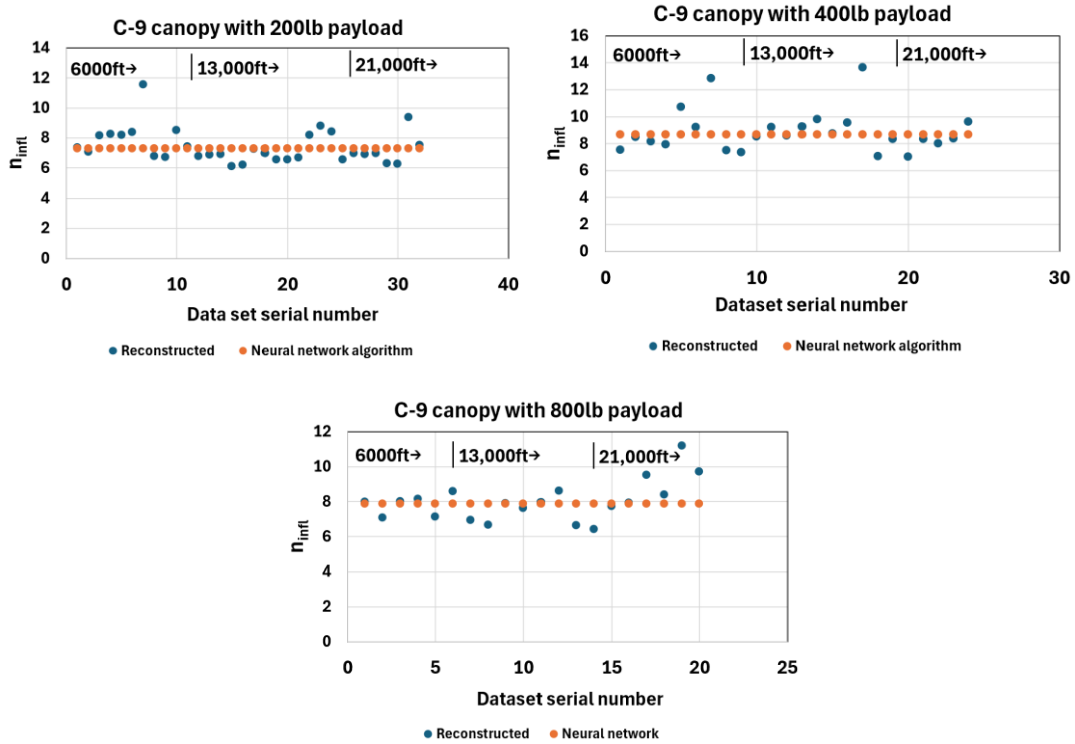


Figure 5. Non-dimensional inflation time reconstructed for each drop test of a USAF parachute drop program [9-13] . The reconstructed values (blue circles) were calculated with Eq. (8) using the experimentally-measured values. The PINN values (orange circles) were obtained from Eq. (10b) or (12b) while using the weights calculated at the last iteration.

The PINN results show a high-level of consistency due to the same $S_{skirt}(t_{init})$ and VOL_{final} used in each dataset (unlike the inflation time). The neural network data placed its prediction through the “gravitational center” of the reconstructed data. The value of the corresponding correction factor “ G ” values (Eq. (10a)) are shown in Table 1. The inferred through-skirt flux $\langle u_{skirt} S_{skirt} \rangle$ were also calculated, via Eqs. (9) and (11), while using again the converged values of the weights. The values also appear in Table 1.

Table 1. PINN-produced algebraic relationships for inflation time and through-skirt flux in the case of unreefed C-9 canopy. Factor G is defined in Eq. (10b).

System total nominal weight (lbs.). Actual weight may differ by a dozen lbs. or so	Non-dimensional inflation time formula derived from PINN.	Time-averaged through-skirt flux (ft ³ /s)
220	$n_{infl} = G \frac{2Vol_{final}}{D_{ref} S_{skirt}(t_{init})} \approx 0.076 \frac{2Vol_{final}}{D_{ref} S_{skirt}(t_{init})}$	$\langle u_{skirt} S_{skirt} \rangle \square 6.57 S_{skirt}(t_{init}) U(t_{init})$
440	$n_{infl} \approx 0.091 \frac{2Vol_{final}}{D_{ref} S_{skirt}(t_{init})}$	$\langle u_{skirt} S_{skirt} \rangle \square 5.52 S_{skirt}(t_{init}) U(t_{init})$
880	$n_{infl} \approx 0.082 \frac{2Vol_{final}}{D_{ref} S_{skirt}(t_{init})}$	$\langle u_{skirt} S_{skirt} \rangle \square 6.08 S_{skirt}(t_{init}) U(t_{init})$

B. NASA CPAS Main Parachute Cluster

NASA’s Orion spacecraft parachute recovery system [22] employs eleven canopies of various sizes and porosities, opening under different reefing conditions over four orders of magnitudes in mass ratios [15]. Of particular interest here is the inflation performance of the three clustered 116 ft-nominal diameter (D_0) ringsail canopies, which open through (timed) inflation stages, namely: inflation under a reefed constraint (the so-called first reefed stage), followed by disreefing and inflation from the 1st state to another reefed state (the so-called 2nd stage reefing), itself followed by disreefing and inflation from the 2nd state to each canopy’s final “full open” configuration. Herein the focus is on the inflation duration of the last two stages, given their well-defined initial skirt areas and partial internal pressurization. Omission of the 1st inflation stage followed from the current lack of empirical determination of the initial skirt area (Section V.B).

These data were collected during 6 drops of the EDU program (“CDT”), 8 drops of the Qualification Test program (“CQT”) and 1 “drop” during the first Orion unmanned flight (“EFT-1”). Counting the data collected with each cluster member (2 or 3 parachutes, depending on the test) lead to 16 (CDT), 23 (CQT) and 3 (EFT-1) datasets, for a grand total of 42 datasets. The reconstructed inflation times for each cluster member were published in [15]. On the other hand, the final inflated volume in each stage were inferred from the data obtained by Ray [21]; and the skirt area calculated from the known lengths of the two reefing lines used in each test and corrected for canopy interference (typically abnormally long inflation times were decreased by 30%). The results are shown in figures 6 and 7. Table 2, on the other hand, displays the functional relationships that arise after using Eqs. (9), (10b) and (11).

In general, the PINN calculations, based on the weights of the last PINN iteration track the test-to-test variations data observed in most of the 1st-to-2nd and 2nd-to-full open disreef cases. Interestingly, the test-to-test variations shown in the figures are more pronounced in the (reconstructed) CDT and EFT-1 data, versus the CQT data, due to (up to) 20% changes in reefing line lengths adjustments from one test to another. (No such adjustments were made during the qualification tests.) On the other hand, and because of deeper internal pressurization in greater portions of the inflating volume, the 2nd-to-full open disreef data exhibited far less test-to-test scatter than with the 1st-to-2nd disreef data. Finally, comparison of the left and right frames in both figures reveals the sensitivity of the PINN analysis on dataset

contents, where small “movement” of the neuron calculations can be seen between the smaller dataset (right frame) and larger dataset (left frame).

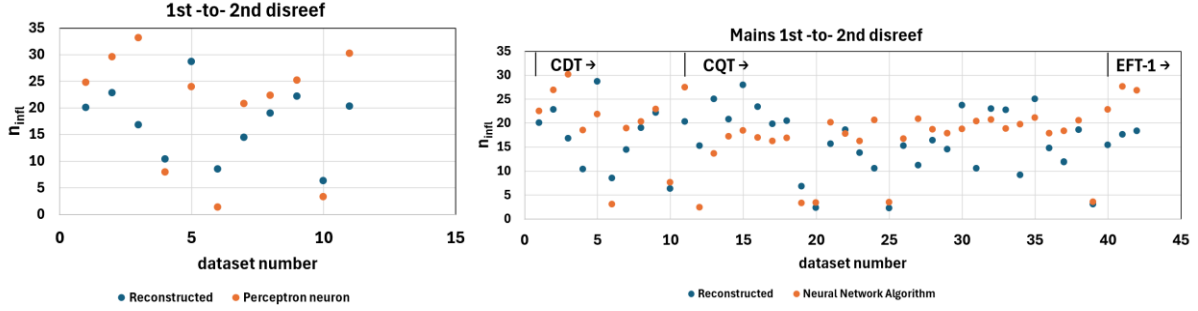


Figure 6. Non-dimensional inflation time reconstructed for each drop test of NASA’s CPAS program (listed in [15]). The reconstructed values (blue circles) were calculated with Eq. (8) using the experimentally-measured values. The PINN values (orange circles) were obtained from Eq. (10b) or (12b) while using the weights calculated at the last iteration. The left frame shows the results of the CDT datasets only, and the right frame, those based on the use of all 43-dataset obtained by combining all three CPAS test drop programs.

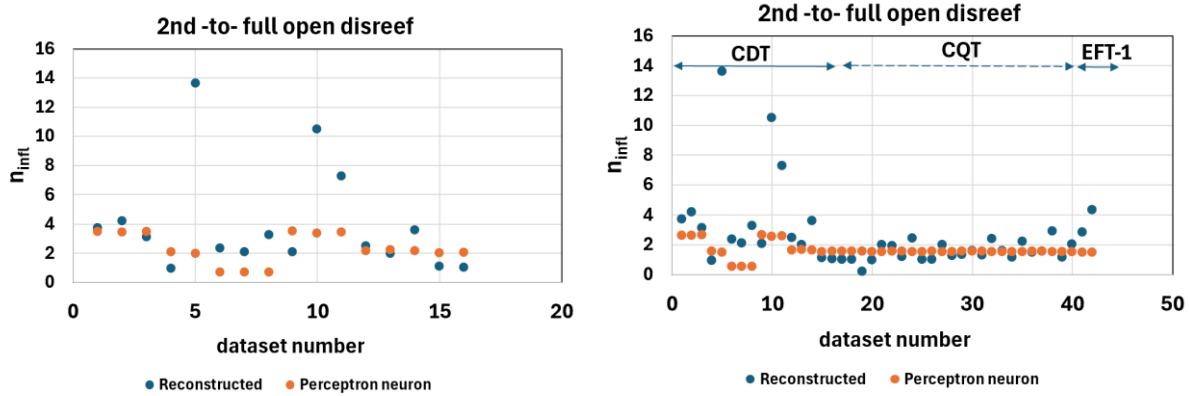


Figure 7. Non-dimensional inflation time reconstructed for each drop test of NASA’s CPAS program (listed in [15]). The reconstructed values (blue circles) were calculated with Eq. (8) using the experimentally-measured values. The PINN values (orange circles) were obtained from Eq. (10b) or (12b) while using the weights calculated at the last iteration. The left frame shows the results of the CDT datasets only, and the right frame, those based on the use of all 43-dataset obtained by combining all three CPAS test drop programs.

Table 2. PINN-produced algebraic relationships for inflation time and through-skirt flux in the case of un-reefed C-9 canopy. Factor G is defined in Eq. (10b).

Category	Non-dimensional inflation time formula derived from PINN	Time-averaged through-skirt flux (ft ³ /s)
1st -to- 2nd disreef CDT, CQT and EFT-1 data	$n_{infl} \propto G \frac{2Vol_{final}}{D_{ref} S_{skirt}(t_{init})} \approx 0.182 \frac{2Vol_{final}}{D_{ref} S_{skirt}(t_{init})}$	$\langle u_{skirt} S_{skirt} \rangle \propto 0.091 S_{skirt}(t_{init}) U(t_{init})$

1st -to- 2nd disreef CDT data only	$n_{\text{infl}} \approx 0.081 \frac{2\text{Vol}_{\text{final}}}{D_{\text{ref}} S_{\text{skirt}}(t_{\text{init}})}$	$\langle u_{\text{skirt}} S_{\text{skirt}} \rangle \approx 0.13 S_{\text{skirt}}(t_{\text{init}}) U(t_{\text{init}})$
2nd -to- full-open disreef CDT, CQT and EFT-1 data	$n_{\text{infl}} \approx 0.182 \frac{2\text{Vol}_{\text{final}}}{D_{\text{ref}} S_{\text{skirt}}(t_{\text{init}})}$	$\langle u_{\text{skirt}} S_{\text{skirt}} \rangle \approx 0.091 S_{\text{skirt}}(t_{\text{init}}) U(t_{\text{init}})$
2nd -to- full-open disreef CDT data only	$n_{\text{infl}} \approx 0.25 \frac{2\text{Vol}_{\text{final}}}{D_{\text{ref}} S_{\text{skirt}}(t_{\text{init}})}$	$\langle u_{\text{skirt}} S_{\text{skirt}} \rangle \approx 0.041 S_{\text{skirt}}(t_{\text{init}}) U(t_{\text{init}})$

V. Discussion

A. Extrapolating the result to other parachute designs

The results of Tables 1-2 clearly improve our ability to predict inflation duration with regards to scaling-up or scaling-down similar canopy designs (but within limits [20]). More interestingly, it can also allow prediction following skirt reefing changes of the same designs. However, one may wonder whether those results can be used to characterize the inflation times of *any* other low-porosity canopy or ringsail clusters. A quick but unwanted answer may be that *it depends*. Obviously, organizing data and PINN-use depending on overall design, e.g., solid vs. porous canopies, flat circular vs. ringsail or even reefing type, etc., as done here, should be a primary consideration given that the inflation dynamics embedded in each category is expected to be different.

The datasets in the USAF C-9 study (Table 1) did not involve clustered canopies and are thus silent on canopy interactions within clusters. Thus, applying those results to other un-reefed and un-clustered low-porosity hemispherical or (perhaps) conical designs may turn out correct at similar initial skirt area ($\sim 0.0024 S_0$) - at the very least. Application to reefed such canopies might work too (as suggested by the CPAS case). On the other hand, *clustered* canopies involve more dynamics and unsurprisingly the inflation time weight ratios w_3/w_1 (or “ G ” in Eq. 10b) turn out twice as large as the un-clustered case. Canopy interactions within clusters are clearly important factors and as such are embedded in the datasets.

Differences between the through-skirt flux equations between the C-9 and CPAS cases are also illuminating. Comparison of the results shown in Tables 1 and 2 highlights the role of the sails (in the CPAS case) functioning as air scoops during the inflation process. This is a feature that has long been suspected by designers. Here one gets a first quantitative exposition of this fact.

Equation (12a) provides the key to studying canopies with non-zero (total) porosity via the use of weight w_2 as with ringslot canopies. For this PINN to work effectively, one needs explicit measurements of the porosity showing up in the datasets – hence larger datasets. When made at the loft, it is hoped that the product $w_2\phi$ would reveal dynamic porosity effects during inflation.

B. Pitfalls

As with any AI-based studies, dataset quality in data point accuracy *and case coverage* are the key to success. In the parachute cases considered here, the quality of the measurement or reconstructions of the initial skirt area and final (inflated) canopy volume is also a determinant factor. In the USAF C-9 case where inflation begins soon after canopy extraction out of a deployment bag (Fig. 3), having no direct access to photographs/movies of each drop forced us to resort to the investigator’s own analysis of the initial skirt area [11]. Given the randomness associated with skirt area shaping after bag-strip (in all un-reefed canopies at least) such data will be accessible only after explicit photogrammetry analysis – preferably via AI-based image analysis. Such limitation is at the root of considering the CPAS inflation stages that (each) began in a reefed stage where the canopy was at least partially pressurized.

Another pitfall is the determination/reconstruction of the final volume in porous canopies. Not so much with the C-9 case in which the final configuration indeed resembles a hemisphere, but certainly with the CPAS ringsail cases for which previously-published values [21] were used.

C. On with the study of inflation drag

Using PINN analysis in the study of inflation drag is also possible since time-averaged drag becomes immediately knowable upon determination of n_{infl} at same total mass and steady descent speed. Indeed and from the Momentum-Impulse Theorem and for a parachute system inflating while moving vertically, one has [14, 15, 18]

$$m_{total}U_{final} - m_{total}U_{init} = \int_{init}^{final} (-F_D(t) + m_{total}g)dt \quad . \quad (13)$$

or, in terms of the time-averaged drag $\langle F_D \rangle$,

$$\langle F_D \rangle = m_{total}g + \frac{m_{total}}{n_{infl}} \frac{U_{init}}{D_{ref}} (U_{init} - U_{final}) \quad . \quad (14)$$

with the known (and fixed) m_{total} , g and U_{final} as the payload-parachute mass, constant of gravity and system's descent speed at the end of inflation respectively (where $U_{final} \sim U_{descent}$).

Peak drag (F_{peak}), on the other hand requires a separate PINN analysis as more weights are required. Given that $\langle F_D \rangle = F_{peak}I$, where I is the so-called “drag integral”:

$$I = \int_i^f \frac{|F_D(t)|dt}{F_D(t_f - t_i)} \quad (15)$$

equation (14) leads to (exactly again)

$$F_{peak}I = m_{total}g + \frac{m_{total}}{n_{infl}} \frac{U_{init}^2}{D_{ref}} \left(1 - \frac{U_{final}}{U_{init}} \right) \quad . \quad (16)$$

Typical values of I – a dynamical variable - vary according to porosity and reefing [14, 15, 18] but are generally known at about 0.4-0.7 in many low-porosity hemispherical parachutes. Approximating I as $I \sim \frac{1}{2} w_3$ may incorporate the complicated and embedded dynamics. Therefore, and after including more weights in defining PINN's summation equation, one gets

$$\Sigma = w_3 m_{total}g + w_1 \frac{m_{total}}{n_{infl}} \frac{U_{init}^2}{D_{ref}} \left(1 - \frac{U_{final}}{U_{init}} \right) - w_2 F_{peak} \frac{1}{2} \quad (17)$$

The sought-after solution (where $\Sigma \rightarrow 0$) would lead to a peak drag written in terms of weight ratios

$$F_{peak} = 2 \left(\frac{w_3}{w_2} \right) m_{total}g + 2 \left(\frac{w_1}{w_2} \right) \frac{m_{total}}{n_{infl}} \frac{U_{init}^2}{D_{ref}} \left(1 - \frac{U_{final}}{U_{init}} \right) \quad (18)$$

As with inflation time, the accuracy and PINN iterative convergence will rest with a most suitable set of trial weights. For this case it is suggested that

$$\left(2 \frac{w_3^{trial}}{w_2^{trial}} \right) = 1 \quad (19)$$

given that the first term (system's weight term) in Eq. (18) is a lower bound. Inserting Eq. (19) into (18) then yields another trial weight constraint

$$1 - \frac{m_{total}g}{F_{peak}} = 2 \left(\frac{w_1^{trial}}{w_2^{trial}} \right) \frac{m_{total}}{n_{infl}} \frac{U_{init}^2}{D_{ref}} \left(1 - \frac{U_{final}}{U_{init}} \right) \quad (20)$$

Solving Eqs. (19) and (20) yields values of w_1^{trial} and w_2^{trial} expressed in terms of w_3^{trial} , meaning that the trial values for two weights are calculated upon choosing the value of a third, user-input. Note also that the calculation of the weight w_2 will inform the user of the integral of Eq. (15) – usually, an un-measurable metric without load cells. Note finally that the peak drag PINN process involves the updating of three weights rather than just two as with inflation time. Further research will show whether the data sets used here will be large enough.

In closing, getting the most likely time evolution of parachute drag will require the implementation of Newtonian differential equations rather than simpler (but exact) algebraic equations as done here (Eqs (10a) and (16)). This, in turns, would demand the use of load cell data as part of the datasets, an inclusion which would end up into a PINN process involving a very large number of weights (in the thousands, e.g., for each time-slice of the integral in Eq. (13)). This isn't an impediment, as shown in ref. [6] for example, given the computer power and memory at hand nowadays. Moreover, and given such large number, deep learning AI techniques will be used instead [1].

VI. Conclusion

This paper has described a novel “parachute-smart” neural network algorithm that finds hard-to-get algebraic relationships tying performance and canopy design data (experimental or computational). Using parachute physics-informed neural networks rather than generic physics-informed algorithms, such as those based on differential equations, has allowed the use of smaller datasets i.e., in the 10^1 -to- 10^2 data point range rather than in the 10^3 -to- 10^6 range. Obviously, the number of weights required by the neural network algorithm at hand will be the ultimate determinant of required datasets size. But exploiting simpler yet exact, and metric-specific physical laws should provide outputs that may be relevant - and accurate-enough - to designers and other parachute technologists. Like CFD/FSI, PINN analysis represents a new class of models/tools which should extend the reach of parachute engineers rather than replacing them.

References

- [1] See for example: Aggarwal, C.C., 2018. “An introduction to neural networks”. In *Neural networks and deep learning: a textbook* (pp. 1-52). Cham: Springer International Publishing. <https://www.charuaggarwal.net/AllSlides.pdf>.
- [2] Narayanan, A. and Kapoor, S., 2025. “AI snake oil: What artificial intelligence can do, what it can't, and how to tell the difference”. Princeton University Press, Princeton, NJ, 2024.
- [3] Hornik, Kurt; Tinchcombe, Maxwell; White, Halbert (1989). "Multilayer feedforward networks are universal approximators". *Neural Networks*. **2** (5): 359–366. doi:10.1016/0893-6080(89)90020-8. ISSN 0893-6080. S2CID 2757547.
- [4] See for example: Raissi, Maziar; Perdikaris, Paris; Karniadakis, George Em (2017). "Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations". arXiv:1711.10561 [cs.AI].
- [5] See for example: He, Q., Barajas-Solano, D., Tartakovsky, G. and Tartakovsky, A.M., 2020. “Physics-informed neural networks for multiphysics data assimilation with application to subsurface transport”. *Advances in Water Resources*, 141, p.103610.
- [6] Yang, Yulong; O'Farrell, Clara; Allen-Blanchette, Christine. “Generating Physically Plausible Parachute Dynamics with Deep Generative Modeling”. AIAA-2026-XXXX. Presented at the 28th AIAA Aerodynamic Decelerator Systems Technology Conference and Seminar, London, UK, June 1-5, 2026.
- [7] Perin, B., Birolleau, A., Hangouet, S., Ginestet, J.B., Patoux, A. and Pastor, P. “Contribution of an Artificial Intelligence approach to the Guidance/Piloting of PADS”. AIAA-2022-2718. Presented at the 26th AIAA Aerodynamic Decelerator Systems Technology Conference, Toulouse, France, 2022.
- [8] Gao, Z., Jiang, Z., Qi, X. and Yang, P., 2025, July. “Integrated identification of parachute aerodynamic parameters and wind field based on artificial neural network”. Presented at the 2025 4th International Symposium on Aerospace Engineering and Systems (ISAES) (pp. 330-338). IEEE.
- [9] Berndt, R. and DeWeese, J., “The opening force of solid cloth, personnel type parachutes”. Paper AIAA-70-1167. Presented at the 3rd aerodynamic deceleration systems conference, Dayton, OH, 1970.
- [10] Berndt R. and Deweese J. “Filling time prediction approach for solid cloth type parachute canopies”. Paper AIAA-1966-1503, presented at the Aerodynamic Deceleration Systems Conference, 1966
- [11] Berndt, R., “Experimental Determination of Parameters for the Calculation of Parachute Filling Times”. Paper presented at the Annual Meeting of the WGLR-DGRR, Berlin, Sept. 1964.
- [12] H. Heinrich Aeronautical J., December 1972, pp. 723-731 (Supplementary paper)
- [13] Heinrich et al. A Parachute Opening Shock Theory Based on Non-Linear Time Histories of Inflow Velocity and Projected Area. June 1974. Natick Report, AD/A-005 271.
- [14] Potvin, J., & Ray, E. (2017). “Extrapolating the Trends of Test Drop Data with Opening Shock Factor Calculations: The Case of the Orion Main and Drogue Parachutes Inflating to 1st Reefed Stage”. Paper AIAA-2017-3228. Presented at the 24th AIAA Aerodynamic Decelerator Systems Technology Conference, 5-9 June 2017, Denver, CO.

- [15] Potvin, J., and Ray, E.; "Opening Shock Factor Calculation for Orion Main Parachute Disreefing". Paper AIAA-2019-3486. Presented at the 25th AIAA Aerodynamic Decelerator Systems Technology Conference and Seminar, Dallas, TX, June 14 - 21, 2019.
- [16] Bevington, P.R. and Robinson, D.K., 2003. Data reduction and error analysis for the physical sciences". 3rd Edition. New York: McGraw-Hill.
- [17] Dubey, S.R., Singh, S.K. and Chaudhuri, B.B., 2022. Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*, 503, pp.92-108.
- [18] Potvin, J. and Peek, G., 2007, May. Testing Without Load Cells-Can Opening Shock Be Estimated From Video Data Only? Paper AIAA-2007-2551. In 19th AIAA Aerodynamic Decelerator Systems Technology Conference and Seminar.
- [19] T. W. Knacke, Parachute Recovery Systems Design Manual; Para Publishing (Santa Barbara, CA 1992).
- [20] Potvin, J., 2023. "Simple rules for parachute inflation testing at underscale and performance extrapolation to full-scale". *Journal of Aircraft*, 60(4), pp.1094-1104.
- [21] Ray, E., 2017. Photographic volume estimation of CPAS main parachutes. Paper AIAA-2017-3229. In 24th AIAA Aerodynamic Decelerator Systems Technology Conference.
- [22] Lichodziejewski, D., Taylor, A., Sinclair, R., Olmstead, R., Kelley, C., Johnson, J., Bledsoe, K. and others (2009, January). Development and Testing of the Orion CEV Parachute Assembly System (CPAS). AIAA-2009-2938. Presented at the 20th AIAA Aerodynamic Decelerator Systems Technology Conference and Seminar.
- [23] Kididis, A.S., *The Art of Drag: A History of US Air Force Parachute Technology Development*. Published by the Parachute Industry Association, 2008.
- [24] Poynter, D. *The parachute manual: a technical treatise on aerodynamic decelerators*. Volume 1. Para Publishing, Santa Barbara, 1984.

Appendix

A.1. Derivatives list for the Cost function

As described in Section III.B, the update of the weights is realized via Eq. (6), reproduced below as Eq. A.1. This is an expression that involves partial derivatives which can be derived from Eqs. (1), (3) and (4) via the chain rule:

$$w_i|_{\text{new}} = w_i|_{\text{old}} - \alpha \frac{1}{N} \sum_{k=1}^N \frac{\partial}{\partial w_i} \text{MSE}^{(k)}(\Sigma^{(k)}) \equiv w_i|_{\text{old}} - \alpha \frac{\partial}{\partial w_i} C \quad (\text{A.1})$$

$$\frac{\partial C}{\partial w_1} = \frac{(-2)}{N} \sum_{k=1}^N \left[\left(\frac{1}{2} - \hat{Y}^{(k)} \right) (1 - \hat{Y}^{(k)}) \hat{Y}^{(k)} X_1^{(k)} (1 - w_2 X_2^{(k)}) \right] \quad (\text{A.2})$$

$$\frac{\partial C}{\partial w_2} = \frac{(-2)}{N} \sum_{k=1}^N \left[\left(\frac{1}{2} - \hat{Y}^{(k)} \right) (1 - \hat{Y}^{(k)}) \hat{Y}^{(k)} (-1) (w_1 X_1^{(k)} X_2^{(k)}) \right] \quad (\text{A.3})$$

$$\frac{\partial C}{\partial w_3} = \frac{(-2)}{N} \sum_{k=1}^N \left[\left(\frac{1}{2} - \hat{Y}^{(k)} \right) (1 - \hat{Y}^{(k)}) \hat{Y}^{(k)} (-1) (X_3^{(k)}) \right] \quad (\text{A.4})$$

A.2. Data Tables - USAF C-9 Inflation times [12]

Total weight 220lbs.

Rho (sl/ft ³)	Vs (ft/s)	opening time (s)	ninfl
0.00198	170	1.21	7.346429
0.00198	180	1.1	7.071429
0.00198	214.5	1.063	8.143339
0.00198	225	1.025	8.236607
0.00198	253	0.905	8.177321
0.00198	264.5	0.885	8.360089
0.00198	291	1.111	11.54646
0.00198	296.5	0.64	6.777143
0.00198	306	0.613	6.699214
0.00198	373	0.638	8.499071
0.00198	377	0.55	7.405357
0.0016	220	0.86	6.757143
0.0016	232	0.828	6.860571
0.0016	240	0.805	6.9
0.0016	247	0.69	6.086786
0.0016	249	0.698	6.207214
0.0016	252	0.805	7.245
0.0016	260	0.75	6.964286
0.0016	262	0.698	6.531286
0.0016	278	0.66	6.552857
0.0016	280	0.666	6.66
0.0016	402	0.57	8.183571
0.0016	404	0.61	8.801429
0.0016	417	0.564	8.399571
0.00121	282	0.65	6.546429
0.00121	286.5	0.68	6.957857
0.00121	288	0.67	6.891429
0.00121	295	0.66	6.953571
0.00121	295	0.597	6.289821
0.00121	309	0.568	6.268286
0.00121	387	0.678	9.370929
0.00121	396	0.53	7.495714

Total weight 440lbs.

Rho (sl/ft ³)	Vs (ft/s)	opening time (s)	n_{infl}
0.00198	229	0.92	7.524286
0.00198	255	0.93	8.469643
0.00198	272	0.839	8.150286
0.00198	275	0.805	7.90625
0.00198	319	0.94	10.70929
0.00198	397	0.65	9.216071
0.00198	408.5	0.88	12.83857
0.0016	228	0.92	7.491429
0.0016	248	0.83	7.351429
0.0016	259	0.92	8.51
0.0016	302	0.855	9.221786
0.0016	302.5	0.793	8.567232
0.0016	336.5	0.77	9.25375
0.0016	362.5	0.755	9.774554
0.0016	371.5	0.659	8.743518
0.0016	452	0.59	9.524286
0.0016	466	0.82	13.64714
0.00121	303.5	0.648	7.023857
0.00121	305.5	0.764	8.335786
0.00121	344	0.57	7.002857
0.00121	376.5	0.62	8.336786
0.00121	406.5	0.55	7.984821
0.00121	441	0.53	8.3475
0.00121	498.5	0.54	9.613929

Total weight 880lbs.

Rho (sl/ft ³)	Vs (ft/s)	opening time (s)	n_{infl}
0.00198	240	0.93	7.9714286
0.00198	241	0.82	7.0578571
0.00198	287.5	0.78	8.0089286
0.00198	299.5	0.76	8.1292857
0.00198	322	0.62	7.13
0.00198	343.5	0.7	8.5875
0.0016	299	0.65	6.9410714
0.0016	305	0.61	6.6446429
0.0016	316	0.7	7.9
0.0016	318.5	0.67	7.62125
0.0016	348	0.64	7.9542857
0.0016	360	0.67	8.6142857
0.00121	290.5	0.64	6.64
0.00121	299	0.6	6.4071429
0.00121	338	0.64	7.7257143
0.00121	364	0.61	7.93
0.00121	386.5	0.69	9.5244643
0.00121	404.5	0.58	8.3789286
0.00121	412	0.76	11.182857
0.00121	417.8	0.65	9.6989286