

N-version programming approach with implicit safety guarantee for complex dynamic system stabilization applications

Measurement and Control
2021, Vol. 54(3-4) 269–278
© The Author(s) 2020
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/0020294019887473
journals.sagepub.com/home/mac


Nadir Subasi¹ , Ufuk Guner² and Ilker Ustoglu³

Abstract

Safety-critical systems are widely used in many sectors to prevent fatal accidents and prevent loss of life, damage of property, or deterioration of the environment. Implementation of software safety standards as part of the development of safety-critical software is generally considered an essential element of any safety program. Therefore, it has become more critical to produce highly reliable software to meet the safety requirements established by functional safety standards, such as IEC 61508, ISO 26262, and EN 50128. IEC 61508 supports well-known safety mechanisms such as design diversity like N-version (multi-version) programming. N-version (multi-version) programming is a method where multiple functionally equivalent programs are independently developed from the same software specifications. N-version (multi-version) programming is particularly an effective approach to increase the quality of software in a safety-critical system. In this paper, one of the well-known and widely used algorithms in the field of N-version (multi-version) programming, the majority voting algorithm, has been modified with an online stability checker where the decisions of the voter are judged against the stability of the underlying system. The plant where all the theoretical results are implemented is a tilt-rotor system with the proposed N-version (multi-version) programming-based controller. The experimental results show that the modified majority voter-based N-version (multi-version) programming controller provides more reliable control of the plant.

Keywords

Safety-critical software, fault-tolerant systems, N-version programming, proportional control, dual tilt-rotor system

Date received: 1 August 2019; accepted: 17 October 2019

Introduction

Design of safety-critical systems is of particular importance in processes which might cause loss of life, injuries, or environmental damage. The software which is used in sectors such as aviation, railway, nuclear, and machine automation also must be safety-critical. Industry-specific safety standards that reside with IEC-61508 (The International Electrotechnical Commission) umbrella standard direct how safety-critical processes should be managed. N-version (multi-version) programming (NVP) that uses multiple different versions of the same software to satisfy the need for variation in software design is one of the methods recommended in these standards.

In the literature, the successful applications of the NVP technique include space,^{1,2} railway signaling systems,³ message transmission systems,⁴ e-voting,⁵ plagiarism detection algorithms,⁶ and network services.^{7,8} In addition, the software requirements in the N-version

programming technique are described in the literature.^{9–12} These studies have shed light on the results that the software to be developed should work in different software development environments by using different software languages by different working groups. The NVP method suggests that errors in functionally equivalent modules can occur at various points, so errors can be detected and actual results can be obtained.¹³

¹Department of Computer Programming, Kırklareli University, Kırklareli, Turkey

²Department of Electrical and Electronics Engineering, Erzurum Technical University, Erzurum, Turkey

³Department of Control and Automation Engineering, Istanbul Technical University, Istanbul, Turkey

Corresponding author:

Nadir Subasi, Department of Computer Programming, Kırklareli University, Kırklareli 39100, Turkey.

Email: nadir.subasi@klu.edu.tr



Creative Commons CC BY: This article is distributed under the terms of the Creative Commons Attribution 4.0 License (<https://creativecommons.org/licenses/by/4.0/>) which permits any use, reproduction and distribution of the work without

further permission provided the original work is attributed as specified on the SAGE and Open Access pages (<https://us.sagepub.com/en-us/nam/open-access-at-sage>).

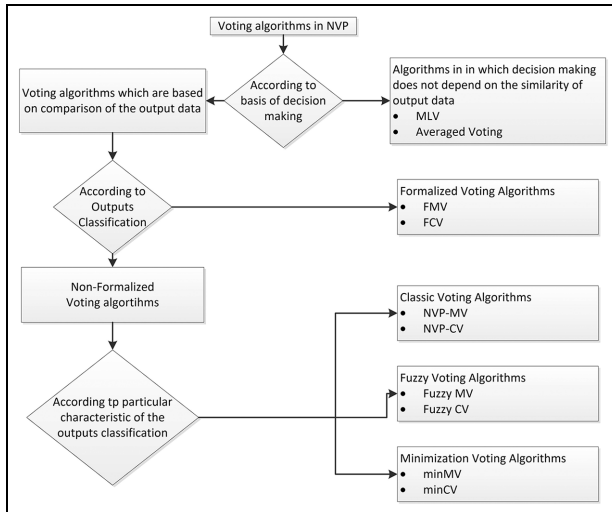


Figure 1. Voting algorithms in NVP.

The most significant benefit of this approach is to maintain software error tolerance.¹⁴ In the event of any version failure, the remaining active versions will generate the desired output, and the system will keep running. In this respect, the regular operation of the system, generated during the software development and testing, is insured against unpredictable errors.^{15,16} The use of the NVP method, along with the available test methods and program accuracy, guarantees a high level of software reliability.^{17,18}

The commonly used algorithms for voting differ in the requirements management of the original data and voting schemes.^{19–24} Some of the algorithms depending on the version given by the data set may be ineffective. The voting algorithms are mainly classified into two categories: voting algorithms established by the output data comparison and voting algorithms where the decision making does not rely on the likeness of the output. Voting algorithms that are based on output data comparison are divided further into two categories, such as formalized and non-formalized algorithms. Note that, when the outputs of multiple versions are compared, the approach of equivalent outputs is used. Thus, for example, if two outputs are in the neighborhood of a fixed number called the tolerance value, the outputs are said to be equivalent. As a rule, the equivalent output is considered as the correct output. Here, selecting the correct output set for the versions is done using subsets of the approved versions or using the so-called agreement matrix. The classification of the voting algorithms applied in NVP method is shown in Figure 1 and it reveals that these algorithms depend on the decision-making principle, classification of the output data, and individual classification characteristics of the output data.²⁵ A list that suits this classification is given below:

1. Absolute majority voting (MV) algorithm (N-version programming with majority voting; NVP-MV);

2. Consensus voting (CV) algorithm (NVP with consensus voting, NVP-CV);
3. Fuzzy MV;
4. Fuzzy CV;
5. Absolute MV algorithm with minimization (minMV);
6. CV algorithm with minimization (minCV);
7. Formalized MV (FMV);
8. Formalized CV (FCV);
9. Maximum likelihood voting;
10. Averaged voting.

This study explains how to use NVP in a new way. With NVP, several versions of the same controller will be used for the next action. However, the majority voter can vote to put the system in an unstable configuration (e.g. it could cause an unmanned aerial vehicle (UAV) crash). This study allows the NVP framework to select the input from the minority, which will still result in a stable system, by combining the NVP with an instability detector that marks such inputs as invalid.

NVP-MV algorithm is explained in detail. For a real-time experiment, a tilt-rotor stabilization platform is built, and here, the mathematical model of the system is given. The system has 3 degrees of freedom. Therefore, it can freely move around the roll, pitch, and yaw axes. The platform has proportional–integral–derivative (PID) controllers for each rotation axes. Without loss of generality, the NVP-MV structure is implemented on only roll and pitch controllers. Furthermore, the NVP-MV algorithm is modified by adding a stability checking feature to the system. Experimental results and concluding remarks are discussed at the end of the paper.

N-version programming

The voting algorithms presented in NVP problems are different in dependency on the initial data and the work program. It is crucial to select the most appropriate voting algorithm for a data set. However, the implementation of such algorithms, which require the division of data into subsets of items, is equivalent to each other.^{1,9,26}

In the NVP technique, the architecture consists of N program versions V_j that are independently designed as given in:

$$(V_1, V_2, \dots, V_{\lceil N/2 \rceil + 1}, \dots, V_{N-1}, V_N) \quad (1)$$

The output of the NVP algorithm is considered to be reliable if at least $(\lceil N/2 \rceil + 1)$ versions agree on the same output.²⁷ This is demonstrated in Figure 2.

The agreement matrix for NVP-MV

The most critical point in choosing the right set of output is based on the creation and analysis of the so-called agreement matrix R . It is an $N \times N$ Boolean matrix

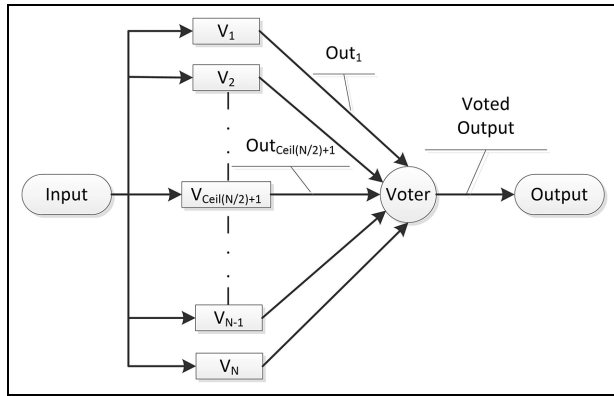


Figure 2. Basic diagram of NVP-MV.

where N is the number of versions, and it reflects the equivalence among outputs. The elements of R are determined as follows

$$r_{ij} = \begin{cases} 1, & |x_i - x_j| \leq \varepsilon \\ 0, & |x_i - x_j| > \varepsilon \end{cases} \quad (2)$$

where i and j indicate the rows and columns of R , respectively, x_i and x_j denote the outputs of versions, and ε is the tolerance threshold.

The following additional terms apply to the agreement matrix R . Equivalence relation on R shall be provided, including reflection (equation (3)), symmetry (equation (4)), and transition (equation (5)) properties

$$r_{ii} = 1, \forall i \quad (3)$$

$$r_{ij} = r_{ji}, \forall i \neq j \quad (4)$$

$$\text{if } r_{ik} = 1 \text{ and } r_{kj} = 1 \text{ then } r_{ij} = 1, \forall i, j \quad (5)$$

The purpose of the Boolean compositions on R is to convert it into a suitable form in which the equivalence relationship holds. Overall, studies of the composition for Boolean matrices are defined as follows.

Given two matrices A and B ; where their entries take values 0 or 1, then the Boolean composition of matrices A and B is as follows

$$C = A \circ B, \text{ where } c_{ij} = \bigoplus_{k=1}^N (a_{ik} \otimes b_{kj}) \quad (6)$$

where \oplus and \otimes represent the Boolean OR and AND operations, respectively. For the fulfillment of the equivalence relationship (3)–(5) on the agreement matrix R , the application of the Boolean compositions of R should be carried out in conjunction with the following principle

$$E = R^1 \cup R^2 \cup \dots \cup R^Q, \quad 1 \leq Q \leq N-1 \quad (7)$$

where Q is the number of results of Boolean composition and N is the number of versions. Thus, if the result is not satisfactory, then using

$$E^2 = R \cup R^{\circ} R \quad (8)$$

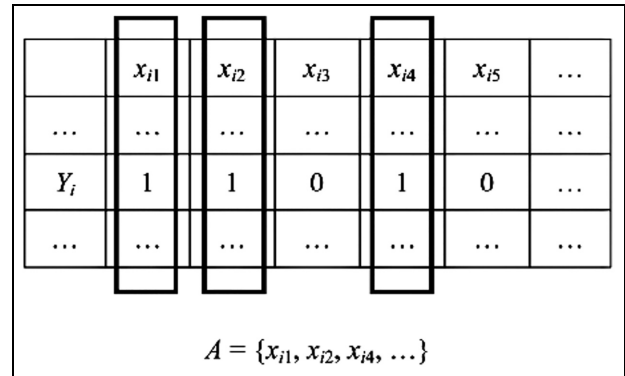


Figure 3. Selection of correct answers from R .

the Boolean combination can be rearranged. If the result of equation (8) is still not satisfactory, then the following Boolean combination can be used

$$E^3 = R \cup R^{\circ} R \cup R^{\circ} R^{\circ} R = E^2 \cup R^3 \quad (9)$$

The NVP-MV algorithm

Assume that each one of N versions is independent and the output values generated by each version are specified by x_1, x_2, \dots, x_N . After choosing the tolerance value ε , the steps of the algorithm are applied:

- Step 1. Build the agreement matrix R using equation (2).
- Step 2. Analyze the equivalence relation on R under the conditions (3)–(5). If it holds, go to Step 4, else, go to Step 3.
- Step 3. Equation (7) is carried out until the equivalence ratio (3)–(5) for R does not hold.
- Step 4. The correct output set shall be defined. In each row of R , the number of elements is determined. Y_i shows the number of elements in row i . If there is such row i with

$$Y_i \geq \left\lceil \frac{N+1}{2} \right\rceil \quad (10)$$

then the list of correct results is created from the corresponding units in row i . Here, $\lceil \cdot \rceil$ in equation (10) denotes the ceiling operator.

Figure 3 shows the principle how the results of the versions are selected, with A being the set of correct results.

The tilt-rotor system and the controller structure

The mechanical structure of the system has two main parts. One is a fixed carrier, and the other is dual tilt-rotor system which is mounted on the fixed carrier. The tilt-rotor frame can be freely rotated about three orthogonal axes according to the limitation of the

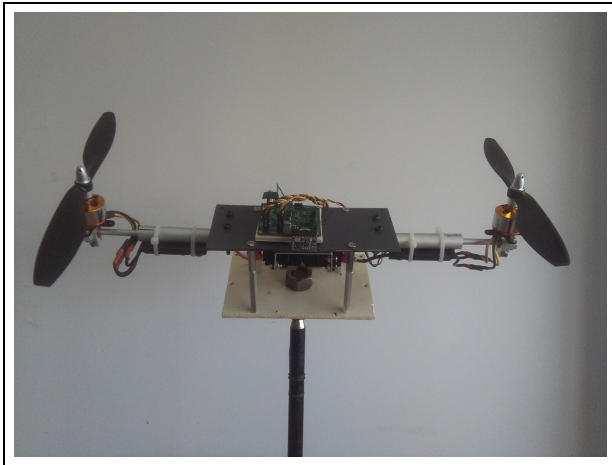


Figure 4. The test system.

platform. Therefore, the system is 3 degrees of freedom and movable on the roll, pitch, and yaw axes. The tilt system contains brushless direct current (BLDC) and servo motors as actuators.²⁸ The servos are responsible for yaw and pitch torques, and the BLDC motors are used for roll control. Figure 4 shows the system under control. The mathematical model of the system is nonlinear. In this study, a linear system approximation is performed, which makes the controller design much more comfortable.²⁹ The controller is chosen to be a PID controller. Any hardware failure of the PID controller causes undesired control signals which will affect the performance or even the stability of the plant. To overcome this problem, an NVP-MV-based structure is considered.

Mathematical model

In order to stabilize the system, the roll, pitch, and yaw torques are used. For roll control of the system, the BLDC motor speed difference is used. The servos provide pitch and yaw torques with tilting the BLDC motors and changing the resultant thrust force. For modeling, frames of the platform are defined as follows: the tilt-rotor part is the inertial frame, and the fixed carrier is the body frame. Besides, equation (11) denotes coordinate of the inertial and body frame

$$\begin{aligned} O_I &= \{x_I, y_I, z_I\} \\ O_B &= \{x_B, y_B, z_B\} \end{aligned} \quad (11)$$

Because of the tilt mechanism, the BLDC motors have their own frame. The counter tilting causes yaw torque and represents O_{y1} and O_{y2} . The parallel tilting, which is denoted by O_p , produces pitch torque

$$\begin{aligned} O_{y1} &= \{x_{y1}, y_{y1}, z_{y1}\} \\ O_{y2} &= \{x_{y2}, y_{y2}, z_{y2}\} \\ O_p &= \{x_p, y_p, z_p\} \end{aligned} \quad (12)$$

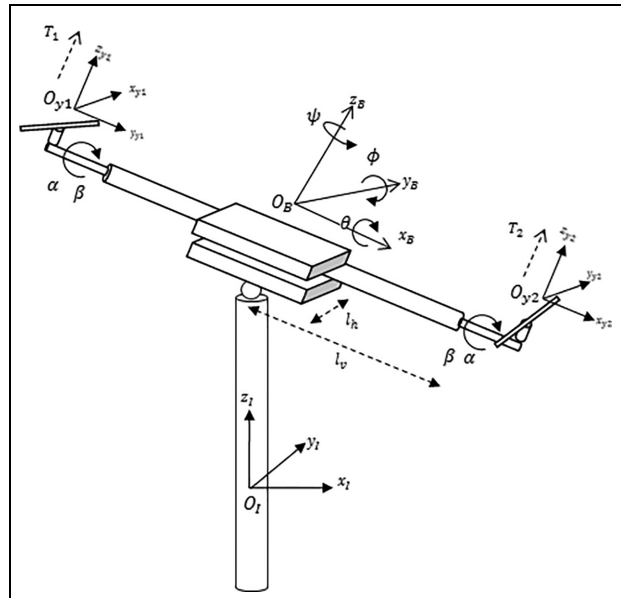


Figure 5. Inertial and body frame of the platform.

The detailed description of system axes can be seen in Figure 5. α , β denote the counter and the parallel tilting angles, respectively.

l_v , l_h are the transverse and the longitudinal distances from stabilization point, respectively. The trust forces, produced by propellers, are represented with T_1 and T_2 . Note that the propellers generate the main trust force. The parallel and counter tilting produce the pitch and the yaw torques. The roll torque is obtained from the trust difference of the rotors. The rotational displacement is defined as $\xi = \{\theta, \phi, \psi\}$. The nonlinear rotational dynamics of the system can be obtained using the Newton–Euler method

$$\vec{\tau} = I\dot{\vec{\Omega}} + \vec{\Omega} \times (I\vec{\Omega}) \quad (13)$$

In the equation, $\vec{\tau}$ is total torque vector, I denotes inertia matrix, and the $\vec{\Omega}$ is angular velocity vector. The total torque contains gyroscopic, trust, and weight torques

$$\vec{\tau} = \vec{\tau}_c + \vec{\tau}_g + \vec{\tau}_w \quad (14)$$

However, in order to reduce the model, the gyroscopic torques, which are produced by tilting, are disregarded. Because of rotational dynamics, the necessary transformation matrices are defined as

$$R^{B \rightarrow I} = \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix}$$

$R^{B \rightarrow I}$ is defined for body to inertial frame transformation, and $c = \cos$ and $s = \sin$

$$R^{y_1 \rightarrow B} = \begin{bmatrix} c_\alpha & 0 & s_\alpha \\ 0 & 1 & 0 \\ -s_\alpha & 0 & c_\alpha \end{bmatrix}, \quad R^{y_2 \rightarrow B} = \begin{bmatrix} c_\alpha & 0 & -s_\alpha \\ 0 & 1 & 0 \\ s_\alpha & 0 & c_\alpha \end{bmatrix}$$

$R^{y_1 \rightarrow B}$, $R^{y_2 \rightarrow B}$ represent the counter tilt effect of rotors. The transformation matrix of rotors pitch change to body frame is defined as

$$R^{p \rightarrow B} = \begin{bmatrix} c_\beta & 0 & -s_\beta \\ 0 & 1 & 0 \\ s_\beta & 0 & c_\beta \end{bmatrix}$$

In this context, using the transformation matrices, the force on the center of the body frame can be defined as

$$\begin{cases} T_1^B = R^{y_1 \rightarrow B} R^{p \rightarrow B} T_1 \\ T_2^B = R^{y_2 \rightarrow B} R^{p \rightarrow B} T_2 \end{cases} \quad (15)$$

Now, let define the actuator torque

$$\tau_c = l_{y_1}^B \times T_1^B + l_{y_2}^B \times T_2^B \quad (16)$$

In the equation, $l_{y_1}^B$, $l_{y_2}^B$ represent distances from the stabilization point: $l_{y_1}^B = [l_h, -l_v, 0]^T$, $l_{y_2}^B = [-l_h, -l_v, 0]^T$.

The weight torque is provided by the center of gravity distance on the body frame and defined as

$$\tau_w = l_w^B \times R^{I \rightarrow B} m G^I \quad (17)$$

where l_w^B is distance of center of gravity from stabilization point and defined as $l_w^B = [0, -l_v, 0]^T$. m is the mass of the body frame and G^I is the gravity vector according to the inertial frame.

So, deriving equations (13), (16), and (17), nonlinear dynamic of the system can be modeled with following equations

$$\begin{aligned} I_{xx} \ddot{\phi} &= \dot{\theta} \dot{\psi} (I_{yy} - I_{zz}) - (T_1 + T_2) l_h \\ &\quad s_\beta s_\alpha - (T_1 + T_2) l_h c_\beta c_\alpha \\ I_{yy} \ddot{\theta} &= \dot{\phi} \dot{\psi} (I_{zz} - I_{xx}) - (T_1 + T_2) l_v \\ &\quad c_\alpha c_\beta - (T_1 + T_2) l_v s_\beta s_\alpha \\ &\quad + c_\psi c_\theta l_v m g \\ I_{zz} \ddot{\psi} &= \dot{\theta} \dot{\phi} (I_{xx} - I_{yy}) - (T_1 + T_2) l_v \\ &\quad s_\beta c_\alpha - (T_1 - T_2) l_v s_\alpha c_\beta \\ &\quad + s_\theta l_v m g \end{aligned} \quad (18)$$

In order to obtain a linear model around the equilibrium point, a linear approximation is applied to the dynamic equations. The roll, pitch, and yaw displacement, and velocities are all equal to zero. So, three sub-systems can be defined to provide linear equations. For roll equilibrium, we have $\alpha = \beta = \theta = \psi = 0$, and the roll equation is simplified as

$$I_{xx} \ddot{\phi} = \dot{\theta} \dot{\psi} (I_{yy} - I_{zz}) - (T_1 - T_2) l_h \quad (19)$$

where if control signal is defined as $u_1 = (T_1 - T_2)$ and for small deviations of $\delta\phi$, $\delta\dot{\phi}$, the linear approximation of roll dynamic can be defined as

$$I_{xx} \delta \ddot{\phi} = -u_1 l_h \quad (20)$$

For the pitch dynamics, assuming $\alpha = \psi = \phi = 0$ for the equilibrium, the pitch equation takes following form

$$I_{yy} \ddot{\theta} = \dot{\phi} \dot{\psi} (I_{zz} - I_{xx}) - (T_1 + T_2) l_v c_\beta + l_v m g c_\theta \quad (21)$$

Here, if the control signal is defined as $u_2 = (T_1 + T_2) c_\beta$ and for small deviation of $\delta\theta$, $\delta\dot{\theta}$, then the equation can be reduced to

$$I_{yy} \delta \ddot{\theta} = -u_2 l_v + l_v m g \delta\theta \quad (22)$$

For the yaw dynamics, assuming $\beta = \theta = \phi = 0$ for the equilibrium, the yaw equation can be defined as

$$I_{zz} \ddot{\psi} = \dot{\theta} \dot{\phi} (I_{xx} - I_{yy}) - (T_1 + T_2) l_v s_\alpha \quad (23)$$

where, if the control signal is defined as $u_3 = (T_1 + T_2) s_\alpha$ and for small deviation of $\delta\psi$, $\delta\dot{\psi}$, the equation can be reduced to

$$I_{zz} \delta \ddot{\psi} = -u_3 l_v \quad (24)$$

Defining the states of the system

$$x_1 = \delta\theta \quad x_2 = \delta\dot{\theta} \quad x_3 = \delta\dot{\psi} \quad (25)$$

The simplified linear model of the system is as follows

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \quad (26)$$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} \frac{l_v m g}{I_{yy}} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} \frac{-l_h}{I_{yy}} & 0 & 0 \\ 0 & \frac{-l_v}{I_{xx}} & 0 \\ 0 & 0 & \frac{-l_v}{I_{zz}} \end{bmatrix} u \quad (27)$$

$$y = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} x$$

$$x_0 = [40^\circ (\text{Pitch}) \quad 0^\circ (\text{Roll}) \quad 0^\circ (\text{Yaw})] \quad (28)$$

where system parameters are shown in Table 1. The initial states are given below

The linearized model shows that the system can be regulated using low-order controllers such as the PID

Table 1. Parameter values of the test environment.

Parameters	Values
I_{xx}	$32 \times 10^{-3} \text{ kg m}^2$
I_{yy}	$102 \times 10^{-3} \text{ kg m}^2$
I_{zz}	$72 \times 10^{-3} \text{ kg m}^2$
m	$134 \times 10^{-3} \text{ kg}$
l_h	$30 \times 10^{-3} \text{ kg m}^2$
l_v	$90 \times 10^{-3} \text{ kg m}^2$
g	9.8 m/s^2

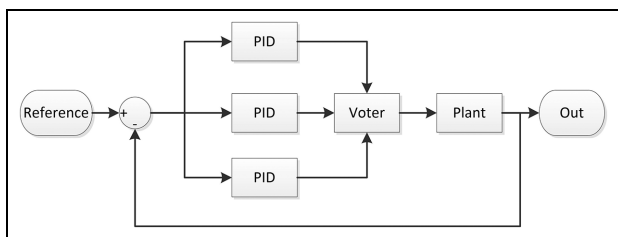


Figure 6. NVP-MV PID controlled system.

controller. The controller transfer function is given in equation (29)

$$PID = P + I\frac{1}{s} + D\frac{N}{1 + N\frac{1}{s}} \quad (29)$$

Controller structure

This section will present a real-time controller design procedure for a tilt-rotor UAV based on a modified NVP-MV algorithm. Without loss of generality, we choose the PID controller to meet satisfactory performance and closed-loop stability. Generally, NVP-MV algorithm is a 2 out of 3 structure, which means that, if two versions agree, majority voter takes this decision as the correct output. This general approach is demonstrated in Figure 6.

In this study, we present an algorithm that makes the voter more intelligent in the sense of detecting stabilizing decisions of the controllers. In this modified voter design, we have implemented an instability detector and a memory which stores the previous decision. So, this type of NVP-MV voter knows whether the decision stabilizes the system or not. A basic block diagram of the novel NVP-MV is demonstrated in Figure 7.

Instability detector needs system output value, system states, and reference of the controlled system. The detector output which is the input of the voter is 0 (False) when the system is stable. On the other hand, when the system’s output diverges (unstable), the detector’s output is 1 (True).

Wang et al.³⁰ proposed that an online Lyapunov stability analysis feature can be integrated to the architecture to achieve a safety-critical controller. This idea influenced the authors of this paper to modify the voter with such a feature, which they call the instability detector.

For input-to-output stability, both the storage and supply functions have to be constructed. Figure 8 shows the principle for L_2 gain stability of tilt-rotor system where

$$\dot{V}(x) - \alpha^2 w^T w + y^T y \leq 0 \quad (30)$$

holds, with w being the unit-peak uniform noise and y being the output of the system. Here, $x^T P x$ represents the (quadratic Lyapunov-like) storage function. The choice of the P matrix and alpha is not straightforward

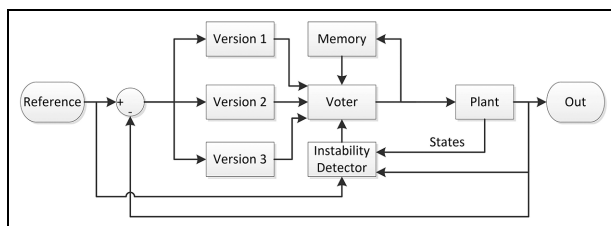


Figure 7. Diagram of designed voter.

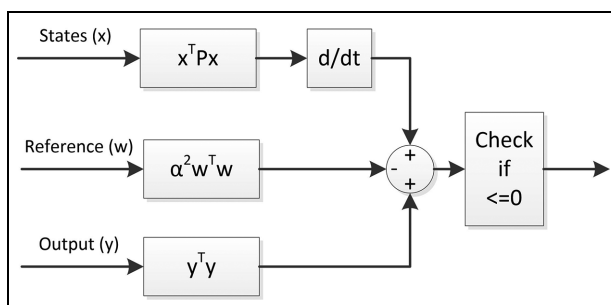


Figure 8. Instability detector.

and requires some effort. In this study, the appropriate P matrix and α value is chosen after some domain knowledge and trial-and-error like simulations.

The proposed modified NVP-MV algorithm is shown in Figure 9.

Experimental results

The performance of the proposed modified NVP-MV-based PID controller has been assessed by simulations executed on a tilt-rotor system. This section describes the simulation scenarios and the design of high availability PID controllers. As the simulation environment, MATLAB R2017b Simulink has been used, which is based on real-time behavior and the mathematical model of the system. In the simulation scenario, the system has got three PID controllers for each two states: roll and pitch (yaw behavior is neglected). Each PID controller parameter is calculated using the Ziegler–Nichols method. Recall that the simulation results are not plotted here, because the real-time experimental results are given at the end of this section. The simulation duration is chosen to be 40 s, and every 5 s, one or more of the PID parameters are replaced with such values that make the system unstable. The reason why 5-s intervals are chosen is that the settling time of the system is 3 s for stabilizing controller sets. For PID parameters which make the system stable, the health of PID is defined as 1 (True). Otherwise, PID parameters leading to instability of the system are defined as 0 (False). Table 2 and Figure 10 give information about the simulation details of the scenario, where Figure 10 shows a Markov Diagram to explain the possible states and transitions. Here, common cause effects, the effect that

Table 2. Simulation scenario.

States	Stable			Instability Detector	NVP-MV Decision	Our designed Voter decision	Interval (s)
	V1	V2	V3				
S0	1	1	1	0	1-2-3	1-2-3	0-5
S1	1	1	0	0	1-2	1-2	5-10
S2	1	0	1	0	1-3	1-3	10-15
S3	0	1	1	0	2-3	2-3	15-20
S4	1	0	0	1	2-3	1	20-25
S5	0	1	0	1	1-3	2	25-30
S6	0	0	1	1	1-2	3	30-35
S7	0	0	0	1	1-2-3	SF	35-40

NVP-MV: N-version programming-multi-version; SF: safety function.

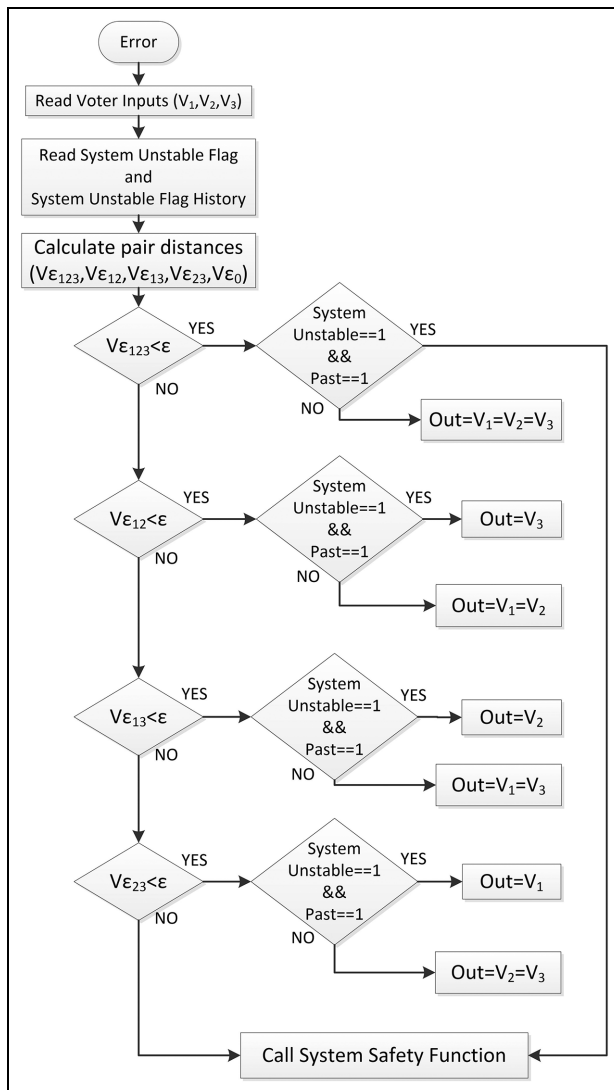


Figure 9. Algorithm of designed voter.

two or three controllers fail at the same time due to a common cause, are neglected.

State S0 indicates that all versions stabilize the system, and $V_{\epsilon 123} < \epsilon$, meaning that all versions are within ϵ -neighborhood which is a sufficiently small number,

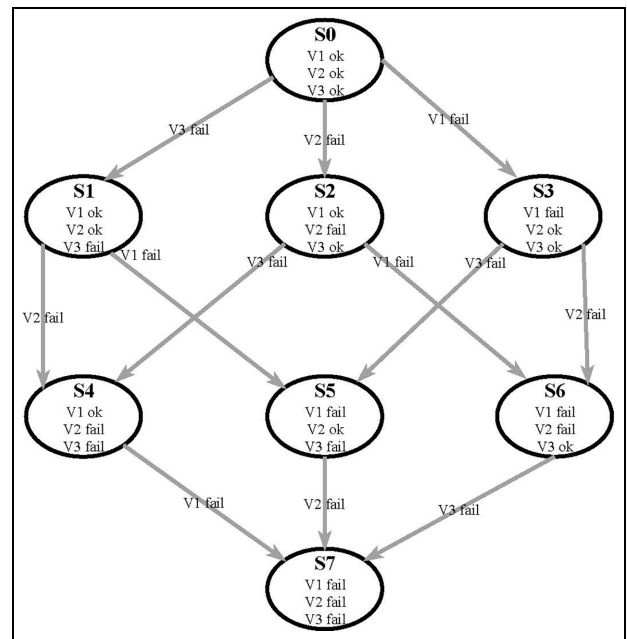


Figure 10. Markov diagram of system.

the instability detector's output is False. On the other hand, for example, state S4 (which is valid between 20th and 25th second) tells us that only Version 1 (PID 1) produces a stabilizing control, Version 2 (PID 2) and Version 3 (PID 3) make the system unstable; however, since they are the majority, NVP-MV chooses the output of Version 2 and Version 3. Our modified NVP-MV immediately switches to the minority's decision, which is Version 1. The voter changes the final decision within the next sampling time: 0.01 s. According to the scenario starting at 25th second and ending at 30th second (State S5), the controller parameters of Version 1 and Version 3 do not stabilize the system while Version 2 results in a stable closed-loop system. The modified voter identifies this problem immediately and switches to the stabilizing minority decision. Finally, if we analyze the last state, S7, we conclude that all versions cause instability, so a safety function (SF)³¹ shall be called.

Table 3. PID parameters.

			Stable value	Unstable value
Roll	(P)	Proportional	3.2	0
	(I)	Integral	0.9	500
	(D)	Derivative	0.4	0
Pitch	(P)	Proportional	3.5	0
	(I)	Integral	1.2	500
	(D)	Derivative	0.6	0

PID: proportional–integral–derivative.

For the real-time experiment, a controller board is build, and a microprocessor is used to implement NVP-MV algorithm. For inertial measurement, 9-degree-of-freedom (DOF) sensor board is added to the controller board. The sensor board has three axes gyroscope, accelerometer, and magnetometer for measuring inertial variations along these axes. The sensor fusion algorithm and the filter are also implemented to increase the reliability of sensor data. In the platform, the servos and BLDCs are controlled by pulse width modulation (PWM) signals.

In the experiment, BLDC's starting PWM value is $1200\ \mu\text{s}$ and the controllable trust range is defined within 1280 and $1380\ \mu\text{s}$ intervals.

The $1280\ \mu\text{s}$ PWM value is representing the base trust for pitch moving of the platform. Therefore, the PID output of pitch control is set at 0 to 100 intervals. In the same manner, the roll PID output range is settled for -20 to 20 . The servos, in the test platform, are settled at its PWM midpoint ($1800\ \mu\text{s}$) for vertical position of BLDCs. The servo PWM operation interval is defined as $-50\ \mu\text{s}$ to $+50\ \mu\text{s}$ from the midpoint. In this way servos provide ± 5 degree tilt angle change for BLDCs. In addition, PID controllers have dead band around equilibrium points.

The NVP-MV algorithm is implemented for roll and pitch PID controllers. Both PID controllers are

simultaneously examined with NVP-MV algorithm. Platform stabilization point is arranged as roll and pitch angles equal to zero. Therefore, the PID controller's desired reference value is also set to zero for roll and pitch. Initially, the system is aligned with zero roll and approximately -40° pitch angles.

In the experiment, three individual PID controllers, which have the same parameters, are designed for roll and pitch controls. PID controller parameters are determined. In addition, PID parameters which can lead to system instability are also determined using the same method (Table 3).

The control board has a frequency of $100\ \text{Hz}$ for reading the sensors and calculating the PID outputs. Therefore, the PWM signals of electronic speed controllers (ESC) and servos can be updated every $10\ \text{ms}$. Besides, all system parameters are monitored every $10\ \text{ms}$ over a serial interface. Figure 11(a) and (b) shows the roll and pitch response of the system and the corresponding control signals, respectively.

In Figure 11(a) and (b), the control signal outputs are correlated with PWM input of ESCs and servos. The system output is directly representing the roll and pitch angle of system. For roll control, the roll PID output is added and subtracted from corresponding PWM value of BLDCs. On the other hand, the pitch PID output multiplication with servos' tilt angle is added ESCs base PWM value, in order to provide necessary trust.

In Figure 11 for state S0 to S3, instability detector outputs are 0 (False), because always two controllers are producing a stabilizing control. S4 state tells us that only Version 1 (PID 1) produces a stabilizing control, and Version 2 (PID 2) and Version 3 (PID 3) make the system unstable; however, since they are the majority, NVP-MV chooses the output of Version 2 and Version 3. Our modified NVP-MV immediately switches to the minority's decision, which is Version 1. The voter changes the final decision within the next sampling

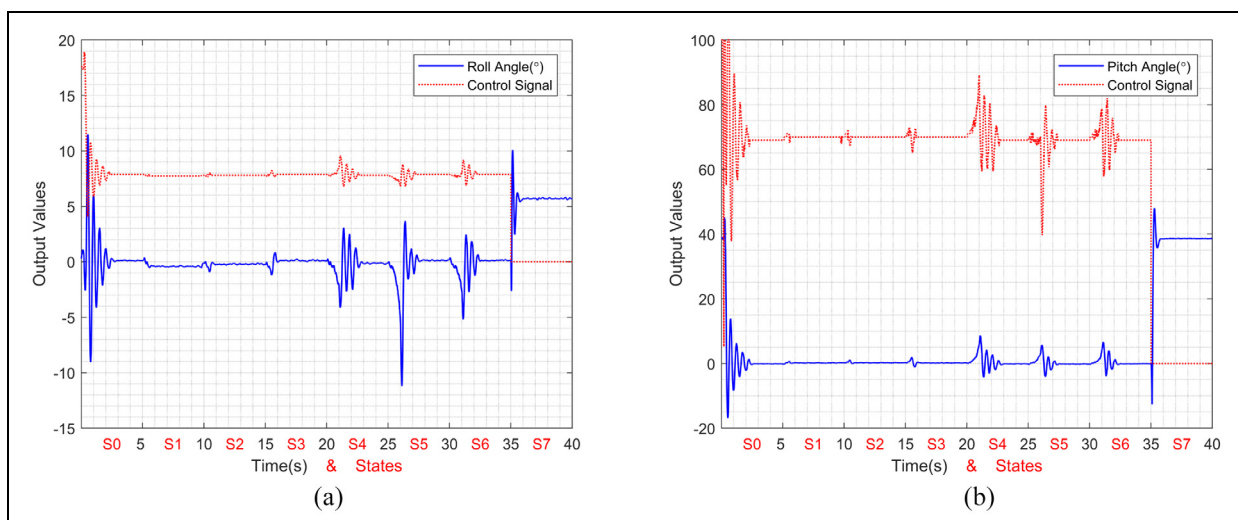


Figure 11. Roll–pitch experimental results: (a) roll response and (b) pitch response.

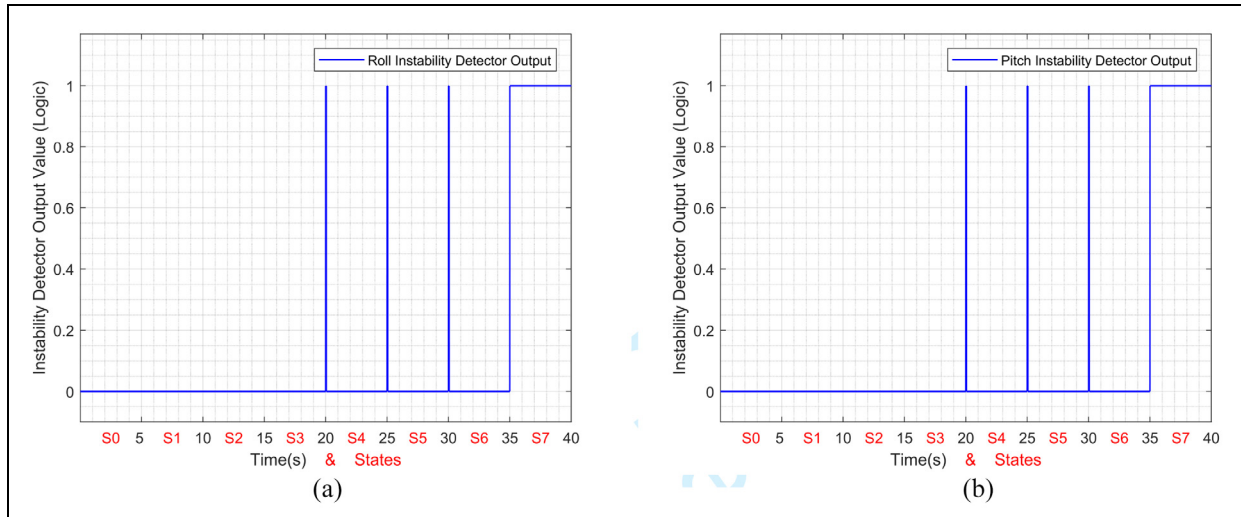


Figure 12. Roll–pitch instability detector outputs: (a) roll instability detector output and (b) pitch instability detector output.

time, which is 0.01 s. Like S4 state, S5 state only Version 2 (PID 2) and S6 state only Version 3 (PID 3) produce a stabilizing control signal which makes the system stable.

In Figure 12(a) and (b), Roll and Pitch instability detectors' outputs are plotted. For S0, S1, S2, and S3 states, instability detectors output cannot be true because majority voter chooses right pair of controllers. But in the S4, S5, and S6 states, majority voter cannot choose controller which makes the system stable. With the instability detector becoming true, voter changes decision with minority of controllers' output. If the system is in S7 state, all controller cannot produce a stabilizing control signal and instability detectors output is true. Then the system calls safety function.

Conclusion and future work

NVP-MV is an effective approach to improve the reliability of a software and it requires an accurate decision of correct and failed versions. In order to do so, using algorithms rating, the correct answer needs to be selected among the set of the plurality of calculation results. Furthermore, NVP-MV is a practical approach to enhance the quality of software for safety-critical applications. However, if the NVP-MV chooses a wrong decision, in other words, the majority is producing a faulty output, then this may lead to instability of the system. In this paper, the NVP-MV is modified in such a way that the voter checks the stability of the system and does not always allow the majority to win if they make the system unstable. The idea is demonstrated on an experimental setup, the tilt-rotor system, and the success of the proposed voter is shown. As a future work, we will study the modified fuzzy voting algorithms and modify the voter further with weighted inputs. Furthermore, we will investigate how the system

will benefit from multiple instability detectors where the decisions of instability detectors are also voted.

Acknowledgements

The authors would like to thank all the editors and anonymous reviewers for improving this article.


Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) received no financial support for the research, authorship, and/or publication of this article.

ORCID iD

Nadir Subasi  <https://orcid.org/0000-0002-5657-9002>

References

1. Avizienis A. A fault tolerance infrastructure for high-performance COTS-based computing in dependable space systems. In: *Proceedings of the 10th IEEE Pacific Rim international symposium on dependable computing*, Papeete, 3–5 March 2004, pp. 336–336. New York: IEEE.
2. Yatsynovich MN, Litvinko PA, Kovalev KA, et al. Evaluation of interference signal from the space allocated abonement stations. In: *Proceedings of the 2012 22nd international Crimean conference "microwave & telecommunication technology,"* Sevastopol, 10–14 September 2012. New York: IEEE.
3. Dincel E, Eris O and Kurtulan S. Automata-based railway signaling and interlocking system design. *IEEE Antenn Propag M* 2013; 55(4): 308–319.
4. Chitsaz B and Razzazi M. Non-blocking N-version programming for message passing systems. In: *2012*

- Proceedings of the 35th international convention MIPRO*, 21–25 May 2012. New York: IEEE.
5. Liburd SSD. *An N-version electronic voting system*. PhD Thesis, Massachusetts Institute of Technology, Cambridge, MA, 2004.
 6. Zhu S, Wu D, Jhi YC, et al. A first step towards algorithm plagiarism detection. In: *Proceedings of the 2012 international symposium on software testing and analysis*, Minneapolis, MN, 15–20 July 2012, pp. 111–121. New York: ACM.
 7. Aghaei S, Khayyambashi MR and Nematbakhsh MA. A fault tolerant architecture for Web services. In: *Proceedings of the 2011 international conference on innovations in information technology*, Abu Dhabi, United Arab Emirates, 25–27 April 2011, pp. 53–56. New York: IEEE.
 8. Nourani E. A new architecture for dependable web services using N-version programming. In: *Proceedings of the 2011 3rd international conference on computer research and development*, Shanghai, China, 11–13 March 2011, pp. 333–336. New York: IEEE.
 9. Xie M, Xiong C and Ng SH. A study of N-Version programming and its impact on software availability. *Int J Syst Sci* 2014; 45(10): 2145–2157.
 10. Peng KL, Huang CY, Wang PH, et al. Enhanced N-version programming and recovery block techniques for web service systems. In: *Proceedings of the international workshop on innovative software development methodologies and practices*, Hong Kong, China, 16 November 2014, pp. 11–20. New York: ACM.
 11. Salako K and Strigini L. When does “diversity” in development reduce common failures? Insights from probabilistic modeling. *IEEE T Depend Secure* 2014; 11(2): 193–206.
 12. Mlynarski S, Pilch R, Smolnik M, et al. Ocena Poziomu NienaruszalnoSci Bezpieczenstwa (SIL) Wg Normy EN 61508 Oraz Z Zastosowaniem Procesow Markowa. *J KONBiN* 2015; 35(1): 73–84.
 13. Cai X, Lyu MR and Vouk MA. An experimental evaluation on reliability features of N-version programming. In: *Proceedings of the international symposium on software reliability engineering*, Chicago, IL, 8–11 November 2005, pp. 161–170. New York: IEEE.
 14. Dai YS, Xie M, Poh KL, et al. A model for correlated failures in N-version programming. *IIE Trans* 2004; 36: 1183–1192.
 15. Chatterjee S, Misra RB and Alam SS. N-version programming with imperfect debugging. *Comput Electr Eng* 2004; 30(6): 453–463.
 16. Williams DP. Study of the warranty cost model for software reliability with an imperfect debugging phenomenon. *Turk J Electr Eng Co* 2007; 15(3): 369–381.
 17. Littlewood B, Popov PT, Strigini L, et al. Modeling the effects of combining diverse software fault detection techniques. *IEEE T Software Eng* 2000; 26(12): 1157–1167.
 18. Parhami B. Approach to component-based synthesis of fault-tolerant software. *Informatica* 2001; 25(4): 533–543.
 19. Kumar A and Malik K. Voting mechanisms in distributed systems. *IEEE T Reliab* 1991; 40(5): 593–600.
 20. Armoush A, Salewski F and Kowalewski S. Recovery block with backup voting: a new pattern with extended representation for safety critical embedded systems. In: *Proceedings of the 11th international conference on information technology*, Bhubaneswar, India, 17–20 December 2008, pp. 232–237. New York: IEEE.
 21. Vouk M, McAllister D, Eckhardt D, et al. An empirical evaluation of consensus voting and consensus recovery block reliability in the presence of failure correlation. *J Comput Softw Eng* 1993; 1(4): 367–388.
 22. Latif-Shabgahi G, Bass JM and Bennett S. A taxonomy for software voting algorithms used in safety-critical systems. *IEEE T Reliab* 2004; 53(3): 319–328.
 23. Mohamed A and Zulkernine M. Improving reliability and safety by trading off software failure criticalities. In: *Proceedings of the 10th IEEE high assurance systems engineering symposium (HASE'07)*, Plano, TX, 14–16 November 2007, pp. 267–274. New York: IEEE.
 24. Yacoub S. Analyzing the behavior and reliability of voting systems comprising tri-state units using enumerated simulation. *Reliab Eng Syst Safe* 2003; 81(2): 133–145.
 25. Tsarev RY, Durmus MS, Üstoglu I, et al. Classification of voting algorithms for N-version software. *J Phys* 2018; 1015: 042060.
 26. Looker N, Munro M and Xu J. Increasing web service dependability through consensus voting. In: *Proceedings of the international computer software and applications conference*, Edinburgh, 26–28 July 2005, pp. 66–69. New York: IEEE.
 27. Batawi Y and Abulnaja O. Accuracy evaluation of Arabic optical character recognition voting technique: experimental study. *Int J Elec Comput Sci* 2012; 12(1): 29–33.
 28. Tran HK and Nguyen TN. Flight motion controller design using genetic algorithm for a quadcopter. *Meas Control* 2018; 51(3–4): 59–64.
 29. Sanchez A, Escareno J, Garcia O, et al. Autonomous hovering of a noncyclic tiltrotor UAV: modeling, control and implementation. *IFAC P Vol* 2008; 41(2): 803–808.
 30. Wang T, Jobredeaux R and Feron E. A graphical environment to express the semantics of control systems. arXiv, <https://arxiv.org/abs/1108.4048>
 31. Easton C. Safety integrity verification of legacy systems. *Meas Control* 2009; 42(6): 185–189.