小型衛星の姿勢決定制御システムの設計と評価

中須賀船瀬五十里研究室

発表者名

2025 年 10 月 19 日

1 はじめに

本資料では,小型衛星における姿勢決定制御システム(ADCS: Attitude Determination and Control System)の設計手法と性能評価について説明する [1] .

1.1 研究背景

近年,CubeSat をはじめとする小型衛星の開発が盛んに行われている.小型衛星は限られたリソースの中で高精度な姿勢制御を実現する必要があり,効率的な ADCS の設計が重要である.

1.2 目的

本研究の目的は以下の通りである:

1. 小型衛星向け ADCS の設計手法の確立
2. センサーとアクチュエータの最適な組み合わせの検討
3. シミュレーションによる性能評価

2 ADCS の構成要素

2.1 センサー系

姿勢決定に用いられる主なセンサーは以下の通り:

- **太陽センサー**: 太陽方向を検出
- **地磁気センサー**: 地球磁場ベクトルを測定
- **ジャイロセンサー**: 角速度を測定
- **スタートラッカー**: 恒星の位置から高精度な姿勢を決定

重要: センサーの組み合わせにより,姿勢決定精度とコストのトレードオフが生じる.

2.2 アクチュエータ系

姿勢制御に用いられる主なアクチュエータ:

- **リアクションホイール**: 角運動量交換による姿勢制御
- **磁気トルカ**: 地球磁場との相互作用によるトルク生成
- **推進器**: 直接的な力・トルクの生成

2.3 姿勢決定アルゴリズム

代表的な姿勢決定手法として,TRIAD 法やカルマンフィルタがある.観測ベクトル$\boldsymbol{v}_i$と参照ベクトル$\boldsymbol{w}_i$から姿勢行列$\boldsymbol{A}$を求める:

$$\boldsymbol{v}_i = \boldsymbol{A}\boldsymbol{w}_i + \boldsymbol{\varepsilon}_i$$

ここで,$\boldsymbol{\varepsilon}_i$は観測誤差である.

3 設計手法

3.1 システム設計プロセス

設計手順

1. ミッション要求の定義
2. センサー・アクチュエータの選定
3. 制御アルゴリズムの設計
4. シミュレーションによる検証

CubeSat プロジェクトでは,限られた予算と開発期間の中で効果的な ADCS を実現する必要がある.先行研究 [2] で報告されているように,Hardware-in-the-Loop Simulation (HILS)を用いた事前検証が有効である.

3.2 制御手法の検討

姿勢制御に用いられる代表的な制御手法:

3.2.1 PD 制御

- **メリット**: 実装が簡単,計算負荷が小さい
- **デメリット**: 外乱に対する応答性が低い

3.2.2 LQR 制御

- **メリット**: 最適制御理論に基づく,性能保証
- **デメリット**: モデル化誤差に敏感

3.2.3 スライディングモード制御

- **メリット**: ロバスト性が高い,外乱に強い
- **デメリット**: チャタリングが発生する可能性

注意: 制御系設計では,センサーノイズやアクチュエータの飽和を考慮する必要がある.

4 図表の例

4.1 図の挿入方法

図を挿入する場合は,`#figure()`関数を使用します:

画像の配置例

```
#figure(
  image("images/example.png", width: 70%),
  caption: [図の説明文をここに書く]
)
```

実際の画像ファイルを使った例:

図 1 Typst のロゴ

注意: 画像ファイルは相対パスで指定します。`Figures/`ディレクトリに画像を配置してから参照してください。

画像がない場合は,プレースホルダーを使用できます:

図 2 衛星システム構成図（プレースホルダー）

4.2 複数の画像を並べる

複数の画像を横に並べる場合:

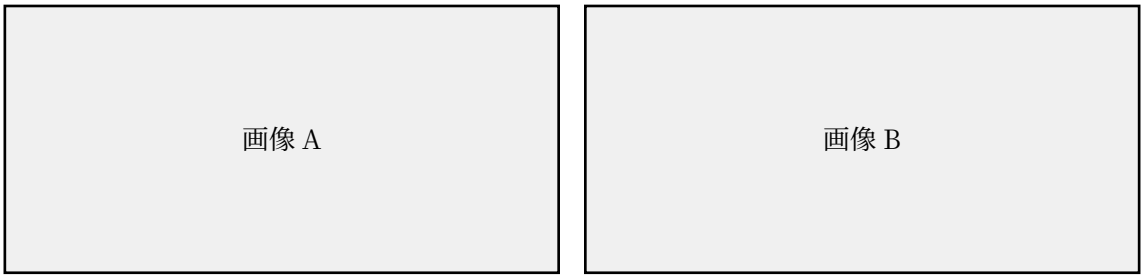

図 3 2 つの画像を並べた例

5 シミュレーション例

5.1 姿勢制御シミュレーション

以下は,Python での簡単な姿勢制御シミュレーションの実装例である:

```python
import numpy as np
from scipy.spatial.transform import Rotation

class AttitudeController:
    """簡易的なPD制御器"""

    def __init__(self, kp=0.1, kd=0.05):
        """
        Parameters:
        -----------
        kp : float
            比例ゲイン
        kd : float
            微分ゲイン
        """
        self.kp = kp
        self.kd = kd

    def compute_torque(self, q_current, q_target, omega):
        """
        制御トルクを計算

        Parameters:
        -----------
        q_current : ndarray
            現在のクォータニオン
        q_target : ndarray
            目標クォータニオン
        omega : ndarray
            角速度ベクトル

        Returns:
        --------
        torque : ndarray
            制御トルク
        """
        # 姿勢誤差の計算
        q_error = self.quaternion_error(q_current, q_target)

        # PD制御則
        torque = -self.kp * q_error[1:4] - self.kd * omega

        return torque

    @staticmethod
    def quaternion_error(q1, q2):
        """クォータニオン誤差を計算"""
```

```
        r1 = Rotation.from_quat(q1)
        r2 = Rotation.from_quat(q2)
        r_error = r2 * r1.inv()
        return r_error.as_quat()

# 使用例
controller = AttitudeController(kp=0.1, kd=0.05)
q_current = np.array([0, 0, 0, 1])  # 現在姿勢
q_target = np.array([0.1, 0, 0, 0.995])  # 目標姿勢
omega = np.array([0.01, 0.02, 0.01])  # 角速度

torque = controller.compute_torque(q_current, q_target, omega)
print(f"制御トルク: {torque}")
```

6 シミュレーション結果

6.1 姿勢制御性能

ステップ応答シミュレーションの結果:

表 1 各制御手法の性能比較

制御手法	整定時間 (s)	オーバーシュート (%)	定常偏差 (deg)
PD 制御	45.2	8.5	0.12
LQR 制御	38.7	3.2	0.05
SMC	32.1	12.8	0.02

シミュレーション結果から,LQR 制御が最もバランスの取れた性能を示すことが確認された.

6.2 考察

6.2.1 制御性能について

制御性能は以下の要因に影響される:

1. センサーノイズの大きさ
2. アクチュエータの性能限界
3. 外乱トルクの大きさ

今後,実機試験による検証を行う予定である.

7 まとめ

本研究では,小型衛星向け ADCS の設計手法とシミュレーションによる性能評価を行った.主な成果は以下の通り:

- ADCS の構成要素とセンサー・アクチュエータの選定基準を整理
- 複数の制御手法の特徴と適用範囲を比較
- シミュレーションによる性能評価で LQR 制御の有効性を確認

今後は実機を用いた Hardware-in-the-Loop 試験により,設計の妥当性を検証する予定である.

8 参考文献

[1] J. R. Wertz, Spacecraft Attitude Determination and Control, 1st 版. Springer, 2011.

[2] J. Kiesbye ほか, 「Hardware-in-the-loop and software-in-the-loop testing of the move-ii cubesat」, Aerospace, vol. 6, no. 12, p. 130, 2019.