VEREINIGUNG DER STEUERUNG VON AKTUATOREN MIT UNTERSCHIEDLICHEN ZEITHORIZONTEN FÜR KI-BASIERTE SATELLITEN-LAGEREGELUNG MITTELS SUBNETZ-POLITIK

K. Djebko*†, T. Baumann†, E. Dilger†, F. Puppe*, S. Montenegro†

* Julius-Maximilians-Universität Würzburg, Center for Artificial Intelligence and Data Science, Emil-Fischer-Straße 50, 97074 Würzburg, Deutschland

† Julius-Maximilians-Universität Würzburg, Lehrstuhl für Informatik VIII - Informationstechnik für Luft- und Raumfahrt, Emil-Fischer-Str. 70, 97074 Würzburg, Deutschland

Zusammenfassung

Zuverlässige Lageregelung ist essenziell für Satellitenmissionen. Die Entwicklung klassischer Regler ist jedoch zeitaufwendig und anfällig für Abweichungen zwischen Erwartung und Realität. Deep Reinforcement Learning (DRL) bietet eine vielversprechende Alternative, mit der ein Regler durch selbstständige Interaktion mit einer Umgebung eine angepasste Regelungsstrategie lernen kann, indem eine Belohnungsfunktion maximiert wird. In der Praxis werden häufig Reaktionsräder und Magnettorquer kombiniert, wobei Reaktionsräder für schnelle Lageänderungen und Magnettorquer für langsames Momentum-Management eingesetzt werden. Da sich beide Aktuatoren gegenseitig beeinflussen, ist ein vereinigter, auf beide Aktuatoren abgestimmter KI-Regler wünschenswert. Das Training eines solchen vereinigten Reglers ist jedoch anspruchsvoll, da die Aktuatoren widersprüchliche Anforderungen an die Belohnungsfunktion und die Trainings-Hyperparameter haben. Reaktionsräder ermöglichen schnelle Lageänderungen und erfordern während des Trainings primär den Fokus auf unmittelbare Aktionen, während Magnettorquer langsame Änderungen bewirken und für sie lange Zeithorizonte betrachtet werden müssen. Erschwerend kommt hinzu, dass beide Aktuatortypen zwar stark unterschiedliche Verhaltenseigenschaften aufweisen, sich jedoch gegenseitig beinflussen.

In unserer Arbeit vereinen wir die Regelung beider Aktuatoren durch den Einsatz von Subnetzwerken, die während des Trainings aktiviert, eingefroren oder deaktiviert werden können. Dies löst den Trainingskonflikt auf und vereinfacht das zugrundeliegende Optimierungsproblem. Wir haben unser Verfahren angewendet, um mithilfe eines realistischen Simulators einen KI-Regler für den InnoCube-Satelliten zu trainieren. InnoCube ist ein 3U-CubeSat, der von der Julius-Maximilians-Universität Würzburg in Kooperation mit der Technischen Universität Berlin entwickelt und im Januar 2025 erfolgreich gestartet wurde. Ziel ist es, den Regler zukünftig direkt an Bord des Satelliten im All zu erproben und iterativ zu verbessern.

Keywords

Künstliche Intelligenz; Deep Reinforcement Learning; Nanosatelliten; CubeSats; Lageregelung

1. EINLEITUNG UND HINTERGRUND

Aufbauend auf unseren bisherigen Arbeiten [1–3] untersuchen wir die Entwicklung eines auf Künstlicher Intelligenz (KI) basierten Lagereglers, der zwei Aktuatortypen mit widersprüchlichen Traingsanforderungen gleichzeitig steuern kann und robust gegenüber Variationen von Rahmenbedingungen ist. Wir setzen Deep Reinforcement Learning (DRL) ein, eine Variante des klassischen Reinforcement Learnings [4], bei der ein sogenannter KI-Agent durch selbstständige Interaktion mit einer Umgebung eine Regelungsstrategie lernt, die eine gegebene Belohnungsfunktion maximiert. Ein Nachteil von DRL liegt in der Tendenz neuronaler Netze, zwar gute Ergebnisse bei Interpolationsproblemen zu erreichen, jedoch bei Extrapolationsproblemen zu versagen [5]. Für praktische Anwendungen ist es daher notwendig, die Rahmenbedingungen während des Trainings zu variieren, um sogenannte Out-of-Distribution-Fehler zu vermeiden. Dies umfasst insbesondere die Trägheitsmomente, die wir bereits in [1, 2] erfolgreich betrachtet haben. Ziel unserer Arbeit ist es einen KI-basierten Lageregler am Boden zu trainieren, auf den InnoCube-Satelliten hochzuladen und in Betrieb zu nehmen, sobald InnoCube seine Hauptmission abgeschlossen hat. InnoCube ist ein 3U-Nanosatellit, der von der Julius-Maximilians-Universität Würzburg und der Technischen Universität Berlin [6–8] entwickelt und am 14. Januar 2025 erfolgreich gestartet wurde. Während ursprünglich eine reine Reaktionsradregelung vorgesehen war, stellte sich eine unerwartete Einschränkung bezüglich der Reaktionsraddrehzahlen heraus. Ein längerer Betrieb der Räder im Bereich von [-350, 350] U/min sollte vermieden werden, da

doi: 10.25967/650051

dies zu ungenauer Geschwindigkeitsschätzung und -steuerung führen kann. Als Vorsichtsmaßnahme haben wir daher Momentum-Management mittels Magnettorquer integriert. Dies erhöhte jedoch die Trainingskomplexität erheblich, da beide Aktuatoren sehr unterschiedliche Trainingsanforderungen haben, die in den folgenden Abschnitten erläutert werden.
Dieses Paper ist wie folgt gegliedert: In Abschnitt 2 wird die verwendete Methodik detailliert beschrieben. Dies umfasst eine Darstellung des Simulationsmodells, der Simulationsrahmenbedingungen sowie eine Beschreibung der Subnetz-Politik und des verwendeten SkipPPO-Trainers. Darüber hinaus wird der eigentliche Trainingsprozess erläutert. Abschnitt 3 präsentiert die Ergebnisse unserer Arbeit, die wir in Abschnitt 4 diskutieren und mit einem Ausblick auf zukünftige Arbeiten abschließen.

2. METHODIK

Im Vorfeld des Trainings wurde ein Simulationsmodell erstellt und iterativ verbessert. Mit der Verfügbarkeit von Telemetriedaten nach dem Start des Satelliten konnten zusätzlich die Trägheitsmomente verfeinert, Variationen der Orbitparameter eingefügt und das magnetische Restdipolmoment sowie der Gyro-Bias bestimmt werden. Zusätzlich wurde ein Sensorrauschmodell für die Gyroskope und Magnetometer erstellt und die Simulation damit erweitert. Für das Training wurde die Stable-Baselines3 [9]-Implementierung des Proximal Policy Optimization (PPO)-DRL-Algorithmus [10] innerhalb des Gymnasium-Frameworks [11] verwendet und anschließend modifiziert. Als Simulator kam das Basilisk Astrodynamics Simulation Framework [12] zum Einsatz.

2.1. Simulationsmodell

Ein Simulationsmodell des InnoCube wurde auf Grundlage von Spezifikationen, Komponenten-Datenblättern, CAD-Daten für die Trägheitsmomente sowie Experimentdaten aus der Entwicklung des InnoCube-Satelliten erstellt und parametrisiert. TAB 1 zeigt die wichtigsten für die Aktuatoren relevanten Eigenschaften des Simulationsmodells.

Parameter	Wert
Sat. Trägheit	$[4.28, 4.22, 0.985] \cdot 10^{-2}$ kg·m^2
RW max Omega	$[1.64, 1.64, 1.64] \cdot 10^{4}$ U/min
RW Trägheit	$[5.68, 5.68, 5.68] \cdot 10^{-5}$ kg·m^2
RW min Drehmoment	$[1.0, 1.0, 1.0] \cdot 10^{-5}$ Nm
RW max Drehmoment	$[2.0, 2.0, 2.0] \cdot 10^{-3}$ Nm
MT max Dipolmoment	$[2.0, 2.0, 2.0] \cdot 10^{-1}$ Am^2

TAB 1. Wesentliche Modellparameter, die für die Dynamik der Lageregelung relevant sind.

Als Rahmenbedingungen für die Simulationen wurden die initialen Drehzahlen der Reaktionsräder (Reaction Wheels; RWs) für jedes Rad unabhängig voneinander zufällig auf -500 oder 500 U/min gesetzt, wobei sich der Satellit in Ruhelage befand. Darüber hinaus wurden während des Trainings die Trägheitsmomente des Satelliten um $\pm$15 % pro Achse variiert. Die Ziel-RW-Drehzahlen wurden dynamisch auf den nächstgelegenen Wert (500 U/min oder -500 U/min) gesetzt. Der Orbit des Satelliten wurde auf der Grundlage von InnoCube TLE-Daten von CelesTrak [13] parametrisiert und simuliert. Diese Orbitparameter wurden während des Fine-Tunings durch Nachtraining und der Evaluation auf Basis dieser Daten variiert. TAB 2 führt die Orbitparameter des InnoCube-Satelliten und deren Variationen auf.

Parameter	Wert	Variation (offset)
Perigäum	508 km	$U \sim$(-5, 5) km
Apogäum	519 km	$U \sim$(-5, 5) km
Exzentrizität	$7.630 \cdot 10^{-4}$	$U \sim(-1 \cdot 10^{-4}, 3 \cdot 10^{-4})$
Inklination	97.43°	$U \sim$(-0.03, 0.03)°
RAAN	$U \sim$(0, 360)°	
Arg. der Periapsis	$U \sim$(0, 360)°	
Wahre Anomalie	$U \sim$(0, 360)°	

TAB 2. Übersicht der Orbitparameter und ihrer Variationen.

Die Zeitauflösung der Simulation betrug 1 Hz, während die Dynamik des Systems mit 10 Hz aktualisiert wurde.

2.2. Restdipolmomentschätzung

Für das Momentum-Management mittels der Magnettorquer (MTs) ist das Wissen über das magnetische Restdipolmoment des Satelliten von entscheidender Bedeutung. Ein nicht kompensiertes Restdipolmoment führt zu ungewollten Wechselwirkungen mit dem Erdmagnetfeld, wodurch das Regelverhalten des Systems erheblich gestört werden kann. Die verwendeten Magnettorquer im InnoCube-Satelliten haben ein max. Dipolmoment von ca. 0.35 Am^2, welches durch Regelung der Spannung bzw. des Stroms durch die Spulen verändert werden kann. Die Spulen sind in zweifacher Ausführung in jeder Achse vorhanden, wodurch insgesamt max. etwa 0.7 Am^2 an Bord zur Verfügung stehen. Die Magnettorquer müssen zum einen das Restdipolmoment ausgleichen können, zum anderen muss noch genug regelbares Dipolmoment zur aktiven Regelung vorhanden sein.
Zur Schätzung des Restdipolmoments wurde auf das in [14] beschriebene Verfahren aufgebaut und um Gyro-Bias erweitert. Die magnetischen Restdipolmomente und der Gyro-Bias wurden auf Basis dieses Verfahrens durch ein PyTorch [15]-Skript mithilfe von Gradient Descent gelernt. Zur ersten Überprüfung wurden Restdipolmomentdaten mit Gyro-Bias mit bekannter Ground Truth durch Simulation erzeugt und durch das Lernverfahren erfolgreich bestimmt. Anschließend wurde das Verfahren auf Telemetriedaten des InnoCube-Satelliten angewandt. Hierbei wurde das magnetische Restdipolmoment sowie der Gyro-

Bias aus einer Menge von Telemetrie-Datensätzen ohne Aktuator-Einfluss ermittelt und als Referenzwert (Baseline) genutzt. Die gleichzeitige Schätzung des Gyro-Bias war notwendig, da eine On-Board Kalibrierung zum Zeitpunkt der Experimente noch nicht durchgeführt worden war.
Zur weiteren Plausibilitätsprüfung wurden im Anschluss systematisch Experimente mit den Magnettorquern des Satelliten durchgeführt, bei denen die Magnettorquer sequenziell jeweils in einer Richtung (z. B. positive X-Achse, negative X-Achse, ...) aktiviert wurden. Die daraus resultierenden Änderungen in den Magnetfeldmessdaten wurden anschließend genutzt, um neue Dipolmomente unter Aktuatoreinfluss zu bestimmen. Diese Dipolmomente wurden paarweise in der relevanten Achse mit den Baseline-Restdipolmomenten verglichen, und die durch die Magnettorquer erzeugten Dipolmomente berechnet. Diese lagen zwischen 0.31 Am^2 und 0.39 Am^2 und entsprachen damit den theoretischen berechneten Dipolwerten. Die gemessenen Reaktionen des Systems standen damit im Einklang mit den theoretisch zu erwartenden Werten, wodurch die Korrektheit der Torquer-Kennwerte bestätigt werden konnten. Der gelernte Gyro-Bias wurde als Korrekturterm für die Verarbeitung der bisherigen Telemetriedaten sowie zur Kalibrierung an Bord verwendet und deckt sich mit dem beobachteten Verhalten des Satelliten.
Die bestimmten Restdipolmomente waren [-0.459, -0.024, 0.069] Am^2. Wie zuvor während des InnoCube-Betriebs beobachtet, wurde ein erhöhtes magnetisches Restdipolmoment in negativer X-Achse bestätigt. Dieses war mit -0.459 Am^2 höher als das Dipolmoment, das durch einen einzelnen Magnettorquer erzeugt werden konnte. Durch ein Softwareupdate wurde die Möglichkeit geschaffen, die redundanten Torquer als zusätzlichen Aktuator für den Ausgleich des Restdipolmoments mitzubenutzen. Durch den Ausgleich des Restdipolmoments verringert sich der störende Einfluss der erzeugten Drehmomente auf den Satelliten, was für die Regelungsqualität sowie für das Vermeiden von zu schneller Sättigung der Reaktionsräder wichtig ist. Für den eigentlichen Regler stehen nach dem Dipolausgleich weiterhin mindestens 0.2 Am^2 in allen Achsen zur Verfügung. Für die Simulation wird ein zufälliger Fehler von bis zu $\pm 10\,\%$ beim Ausgleich des Restdipolmoments angenommen und pro Episode (entspricht einem Manöver) gleichverteilt gesampled, um Unsicherheiten in der Dipolmomentschätzung und -kompensierung einzuschließen.

2.3. Trägheitsmomentschätzung

Die nach dem CAD-Modell berechneten Hauptträgheitsmomente, als Diagonalelemente ausgedrückt, lagen bei [0.0421, 0.0423, 0.0101] $kg{\cdot}m^2$. Hierbei ist die Z-Achse die Längsachse des Satelliten. Eine experimentelle Messung der Trägheitsmomente am Boden wurde nicht durchgeführt. Für die Schätzung der Trägheitsmomente wurden unabhängig die Trägheitsmomente aus dem CAD-Modell, eine manuelle Schätzung aus den Telemetriedaten und eine analytische Berechnung (ebenfalls auf Basis von Telemetriedaten) durchgeführt und abgeglichen. Die Trägheitsmomente wurden daraufhin als Ergebnis korrigiert zu [0.0428, 0.0422, 0.00985] $kg{\cdot}m^2$. Die Abweichung zwischen den aus den Telemetriedaten und durch das CAD-Modell bestimmten Trägheitsmomenten betrug maximal 2.5 % pro Achse. Für das Training des KI-Reglers wurden die Trägheitsmomente zusätzlich um $\pm 15\,\%$ pro Achse pro Episode variiert.

2.4. Sensorrauschmodell

Zum Erstellen des Sensorrauschmodells mussten zunächst Daten erhoben werden. In die Regelung gehen die Daten der Drehratensensorik (MEMS-Gyroskope) sowie der Magnetometer (magneto-induktiv) direkt ein. Das Rauschverhalten sowie die allgemeine Sensorcharakteristik in Bezug auf Umwelteinflüsse musste daher bestimmt werden. Hierzu wurde das Qualifikationsmodell (EQM) des InnoCube-Satelliten in einer Thermal-Vakuum-Kammer (TV-Kammer) aufgebaut und Messdaten aufgezeichnet. Um Effekte durch Konvektion zu vermeiden, wurde die TV-Kammer auf einen Restdruck von 10^{-4} mBar evakuiert. Anschließend wurde die Kammer aktiv abgekühlt. BILD 1 zeigt das angebrachte InnoCube-EQM in der TV-Kammer. Der thermische Kontakt besteht hauptsächlich zwischen den unteren Schienen des Satelliten sowie der Kupferaufnahme-Schienen der TV-Kammer. BILD 2 zeigt den gesamten TV-Kammer-Testaufbau. Es ist zu erkennen, dass der Abstand der Kühlaggregate (quadratische Maschine rechts unten) zum EQM zu gering ist, um Messungen ohne magnetische Störeinflüsse zu gewährleisten. Um dies zu vermeiden, wurde die Kammer nach Erreichen der Zieltemperatur vom aktiven Vakuum-Pumpensystem getrennt, die Temperierung abgeschaltet, und das EQM erwärmte sich in der TV-Kammer auf natürliche Weise (hauptsächlich durch Thermalübertrag von Struktur zu TV-Kammer-Grundplatte und Strahlungsabgabe) wieder auf Raumtemperatur (ca. 30 °C). Die natürliche Erwärmung wurde gewählt, da die gleichzeitige Nutzung der aktiven Temperierung störende Einflüsse auf die Sensordaten durch Vibration sowie magnetische Störfelder zur Folge hätte. Die dadurch entstandenen Temperaturkurven zeigten eine asymptotische Annäherung an die Raumtemperatur. Dieser Verlauf wurde bei der anschließenden Datenauswertung berücksichtigt und ausgeglichen.
Die Daten wurden so zugeschnitten, dass der gemessene Temperaturbereich der verbleibenden Daten den im Orbit gemessenen Temperaturbereich von -3.55 °C bis 15.6 °C abdeckt. Die Experimente wurden mehrfach wiederholt und die Sensordaten inklusive der Temperatur aufgezeichnet. Für die Auswertung wurden Daten von insgesamt fünf Experimenten verwendet.

Das Sensorrauschmodell für die Magnetometer und Gyros modelliert deren Rauschverhalten und besteht aus zwei Komponenten:

- Bias (v_b): Ein langsam variierender Fehler oder eine Verschiebung des Sensorwerts (gesampled pro Episode).
- Weißes Rauschen (v_w): Zufälliges, schnell variierendes Rauschen (gesampled pro Zeitschritt).

Die Rohdaten jedes Sensors wurden in aufeinanderfolgende Segmente (künstliche “Manöver”) zu je 60 Sekunden Länge unterteilt. Es wird angenommen, dass der Bias innerhalb eines Segments vergleichsweise konstant bleibt, aber über die Segmente hinweg variieren kann.

Berechnung des Bias (v_b) und seiner Standardabweichung (σ_b):

- Für jedes Segment wurde der Mittelwert berechnet. Diese Mittelwerte repräsentieren die geschätzten “Bias”-Werte für jedes einzelne Manöver.
- Die Standardabweichung dieser Segmentmittelwerte wurde dann als σ_b (Bias-Standardabweichung pro Manöver) berechnet. Ein großer Wert für σ_b deutet darauf hin, dass der Bias des Sensors stark zwischen den verschiedenen künstlichen Manövern variiert.
- Da aufzeichnungsbedingt mehr Daten für Temperaturen nahe der Raumtemperatur verfügbar waren, wurden die Segmentmittelwerte temperaturbedingt gewichtet. Hierbei wurde die Häufigkeitsverteilung der mittleren Temperaturen der Segmente über ein Histogramm analysiert und gewichtet. Dadurch wurde sichergestellt, dass die Berechnung von σ_b nicht von überrepräsentierten Temperaturbereichen dominiert und die Variabilität des Bias über den gesamten relevanten Temperaturbereich besser erfasst wird.

Berechnung des weißen Rauschens (w_w) und seiner Standardabweichung (σ_n):

- Für jeden Datenpunkt innerhalb eines Segments wurde das Residuum durch $Datenpunktwert - Segmentmittelwert$ berechnet. Diese Residuen stellen das “Restrauschen” dar, nachdem der Bias des Segments entfernt wurde. Wir nehmen an, dass dieses Restrauschen hauptsächlich weißes Rauschen ist.
- Die Standardabweichung aller dieser Residuen über den gesamten Datensatz (aller Segmente) wurde als σ_n (Standardabweichung des weißen Rauschens) berechnet. Ein großer Wert für σ_n deutet auf ein hohes Maß an zufälligem, schnellem Rauschen in den Sensormessungen hin.

Aus den berechneten Werten können nun neue Rauschwerte v_b und v_w gesampled werden. Für die Simulation wurden die σ_b und σ_n zusätzlich noch mit dem Faktor 1.2 skaliert, um die Rahmenbedingungen weiter zu fassen. Anschließend wurden zu den exakten simulierten Werten v_{sim} die Rauschwerte v_b und v_w addiert, um den effektiven simulierten Sensorwert v_{eff} zu erhalten, wie in den Gleichungen (1)–(3) dargestellt. Hierbei wurde v_b pro Episode und v_w pro Zeitschritt gesampled.

$$v_b \sim \mathcal{N}(0, \sigma_b^2) \tag{1}$$

$$v_w \sim \mathcal{N}(0, \sigma_n^2) \tag{2}$$

$$v_{eff} = v_{sim} + v_b + v_w \tag{3}$$

Es ist zu beachten, dass eine Kalibrierung der realen Sensoren, wie in Abschnitt 2.2 beschrieben, weiterhin notwendig ist. Das Rauschmodell wird auf Basis von Residuen berechnet und beschreibt daher das Restrauschen, das auch die temperaturabhängigen Variationen der Gyroskope beinhaltet.

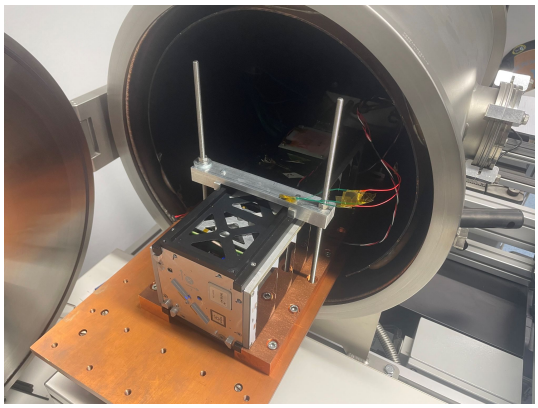


BILD 1. InnoCube-EQM in der Thermal-Vakuum-Kammer.

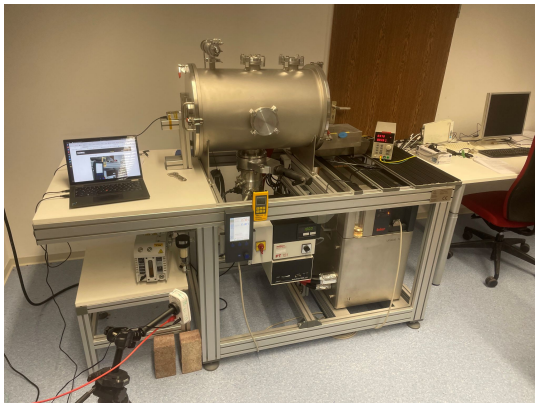


BILD 2. InnoCube-EQM in der Thermal-Vakuum-Kammer (Außenansicht).

2.5. Beobachtungs- und Aktionsraum

Der Beobachtungsraum wurde als 39-dimensionaler Vektor definiert, der aus den in TAB 3 aufgeführten Komponenten besteht. Alle Werte der Beobachtungen wurden so skaliert, dass sie innerhalb des Intervalls $[-1, 1]$ liegen. Der Aktionsraum ist ein sechsdimensionaler Vektor mit drei Elementen für die zu kommandierenden Drehmomente der Reaktionsräder und drei Elementen für die zu kommandierenden magnetischen Dipolmomente der Magnettorquer. Der Wertebereich der vom KI-Agenten produzierten Aktionen liegt in $[-1, 1]$ und wird während der Simulation

mit den jeweiligen maximalen Drehmomenten bzw. Dipolmomenten skaliert, um die effektiven Aktionen zu erhalten, die anschließend umgesetzt werden.

Komp. (Länge)	Beschreibung
$err\_quat_{t_i}$(4)	Aktuelles Lagefehler-Quaternion
$err\_rr_{t_i}$(3)	Aktueller Satelliten-Drehratenfehler
$a\_rw_{t_{i-1}}$(3)	Zuletzt durchgeführte RW-Aktion
$err\_quat_{t_{i-1}}$(4)	Vorheriges Lagefehler-Quaternion
$err\_rr_{t_{i-1}}$(3)	Vorheriger Satelliten-Drehratenfehler
rwr_{t_i}(3)	Aktuelle RW-Drehzahlen
$rwr_{t_{i-1}}$(3)	Vorherige RW-Drehzahlen
$mag\_b_{t_i}$(3)	Aktuelles Magnetfeld
$mag\_b_{t_{i-1}}$(3)	Vorheriges Magnetfeld
$err\_rwr_{t_i}$(3)	Aktueller RW-Drehzahlfehler
$err\_rwr_{t_{i-1}}$(3)	Vorheriger RW-Drehzahlfehler
crm_{t_i}(3)	Kreuzprodukt aus RW-Drehzahlen und Magnetfeld, geteilt durch 2
bn_{t_i}(1)	Norm des aktuell gemessenen Magnetfelds

TAB 3. Komponenten des Beobachtungsraums mit Länge und Beschreibung.

Der Satelliten-Drehratenfehler, $err\_rr_{t_i}$, entspricht hierbei den Satelliten-Drehraten, da die Zieldrehraten 0 sind. Der RW-Drehzahlfehler, $err\_rwr_{t_i}$, ist der Betrag der Differenz zwischen den aktuellen und den Zieldrehzahlen der Reaktionsräder. Die Magnetfelddaten stammen von (simulierten) Magnetometern und sind lageabhängig.

2.6. Belohnungsfunktionen und Hyperparameter

Der KI-Agent wird trainiert, Inertial-Pointing-Manöver durchzuführen, wobei das Verfahren grundsätzlich auch zum Lernen anderer Lageregelungsmodi angewandt werden kann. Für das Training der Reaktionsräder haben wir eine modifizierte Version unserer Belohnungsfunktion aus [1, 3] verwendet, wie in den Gleichungen (4)–(9) aufgeführt.

(4) $$err\_att_{t_i} = 1 - |err\_quat_{t_i}[0]|$$

(5) $$\Delta\theta_{t_i} = err\_att_{t_i} - err\_att_{t_{i-1}}$$

(6) $$p\_norm_{t_i} = \|err\_rr_{t_i}\|$$

(7) $$p\_att_{t_i} = \frac{err\_att_{t_i}}{10}$$

(8)
$$r_{t_i} = \begin{cases} 1 + \frac{1}{p\_norm_{t_i}+0.1}, & \text{if } err\_att_{t_i} < 3.8 \cdot 10^{-5} \\ e^{-\frac{err\_att_{t_i}}{0.14}}, & \text{elif } err\_att_{t_i} < err\_att_{t_{i-1}} \\ 0.1 \cdot e^{-\frac{\Delta\theta_i}{0.14}} - 1 & \text{elif } \Delta\theta_{t_i} < \Delta\theta_{t_{i-1}} \\ e^{-\frac{err\_att_{t_i}}{0.14}} - 2, & \text{else} \end{cases}$$

(9) $$reward\_rw_{t_i} = r_{t_i} - p\_norm_{t_i} - p\_att_{t_i}$$

Hierbei ist $|err\_quat_{t_i}[0]|$ die skalare Komponente des Fehlerquaternions zum Zeitpunkt t_i. Die Terme $p\_norm_{t_i}$ und $p\_att_{t_i}$ sind regularisierende Strafterme (penalty terms; p) für die Satellitendrehraten und den Lagefehler. Die erste Komponente von r_{t_i} belohnt den Agenten stark, wenn dieser die Ziellage mit einer Toleranz von ca. 1° erreicht. Die zweite Komponente belohnt den Agenten geringfügig, wenn dieser sich der Ziellage annähert. Die dritte Komponente bestraft den Agenten geringfügig, wenn dieser sich zwar von der Ziellage wegbewegt, die Zunahme der Distanz jedoch abnimmt. Dies dient dazu, den Agenten zur Korrektur bei Übersteuerung zu bringen, ohne jedoch einen Anreiz zu geben um die Ziellage zu oszillieren. Die vierte Komponente bestraft den Agenten stark, wenn dieser sich von der Ziellage wegbewegt, ohne die Drehgeschwindigkeit zu reduzieren. Die kumulative RW-Belohnung ist maximal, wenn die Ziellage so schnell wie möglich erreicht und dann konstant gehalten wird.
Die Belohnungsfunktion für die Magnetorquer wird durch die Gleichungen (10)–(12) angegeben.

(10) $$err\_deviation_{t_i} = \sum err\_rwr_{t_i} \cdot 16384.0$$

(11) $$p\_action_{t_i} = \frac{\sum |a\_m_{t_i}|}{0.6 \cdot 50}$$

(12) $$reward\_mt_{t_i} = \frac{1 - p\_action_{t_i}}{\sqrt{err\_deviation_{t_i} + 1}}$$

Die Multiplikation von $err\_rwr_{t_i}$ mit 16384.0 dient der Ent-Normalisierung. $a\_m_{t_i}$ bezeichnet die MT-Aktionen. Der Strafterm $p\_action_{t_i}$ wird als Regularisierungsterm verwendet, um den Agenten dazu anzuregen, die Dipolmomente möglichst gering zu halten. Die Belohnung ist reziprok abhängig von den RW-Drehzahlabweichungen und wird um die Stärke der MT-Aktionen skaliert. Die kumulative MT-Belohnung ist maximal, wenn die RW-Drehzahlabweichung so schnell wie möglich minimiert wird und mit möglichst geringen MT-Aktionen gehalten wird. Für das Fine-Tuning durch Nachtraining wurde eine gemeinsame Belohnungsfunktion verwendet, die in Gleichung (13) gezeigt ist.

(13) $$reward\_c_{t_i} = \frac{reward\_rw_{t_i}}{11} + reward\_mt_{t_i}$$

Der Anteil der RW-Belohnung wurde hier mit 11 skaliert, um die Wertebereiche der RW- und MT-Belohnungen aneinander anzugleichen. Dies soll einen Bias in Richtung eines Aktuatortyps verhindern. Um oszillierende Steuersignale zu vermeiden, wurde generalized State Dependent Exploration (gSDE) [16, 17] verwendet.
Eine bedeutende Herausforderung ergab sich aus den widersprüchlichen Trainingsanforderungen der RWs und MTs. Während die RWs die Ziellagen in etwa 20 Sekunden einnehmen können, benötigen die MTs etwa zwei Größenordnungen länger, um die

Zieldrehzahlen zu erreichen. Folglich konzentriert sich das Training der RWs auf unmittelbare Belohnungen, während das Training der MTs langfristige Belohnungen in den Vordergrund stellt. Darüber hinaus wird das MT-Training durch zwei weitere Probleme erschwert. Erstens hat eine einzelne Aktion der MTs für sich genommen nur minimale Auswirkungen, und jede Aktion kann leicht von den RWs konterkariert werden. Zweitens sind die MTs mit der Einschränkung der Unteraktuierung konfrontiert. Selbst eine korrekte MT-Aktion kann wirkungslos bleiben, wenn die Lage des Satelliten zum Erdmagnetfeld ungünstig ist, oder sogar bestraft werden, wenn eine negative Belohnung folgt, auch wenn die Ursache hierfür nicht in der MT-Aktion liegt. Dies führt zum sogenannten Credit-Assignment-Problem, bei dem nicht eindeutig bestimmt werden kann, inwieweit einzelne Aktionen während des Trainings zur Erreichung des Ziels beigetragen haben.
Um diese Herausforderungen zu bewältigen, verwenden wir ein Policy-Netzwerk, bestehend aus dedizierten Subnetzwerken für jeden Aktortyp. TAB 4 zeigt die verwendeten Hyperparameter und Netzwerkarchitekturen, während die Abschnitte 2.7–2.10 den Trainingsablauf beschreiben.

Hyperparameter	RW Training/ Nachtraining	MT Training/ Nachtraining
total_timesteps	$5 \cdot 10^8$	$5 \cdot 10^8$
n_steps (rollout)	$2 \cdot 10^3$ / $4 \cdot 10^4$	$4 \cdot 10^4$
gamma	0.95	0.97 / 0.95
batch_size	64	64
clip_range	0.2	0.2
target_kl	0.2	0.2
learning_rate (init)	$1 \cdot 10^{-4}$ / $1 \cdot 10^{-5}$	$5 \cdot 10^{-5}$
sde_sample_freq	1 pro rollout	1 pro rollout
log_std_init	-2.0	0.0 / -2.0
episode_length	50 / 5000	5000
num_skip	0	9 / 0
Gewichtsinitialisierung	Orthogonale Gewichtsinitialisierung mit 0.01 für das RW action network, 10.0 für das MT action network und 1.0 für alle anderen Netzwerke	
Netzwerke	RW Policy / Value Netzwerk: [64, 64, 64] / [64, 64] MT Policy / Value Netzwerk: [64, 64, 64, 64] / [64, 64, 64, 64] Aktivierungsfunktion: SiLU	

TAB 4. Hyperparameter und Netzwerkarchitekturen der RW- und MT-Subnetzwerke. Änderungen der Parameter beim Nachtraining sind nur angegeben, wenn diese sich von den Hyperparametern des regulären Trainings unterscheiden.

Während des Fine-Tunings durch Nachtraining wird, wie in Abschnitt 2.10 beschrieben, das Value-Netzwerk [64, 64, 64, 64] verwendet und mit den Werten aus dem MT-Training initialisiert.

2.7. Subnetz-Politik

Ein KI-Agent, der mit Proximal Policy Optimization (PPO) trainiert wurde, umfasst typischerweise drei Hauptkomponenten: einen Feature-Extraktor (das Policy-Netzwerk), der Beobachtungen in latente Features (einen Vektor von Gleitkommazahlen) umwandelt; ein Action-Netzwerk, das diese Features in eine (gaußsche) Verteilung abbildet, aus der die Aktionen gesampled werden; und ein Value-Netzwerk, das die erwartete zukünftige Belohnung für eine gegebene Beobachtung unter der aktuellen Politik schätzt. Das Trainingsverfahren folgt dem in [10] beschriebenen Algorithmus. PPO optimiert die Politik mithilfe einer "geclippten" Surrogat-Zielfunktion, die große Politikänderungen begrenzt, um die Stabilität des Trainings zu erhöhen. Der Agent sammelt Erfahrungen durch Interaktion mit der Umgebung und speichert diese in einem sogenannten "Rollout-Buffer". Anschließend werden die Politik-Parameter mithilfe von Mini-Batch-Gradient-Ascent aktualisiert, mit dem Ziel, die erhaltenen kumulativen Belohnungen zu maximieren. Gleichzeitig wird das Value-Netzwerk aktualisiert, um die sogenannten "Advantage-Schätzungen" zu verbessern. Diese quantifizieren, wie viel besser oder schlechter eine durchgeführte Aktion im Vergleich zum erwarteten Wert des Zustands (Beobachtung) war. Diese Advantage-Werte skalieren die Politik-Gradienten, verstärken günstige Aktionen und bestrafen ungünstige. Für eine ausführlichere Beschreibung verweisen wir auf [10].
Im Folgenden werden das Policy- und Action-Netzwerk gemeinsam als Politiknetzwerk bezeichnet, welches für die Erzeugung der Steuersignale für die Aktuatoren verantwortlich ist. Wir teilen das Politiknetzwerk und verwenden ein Subnetzwerk für die Reaktionsräder (vergleichbar mit einem RW-Agenten) und ein Subnetzwerk für die Magnettorquer (vergleichbar mit einem MT-Agenten), wobei jedes nur Aktionen für dessen jeweiligen Aktuatortyp erzeugt. Die Ausgaben beider Subnetzwerke werden dann konkateniert, was einen sechselementigen Aktionsvektor mit drei Komponenten für die RW-Drehmomente und drei Komponenten für die MT-Dipolmomente ergibt. Ebenso sind die gSDE-relevanten Komponenten wie die Explorationsmatrizen geteilt. Jedes Subnetzwerk kann sich in einem von drei Zuständen befinden: DISABLED, FROZEN oder ACTIVE. Im Zustand DISABLED produziert das Subnetzwerk keine Ausgabe (einen Nullvektor), und seine Aktionswahrscheinlichkeiten sind auf 1 gesetzt, während seine Entropie 0 ist. Zusätzlich sind die Gewichte eines Subnetzwerks in diesem Zustand nicht trainierbar. Im Zustand FROZEN produziert das Subnetzwerk Aktionen basierend auf seiner Politik, aber wie im DISABLED-Zustand trägt es nicht zum Training bei, und seine Gewichte sind nicht veränderbar. Im Zustand ACTIVE erzeugt das Subnetzwerk Aktionen entsprechend seiner Politik, nimmt vollständig am Training teil, und seine Gewichte sind anpassbar.

2.8. SkipPPO-Trainer

Neben den Unterschieden in den Anforderungen der Aktuatoren an die Trainings-Hyperparameter und die Belohnungsfunktion bleibt die Herausforderung des langen Zeithorizonts der Magnettorquer bestehen. Um diesem Problem zu begegnen, haben wir den PPO-Algorithmus modifiziert, um ein optionales Einfrieren von Aktionen zu ermöglichen. Wenn das Skip-Training genutzt wird, werden während des Trainings initial Steuersignale für die RWs und MTs generiert, und anschließend werden die MT-Aktionen für num_skip Zeitschritte eingefroren, während für die RWs weiterhin zu jedem Zeitschritt neue Aktionen generiert werden. Dies reduziert die effektive Episodenlänge für das MT-Subnetzwerk auf $episode\_length/(num\_skip + 1)$. Belohnungen, die während der Skip-Schritte erzielt werden, werden im Rollout-Buffer aggregiert und der entsprechenden initialen Aktion zugeordnet, wodurch das Belohnungssignal verstärkt und das Credit-Assignment-Problem abgemildert wird.

2.9. Training

Das Training des KI-Reglers war unterteilt in das initiale Training des Basis-Agenten und das anschließende Fine-Tuning. Für das initiale Training wurden nur die Orbitparameter- und Trägheitsmomentvariationen berücksichtigt. Für das Fine-Tuning durch Nachtraining und die Evaluationen aus Abschnitt 3 wurden alle Variationen der Rahmenbedingungen angewendet. Hintergrund war, dass der KI-Agent zunächst die grundlegende Problemstruktur lernen sollte, um einen Basis-Agenten zu erhalten, der dann einfacher mit unterschiedlichen Störmomenten nachtrainiert werden kann.
Während des initialen Trainings wurde das RW-Subnetzwerk mit der RW-Belohnungsfunktion und den Hyperparametern aus Abschnitt 2.6 trainiert. Zu Beginn jeder Episode wurde das Satellitenmodell mit einer zufälligen Ausgangslage initialisiert und musste eine zufällige Ziellage ohne Berücksichtigung der RW-Drehzahlbeschränkungen erreichen. Anschließend wurde das RW-Netzwerk eingefroren, und das MT-Netzwerk mit den MT-Hyperparametern und der MT-Belohnungsfunktion trainiert. Hierbei hatte der Agent das Ziel, RW-Drehzahlen von $\pm$500 U/min zu erreichen, während das RW-Subnetzwerk weiterhin zufällige Ziellagen einnahm. Dieser Trainingsansatz sorgte dafür, dass die Konflikte zwischen den Aktuatoranforderungen, Hyperparametern und den Belohnungsfunktionen aufgelöst wurden. Das Ergebnis des Trainings war ein Basis-Agent.

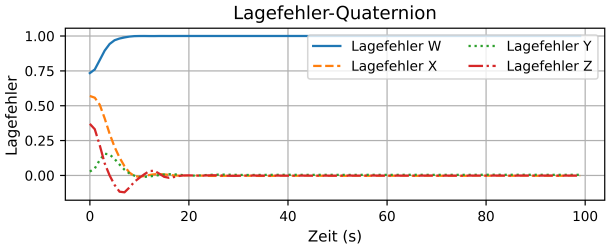
2.10. Nachtraining

Die Adaption an das Rauschen und die Restdipolmomente erfolgte durch das anschließende Nachtraining. Nach einem initialen Trainingslauf sind zwar beide Subnetzwerke trainiert, jedoch sind die Aktionen des MT-Subnetzwerks noch effektiv Störmomente für das RW-Subnetzwerk, da diese nicht Teil dessen initialen Trainings sind. Durch das Nachtraining soll das RW-Subnetzwerk lernen, damit umzugehen. Hierdurch ändert sich wiederum das Verhalten des RW-Subnetzwerks, sodass eine Anpassung des MT-Subnetzwerks vorteilhaft wäre, etc.
Für das Nachtraining wurden mehrere Herangehensweisen untersucht. In der ersten Variante wurde nach dem initialen Training das RW-Subnetzwerk aktiviert, während das MT-Subnetzwerk eingefroren war, und dediziert das RW-Subnetzwerk mit der RW-Belohnungsfunktion nachtrainiert. Anschließend wurde das RW-Subnetzwerk wieder eingefroren, das MT-Subnetzwerk aktiviert und ebenfalls mit dessen eigener Belohnungsfunktion nachtrainiert. In beiden Schritten wurde das Value-Netzwerk aus dem initialen MT-Training verwendet und nach jedem RW-MT-Wechsel neu initialisiert. Der Prozess wurde mehrfach wiederholt. Dieser Ansatz wurde jedoch wegen des hohen Overheads, der sich aus dem erneuten Trainieren des Value-Netzwerks ergibt, und der weiterhin verbleibenden Spezialisierung auf zwei getrennte Belohnungsfunktionen verworfen.
Um einen höherfrequenten Wechsel zu ermöglichen und damit eine häufigere Abstimmung der Subnetzwerke aufeinander, wurde der Trainer weiter modifiziert, um zwei Konfigurationen gleichzeitig entgegenzunehmen, und um die Definition einer Wechselfrequenz erweitert. Hiermit kann der Trainer den Wechsel zwischen RW- und MT-Subnetzwerktraining selbstständig während eines einzigen Trainingslaufs mit einer einstellbaren Frequenz durchführen. Aus Stabilitätsgründen wird die gemeinsame Belohnungsfunktion aus Gleichung (13) genutzt, die sich aus einer gewichteten Summe der Original-Belohnungsfunktionen der Subnetzwerke ergibt. Das Optimierungsziel schließt damit stets beide Aktuatoren ein, und da das zugrundeliegende Lageregelungsproblem bereits durch das initiale Training gelernt wurde, ist eine Adaption von beiden Aktuatoren gleichzeitig wesentlich einfacher als ein Training von Grund auf. Weiterhin kann, da keines der Subnetzwerke deaktiviert ist und die Belohnungsfunktion nicht wechselt, das Value-Netzwerk weiterhin zuverlässig die erwartete zukünftige Belohnung vorhersagen, ohne ausgetauscht oder neu trainiert werden zu müssen. Das Value-Netzwerk wird für das Nachtraining mit den Gewichten aus dem MT-Training des Basis-Agenten initialisiert.

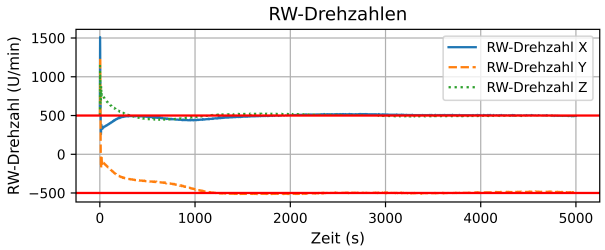
3. ERGEBNISSE

Die Subnetzwerke wurden unter Verwendung der in den Abschnitten 2.7–2.10 beschriebenen Politik und des SkipPPO-Trainers trainiert. Jede Konfiguration wurde alternierend für jeweils 10 Rollouts für das RW-Subnetzwerk und 100 Rollouts für das MT-Subnetzwerk mit einer Episodenlänge von 5000 Zeitschritten und num_skip = 0 angewandt. Der Gamma-Hyperparameter wurde aus Stabilitätsgrün-

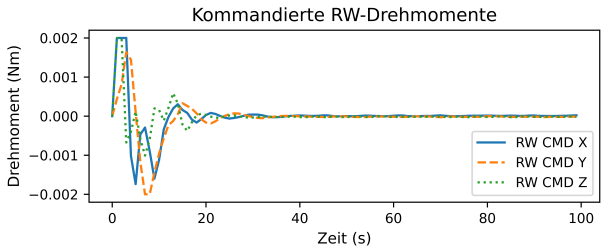
den für beide Konfigurationen auf 0.95 gesetzt. Während des Trainings wurde die initiale Lernrate quadratisch reduziert, da (g)SDE dazu neigt, zunächst zu einer effektiven Politik zu konvergieren, diese jedoch anschließend wieder zu verlernen, was vermutlich auf Überexploration zurückzuführen ist [16]. BILD 3 zeigt beispielhaft den Verlauf des Lagefehlerquaternions, der Reaktionsraddrehzahlen, der kommandierten RW-Drehmonente und der kommandierten MT-Dipolmomente für eine Episode für den nachtrainierten KI-Regler.



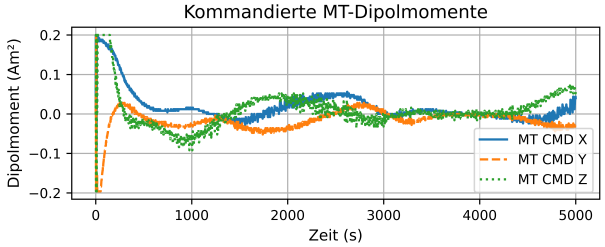
(a) Lagefehler-Quaternion über 100 Zeitschritte.



(b) RW-Drehzahlen über 5000 Zeitschritte.



(c) Kommandierte RW-Drehmomente über 100 Zeitschritte.



(d) Kommandierte MT-Dipolmomente über 5000 Zeitschritte.

BILD 3. Lagefehler-Quaternion über 100 Zeitschritte (a), RW-Drehzahlen über 5000 Zeitschritte (b), kommandierte RW-Drehmomente über 100 Zeitschritte (c), und kommandierte MT-Dipolmomente über 5000 Zeitschritte (d) für eine repräsentative Episode des nachtrainierten KI-Reglers.

Der KI-Agent war in der Lage, die Ziellagen zuverlässig einzunehmen und beizubehalten. Außerdem wurde das Momentum-Management erfolgreich durchgeführt, und die RW-Drehzahlen in die Nähe von ±500 U/min gebracht und gehalten. TAB 5 zeigt die Ergebnisse des vereinigten KI-Reglers vor dem Fine-Tuning durch Nachtraining und TAB 6 zeigt die Ergebnisse nach dem Fine-Tuning. Die Ergebnisse wurden jeweils über 10.000 Episoden mit einer Episodenlänge von 5.000 Zeitschritten mit num_skip = 0, zufälligen Initial- und Ziellagen sowie mit den in den Abschnitten 2.1–2.4 beschriebenen Variationen, berechnet. Die Metriken umfassen Rise Time (Lage), Einschwingzeit (Lage), Steady-State-Fehler (Lage), Einschwingzeit (RW-Drehzahlen) und den Regelungsaufwand (MT).

Metrik	μ	σ
Lage Rise Time (s; 90 % → 10 %)	8.86	2.68
Lage Einschwingzeit (s)	44.80	349.95
Lage Steady-State Fehler (°)	0.97	0.24
RW-Drehzahl Einschwingzeit (min)	16.30	14.17
MT-Regelungsaufwand (Am²s)	453.75	145.94

TAB 5. Mittelwert und Standardabweichung für verschiedene Metriken für den Basis-KI-Regler über 10.000 Episoden mit 5.000 Zeitschitten, num_skip = 0, und zufälligen Initial- und Ziellagen, zusammen mit den Variationen aus den Abschnitten 2.1–2.4. Toleranz Lage: 1°; Toleranz RW-Drehzahlen: 100 U/min.

Metrik	μ	σ
Lage Rise Time (s; 90 % → 10 %)	10.95	3.58
Lage Einschwingzeit (s)	22.24	5.46
Lage Steady-State Fehler (°)	0.83	0.07
RW-Drehzahl Einschwingzeit (min)	16.02	13.11
MT-Regelungsaufwand (Am²s)	450.89	141.77

TAB 6. Mittelwert und Standardabweichung für verschiedene Metriken für den nachtrainierten KI-Regler über 10.000 Episoden mit 5.000 Zeitschitten, num_skip = 0, und zufälligen Initial- und Ziellagen, zusammen mit den Variationen aus den Abschnitten 2.1–2.4. Toleranz Lage: 1°; Toleranz RW-Drehzahlen: 100 U/min.

Es ist zu erkennen, dass der nachtrainierte Regler, bedingt durch Unsicherheiten aus dem zusätzlichen Rauschen, ein konservativeres Lageregelungsverhalten zeigt. Dadurch stieg die Rise Time leicht von 8.86 s auf 10.95 s an. Im Gegenzug verbesserten sich die mittleren Lage- und RW-Drehzahl-Einschwingzeiten. Die mittlere Lage-Einschwingzeit verringerte sich erheblich von 44.80 s auf 22.24 s, während sich die Standardabweichung sogar von 349.95 s auf 5.46 s verringerte. Dies weist was auf

ein wesentlich stabileres und zuverlässigeres Regelungsverhalten in Anwesenheit des Rauschens und der Variationen der Rahmenbedingungen hin. Die Einschwingzeit der RW-Drehzahlen (Effekt des Momentum-Managements) verbesserte sich leicht, von im Mittel 16.30 min auf 16.02 min. Die zugehörige Standardabweichung verringerte sich von 14.17 min auf 13.11 min. Auch die restlichen Werte verbesserten sich geringfügig. Insgesamt zeigen die Ergebnisse, dass das Nachtraining erfolgreich war und der KI-Regler das Lageregelungsproblem unter Berücksichtigung der Drehzahlbeschränkungen effektiv bewältigen kann.

4. DISKUSSION UND AUSBLICK

In dieser Arbeit haben wir ein Verfahren vorgestellt, das Reaktionsrad- und Magnettorquerregelung mittels Subnetzpolitik und einem angepassten Trainer, in Anwesenheit von Variationen der Rahmenbedingungen und Einschränkungen bei den Reaktionsraddrehzahlen, in einem einzigen KI-Regler für Inertial-Pointing vereinigt. Unser Trainings-Verfahren adressiert erfolgreich die Anforderungen der stark unterschiedlichen Aktuatoren. Während die RWs schnelle und präzise Lageänderungen ermöglichen, dabei jedoch den Fokus auf unmittelbare Belohnungen legen, werden die MTs für vergleichsweise langfristiges Momentum-Management verwendet und erfordern die Betrachtung langer Zeithorizonte. Durch die Verwendung von Subnetzen konnten die widersprüchlichen Anforderungen an die Trainings-Hyperparameter und die Belohnungsfunktionen der RWs und MTs gelöst werden. Die Modifikation des PPO-Algorithmus mit partiellem Einfrieren der MT-Aktionen trug maßgeblich zur Bewältigung des langen Zeithorizonts der MTs bei, indem die effektive Episodenlänge für das MT-Subnetzwerk reduziert und die Belohnung durch Aggregation verstärkt wurde. Wir evaluierten den entwickelten KI-Regler durch Simulation des InnoCube-Nanosatelliten unter Berücksichtigung realistischer und variierender Rahmenbedingen, wie die der Trägheitsmomente und Orbitparameter, sowie fehlerbehaftetem Restdipolmomentausgleich und Sensorrauschen. Der KI-Regler war in der Lage das Lageregelungsproblem erfolgreich zu lösen und sowohl die Ziellagen zuverlässig zu erreichen, als auch die Ziel-RW-Drehzahlen umzusetzen. Zukünftige Arbeiten werden sich auf die weitere Optimierung des Fine-Tunings durch Nachtraining, die Verfeinerung des Trainingsverfahrens und insbesondere auf den Sim2Real-Transfer konzentrieren, um den KI-Lageregler erfolgreich auf den InnoCube-Nanosatelliten hochzuladen und ihn in Q3 2025 im Orbit zu evaluieren. Das InnoCube-Projekt wird vom Bundesministerium für Wirtschaft und Energie (BMWE) unter dem Förderkennzeichen (FKZ) 50RU2000 gefördert.

Förderung

Diese Arbeit wurde im Rahmen des LeLaR-Projekts (FKZ: 50RA2403) durchgeführt, das vom Bundesministerium für Wirtschaft und Energie (BMWE) auf Grundlage eines Beschlusses des Deutschen Bundestages gefördert wird.

Kontaktadresse:

kirill.djebko@uni-wuerzburg.de

Literatur

[1] K. Djebko, F. Puppe, S. Montenegro, T. Baumann, and M. Faisal. Learning attitude control. In *Proceedings of the 14th IAA Symposium on Small Satellites for Earth System Observation*, Berlin, Germany, May 2023. Presented at the 14th IAA Symposium on Small Satellites for Earth System Observation, May 7–12, 2023.

[2] R. Gerlich, R. Gerlich, S. Montenegro, F. Puppe, K. Djebko, C. Plasberg, and M. Bädorf. It's the data, stupid! constructive and analytical quality-assurance for ai-based space systems. Presented at DASIA 2023, June 6–8, Sitges, Spain, 2023.

[3] K. Djebko, T. Baumann, E. Dilger, F. Puppe, and S. Montenegro. Ai-based attitude control for restricted reaction wheels. In *Proceedings of the 15th IAA Symposium on Small Satellites for Earth System Observation*, Berlin, Germany, May 2025. Presented at the 15th IAA Symposium on Small Satellites for Earth System Observation, May 4–8, 2025.

[4] R. S. Sutton, and A. G. Barto. *Reinforcement learning: An introduction*. Number 1. MIT press Cambridge, 1998.

[5] S. Tsimenidis. Limitations of deep neural networks: a discussion of g. marcus' critical appraisal of deep learning. *arXiv preprint arXiv:2012.15754*, 2020.

[6] B. Grzesik, T. Baumann, T. Walter, F. Flederer, E. Dilger, F. Sittner, S. Gläsner, J. L. Kirchler, M. Tedsen, S. Montenegro, and E. Stoll. Innocube—a wireless satellite platform to demonstrate innovative technologies. *Aerospace*, 8(5), 2021. ISSN: 2226-4310. DOI: 10.3390/aerospace8050127.

[7] S. Montenegro, T. Baumann, E. Dilger, F. Sittner, M. Strohmaier, T. Walter, and S. Gläsner. InnoCubE: Der erste drahtloser Satellit. *Deutsche Gesellschaft für Luft- und Raumfahrt - Lilienthal-Oberth e.V.*, 2022. URN: urn:nbn:de:101:1-20221118110824999971 54. DOI: 10.25967/570007.

[8] T. Baumann, E. Dilger, S. Montenegro, F. Sittner, M. Arbab, and T. Walter. InnoCube – First In-Orbit Results of the Fully Wireless Satellite Data Bus. In *Proceedings of SmallSat 2025, Salt Lake City, USA*. Presented at SmallSat 2025, August 10–13, 2025. DOI: 10.26077/956f-62d3.

[9] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[10] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[11] M. Towers, A. Kwiatkowski, J. Terry, J. Balis, G. De Cola, T. Deleu, M. Goulão, A. Kallinteris, M. Krimmel, A. KG, R. Perez-Vicente, A. Pierré, S. Schulhoff, J. Tai, H. Tan, and O. Younis. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

[12] P. W. Kenneally, S. Piggott, and H. Schaub. Basilisk: A flexible, scalable and modular astrodynamics simulation framework. *Journal of aerospace information systems*, 17(9):496–507, 2020.

[13] CelesTrak. SatCat Table: INNOCUBE. https://celestrak.org/satcat/table-satcat.php?NAME=INNOCUBE, 2025. Last visited February 14, 2025.

[14] S. Busch, P. Bangert, S. Dombrovski and K. Schilling. Uwe-3, in-orbit performance and lessons learned of a modular and flexible satellite bus for future pico-satellite formations. *Acta Astronautica*, 117:73–89, 2015. ISSN: 0094-5765. DOI: doi.org/10.1016/j.actaastro.2015.08.002.

[15] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

[16] T. Rückstieß, M. Felder, and J. Schmidhuber. State-dependent exploration for policy gradient methods. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 234–249. Springer, 2008.

[17] A. Raffin, J. Kober, and F. Stulp. Smooth exploration for robotic reinforcement learning. In *Conference on robot learning*, pages 1634–1644. PMLR, 2022.