A Smart Landing Platform With Data-Driven Analytic Procedures for UAV Preflight Safety Diagnosis

ZHENYU ZHOU AND YANCHAO LIU
Department of Industrial and Systems Engineering, Wayne State University, Detroit, MI 48201, USA
Corresponding author: Yanchao Liu (yanchaoliu@wayne.edu)

This work was supported by the National Science Foundation under Grant CMMI-1944068.

ABSTRACT Due to the limitation imposed by hardware and form factor considerations, multirotor unmanned aircraft (drones) are unable to conduct preflight physical checks on their own capacity. Critical safety checks involve detecting various anomalies such as imbalanced payload, damaged propellers, malfunctioning motors and poorly calibrated compass, etc. Human efforts are currently required for performing such tasks, which impedes large-scale deployments of drones and increases the operational costs. In this work, we propose a weight-measuring landing platform along with a set of statistical inference algorithms aimed at performing safety checks for any multicopter aircraft that lands on the platform. We develop a nonconvex nonlinear least squares model for estimating the center of gravity and orientation of the aircraft, and derive a recursive formula for calculating the optimal solution. In numeric experiments, our analytical solution method has been able to find the global solution orders-of-magnitude faster than a global optimization solver. We have conducted real-system tests on a quadcopter drone deliberately configured to carry misplaced payload, and to use damaged propellers. Experiment results show that the platform is able to detect and profile these common safety issues with high accuracy.

INDEX TERMS Anomaly detection, nonlinear least squares, statistical inference, unmanned aircraft.

I. INTRODUCTION
Unmanned aircraft (UA), or drones, have been envisioned by many to transform the way people live and work. Large-scale deployment of autonomous drones for tasks such as package delivery [1], security patrol and traffic monitoring [2], [3], etc., is believed to contribute an integral part of the smart city infrastructure.

Among the plethora of air frame types, the multirotor and vertical takeoff and landing (VTOL) designs account for the vast majority of small UA that exist today. Multirotor aircraft are also called multicopters. Compared to fixed-wing aircraft, multicopters are preferred for low-altitude applications, because they are nimbler, easier to control and more permissible for airway limitations.

Safety is the utmost concern in aviation, be it for manned and unmanned operations. Due to hardware limitation, small multicopters are unable to conduct physical check-ups autonomously using onboard sensors. Human operators’ visual examination, sometimes aided by tools, are currently required to identify obvious problems such as propeller damages, motor malfunction and payload misplacement. The reliance on human efforts for preflight checks not only increases the operational cost but also limits where drones can land. For instance, building rooftops that are widely envisioned to provide temporary landing space for drones might not be suitable to station a human operator, and hence could not be used for drone landing.

To close this gap, this paper presents a novel platform along with a set of statistical inference algorithms to perform automated, unmanned pre-flight checks for multicopter drones. The proposed method is able to answer the following questions.

- Is the drone overloaded with heavy payload?
- Is the center of gravity aligned with the geometric center of the air frame? Weight imbalance will cause certain motors spin harder than others and lead to motor failure due to overheat.
- Are all motors able to spin as directed by the flight controller? Malfunctioning of the electronic speed
controller (ESC) may render the motor unresponsive to flight control signals.

- Are all propellers intact and able to generate the expected thrust? Propellers may have been damaged due to impact with foreign objects, such as birds and twigs, during prior operation.
- Does the compass need re-calibration or replacement?

To realize these functions, we design a digital scale equipped with multiple load measuring tips (load cells) beneath a flat and rigid platform surface with grid holes. The platform is able to measure the weight and weight distribution of any drone that lands on it. When a motor spins at a given output level, the propeller, if in good condition, is expected to generate a calculable amount of upward lift, which would lead to a reduction and a redistribution of the drone’s weight as measured by the platform. By analyzing the instantaneous weight measurements within a statistical inference framework, anomalies (if exist) in weight distribution and in the actuation and propelling system can be detected and properly attributed. A prototype platform is demonstrated in Figure 1.

![FIGURE 1. A prototype platform (with a multicopter on top) constructed for experimentation and method validation. Four digital load cells are placed beneath the platform, which provide accurate measurements of the weight and weight distribution.](image1)

We propose a nonlinear least squares model with air frame geometry constraints to analyze the measurement data and draw accurate inferences. Since the load cells are able to take instantaneous measurements quickly, a large amount of samples may be recorded in a short diagnosis period. As the sample size increases, it becomes increasingly difficult for a numeric solver to find the global optimal solution (i.e., the best fit) within a practical diagnosis time frame. To overcome this difficulty, we derive the closed-form analytical solution to the nonlinear optimization problem and prove the uniqueness of the solution in the operating range. As a result, the computing time is reduced from many minutes to a few seconds. The effectiveness and practical value of the proposed platform and algorithms are validated via an extensive set of experiments.

The proposed apparatus and the diagnostic model can be integrated with other innovations aimed at increasing the level of automation in drone-enabled systems, such as the remote charging technology, mobile launch pads (or containers), and precision landing gears, etc. The intended use case is illustrated in Figure 2. The present work also provides a hardware platform for implementing our previous research on the optimized drone fleet landing problem [4] and the drone landing depot location problem [5].

![FIGURE 2. Envisioned integration of the smart landing platforms in large-scale UAV deployments.](image2)

The remainder of the paper is organized as follows. In Section II, we review the related literature on aircraft fault detection and load balancing problems. In Section III, the main models and solutions are derived and the test procedure is outlined. In Section IV, we conduct extensive simulated and real-world experiments to validate the effectiveness of the proposed approach and analyze the solution properties under different realistic settings. Section V concludes the paper with pointers for future work.

II. RELATED LITERATURE

Many passive fault detection methods are based on detecting abnormal vibration emitted when the target component (e.g., the electric motor) is set to the working state. For instance, Iannace et al. [6] applied paper strips on a blade to simulate unbalanced propeller that would generate noise in rotation, and they used a neural network to characterize and analyze the noise and detect propeller fault. However, as mentioned in that paper, the test of the model must be performed indoors; this indicates that the acoustic analysis method might not be directly applicable in real operating environments, which are primarily outdoors. Ghalamchi and Mueller [7] employed spectral analysis on data collected by the drone’s built-in accelerometer to locate the problematic propeller during flight. In experiments, they made a damaged propeller by cutting off a small end of a propeller which would result in vibration in rotation. We adopt a similar experimental method to create broken propellers in our
validation tests. Furthermore, as demonstrated via experiments, our method is sensitive enough to accurately portray the severity of the propeller damages. Saied et al. [8] introduced a real-time error detection, fault isolation and system recovery algorithm. They monitored the rotation speed and current draw of each motor on an octocopter, and they used the support vector machine method to detect motor failure and propeller loss. Baskaya et al. [9] simulated a model to generate data for a small fixed-wing UAV and used support vector machines (SVM) to classify the faulty and nominal flight conditions. The result showed that their method could give accurate classifications for the loss of effectiveness faults on the control surfaces. Wang et al. [10] introduced a fault detection model acceleration engine to detect UAV faults. They integrated the principal component analysis method and the long short-term memory method, and deployed the algorithm on an onboard computer. The computer could reportedly collect the roll rate data in real time from a small fixed-wing UAV and perform fault detection at an accuracy of 0.986. Chelly et al. [11] presented a quadcopter gyroscope fault-tolerant control based on Thau observer and flatness theory. The experiment results showed that the observer could detect and isolate the fault on the gyroscope sensor. They used a fault-tolerant control strategy to correct the detected fault. Liang et al. [12] collected actual flight data from multiple sensors (such as course sensor, attitude angle sensor and throttle actuator, etc.) from a fixed-wing UAV, and used shared nearest neighbor-based algorithms to classify and recognize the conditions. They proposed a multiple conditions oriented dynamic kernel principal component analysis with weighted sliding window to diagnose the faults. López-Estrada et al. [13] considered a linear parameter varying system to perform sensor fault detection and isolation on a quadcopter UAV. When sensor fault appeared, the magnitude of the residuals would exceed a predefined threshold level. The authors showed that the residuals were sensitive to only one fault and that the method was robust against disturbances.

As the application of UAVs in the civilian domain ramps up in recent years, many novel fault diagnosis approaches have been developed for UAV self-inspection during airborne operations. These approaches generally fall into two types, model-based and data-driven. The model-based approaches are based on physical principles and use mathematical models to represent the system [14], [15]. Maqsood et al. [16] proposed a fault diagnosis approach based on an improved high gain observer on quadrotor system that exhibited better accuracy and lower gain requirement compared to traditional observers. Their method was proved from simulation results under various fault scenarios and was shown to be useful for fault detection in angular rate sensors on quadrotor UAVs. Lee and Choi [17] presented a fault detection and isolation method targeting application on quadcopters’ sensors and actuators. The method integrated Neural Adaptive Observers (NAOs) into an Interactive Multiple Model (IMM) framework. The authors constructed two NAOs for sensor faults and actuator faults respectively. Their method was demonstrated to be effective via simulation studies. Nguyen et al. [18] used nonlinear Thau observer technique to detect actuator faults of a hexacopter and to generate the residuals. They also used a sliding mode and disturbance observer to tackle the disturbance issue. Experimental results showed that their approach could keep a hexacopter safely airborne in cases when up to two motors fail. Saied et al. [19] proposed two approaches for octorotor UAV actuator failures: robust control based on high-order super-twisting sliding mode techniques and reconfigurable control based on redistributing the control signals among the healthy actuators using reconfigurable multiplexing and pseudo-inverse control allocation. The proposed techniques were evaluated through different real experiments on a coaxial octorotor UAV. Saied et al. [20] used flatness techniques for residual generation, whereas the residual signals were used for fault diagnosis for both sensors and actuators. Their approach was analytically and experimentally demonstrated on Matlab/Simulink and on a hexacopter UAV, respectively. Wang et al. [21] used extended Kalman filter (EKF) technique to design a fault observer group with flight data provided by the multi-sensor navigation unit (MSNU) for fault detection and isolation on a hex-rotor UAV. The reconfiguration controller was able to guarantee the attitude control stability and control quality of the hex-rotor UAV after fault was detected. Their approach was validated by numerical simulation and actual flight experiments. Compared to model-based approach, a data-driven approach for fault diagnosis is preferred when system models are not available [22], in which case faults should be classified or detected by machine learning methods. Along this line, Wang et al. [23] proposed a data-driven method for flight data anomaly detection using Relevance Vector Machine (RVM)-based multimodal regression model. They built a regression model engine to realize anomaly detection by switching detection model in different flight modes. Their approach was tested against real flight data and the results have demonstrated high accuracy. Sun et al. [24] proposed a data-driven Adaptive Neuron Fuzzy Inference System (ANFIS)-based approach to detect on-board navigation sensor faults in UAVs. Their algorithm obtained high-accuracy residual estimations, capturing the characteristics of the data and feeding the ANFIS-based decision system. Field test results demonstrated faster and more precise fault detection than the Particle Filter (PF)-based algorithm and the Fuzzy Inference System (FIS)-based algorithm. Park et al. [25] used multivariate statistical analysis techniques on the inertial measurement unit (IMU) and the motor input measurements of a quadrotor to detect an actuator fault in real-time. In the experiment, they blocked the motor input signal to simulate actuator fault condition. Among the techniques they used, the partial least squares regression showed the best performance with the highest accuracy on actuator fault detection. Some researchers combined model-based and data-driven approaches for UAV fault detection. For instance, Ouadine et al. [26] proposed an approach based on a Hammerstein-Wiener model for
generating residuals and then used neural networks to perform classification and automatic diagnosis. Their model was validated by the simulated test data in MATLAB. Furthermore, Wang et al. [27] proposed an adaptive sliding mode control technique to generate appropriate control signals to compensate model uncertainties for a quadrotor UAV. In that work, recurrent neural networks were designed to estimate the severity of the actuator faults with a certain level of accuracy. The effectiveness of the method was compared to that of a model-based two-stage extend Kalman filter (TSEKF) fault estimator and superior performance was demonstrated.

Overall, model-based fault detection techniques have been mostly based on specific airframe design and actuator characteristics (for example, a model that is suitable for a quadcopter will not work on a hexacopter). In contrast, the method presented in this paper is agnostic of the airframe, motor type and propeller configurations, and thus will in theory work for any VTOL aircraft. Another critical difference between our work and most of the literature work on UAV fault diagnosis is the intended application scenario. Specifically, our proposed platform intends to perform unmanned preflight check before an autonomous (or remote-controlled) UAV takes off, while most works reviewed above aim to identify, isolate and contain sensor and/or actuator faults while the UAV is in airborne operation. From both application and methodology perspectives, the proposed work is novel and unique.

Besides motor and propeller functional and physical integrity, another factor that affects flight safety is the payload’s weight and placement. Overweight or imbalanced loading not only affects dynamic stability, but also causes overheating of motors and ESC modules. Imbalanced power distribution for motors can also reduce the drone’s agility in responding to power-intensive maneuvers such as dodging obstacles and fighting against gusts of wind. In air cargo transportation, load planning is a critical task to ensure the payload’s weight and placement. Overweight or imbalanced loading not only affects dynamic stability, but also causes overheating of motors and ESC modules. The effectiveness of the method was compared to that of a model-based two-stage extend Kalman filter (TSEKF) fault estimator and superior performance was demonstrated.

### TABLE 1. Definition of index sets.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( K )</td>
<td>set of scale status, ( k \in K ), ( -1 ): stable status when the platform is placed on the load cells, before the drone is landed on the platform; ( 0 ): stable status when the drone is landed on the platform, propellers not spinning; ( 1-4 ): status when propeller 1 - 4 spins at the respective stable level of output; ( T_k )</td>
</tr>
<tr>
<td>( \mathcal{Z} )</td>
<td>set of pairs of propellers, ( i \in \mathcal{Z} )</td>
</tr>
<tr>
<td>( \mathcal{J} )</td>
<td>set of load cells, ( j \in \mathcal{J} )</td>
</tr>
<tr>
<td>( \mathcal{S} )</td>
<td>set of samples, ( s \in \mathcal{S} )</td>
</tr>
</tbody>
</table>

### TABLE 2. Definition of parameters.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( w_{ij} )</td>
<td>reading value from load cell ( j ) at time ( t )</td>
</tr>
<tr>
<td>( \Delta w_{ijk} )</td>
<td>change in the reading value from load cell ( j ) at time ( t ) under status ( k )</td>
</tr>
<tr>
<td>( M_k )</td>
<td>moment of force generated in status ( k )</td>
</tr>
<tr>
<td>( W_k )</td>
<td>force generated in status ( k )</td>
</tr>
<tr>
<td>( l )</td>
<td>distance from the frame center to each propeller for the one- propeller-pair situation (cm)</td>
</tr>
<tr>
<td>( l_i )</td>
<td>distance from the frame center to the ( i )-th propeller (cm)</td>
</tr>
<tr>
<td>( \alpha )</td>
<td>angle between ( x )-axis and the vector pointing from the frame center to the propeller above ( x )-axis, in the one-propeller-pair situation</td>
</tr>
<tr>
<td>( \alpha_i )</td>
<td>angle between ( x )-axis and the vector pointing from the frame center to the ( i )-th propeller in the 1st or 2nd quadrant</td>
</tr>
<tr>
<td>( x_{ij}^x, y_{ij}^y )</td>
<td>( x ) and ( y )-coordinate of load cell ( j ) (cm)</td>
</tr>
</tbody>
</table>

III. METHODS

We first develop a nonlinear least squares model for estimating the center position and orientation of a single airframe arm based on the location samples of two propellers installed on both ends of the arm. This is the baseline model on which the analytical solution is derived in Proposition 1. In subsection III-B, the model is extended to a multi-armed architecture which can be instantiated to have an arbitrary number of propeller pairs. The analytical solution is presented in Proposition 2. In subsection III-C, using a quadcopter as example, we describe the diagnosis procedure for obtaining the location samples of the propellers, which serve as raw data for the statistical inference models. The index sets, parameters and decision variables involved in the mathematical models are listed in Tables 1 to 3, respectively.
of the squared errors between the estimate values (variables) and the measurement values (parameters), leading to a least squares model with nonlinear geometric constraints.

\[
\begin{align*}
\text{Minimize} & \quad \sum_s (x^P_1 - x_{1s}^P)^2 + (y^P_1 - y_{1s}^P)^2 \\
\text{s.t.} & \quad x^P_1 = x_{DC} + l_1 \cos(\alpha + \theta) \\
& \quad y^P_1 = y_{DC} + l_1 \sin(\alpha + \theta) \\
& \quad x_{1s}^P = x_{DC} - l_1 \cos(\alpha + \theta) \\
& \quad y_{1s}^P = y_{DC} - l_1 \sin(\alpha + \theta) \\
& \quad \theta \in (-\pi, \pi]
\end{align*}
\]

(1)

The above problem can be written as an (almost) unconstrained nonlinear optimization problem with bounds on the variable \(\theta\), as follows.

\[
\begin{align*}
\text{Minimize} & \quad \sum_s (x_{DC} + l_1 \cos(\alpha + \theta) - x_{1s}^{IS})^2 \\
& \quad + (y_{DC} + l_1 \sin(\alpha + \theta) - y_{1s}^{IS})^2 \\
& \quad + (x_{DC} - l_1 \cos(\alpha + \theta) - x_{1s}^{IS})^2 \\
& \quad + (y_{DC} - l_1 \sin(\alpha + \theta) - y_{1s}^{IS})^2, \\
& \quad -\pi < \theta \leq \pi
\end{align*}
\]

(2)

This problem can be solved by a numeric optimization solver. To simplify the solution process, we derive the analytical solution for this problem, as stated in Proposition 1.

**Proposition 1:** The optimal solution \((x^{DC*}, y^{DC*}, \theta^*)\) to problem (2) is:

\[
\begin{align*}
x^{DC*} & = \sum_s \left(x_{1s}^{IS} + x_{1s}^{IS}\right)/(2|S|) \\
y^{DC*} & = \sum_s \left(y_{1s}^{IS} + y_{1s}^{IS}\right)/(2|S|) \\
\theta^* & = \begin{cases} 
\frac{n \pi - \alpha + \frac{b}{\sqrt{b^2 - a}} \arccos \frac{a}{\sqrt{a^2 + b^2}}}{b} & b \neq 0 \\
\frac{n \pi - \alpha + \arccos \frac{a}{\sqrt{a^2 + b^2}}}{b} & b = 0
\end{cases}
\end{align*}
\]

where \(a = \sum_s (x_{1s}^{IS} - x_{1s}^{IS}), b = \sum_s (y_{1s}^{IS} - y_{1s}^{IS})\) and \(n\) is any nonnegative even integer that makes \(\theta^*\) fall in the interval \((-\pi, \pi]\).

The (sufficient) optimality condition states that any point \(x^*\) at which \(\nabla f(x^*) = 0\) and \(\nabla^2 f(x^*)\) is positive definite is a strong local minimizer of \(f\) [40]. The formulae for \((x^{DC*}, y^{DC*}, \theta^*)\) in the above proposition are derived by setting the partial derivatives of equation (2) with respect to each variable to zero. To show that the first-order solution is indeed a minimizer of the function, we show that the Hessian matrix is positive semi-definite at the calculated \((x^{DC*}, y^{DC*}, \theta^*)\) values. The detailed proof is given in Appendix B.

**B. MODEL AND ANALYTICAL SOLUTION FOR MULTIPLE PROPELLER PAIRS**

We now extend the above model to account for a general air frame architecture with multiple pairs of propellers. Figure 4
we can write min

point location and air frame orientation can be formulated
demonstrates such a generic set up with n ≥ 2 pairs of
propellers \( l_i \) to \( l'_i \). For each propeller pair \( l_i \), the distance to
its symmetric center is \( l_i \) and the angle between \( l_i \) and the
x-axis \( \alpha \). Note that we do not require that all arms are of equal
length. The least squares model for estimating the center
point location and air frame orientation can be formulated
similarly to the single pair of propellers condition, then define
\( f(x^{DC}, y^{DC}, \theta) \) to denote the general function to optimize,
we can write \( \min f(x^{DC}, y^{DC}, \theta) \) as follows.

\[
\text{Minimize } \sum_{i} \sum_{s} (x^{DC} + l_i \cos(\alpha_i + \theta) - x^{IS})^2 \\
+ (y^{DC} + l_i \sin(\alpha_i + \theta) - y^{IS})^2 \\
+ (\gamma^{DC} - l_i \sin(\alpha_i + \theta) - y^{IS})^2 \\
- \pi < \theta \leq \pi
\]

(3)

The analytical solution for problem (3) is not as
straightforward to express as that for the single-arm problem (2).
It requires a recursive functional evaluation. Furthermore,
in each recursive step the intermediate variable \( \beta_i \) must be
projected into the desired angular range of \([-\pi/2, \pi/2]\).
We put forward the following lemma to justify the uniqueness
of the results from the range projection operator \([ \cdot ]\), which is
defined and used subsequently.

Lemma 1: Given \( l, u \in \mathbb{R} \), and \( l < u \), for any \( a \in \mathbb{R} \), there
exists a unique \( n \in \mathbb{Z} \) such that

\[
l < a + n(u - l) < u
\]

Proof: Assume that \( \exists m, n \in \mathbb{Z} \) such that

\[
l < a + n(u - l) < u
\]

\[
l < a + m(u - l) < u
\]

When \( n > m \), we can look at their extreme value that

\[
n < \frac{u - a}{u - l} \quad \text{and} \quad m > \frac{l - a}{u - l}
\]

So, we have

\[
n - m < \frac{u - a}{u - l} - \frac{l - a}{u - l}
\]

and thus, we have \( n - m < 1 \). On the other hand, when \( n < m \),
we can obtain \( n - m > -1 \) by the same way, thus we have
\(-1 < n - m < 1 \). Since \( m, n \in \mathbb{Z} \), we must have \( n - m = 0 \).
Thus, the uniqueness of \( n \) is established.

According to Lemma 1, we can denote such an \( n \) as \( n^* \).
Define \( [a]^n \) to be \( a + n^*(u - l) \), i.e. \( l < [a]^n < u \). Then for
every \( a \), there exists a unique value for \( [a]^n \).

Proposition 2: The optimal solution \((x^{DC*}, y^{DC*}, \theta^*)\) to
problem (3) is:

\[
\begin{align*}
x^{DC*} &= \sum_{i} \sum_{s} \left( x^{IS} + x^{IS}_i \right) / (2|IS|) \\
y^{DC*} &= \sum_{i} \sum_{s} \left( y^{IS} + y^{IS}_i \right) / (2|IS|) \\
\theta^* &= n\pi - \beta_i
\end{align*}
\]

where

\[
a_i = \sum_{s} (x^{IS}_i - x^{IS}) \\
b_i = \sum_{s} (y^{IS}_i - y^{IS}) \\
\gamma_i = \begin{cases} 
\arccos \frac{a_i}{\sqrt{a_i^2 + b_i^2}}, & b_i \neq 0 \\
\arccos \frac{a_i}{\sqrt{a_i^2 + b_i^2}}, & b_i = 0
\end{cases}
\]

\[
A_i = \frac{(A_{i-1} \cos \beta_{i-1} + B_{i-1} \cos \mu_{i-1})}{\sqrt{(A_{i-1} \sin \beta_{i-1} + B_{i-1} \sin \mu_{i-1})^2}} \quad i = 2, 3, \ldots, I
\]

\[
B_i = 2l_{i-1} \sqrt{a_{i+1}^2 + b_{i+1}^2} \quad i \leq I - 1
\]

\[
\mu_i = \alpha_{i+1} - \gamma_{i+1}, \quad i \leq I - 1
\]

\[
\beta_1 = \left[ (\alpha_i - \gamma_i) \right]_0^{\pi}
\]

The proof of Proposition 2 is similar to the proof of
Proposition 1. We can obtain all the partial derivatives
to calculate critical point, and obtain all the second partial
derivatives to get the Hessian matrix \( H_{II} \). Finally we can prove
\((x^{DC*}, y^{DC*}, \theta^*)\) is the local minimum point for problem (3).
The detailed proof is given in Appendix C.

C. MEASUREMENT PRINCIPLES AND PROCEDURE

Theorem 1 (Equilibrium [41], Page 109): When a body is
in equilibrium, the resultant of all forces acting on it is zero.
Thus, the resultant force \( \textbf{R} \) and the resultant couple \( \textbf{M} \) are
both zero, and we have the equilibrium equations:

\[ R = \sum F = 0 \]  \hspace{1cm} (4)

and

\[ M = \sum M = 0 \]  \hspace{1cm} (5)

These requirements are both necessary and sufficient conditions for equilibrium.

Theorem 1 implies that, at equilibrium, we can always obtain the magnitude and position of an unknown force by the other known forces in the system. So we define \( F_k(\Delta w_{jk}) = (x_t, y_t, W_t) \) as a function that given change of scale reading at time \( t \) under status \( k \) (\( \Delta w_{jk} \), the “known forces”), we can calculate the new force to the system at time \( t \) compared with time 0 with magnitude \( W_t \) and the \( x \) and \( y \)-coordinates \( x_t, y_t \). Then we can obtain:

\[
W_t = \sum_{j \in J} \Delta w_{jk} - \sum_{k''=-1}^{k=1} W_{k''},
\forall t \in T_k, k' = \min(1, k)
\]  \hspace{1cm} (6)

\[
x_t = \frac{1}{W_t} \left( \sum_{j \in J} \Delta w_{jk} x_j^S - \sum_{k''=-1}^{k=1} M_{k''} \right),
\forall t \in T_k, k' = \min(1, k)
\]  \hspace{1cm} (7)

\[
y_t = \frac{1}{W_t} \left( \sum_{j \in J} \Delta w_{jk} y_j^S - \sum_{k''=-1}^{k=1} M_{k''} \right),
\forall t \in T_k, k' = \min(1, k)
\]  \hspace{1cm} (8)

Figure 5 shows a sketch of the platform, and the platform instantiates the “body” in equilibrium referred to in Theorem 1. At equilibrium, equation (4) is instantiated by (6), which states that the resultant force of the support forces from four measure points \( w_1, w_2, w_3 \) and \( w_4 \) and all forces on the platform \( W \) is zero. Equation (5) is instantiated by (7) and (8), which states that the resultant moment of support forces and all forces on the platform is also zero. For simplicity, we ignored the propellers’ wind effect on the platform, which is indeed negligible in practice as evidenced in our physical demonstration.

Equations (6), (7) and (8) can be summarized as a general function to calculate the new force and its position, given in Algorithm 1. The complete testing procedure for a quadcopter drone is given in Algorithm 2. Calculation details for each status \( k \) are given in Appendix A.

**Algorithm 1 Calculate the New Force and Its Location**

1: function \( F(\Delta w_j) \) \hspace{1cm} \( \triangleright \) The change of \( w_j \)
2: \( W \leftarrow \sum_{j \in J} \Delta w_j \)
3: \( x \leftarrow \frac{1}{W} \sum_{j \in J} \Delta w_j x_j^S \)
4: \( y \leftarrow \frac{1}{W} \sum_{j \in J} \Delta w_j y_j^S \)
5: return \( (x, y, W) \)

**IV. EXPERIMENTS**

In this section, we perform experiments with both simulated data and a real system under different parameter settings to validate Proposition 2 and demonstrate the practical use and effectiveness of the proposed procedures.

**A. SIMULATION EXPERIMENTS**

Our simulation experiments were performed on a Dell Precision Tower 3420 computer with an Intel® Core™ i7-7700 CPU @ 3.60 GHz, 32.0 GB RAM and 64-bit Windows 10 Enterprise Operating System. We used LINDOGlobal solver [42] (via GAMS 30.1.0) for global optimizer. Relative optimality threshold was set to \( 10^{-7} \). We used Python 3.7 for implementing the analytical method. Note that in practical use, the Python program that implements the analytical method (i.e., Proposition 2) does not rely on an external optimization solver and thus can be run as well on small, system-on-chip (SoC) computers such as a Raspberry Pi computer.
Algorithm 2 Measurement Procedure for Quadcopter Drones

1: for \( t \in T_{-1} \) do >> Get weight of the platform
2: \( w_{jt} \leftarrow w_j, \forall j \in J \)
3: \( \bar{w}_{-1j} \leftarrow \frac{1}{|T_{-1}|} \sum_{t \in T_{-1}} w_{jt} \)
4: for \( t \in T_0 \) do
5: \( w_{jt} \leftarrow w_{j}, \forall j \in J \)
6: \( \bar{w}_{0j} \leftarrow \frac{1}{|T_0|} \sum_{t \in T_0} w_{jt} \)
7: \( \Delta w \leftarrow \bar{w}_{0j} - \bar{w}_{-1j} \)
8: \((x_0, y_0, W_0) \leftarrow F(\Delta w) \) >> Get CG and weight of the drone
9: \( \chi^{DCG} \leftarrow x_0 \)
10: \( \chi^{DCG} \leftarrow y_0 \)
11: \( W^D \leftarrow W_0 \)
12: for \( i = 1 \) to 4 do
13: for \( t \in T_i \) do
14: \( w_{jt} \leftarrow w_j, \forall j \in J \)
15: \( \Delta w \leftarrow \Delta w_{jt} - W_{0j} \)
16: \((x_i, y_i, W_i) \leftarrow F(\Delta w) \) >> Get position and force of propeller \( i \)
17: \( x_{is}^{IS} \leftarrow x_i \)
18: \( y_{is}^{IS} \leftarrow y_i \)
19: \( F_{is} \leftarrow W_i \)
20: return \((x_i, y_i, W_i), i \in \{0, 1, 2, 3, 4\}\)

1) EXPERIMENTS ON DIFFERENT NUMBERS OF PROPELLERS

We generated two datasets. In order to observe the effects of the number of propellers on solution time, we simulated 6 multicopters with 2, 4, 6, 8, 10, and 12 propellers, and used a sample of 50 for each propeller. The arm length \( l_i \) was chosen randomly between 10 cm and 30 cm, and the propeller angle \( \alpha_i \) for each pair \( i \) was randomly chosen in the interval \((0, \pi)\). For each of simulated multicopter, we generated 20 batches \((b = 1, 2, \ldots, 20)\). In each batch, the \( x \) and \( y \)-coordinates \((x_b^{DC}, y_b^{DC})\) of the geometric centers of the multicopter were randomly generated within \([-2, 2]\)cm and the yaw angle of drone \( \theta_b \) was randomly generated within \([-0.99\pi, 0.99\pi]\). Then the \( x \) and \( y \)-coordinates for different propeller positions \( x_{is}^{IS}, y_{is}^{IS}, x_{is}^{JS}, y_{is}^{JS} \) can be generated as:

\[
x_{is}^{IS} = x_b^{DC} + l_i \cos(\alpha_i + \theta_b) + \epsilon_{is} \quad (9)
y_{is}^{IS} = y_b^{DC} + l_i \sin(\alpha_i + \theta_b) + \epsilon_{is} \quad (10)
x_{is}^{JS} = x_b^{DC} - l_i \cos(\alpha_i + \theta_b) + \epsilon_{is} \quad (11)
y_{is}^{JS} = y_b^{DC} - l_i \sin(\alpha_i + \theta_b) + \epsilon_{is}
\]

\[i \in \{1, 1, 2, \ldots, 1, 2, \ldots, 6\}; \quad s = 1, 2, \ldots, 50; \quad b = 1, 2, \ldots, 20 \quad (12)\]

where \( \epsilon_{is} \) is random error uniformly distributed in the range \([-1.5, 1.5]\) cm. We solved the model using both the analytical method given in Proposition 2 and the numerical solver LINDOGlobal. The solution time for each case was then recorded. Taking the average solution time of all batches within each propeller count setting, we obtained the results shown in Figure 7.

As shown in Figure 7, the time taken for the global optimization method increases significantly as the structure of the multicopter becomes more complex. A multicopter with 12 propellers requires 125.1 seconds for the numerical solver to get the exact position, while our analytical method requires less than 0.1 second to achieve the same result.

The results are compared with the ground-truth values used for generating the datasets, and the differences are summarized in Table 4. The distribution of the differences is not significantly different from normal distribution, which is expected.

To control for the effect of sample size, we designed experiments with the same total sample size (1680). So for this experiment, we adjusted the sample size to keep the total sample size constant for different number of propellers. Values for \( x_{is}^{IS}, y_{is}^{IS}, x_{is}^{JS}, y_{is}^{JS} \) were generated by the same method in equations (9) to (12), where \( i \in \{1, 1, 2, \ldots, 1, 2, \ldots, 6\}; \quad s = 1, 2, \ldots, 420/1 \) and \( b = 1, 2, \ldots, 20 \). We simulated 20 batches of multicopters with the number of propellers ranging from 2 to 18, all having an arm length of \( 10\sqrt{2} \) cm, all propellers being located at the vertices of a regular polygon, and the initial angle \( \alpha_1 \) being set at a position such that the \( x \)-axis bisects the angle between arm \( L_1 \) and \( L'_1 \). In each batch, \( x \) and \( y \)-coordinate of the geometric centers \( x_b^{DC} \) and \( y_b^{DC} \).
of the multicopter were randomly generated within \([-2, 2]\) cm and the yaw of drone \(\theta_b\) was randomly generated within \([-0.99\pi, 0.99\pi]\). For each propeller number setting, we used the average value of the distance difference (mean of distance error) between the model-predicted translation and the actual position, and the average value of the angle difference (mean of angle error) between the model-predicted rotation and the actual rotation angle to obtain the experimental results in Figure 8 and Figure 9, respectively. It can be seen that the number of propellers of the multicopter does not affect the accuracy of the predicted positions when the total sample size is held constant.

Figure 10 shows the solution time of the analytical method and numerical solver for each number of propellers and the same total number of samples. Since the solution time for some batches in numerical solver was longer than 3600 seconds (we terminated the process when solution time reaches 3600 seconds for efficiency and any solution time longer than 3600 seconds were recorded as 3600s), we used the median to present the results. We can find that the number of propellers and the solution time are roughly positively correlated for numerical solver, while the solution time in analytical method remains consistently small despite the increasing number of propellers involved in the system.

2) EXPERIMENT ON DIFFERENT SAMPLE SIZES
To understand how the sample size of each propeller affects the solution time, we generated 6 sample sizes: 50, 100, 150, 200, 250 and 300, using a centrosymmetrical quadcopter structure with two pairs of propellers (i.e., \(i = 2\)). The arm length is \(10\sqrt{2}\) cm (i.e., \(l_1 = l_2 \approx 14.1\) cm).

The initial angles between the arms to x-axis are \(\pi/4\) and \(\pi/3\) (i.e., \(\alpha_1 = \pi/4\) and \(\alpha_2 = \pi/3\)). The method to generate \(\Delta x, \Delta y\) and \(\theta\) is the same as above. We obtained propellers position \(\Delta x^{IS}, \Delta y^{IS}, \theta^{IS}\) for the quadcopter by equations (9), (10), (11) and (12), where \(i = 1, 2\), \(s \in \{1, 2, \ldots, 50\}, \{1, 2, \ldots, 100\}, \ldots, \{1, 2, \ldots, 300\}\) and \(b = 1, 2, \ldots, 20\). The solution time of the two methods for this dataset is presented in Figure 18.

In Figure 18, there is a clear increasing trend of computing time as more samples are taken. For a standard quadcopter, it took the numerical solver 11.8 seconds to solve an instance of 50 samples, and the analytical method took only 0.028 seconds on average. The analytical method remained robust in solution time even if the sample size becomes as large as 300.

Table 5 shows that as the sample size increases, there is a decreasing trend in positional and angular errors, though the decrease is not strictly monotone. To get a more detailed picture, we performed an additional set of experiments with increasing sample sizes from 1 to 200 while keeping other settings unchanged. We then obtained the mean and standard deviation of the errors in distance and angle, as presented in Figure 11 and Figure 12. Figure 11 shows that the error in distance is strongly influenced by the number of samples when the number of samples is less than 50; beyond 50, the variation gradually flattens out. Figure 12 shows that the error in angle hardly varies with the number of samples, except for a few outliers, the mean value is within \(\pm 0.001\) and the standard deviation is within 0.01, for a sample size over 10.

In practice, we can choose the optimized number of samples to achieve the accuracy we need based on the relationship between the error in distance and the sample size.

B. EXPERIMENTS ON A REAL DRONE PLATFORM
1) MATERIAL AND APPARATUS
In this experiment, we measured the weight change of the scale system in different statuses, and finally obtained the CG of the drone and propellers’ positions using Algorithm 1 and Algorithm 2.
TABLE 5. Differences between the estimate and true values in $x_{DC}$, $y_{DC}$ and $\theta$ under different sample sizes, for a quadcopter setup.

<table>
<thead>
<tr>
<th>Sample Size</th>
<th>$x_{DC}^{\text{Est}} - x_{DC}^{\text{True}}$</th>
<th>$y_{DC}^{\text{Est}} - y_{DC}^{\text{True}}$</th>
<th>$\theta_{DC}^{\text{Est}} - \theta_{DC}^{\text{True}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>0.0044</td>
<td>0.0207</td>
<td>0.0005</td>
</tr>
<tr>
<td>100</td>
<td>-0.0040</td>
<td>-0.0088</td>
<td>0.0010</td>
</tr>
<tr>
<td>150</td>
<td>0.0042</td>
<td>-0.0013</td>
<td>0.0003</td>
</tr>
<tr>
<td>200</td>
<td>-0.0030</td>
<td>-0.0060</td>
<td>0.0003</td>
</tr>
<tr>
<td>250</td>
<td>-0.0019</td>
<td>0.0044</td>
<td>0.0000</td>
</tr>
<tr>
<td>300</td>
<td>0.0015</td>
<td>-0.0043</td>
<td>0.0005</td>
</tr>
</tbody>
</table>

As shown in Figure 14, four digital load cells were placed beneath the platform to take measurements. Each load cell consists of 1 TAL220 10kg load cell [43], 1 SparkFun OpenScale microcontroller [44], and a wooden base. A USB hub parallel-connected the four scales sets to a personal computer (PC) for synchronized data collection. The platform was made by a 23-3/4” × 23-3/4” polystyrene louver, of dimension 603.25 mm × 603.25 mm × 12.8 mm. Each hole of the platform is 12.55 mm × 12.55 mm, and the slat between each adjacent two holes is 1.65 mm in width.

The platform and the quadcopter used in the experiment are shown in Figure 1, 13, 14 and 15. The load cells used in our experiments are strain gauge load cells, which are capable of accurately measuring small changes in weight with one-gram. The microcontroller connected to each load cell is able to send the instantaneous load cell reading to the PC every 200 ms. The quadcopter structure (shown in Figure 13) was made of 3D printed parts and carbon fiber square tubes. For the quadcopter, we used four 2212/920kV brushless motors, each connected to a Simonk 30A Firmware Brushless ESC. The propellers were DJI 9450, and the battery was a 5000mAh 4-cell LiPo battery pack.
2) PROCEDURE OUTLINE

Before the test process, we set microcontrollers’ baud rate to 11520 bps, set board report interval to 200 ms, varied the output of each propeller (8% - 60%), and set spin time for each propeller to 5 s during the motor test. During the test process, we first tared all load cells to zero, set the platform onto the load cell system, and obtained the initial readings. This step, mentioned in Algorithm 2 lines 1 to 7, is for obtaining the weight of the platform. Afterwards, we put the drone onto the platform (location marked before the process), and obtained the new readings. These readings were used for obtaining the CG and weight of the drone by the steps described in lines 8 to 11 in the Algorithm 2 and the method in Algorithm 1. Finally, the propellers were spun in sequence and the readings were recorded. The readings were used for obtaining the position and force of all propellers by steps described in Algorithm 2 in lines 12 to 19. The action commands were sent to the drone by the Mission Planner software through a direct USB data link.

We changed motor output intensity and duration, drone position, and propeller with different damage situations, and recorded experiment results, as shown below.

3) RESULTS AND ANALYSIS

Table 6 and Table 7 show the errors between the translation obtained by the model and the actual translation for different motor outputs (30%, 40% and 50%) in the experimental results, respectively. The experiment was repeated by different αi settings.

The box plots (Figure 16 and Figure 17) show the average values of translational and rotational errors for different motor output conditions, respectively. The whiskers show extreme values of the results. The plots show that both translational error and rotational error are lower when the motor output is 40%. We can use this finding in future practice to make the detection results more accurate by selecting the appropriate motor output in the testing procedure. For instance, for the quadcopter used in our experiments, a 40% output level has been the most suitable.

4) DAMAGED PROPELLER TESTS

We performed field tests using a set of damaged propellers having damage levels of 0%, 15%, 31% and 55% as shown in Figure 20. The four propellers of different damage levels were mounted on the quadcopter, and after setting a certain motor output rate, the four propellers were rotated one by
FIGURE 19. Propeller Position: motor output = 40%, rotation: 30°, sample size: 9. $x_{DC} = -0.146$ cm, $y_{DC} = -0.222$ cm, $\theta = 0.502$ rad.

FIGURE 20. An intact propeller juxtaposed with broken ones with 15%, 31% and 55% broken area.

one for 5 seconds each. The sensors detected and recorded weight changes at a frequency of 5 Hz when the weight measurements became stable. From the experimental results in Figure 22, we can see that although the propellers are damaged to a large extent, our model is still able to obtain the precise position and the geometric center of the quadcopter. Figure 21 shows the output force for each propeller condition. The differences in the output force are due to different levels of propeller damages - a higher level damage resulted in a greater reduction in output force. If all four propellers had been intact, the sequential points of the same color (same motor output rate) would have been aligned along the same horizontal line at some force level. Each successive aggregated point represents a collection of weight reduction values (output force) over a period of time for a given motor output condition for that propeller. By comparing the measured values of the output force against what is expected for an intact propeller, the algorithm is able to identify the damaged propellers and estimate the degree of damage.

V. CONCLUSION

In this paper, we have investigated the challenge of automatically performing safety checks for multicopter drones to support unmanned deployment of drones in various application domains. The challenge involved acquisition and fusion of the multicopter status data with weight measurement data from a smart landing platform. To detect component anomalies, we have developed a nonlinear least squares model to
estimate the amount of translation and rotation of the multicopter’s airframe with respect to the nominal values. Based on the properties of the model, we have derived the analytical formula for its solution and proved the uniqueness of the solution. Simulation results have validated the superiority of the analytical approach over numerical solvers in solution time, particularly for problem instances with large sample sizes. The experiments conducted on real drone platforms have verified the robustness and the practical value of the proposed approach. Future research can address the impact of propeller-generated wind on the platform, extend the framework to address asymmetric airframe types, improve the design and materials of the platform, and implement the proposed algorithm in a software package for practical use.

**APPENDIX A**

**DETAILS OF CALCULATION IN EACH STATUS k**

Given different status k, we have:

\[ F(w_{jt}) = (x^L_{LCG,j}, y^L_{LCG,j}, W^l_j), \forall t \in T_0 \]
\[ W^L_j = \sum_{j \in J} w_{jt}, \forall t \in T_0 \]
\[ x^L_{LCG,j} = \frac{1}{W^L_j} \sum_{t \in T_0} w_{jt} x^S_{jt}, \forall t \in T_0 \]
\[ y^L_{LCG,j} = \frac{1}{W^L_j} \sum_{t \in T_0} w_{jt} y^S_{jt}, \forall t \in T_0 \]

To obtain \( W^L, x^L_{LCG}, y^L_{LCG} \) and \( M_{-1} \):

\[ W^L = \frac{1}{|T_0|} \sum_{j \in J} \sum_{t \in T_0} w_{jt} \]
\[ x^L_{LCG} = \frac{1}{W^L |T_0|} \sum_{j \in J} \sum_{t \in T_0} w_{jt} x^S_{jt} \]
\[ y^L_{LCG} = \frac{1}{W^L |T_0|} \sum_{j \in J} \sum_{t \in T_0} w_{jt} y^S_{jt} \]
\[ M_{-1} = W^L p_{LCG} \]
\[ F_0(w_{jt}) = (x^l_{D CG,j}, y^l_{D CG,j}, W^D_j), \forall t \in T_0 \]
\[ W^D_j = \sum_{j \in J} w_{jt} - W^L, \forall t \in T_0 \]
\[ x^D_{D CG,j} = \frac{1}{W^D_j} \left( \sum_{j \in J} w_{jt} x^S_{jt} - W^L x^L_{LCG,j} \right), \forall t \in T_0 \]
\[ y^D_{D CG,j} = \frac{1}{W^D_j} \left( \sum_{j \in J} w_{jt} y^S_{jt} - W^L y^L_{LCG,j} \right), \forall t \in T_0 \]

To obtain \( W^D, x^D_{D CG}, y^D_{D CG} \) and \( M_0 \):

\[ W^D = \frac{1}{|T_0|} \sum_{j \in J, t \in T_0} w_{jt} - W^L \]
\[ x^D_{D CG} = \frac{1}{W^D |T_0|} \left( \sum_{j \in J, t \in T_0} w_{jt} x^S_{jt} - W^L x^L_{LCG,j} \right) \]

\[ y^D_{D CG} = \frac{1}{W^D |T_0|} \left( \sum_{j \in J, t \in T_0} w_{jt} y^S_{jt} - W^L y^L_{LCG,j} \right) \]

\[ M_0 = W^D p^D_{D CG} \]

\[ F_j(w_{jt}) = (x^{IS}_{IS,j}, y^{IS}_{IS,j}, F_{it}), \forall t \in T_i \]
\[ F_{it} = \sum_{j \in J} w_{jt} - W^L - W^D, \forall t \in T_i \]
\[ x^{IS}_{it} = \frac{1}{F_{it}} \left( \sum_{j \in J} w_{jt} x^S_{jt} - W^L x^L_{LCG,j} - W^D x^D_{D CG,j} \right), \forall t \in T_i \]
\[ y^{IS}_{it} = \frac{1}{F_{it}} \left( \sum_{j \in J} w_{jt} y^S_{jt} - W^L y^L_{LCG,j} - W^D y^D_{D CG,j} \right), \forall t \in T_i \]

\[ F_{it} = W^L + W^D - \sum_{j \in J} w_{jt}, \forall t \in T_i \]
\[ x^{IS}_{is} = \frac{1}{F_{it}} \left( W^L x^L_{LCG} + W^D x^D_{D CG} - \sum_{j \in J} w_{jt} \right), \forall s \in T_i \]
\[ y^{IS}_{is} = \frac{1}{F_{it}} \left( W^L y^L_{LCG} + W^D y^D_{D CG} - \sum_{j \in J} w_{jt} \right), \forall s \in T_i \]

**APPENDIX B**

**PROOF OF PROPOSITION 1**

Proof of Proposition 1: To prove, we only need to show that the optimal solution is a minimum point for equation (2) in the given domain. Take the derivative of equation (2), we get

\[ \frac{\partial f_1}{\partial x^{DC}} = 4|S| x^{DC} - 2 \sum_s \left( x^{IS}_{1s} + x^{IS}_{2s} \right) \]

\[ \frac{\partial f_1}{\partial y^{DC}} = 4|S| y^{DC} - 2 \sum_s \left( y^{IS}_{1s} + y^{IS}_{2s} \right) \]

\[ \frac{\partial f_1}{\partial \theta} = 2l \sum_s \left( x^{IS}_{1s} - x^{IS}_{2s} \right) \sin(\alpha + \theta) \]

\[ + 2l \sum_s \left( y^{IS}_{1s} - y^{IS}_{2s} \right) \cos(\alpha + \theta) \]

Define \( a := \sum_s (x^{IS}_{1s} - x^{IS}_{2s}), \)

\( b := - \sum_s (y^{IS}_{1s} - y^{IS}_{2s}), \)

\( A := \frac{a}{\sqrt{a^2 + b^2}} = \cos \gamma \) and \( B := \frac{b}{\sqrt{a^2 + b^2}} = \sin \gamma. \) We have

\[ \frac{\partial f_1}{\partial \theta} = 2l_1 (a \sin(\theta + \alpha) - b \cos(\theta + \alpha)) \]

\[ = 2l_1 \sqrt{a^2 + b^2} (A \sin(\theta + \alpha) - B \cos(\theta + \alpha)) \]

\[ = 2l_1 \sqrt{a^2 + b^2} (\cos \gamma \sin(\theta + \alpha) - \sin \gamma \cos(\theta + \alpha)) \]

\[ = 2l_1 \sqrt{a^2 + b^2} \sin(\theta + \alpha - \gamma) \]

Let \( \frac{\partial f_1}{\partial \theta} = 0, \) we have \( \sin(\theta + \alpha - \gamma) = 0 \Rightarrow \theta + \alpha - \gamma = n\pi. \)
Let us define \( \gamma \) to be the angle between the arm and the x-axis after rotating an angle of \( \theta \) from the initial position \( \alpha \), then according to Figure 23, we have

\[
\gamma = \arccos \frac{a}{\sqrt{a^2 + b^2}} \quad \text{when} \quad b = 0,
\]

\[
\gamma = b \arccos \frac{a}{\sqrt{a^2 + b^2}} \quad \text{when} \quad b \neq 0.
\]

Given \( \theta^* = n\pi - \alpha + \gamma \)

\[
\theta^* = \begin{cases} 
  n\pi - \alpha + \frac{b}{\sqrt{b^2}} \arccos \frac{a}{\sqrt{a^2 + b^2}} & b \neq 0 \\
  n\pi - \alpha + \arccos \frac{a}{\sqrt{a^2 + b^2}} & b = 0
\end{cases}
\]

Take the second derivative of equation (38), we get

\[
\frac{\partial^2 f_1}{\partial \theta^2} = \sqrt{a^2 + b^2} (\cos \gamma \cos(\theta + \alpha) + \sin \gamma \sin(\theta + \alpha))
\]

(42)

Substitute \( \theta^* \) for \( \theta \) in the equation, we can obtain:

\[
\frac{\partial^2 f_1}{\partial \theta^2} = 2l_1 \sqrt{a^2 + b^2} (\cos \gamma \cos(n\pi + \gamma) + \sin \gamma (n\pi + \gamma))
\]

(43)

When \( n = 0, 2, 4, \ldots \), \( \frac{\partial^2 f_1}{\partial \theta^2} > 0 \). Here we choose \( n = 0 \) arbitrarily.

Let the derivatives \( \frac{\partial^2 f_1}{\partial x_{DC}^2} \) and \( \frac{\partial^2 f_1}{\partial y_{DC}^2} \) in equations (36) and (37), respectively, to be zero. We can obtain

\[
\begin{cases} 
  x_{DC}^* = \sum_s \left(x_{1s}^{IS} + x_{1s}^{JS}\right) / 2|S| \\
  y_{DC}^* = \sum_s \left(y_{1s}^{IS} + y_{1s}^{JS}\right) / 2|S|
\end{cases}
\]

We can also get

\[
\frac{\partial^2 f_1}{\partial (x_{DC})^2} = 4|S|
\]

\[
\frac{\partial^2 f_1}{\partial (y_{DC})^2} = 4|S|
\]

and \( \frac{\partial^2 f_1}{\partial x_{DC} \partial \theta} = \frac{\partial^2 f_1}{\partial y_{DC} \partial \theta} = 0 \), \( \frac{\partial^2 f_1}{\partial \alpha_{DC} \partial \theta} = 0 \).

Define \( H_{f_1} \) as the Hessian matrix \( H \) of function \( f_1(x_{DC}, y_{DC}, \theta) \),

\[
H_{f_1} = \begin{bmatrix}
\frac{\partial^2 f_1}{\partial (x_{DC})^2} & \frac{\partial^2 f_1}{\partial x_{DC} \partial y_{DC}} & \frac{\partial^2 f_1}{\partial x_{DC} \partial \theta} \\
\frac{\partial^2 f_1}{\partial y_{DC} \partial x_{DC}} & \frac{\partial^2 f_1}{\partial y_{DC} \partial y_{DC}} & \frac{\partial^2 f_1}{\partial y_{DC} \partial \theta} \\
\frac{\partial^2 f_1}{\partial \theta \partial x_{DC}} & \frac{\partial^2 f_1}{\partial \theta \partial y_{DC}} & \frac{\partial^2 f_1}{\partial \theta \partial \theta}
\end{bmatrix}
\]

and the leading principal minors of \( H_{f_1} \) at \( (x_{DC}^*, y_{DC}^*, \theta^*) \):

\[
H_1 = \frac{\partial^2 f_1}{\partial (x_{DC})^2} = 4|S| > 0
\]

\[
H_2 = \frac{\partial^2 f_1}{\partial y_{DC} \partial y_{DC}} = 16|S|^2 > 0
\]

\[
H_3 = \frac{\partial^2 f_1}{\partial \theta \partial \theta} = 16|S|^2 (2l_1 \sqrt{a^2 + b^2}) > 0
\]

So \( (x_{DC}^*, y_{DC}^*, \theta^*) \) is a local minimum point for \( f_1(x_{DC}, y_{DC}, \theta) \).

\[\square\]

**APPENDIX C**

**PROOF OF PROPOSITION 2**

**Proof of Proposition 2:** Similarly, define \( a_i = \sum_s (x_{1s}^{IS} - x_{1s}^{JS}) \) and \( b_i = -\sum_s (y_{1s}^{IS} - y_{1s}^{JS}) \) and

\[
\gamma_i = \frac{b_i}{\sqrt{b_i^2 + a_i^2}} \arccos \frac{a_i}{\sqrt{a_i^2 + b_i^2}}
\]

(44)

when \( b_i = 0 \),

\[
\gamma_i = \arccos \frac{a_i}{\sqrt{a_i^2 + b_i^2}}
\]

(45)

The partial derivatives are:

\[
\frac{\partial f}{\partial x_{DC}} = 4|S| x_{DC} - 2 \sum_s (x_{1s}^{IS} + x_{1s}^{JS})
\]

(46)

\[
\frac{\partial f}{\partial y_{DC}} = 4|S| y_{DC} - 2 \sum_s (y_{1s}^{IS} + y_{1s}^{JS})
\]

(47)

\[
\frac{\partial f}{\partial \theta} = \sum_s \left(2l_1 \sqrt{a_i^2 + b_i^2} \sin(\theta + \alpha_i - \gamma_i) \right)
\]

(48)

let \( \frac{\partial f}{\partial \theta} = 0 \) we can obtain \( \theta^* = n\pi - \beta_i \)
From equation (48) we have
\[
\begin{align*}
\frac{\partial^2 f}{\partial \theta^2} &= \sum_{i=1}^{l} \left( \sqrt{a_i^2 + b_i^2} \cos(\theta + \alpha_i - \gamma_i) \right) \\
\end{align*}
\] (49)
and using \( A_i, \beta_i, B_i, \mu_i \) to substitute terms in the equation, we have
\[
\frac{\partial^2 f}{\partial \theta^2} = A_1 \cos(\theta + \beta_1) + B_1 \cos(\theta + \mu_1) + \sum_{i=3}^{l} \left( \sqrt{a_i^2 + b_i^2} \cos(\theta + \alpha_i - \gamma_i) \right)
\]
\[
= \sqrt{(A_1 \cos \beta_1 + B_1 \cos \mu_1)^2 + (A_1 \sin \beta_1 + B_1 \sin \mu_1)^2} \\
\cos \left( \theta + \arctan \left( \frac{A_1 \sin \beta_1 + B_1 \sin \mu_1}{A_1 \cos \beta_1 + B_1 \cos \mu_1} \right) \right) \\
+ \sum_{i=3}^{l} \left( \sqrt{a_i^2 + b_i^2} \cos(\theta + \alpha_i - \gamma_i) \right) \\
= A_2 \cos(\theta + \beta_2) + B_2 \cos(\theta + \mu_2) + \sum_{i=3}^{l} \left( \sqrt{a_i^2 + b_i^2} \cos(\theta + \alpha_i - \gamma_i) \right) \\
= \ldots \\
= A_i \cos(\theta + \beta_i)
\] (50)
Substitute \( \theta^* \) for \( \theta \) in the equation, we can obtain:
\[
\frac{\partial^2 f}{\partial \theta^2} = A_i \cos(n \pi - \beta_i + \beta_i) = A_i \cos n \pi
\] (51)
When \( n = 0, 2, 4 \ldots \frac{\partial^2 f}{\partial \theta^2} > 0 \). Similarly, we choose \( n = 0 \) arbitrarily.

let \( \frac{\partial f}{\partial x_{DCe}} = 0, \frac{\partial f}{\partial y_{DCe}} = 0 \). We can obtain
\[
\begin{align*}
&\quad x_{DCe} = \sum_{i=1}^{l} \left( x_{IS} + x_{IS} \right) / (2 |I||S|) \\
&\quad y_{DCe} = \sum_{i=1}^{l} \left( y_{IS} + y_{IS} \right) / (2 |I||S|) \\
&\quad \theta^* = n \pi - \beta_i
\end{align*}
\]
And we can take the second partial derivative of \( x_{DC} \) and \( y_{DC} \):
\[
\frac{\partial^2 f}{\partial (x_{DC}^2)} = 4 |I||S| > 0 \\
\frac{\partial^2 f}{\partial (y_{DC}^2)} = 4 |I||S| > 0
\]
The proof that \((x_{DCe}^*, y_{DCe}^*, \theta^*)\) is a local minimum point for \( f(x_{DC}^*, y_{DC}^*, \theta) \) in which \( i \in I \) is similar to the proof of \( i = 1 \) condition. \( \square \)

REFERENCES


ZHENYU ZHOU received the M.S. degree in industrial engineering from Clemson University. He is currently pursuing the Ph.D. degree in industrial engineering with the Department of Industrial and Systems Engineering, Wayne State University (WSU). His current research interests include developing optimization models and algorithms for UAV fleet routing problems. His technical expertise broadly covers operations research, dynamic programming, heuristic, and machine learning methods.

YANCHAO LIU received the B.S. degree in industrial engineering (IE) from the Huazhong University of Science and Technology, in 2006, the M.S. degree in IE from the University of Arkansas, in 2008, and the Ph.D. degree in industrial and systems engineering from the University of Wisconsin-Madison, in 2014. He is currently an Assistant Professor with the Department of Industrial and Systems Engineering, Wayne State University. Prior to joining Wayne State University in 2017, he was a Data Scientist and the Manager of advanced analytics at Sears Holdings Corporation and subsequently the Director of brand marketing analytics at Catalina Marketing Corporation. His research interests include developing novel optimization models to solve pressing transportation problems, including dynamic aerial vehicle routing, automated air traffic control for UAV fleets, and resource planning and management for advanced air mobility systems. He was a recipient of the U.S. National Science Foundation (NSF) Faculty Early Career Development Award, in 2020.