Advanced Control Strategies for Single-Gimbal CMG Systems: Design and Modelling

Jadyn Chowdhury[1]
DP1-AE 353: Aerospace Control Systems, Urbana, IL, 61820, United States of America

This paper presents a comprehensive study on the design and implementation of a closed-loop control system for a Single-Gimbal Control Moment Gyroscope (CMG) system. The research involves a multi-faceted approach that combines theoretical modelling, symbolic mathematics, and practical implementation to achieve precise control of the CMG system's dynamics. The objective is to stabilize the system and control its behaviour effectively.

I. Nomenclature

q_1	= angle (rad) of the platform
$\dot{q_1}$	= angular velocity ($rads^{-1}$) of the platform
q_2	= angle (rad) of the gimbal
$\dot{q_2}$	= angular velocity ($rads^{-1}$) of the gimbal
τ	= torque (Nm) applied to the platform by the gimbal
J_{1z}	= 0.5 m^2 one principal moment of inertia of the platform
J_{2x},J_{2z}	= 0.001 kgm^2 two principal moments of inertia of the gimbal
J_{3x},J_{3y},J_{3z}	= 0.01 kgm^2 principal moments of inertia of the rotor
m	= 1.0 kg the mass of the boom
r	= 2.0 m the length of the boom
g	= 9.81 ms^{-2} the acceleration of gravity
v_{rotor}	= 500 $rads^{-1}$ angular velocity of the rotor

II. Introduction

CMGs[2] have long been instrumental in spacecraft attitude control systems, offering exceptional agility and manoeuvrability. As spacecraft missions become increasingly ambitious and complex, the demand for more robust and efficient CMG systems continues to grow. This paper presents a comprehensive exploration of CMG dynamics, emphasizing the connection between theoretical foundations, numerical analysis, and practical implementation. The key objectives of this study are to assess the model's performance, evaluate the efficacy of CMGs in general, and draw practical conclusions based on in-depth numerical simulations. Through analysis of the simulation results, this research endeavours to shed light on the effectiveness of CMGs in spacecraft control applications across a spectrum of scenarios.

With a firm grounding in numerical theory, advanced computational techniques, and real-world simulations, this paper not only contributes to the understanding of CMG systems but also opens new avenues for optimising their performance. The ensuing sections delve into the theoretical framework, model design, simulation methodologies, and the rich insights gathered from the data, offering a comprehensive examination of the role of CMGs in spacecraft attitude control systems.

[1] Aerospace Engineering and Computer Science Double Major, Grainger School of Engineering at UIUC.
[2] Single-Gimbal Control Moment Gyroscope

III. Model Design

A. Physical Model

Visualising the CMG model will enable more effective analysis by aiding in communicating successes and flaws within the code structure. Consider the following physical layout,

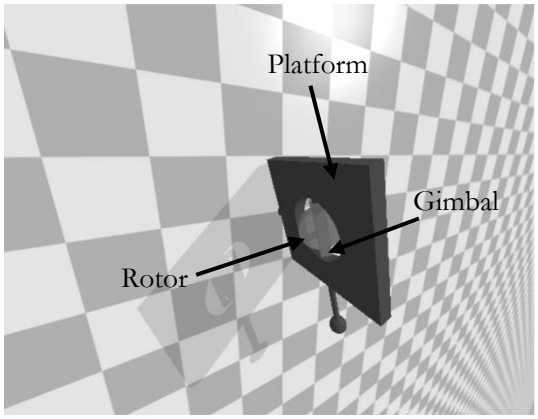

Fig. 1 Physical representation of CMG model

The platform possesses the intrinsic capability for unobstructed rotational motion around its foundational axis. This inherent pivotal capacity enables the platform to execute continuous, frictionless revolutions, thereby facilitating alterations in its angular orientation throughout the entire expanse of its rotational freedom. The gimbal mechanism can be actuated by a motor to effectuate rotational movement along an orthogonal axis with respect to the platform. Similarly, the rotor can be driven by a motor to spin about another perpendicular axis with respect to the gimbal.

In the context of a high-speed rotor rotation, the application of an "input torque" to the gimbal yields an "output torque" imparted onto the platform, as dictated by the fundamental principle of angular momentum conservation. Specifically, this output torque serves as a pivotal mechanism for the deliberate alteration of the platform's orientation.

Indeed, this quality is the variable exploited as to control the model. The investigation aims to use numerical analysis to quantitatively define operated platform orientation and thereby also establishing its limits.

1. Exploration

We imagine that the platform is symbolic of a spacecraft. Control is a necessary requirement for active functionality. Although numerous options exist to serve this purpose, we look to learn how effective a CMG as a non-propulsive actuator is to accomplish this task.

B. Mathematical Context

1. Provided EOMs

The elucidation and regulation of the system's motion are rigorously defined by a system of ODEs[3] meticulously crafted to provide a comprehensive account of how the system's constituent variables evolve over time. These ODEs serve as the foundational framework for characterising the intricate interplay of forces, torques, and inherent physical attributes that dictate the system's dynamic trajectory.

$$\ddot{q_1} = \frac{a_1 \sin(2q_2)\, \dot{q_1}\dot{q}_2 + a_2 \cos(q_2)\, \dot{q}_2 v_{rotor} + a_3 \sin(q_1)}{a_4 + a_5 \cos^2(q_2)} \quad \textbf{(1)}$$

[3] ordinary differential equations

$$\ddot{q_2} = a_6 \sin(2q_2)\, \dot{q_1^2} + a_7 \cos(q_2)\, \dot{q_1} v_{rotor} + a_8 \tau$$

(2)

Symbolising the respective angular accelerations of the system. Where,

$$a_1 = -J_{3y} + 2J_{3z}$$

$$a_2 = 2J_{3y}$$

$$a_3 = -2gmr$$

$$a_4 = 2J_{1z} + 2J_{2z} + 2mr^2$$

$$a_5 = 2J_{3z}$$

$$a_6 = \frac{J_{3y} - J_{3z}}{2(J_{2x} + J_{3x})}$$

$$a_7 = -\frac{J_{3y}}{J_{2x} + J_{3x}}$$

$$a_8 = \frac{1}{J_{2x} + J_{3x}}$$

2. Equilibrium Angles

The primary objective of this investigation is to ascertain the equilibrium state for the model under consideration. This can be resolved as the respective angular accelerations being precisely equal to zero justified by Newtons second law defining that there are therefore no forces acting on the system. Through analysis of the practical implications of these equations, it becomes evident that there exist two distinct equilibrium platform angles that satisfy this requisite condition.

Assume the platform angle is some constant. The torque, τ, must be zero for the system to be in equilibrium. Similarly, the gimbal must also be centred and unmoving, therefore q_2 must also be zero. As a direct consequence of this configuration, the angular velocities associated with the system's components must also converge to zero, given that they represent the time derivatives of constants under these equilibrium conditions. Therefore from (1),

$$\ddot{q_1} = \frac{a_3 \sin(q_1)}{a_4 + a_5}$$

Which can only evaluate to zero when q_1 is 0 or π. It is therefore concluded that the attainable equilibrium points exist at these two angles.

VARIABLE	EQUILIBRIUM VALUE
$\boldsymbol{q_1}$	$0, \pi$
$\boldsymbol{\dot{q_1}}$	0
$\boldsymbol{q_2}$	0
$\boldsymbol{\dot{q_2}}$	0
$\boldsymbol{\tau}$	0

Table 1: Table of equilibrium values for respective variables

3. State Space Model

In the endeavour to design the CMG system, simplifying the given complex equations into a more tractable linear state-space model is necessary. This strategic transformation is pivotal, as it paves the way for the application of advanced control methodologies, streamlines the process of stability analysis, and facilitates the design of purpose-built controllers. Converting the intricate dynamics of the CMG system into a linear state-space representation establishes the essential framework upon which control strategies can be systematically formed with the precision required to achieve the predefined performance goals, all while necessitating the critical criterion of system stability.

Let x be a column vector representing the state,

$$x = \begin{bmatrix} q_1 \\ \dot{q_1} \\ q_2 \\ \dot{q_2} \end{bmatrix}$$

Similarly let $\dot{x}$ be a column vector representing the derivatives of the states,

$$\dot{x} = \begin{bmatrix} \dot{q_1} \\ \ddot{q_1} \\ \dot{q_2} \\ \ddot{q_2} \end{bmatrix}$$

The state space model is given by the form,

$$\dot{x} = Ax + Bu$$

Where A is a matrix that represents the system's dynamics or the system matrix. It describes how the state variables change over time in the absence of control inputs. B is a matrix that represents the effect of control inputs on the state variables. It describes how external inputs or control actions influence the evolution of the system's state. u represents the control input or control action applied to the system. A and B are of interest as they can be used to calculate the control matrix for the model.

Based on the aforementioned equilibrium values calculated above, a new state column vector is defined to account for the differing states as the error state vector,

$$x_e = \begin{bmatrix} q_1 - q_{1_e} \\ \dot{q_1} - \dot{q}_{1_e} \\ q_2 - q_{2_e} \\ \dot{q_2} - \dot{q}_{2_e} \end{bmatrix}$$

The Jacobian matrix of $\dot{x}$ with respect to $q_1, q_2, \dot{q_1}, \dot{q_2}$ and τ is calculated using the partial derivatives of each component. Firstly, the partials for the platform angular velocity,

$$q_1 \rightarrow \frac{\partial \dot{q_1}}{\partial q_1}$$

$$q_2 \rightarrow \frac{\partial \dot{q_1}}{\partial q_2}$$

$$\dot{q_1} \rightarrow \frac{\partial \dot{q_1}}{\partial \dot{q_1}} = 1$$

$$\dot{q_2} \rightarrow \frac{\partial \dot{q_1}}{\partial \dot{q_2}} = 0$$

$$\tau = \frac{\partial \dot{q_1}}{\partial \tau}$$

Similarly for the gimbal angular velocity,

$$q_1 \rightarrow \frac{\partial \dot{q_2}}{\partial q_1}$$

$$q_2 \rightarrow \frac{\partial \dot{q_2}}{\partial q_2}$$

$$\dot{q_1} \rightarrow \frac{\partial \dot{q_2}}{\partial \dot{q_1}} = 0$$

$$\dot{q_2} \rightarrow \frac{\partial \dot{q_2}}{\partial \dot{q_2}} = 1$$

$$\tau = \frac{\partial \dot{q_2}}{\partial \tau}$$

For platform acceleration (1),

$$q_1 \rightarrow \frac{\partial \ddot{q_1}}{\partial q_1} = \frac{a_3 \cos(q_1)}{a_3}$$

$q_2 \rightarrow \frac{\partial \ddot{q_1}}{\partial q_2}$

$$= \frac{2a_1\dot{q_1}\dot{q_2}\cos(2q_2) - a_2\dot{q_2}v_{rotor}\sin(q_2)\,(a_4 + a_5\cos^2(q_2)) + a_5\sin(2q_2)\,(a_1\dot{q_1}\dot{q_2}\sin(2q_2) + a_2a_2\dot{q_2}v_{rotor}\cos(q_2) + a_3\sin(q_1))}{(a_4 + a_5\cos^2(q_2))^2}$$

$$\dot{q_1} \rightarrow \frac{\partial \ddot{q_1}}{\partial \dot{q_1}} = \frac{a_1\dot{q_2}\sin(2q_2)}{a_4 + a_5\cos^2(q_2)}$$

$$\dot{q_2} \rightarrow \frac{\partial \ddot{q_1}}{\partial \dot{q_2}} = \frac{a_1\dot{q_2}\sin(2q_2) + a_2v_{rotor}\cos(q_2)}{a_4 + a_5\cos^2(q_2)}$$

$$\tau = \frac{\partial \ddot{q_1}}{\partial \tau} = 0$$

For gimbal acceleration (2),

$$q_1 \rightarrow \frac{\partial \ddot{q_2}}{\partial q_1} = 0$$

$$q_2 \rightarrow \frac{\partial \ddot{q_2}}{\partial q_2} = 2a_6\dot{q_2}\cos(2q_2) - a_7\dot{q_1}v_{rotor}\sin(q_2)$$

$$\dot{q_1} \rightarrow \frac{\partial \ddot{q_2}}{\partial \dot{q_1}} = a_7v_{rotor}\cos(q_2)$$

$$\dot{q_2} \rightarrow \frac{\partial \ddot{q_2}}{\partial \dot{q_2}} = a_1\sin(2q_2)$$

$$\tau = \frac{\partial \ddot{q_2}}{\partial \tau} = a_8$$

Based on the state base structure, the Jacobian matrix is then given as,

$$\begin{bmatrix} \frac{\partial \dot{q_1}}{\partial q_1} & \frac{\partial \dot{q_1}}{\partial q_2} & \frac{\partial \dot{q_1}}{\partial \dot{q_1}} & \frac{\partial \dot{q_1}}{\partial \dot{q_2}} & \frac{\partial \dot{q_1}}{\partial \tau} \\ \frac{\partial \ddot{q_1}}{\partial q_1} & \frac{\partial \ddot{q_1}}{\partial q_2} & \frac{\partial \ddot{q_1}}{\partial \dot{q_1}} & \frac{\partial \ddot{q_1}}{\partial \dot{q_2}} & \frac{\partial \ddot{q_1}}{\partial \tau} \\ \frac{\partial \dot{q_2}}{\partial q_1} & \frac{\partial \dot{q_2}}{\partial q_2} & \frac{\partial \dot{q_2}}{\partial \dot{q_1}} & \frac{\partial \dot{q_2}}{\partial \dot{q_2}} & \frac{\partial \dot{q_2}}{\partial \tau} \\ \frac{\partial \ddot{q_2}}{\partial q_1} & \frac{\partial \ddot{q_2}}{\partial q_2} & \frac{\partial \ddot{q_2}}{\partial \dot{q_1}} & \frac{\partial \ddot{q_2}}{\partial \dot{q_2}} & \frac{\partial \ddot{q_2}}{\partial \tau} \end{bmatrix}$$

For readability the equations for these derivatives are not substituted in this report, but can be referenced from above. The solved equilibrium values (Table 1) are then substituted which yields the A matrix. The B matrix is calculated using the same reasoning where its Jacobian is given by,

$$\begin{bmatrix} \frac{\partial \dot{q_1}}{\partial \tau} \\ \frac{\partial \ddot{q_1}}{\partial \tau} \\ \frac{\partial \dot{q_2}}{\partial \tau} \\ \frac{\partial \ddot{q_2}}{\partial \tau} \end{bmatrix}$$

Establishing A and B enables the calculation of the necessary control matrices used as inputs of the system.

4. Control Matrices

The gimbal torque is defined as the control input or control action applied to the system, u, such that,

$$u = -Kx \qquad \textbf{(3)}$$

Where K represents the control gain matrix. Ackermann's method is used for this calculation.

Ackermann's formula, also known as the pole placement method, is a technique used to calculate the control gain matrix K for a linear time-invariant control system. The goal is to place the closed-loop poles of the system at desired locations, $[a, b, c, d]$, to achieve the desired system behaviour.

Firstly, the controllability matrix, W, is calculated by,

$$W = [B, AB, A^2B, \ldots, A^{n-1}B]$$

Where n is the dimension of the state vector x, and the function represents the concatenation of the columns of these matrices, such that in this case,

$$W = [B, AB, A^2B, A^3B]$$

Verify that the rank of the controllability matrix is equal to the number of state variables, 4. The system is therefore controllable. Following this, the desired characteristic polynomial $p(s)$ is constructed based on the desired pole locations [a, b, c, d] given by,

$$p(s) = (s - a)(s - b)(s - c)(s - d)$$

Extracting the relative coefficients yields the matrix P such that,

$$K = [0, 0, 0, 1]\, C^{-1}\, P \quad [4]$$

Using this value in (3) completes the requisite computations for effecting control over the CMG model as outlined.

[4] Ackermann's Formula

5. Exploration

While it is possible to replicate this computation through code-based simulations, it is imperative to underscore the significance of grasping the underlying mathematical framework for maximizing the efficacy of the resultant simulation. Being cognisant of the functionality of each constituent element and their interconnections enables valuable discernment into the practical implications of the theoretical foundation.

C. Numerical Analysis with Python

The computations described previously have now been transcribed into Python code, facilitating their integration with the model for conducting simulations. The following walks through each implementation, significant portions of the code have been redacted for readability purposes[5].

Firstly, the variables are defined to be used as global variables in the following code,

```
J_1z = 0.5                  # m^2 one principal moment of inertia of the
platform
J_2x = J_2z = 0.001          # kgm^2 two principal moments of inertia of
the gimbal
J_3x = J_3y = J_3z = 0.01    # kgm^2 principal moments of inertia of the
rotor
m = 1.0                     # kg the mass of the boom
r = 2.0                     # m the length of the boom
g = 9.81                    # ms^-2 the acceleration of gravity
v_rotor = 500               # rad/s angular velocity of the rotor

a_1 = -J_3y+(2*J_3z)
a_2 = 2*J_3y
a_3 = -2*g*m*r
a_4 = 2*J_1z+2*J_2z+2*m*r*r
a_5 = 2*J_3z
a_6 = (J_3y-J_3z)/(2*(J_2x+J_3x))
a_7 = -J_3y/(J_2x+J_3x)
a_8 = 1/(J_2x+J_3x)
```

[5] The complete code can be reviewed by the sister Jupyter file submitted with this report

The class, *StateSpaceModel*, that represents a state-space model for the modelled dynamic system. It calculates symbolic equations for the system's state derivatives, Jacobian matrices, and provides methods to evaluate them numerically at specified equilibrium values that are passed for initialisation.

```
class StateSpaceModel:
    def __init__(self, q1_val,  q1_dot_val, q2_val, q2_dot_val, tau_val):
        # Define symbolic variables
        ... = symbols('...')

        # Assign the symbolic variables to instance variables
        self.q1 = q1
        ...
        # Calculate q1_double_dot and q2_double_dot using symbolic equations
        self.q1_double_dot = self.calculate_q1_double_dot(...)
        self.q2_double_dot = self.calculate_q2_double_dot(...)

        # Create a symbolic matrix for the state derivatives
        self.state_dot = Matrix([...])

        # Define equilibrium values as a dictionary
        self.equilibrium_values = {
            self.q1: q1_val,
            ...
        }

        # Create a symbolic matrix for the state errors
        self.state_error = Matrix([[…] for var, eq_val in self.equilibrium])

        # Calculate Jacobian matrices symbolically
        self.A_matrix = jacobian([q1, q1_dot, q2, q2_dot])
        self.B_matrix = jacobian([tau])

        # Lambda functions for numerical evaluation
        self.A_evaluator = lambdify([...], self.A_matrix)
        self.B_evaluator = lambdify([...], self.B_matrix)

    # Define symbolic equations for q1_double_dot and q2_double_dot
    def calculate_q1_double_dot(self, q1,  q1_dot,q2, q2_dot, tau):
        ...
        return q1_double_dot

    def calculate_q2_double_dot(self, q1,  q1_dot,q2, q2_dot, tau):
        ...
        return q2_double_dot

    # Evaluate Jacobian matrices at equilibrium values
    def evaluate_jacobians_at_equilibrium(self):
        ...
        return A_at_eq, B_at_eq
```

The utilised *evaluate_jacobians_at_equilibrium* function provides the *A* and *B* matrices used by the next class.

The class *ClosedLoopController* for the closed-loop control system calculates the feedback control matrix K to achieve desired pole locations, computes the closed-loop matrix A_cl, checks the stability of the closed-loop system, and calculates the controllability matrix W for the given system. The class is passed the A, B and the poles at each instance.

```
class ClosedLoopController:
    def __init__(self, A, B, desired_poles):
        # Initialize the class with system matrices A and B, and desired poles
        self.A = A
        ...

        # Calculate the feedback control matrix K using desired poles
        self.K = self.calculate_feedback_matrix()

        # Calculate the closed-loop matrix A_cl
        self.A_cl = self.calculate_closed_loop_matrix()

        # Calculate the eigenvalues (poles) of the closed-loop matrix
        self.poles = self.calculate_closed_loop_poles()

        # Calculate the controllability matrix W
        self.W = self.calculate_controllability_matrix()

    def calculate_feedback_matrix(self):
        # Calculate the feedback control matrix K using desired poles
        return control.acker(self.A, self.B, self.desired_poles)

    def calculate_closed_loop_matrix(self):
        # Calculate the closed-loop matrix A_cl as A - B*K
        return self.A - self.B @ self.K

    def calculate_closed_loop_poles(self):
        # Calculate the eigenvalues (poles) of the closed-loop matrix A_cl
        return np.linalg.eigvals(self.A_cl)

    def is_stable(self):
        # Check if all closed-loop poles are stable (have negative real parts)
        for pole in self.poles:
            if pole > 0:
                return False
        return True

    def calculate_controllability_matrix(self):
        # Calculate the controllability matrix W
        n = self.A.shape[0]
        ...

        for i in range(n):
            # Calculate the controllability matrix using matrix powers
            W[:, i * m:(i + 1) * m] = np.linalg.matrix_power(self.A, i) @ self.B

        return W
```

The function *is_stable* checks that all the eigenvalues of the calculated matrix are less than zero therefore affirming that the system is asymptotically stable at that point. The *calculate_controllability_matrix* function is used to verify that the rank matches the state size and thus confirm the system is controllable. The function used for modelling *calculate_feedback_matrix* uses Pythons' control library to calculate K to be assigned such that it can be called.

The *Controller* class designates the control input for modelling the CMG. The primary function is *run* that updates the feedback on each iteration, responsible for calculating the control action (gimbal torque) based on the given inputs.

```
class Controller:
    def __init__(self):
        # Initialize any controller-specific variables here
        ...

    def reset(self):
        # Reset any controller-specific variables here
        ...

    def run(
            self,
            t,
            platform_angle,
            platform_velocity,
            gimbal_angle,
            gimbal_velocity,
        ):
        x = array([platform_angle, ...])
        model = StateSpaceModel(np.deg2rad(0),0,0,0,0)
        A,B = model.evaluate_jacobians_at_equilibrium()
        temp_controller = ClosedLoopController(A, B, [-1,-2,-3,-4])
        print(temp.conroller.is_stable())
        gimbal_torque = np.dot(-temp_controller.K,x)[0]

        return gimbal_torque
```

Firstly, *run* constructs a state vector x containing the platform and gimbal angles and velocities. It then creates an instance of a *StateSpaceModel* class to then call *evaluate_jacobians_at_equilibrium* on the *StateSpaceModel* instance to obtain the state-space matrices A and B and uses them for control calculations. A $ClosedLoopController$ instance named t*emp_controller* is then created with the obtained matrices and a desired set of poles[6] that should be equal to the eigenvalues of the corresponding gain matrix. The stability of each iteration is displayed to ensure that the system is asymptotically stable approaching the equilibrium points. In addition, it then calculates the gimbal torque by multiplying the control gain matrix K from *temp_controller* by the state vector and extracts the torque value from the result, thus completing the feedback loop.

The remainder of the code processes the relevant data for application in the simulation and draw numerical and graphical results.

1. Exploration

In the context of the future of physics, computational proficiency emerges as an indispensable skill. The capacity to swiftly manipulate extensive datasets and generate real-time simulations holds the key to unlocking avenues for enhancing both individual and collective understanding. Traditional mathematical theory, while indispensable in its own right, often falls short in vividly illustrating the implications of control parameter variations, a deficiency notably mitigated by code-based simulations.

[6] Reasoning for these values discussed in following sections

IV. Simulation Results

A. Ideal Parameters

Following a series of experimental iterations, the optimal plot configuration was determined utilising the specified parameters,

PARAMETER	VALUE
INITIAL ANGLE	$30°$
TARGET ANGLE	$0°$
DESIRED POLES	$[-1, -2, -3, -4]$

Table 2: Ideal parameters for stabilisation

These values yield the following results: A snapshot from the end of the simulation showing the platform at rest at the target angle,

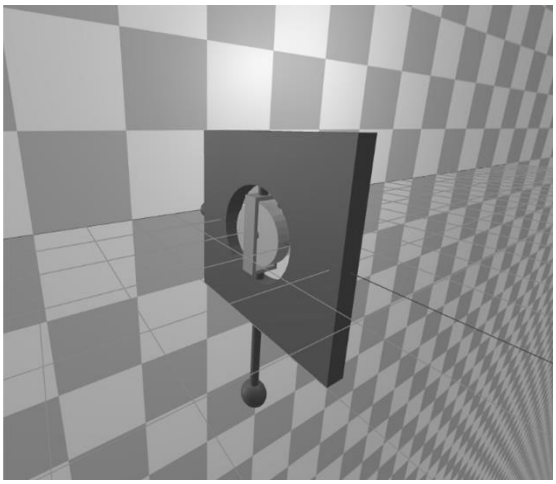

Fig. 2 Snapshot of final ideal parameter position

To better visualise how the system resolved to this final state the variable plots can be reviewed,

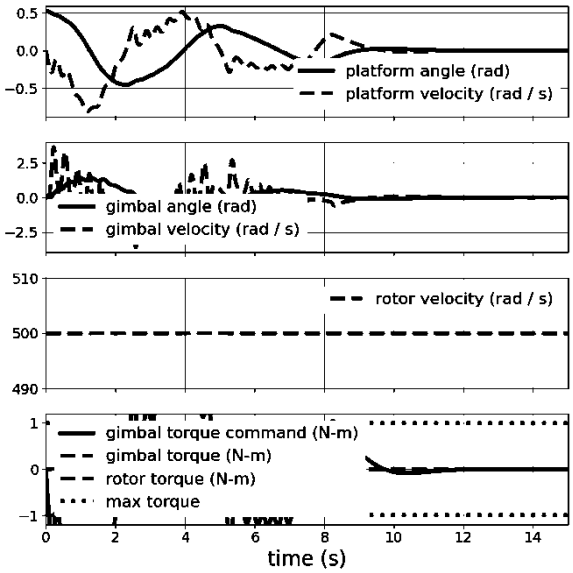

Fig. 3 Graph of state variables from ideal parameters

This demonstration serves to show the successful operation of the CMG controller as envisioned. The platform's angular motion exhibits a controlled oscillation characterized by progressively diminishing amplitudes centred around the desired target angle, ultimately converging to and stabilising at the target angle.

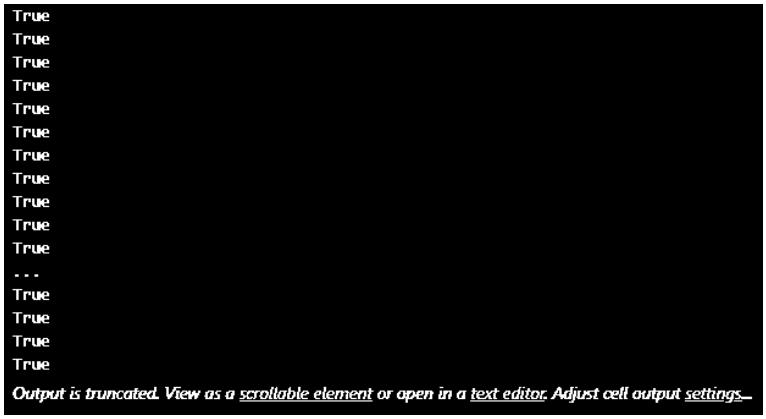

Fig. 4 Output terminal displaying the result from is_stable function

Throughout the duration of the simulation, the system exhibited asymptotic stability. It is worth highlighting that the initial angular position closely approximated the desired target angle.

1. Exploration

The observed success of the controller in specific scenarios presents an encouraging prospect. It suggests the potential for a broader application of control principles when the CMG system is generalised. This outcome sheds light on the prospect of extending these findings into the realm of real-world control systems, where the principles and insights gained from this study may find wider and more versatile applicability.

B. Testing Angle Limits

Systematic investigation of graphical representations generated at varying angles relative to the target angle affords valuable insights into discernible patterns, which, in turn, may provide crucial indications regarding the model's inherent limitations.

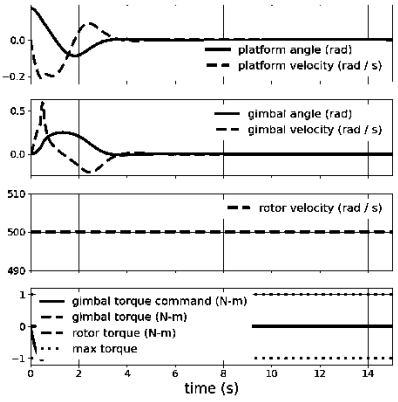

Fig. 7 Graph of state variables at initial angle of 10 deg

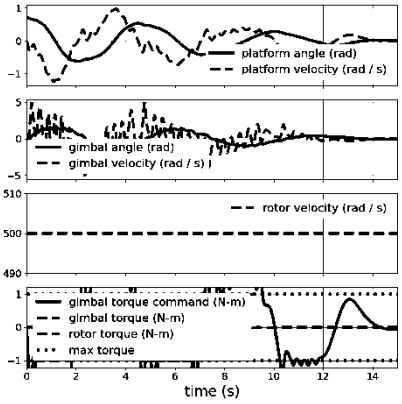

Fig. 6 Graph of state variables at initial angle of 40 deg

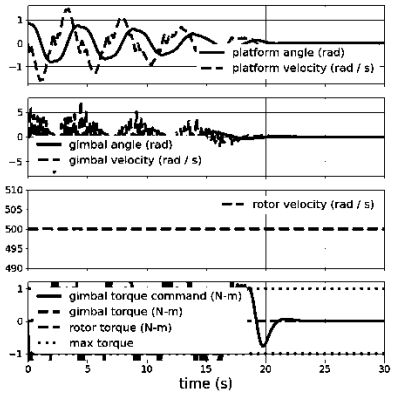

Fig. 5 Graph of state variables at initial angle of 50 deg

At an initial angle of ten degrees, the platform angle exhibits behaviour closely resembling the ideal parameters. It undergoes a smooth oscillatory motion, converging towards the target angle before reaching a state of equilibrium. This stabilisation process is notably swift, concluding approximately seven seconds earlier than when the initial angle is set at thirty degrees.

In contrast, when the initial angle is increased to forty degrees, the platform's stabilisation becomes protracted, requiring a duration of approximately fourteen seconds. Moreover, the oscillation pattern deviates from smoothness, characterized by abrupt jerking motions as the CMG attempts to stabilise evidenced by the rougher platform angle waveform. A similar trend is observed when the initial angle is further elevated to fifty degrees, with the stabilisation process elongating to approximately twenty seconds and exhibiting more erratic behaviour along its path towards equilibrium.

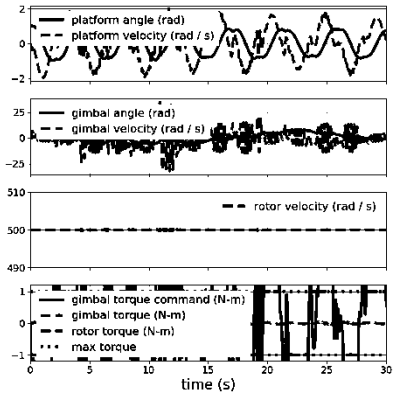

Fig. 10 Graph of state variables at initial angle of 60 deg

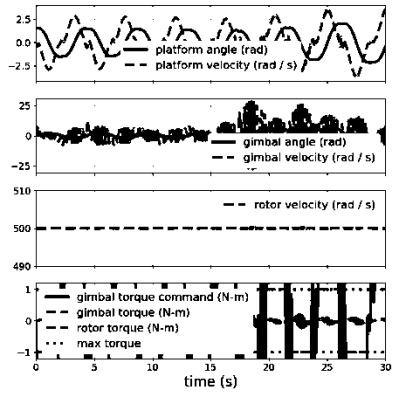

Fig. 9 Graph of state variables at initial angle of 90 deg

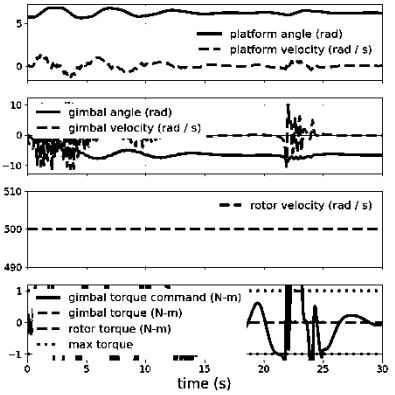

Fig. 8 Graph of state variables at initial angle of 330 deg

When subjected to a sixty-degree initial angle, the system exhibits a notable inability to attain equilibrium within a reasonable time frame. Observations indicate erratic behaviour in the gimbal mechanism, which further complicates stabilisation efforts.

Upon elevating the initial angle to ninety degrees, not only does the system fail to stabilise, but it also displays a discernible divergence from equilibrium. This divergence is indicated in the platform angle waveform, as evidenced by an increasing wave amplitude. Additionally, the resultant torques demonstrate minimal indications of effective control under these more extreme conditions.

When commencing the angle from negative thirty degrees from the target angle or 330 degrees, the CMG system demonstrates a capacity to return to equilibrium. At all tested angles, the system remained asymptotically stable.

The system failed to stabilise at 180 degrees, even with initial angles closely approximating this target angle.

1. Exploration

This demonstrates the influence of the starting conditions on the system's dynamic response, suggesting a potential avenue for control optimisation in challenging operational scenarios.

C. Testing Pole Limits

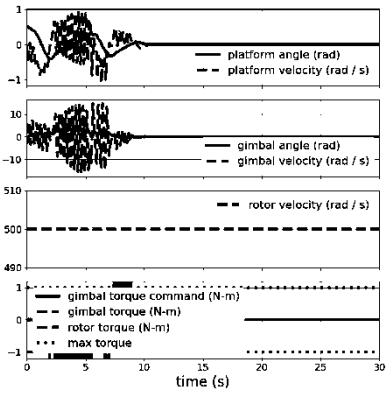

Fig. 13 Graph of state variables at desired poles [-4,-5,-6,-7]

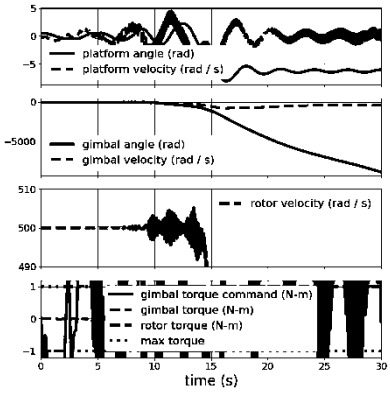

Fig. 12 Graph of state variables at desired poles [-0.1,-0.2,-0.3,-0.4]

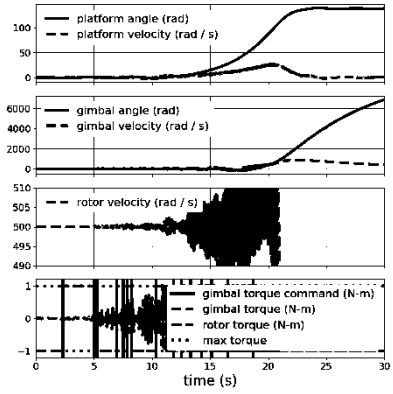

Fig. 11 Graph of state variables at desired poles [-10,-20,-30,-40]

Increasing the magnitude of the system's poles marginally yields equilibrium, albeit lacking the desired smoothness seen by the ideal poles. Indeed, the gimbal's behaviour becomes erratic, causing substantial perturbations on the platform angle waveform.

Conversely, when adjusting the pole magnitudes to more closely align with equilibrium values, the CMG system experiences an alarming loss of control. Notably, the gimbal angle diverges, signifying an incapacity to achieve stabilisation.

Further experiments involving a substantial pole magnitude deviation from equilibrium shows the gimbal angle exhibiting similar divergent behaviour, but in the positive direction. Notably, the platform angle shows no sign of oscillation and appears to stabilise around a different angle.

Figures 11 and 12 depict pronounced deviations in rotor velocity, despite their intended constancy. These deviations can be interpreted as indications of controller instability, with the PID controller succumbing to inaccuracies induced by the pole adjustments. At all tested angles, the system remained asymptotically stable.

1. Exploration

The vast spectrum of potential pole values presents a complex landscape, wherein discerning optimal choices through a trial-and-error approach proves to be an inefficient and impractical strategy for thoroughly exploring the spectrum of compatibility. The need for a more systematic and informed approach to navigate this range becomes evident, as the effectiveness of the control system crucially depends on precise pole value selection.

D. Adjusting Constants

To further establish the CMGs functionality, observations can be made to value adjustments of the constants defined at the beginning of the file.

1. Boom mass

Assuming ideal parameters,

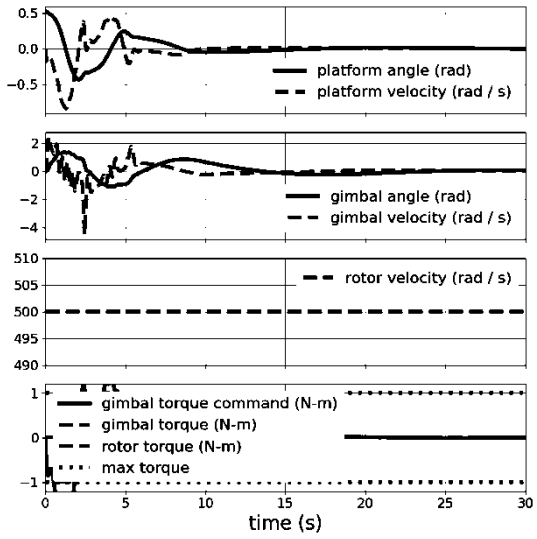

Fig. 15 Graph of state variables at boom mass 10kg

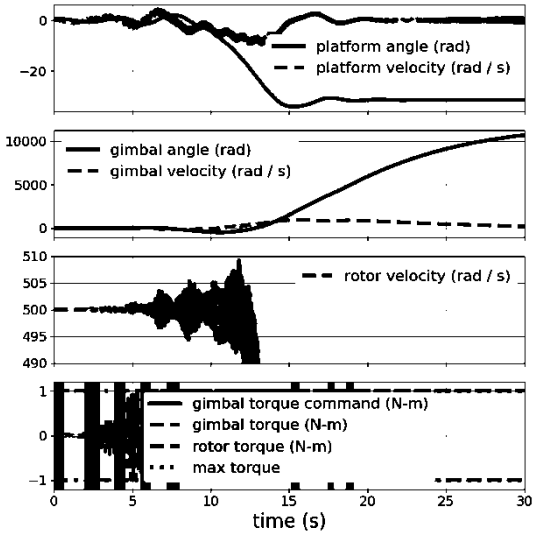

Fig. 14 Graph of state variables at boom mass 1000kg

At a specified boom mass of 10kg, the system demonstrates a notably accelerated stabilisation process compared to the configuration with a 1kg boom mass. However, this expeditious stabilisation comes at the cost of platform angle waveform smoothness.

When the boom mass is substantially increased, an intriguing scenario unfolds. While the system initially appears to stabilise, further analysis reveals complications. There is a discernible loss of rotor velocity consistency, and conflicting indications regarding the equilibrium angle become apparent. These observations collectively suggest a breakdown in the control system, signifying that the controller could no longer maintain the desired performance under the altered boom mass conditions.

2. Rotor Velocity

Assuming ideal parameters,

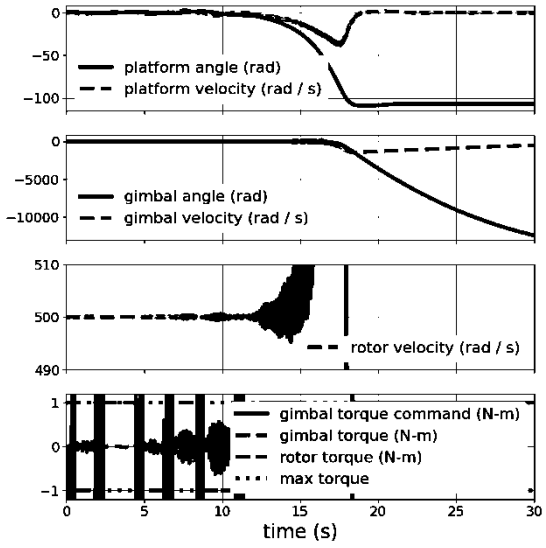

Fig. 17 Graph of state variables at rotor velocity 2500 rad/s

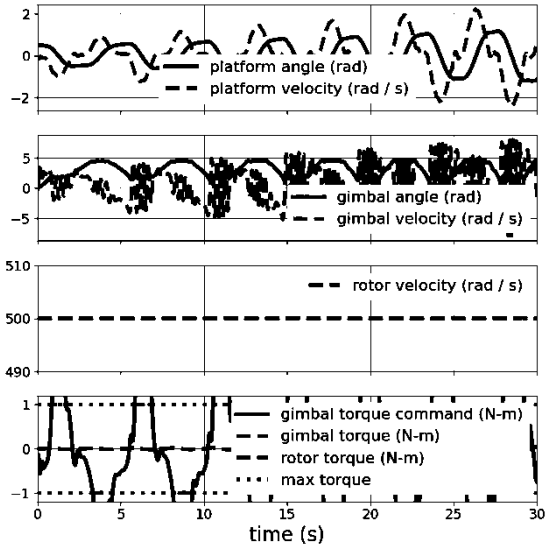

Fig. 16 Graph of state variables at rotor velocity 50 rad/s

When subjected to a rotor velocity of 2500 rad/s, the CMG system exhibits outcomes similar to the scenario involving a substantial boom mass. In this configuration, rotor velocity deviation becomes apparent, and the equilibrium angle diverges from the initial input, culminating in a breakdown of the controller's functionality.

In contrast, when the rotor velocity is reduced to 50 rad/s, the platform angle has a smoothly oscillating waveform. However, an intriguing challenge emerges, as the gimbal mechanism seems to lack the requisite torque to effect significant alterations in the trajectory toward equilibrium.

3. Transitioning Between Equilibrium Angles

Assuming ideal parameters,

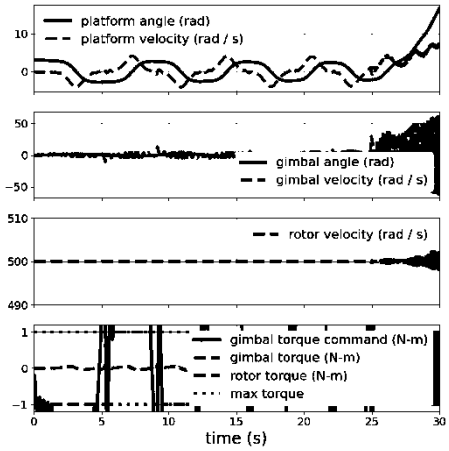

Fig. 18 Graph of state variables at initial angle 180 degrees and final 0 degrees

Extensive experimentation involving various pole configurations led to a consistent observation that there was no scenario in which the CMG system achieved stabilisation or indeed, approached equilibrium. In every instance, the controller ultimately failed, though the timing of these failures did vary across different configurations. However, it is vital to emphasise that these observed limitations should not be construed as definite. Rather, they underscore the inherent challenges related to the constraints discussed in the 'Pole Exploration' section.

4. Exploration

The dichotomy between stability and smoothness highlights the intricate relationship between factors and system performance, stressing the delicate balance required for effective control in such dynamic scenarios.

V. Conclusion

A. Differences Between Linear Model and Nonlinear Simulation

Linear models, by nature, are abstractions crafted to approximate the behaviour of inherently nonlinear systems around specific operating points. Their validity is contingent upon operation within a confined range, and they are intrinsically constrained in their capacity to accurately depict system behaviour when the system operates significantly distant from the equilibrium point.

Nonlinear systems frequently show complex behaviours which lie beyond the descriptive capacity of linear models. The presence of these nonlinear elements within a system can result in deviations from the predictions theorised by linear models.

This conclusion became apparent through the empirical findings, where the model exhibited signs of degradation and erratic behaviour as inputs were progressively moved away from their equilibrium points.

B. Supporting Questions

An empirical investigation into the impact of initial conditions and the selection of goal angles was conducted within the mathematical background and results sections of this study. The outcomes revealed a pronounced volatility of the system to variations in initial conditions, with optimal performance occurring exclusively when the initial conditions closely aligned with equilibrium and ideal parameters. Additionally, it was observed that the system exhibited stability solely at two specific goal angles, thereby imposing constraints on the scope of the simulation and its testing boundaries.

C. Model Viability

The findings presented in this report underscore a critical aspect that, while conditions leading to system stabilisation do exist, the challenges associated with identifying these conditions, coupled with the preponderance of unsuccessful scenarios, cast doubt on the practical viability of this model in real-world applications. Essential factors, such as the need for smooth, payload-protecting movements, appear to be achievable only within a limited set of conditions. Additionally, the imperative need for rapid orientation adjustments and sustained stability, pivotal in real-world control scenarios, becomes exponentially more challenging as the system deviates from its equilibrium points. These observations collectively highlight the inherent limitations and complexities associated with the practical implementation of this model.

D. CMG Viability and Model Conflict

In the realm of spacecraft control, single-gimbal CMGs offer a distinct advantage over reaction wheels through torque amplification, the potential for significantly higher output torques. This characteristic empowers CMGs to generate substantial torques while maintaining relatively low power consumption. As a result, they emerge as a compelling choice for extended space missions, particularly those with strict power and weight limitations, such as geosynchronous satellites, larger spacecraft, and interplanetary probes.

The utilisation of CMGs necessitates the implementation of advanced control strategies to harness their full potential. These strategies often involve intricate feedback control algorithms as a means to attain precise and agile attitude control. The operational success of CMGs is intrinsically tied to the robustness of these control algorithms, as they must contend with a multitude of real-world challenges, including disturbances, uncertainties, and nonlinearities.

Regrettably, the model presented in this report did not encompass the nuanced complexities of these control algorithms. Consequently, it did not account for the critical aspect of algorithmic robustness required for effective CMG operation in practical scenarios.

It is worth noting that the computational intricacies and algorithmic modifications necessary to enhance system robustness extend beyond the scope of this report. Nevertheless, one promising avenue for future research and extension of this work lies in a comprehensive exploration of these optimisation characteristics.

E. Reflection

Being a double major in computer science and aerospace engineering, I find myself drawn to the intersection of these fields. This report represents my initial step into practical applications within this area, offering me valuable insights into its intricacies and challenges.

What is most gratifying is the tangible impact that well-designed algorithms can have, especially in the context of spacecraft control systems. This report has not only expanded my technical knowledge but has also ignited a deep sense of fulfilment, seeing how algorithms can translate into real-world results.

Furthermore, this experience has intensified my curiosity and passion for delving further into the realm of computer-aided numerical analysis in the context of control systems. It has solidified my resolve to pursue a career that allows me to explore the depths of aerospace engineering and advance the field of control systems with practical implementations.