

Orbitwerkstatt

Softwaredokumentation

Version 0.14.0
Anforderungsbasis und Systembeschreibung
Stand 15. Oktober 2026

Zweck dieses Dokuments

Diese Dokumentation beschreibt den vorliegenden Prototyp zur Bestimmung und Veranschaulichung von Bahnen aus astrometrischen Beobachtungen. Sie verbindet den fachlichen Einsatz in der Lehre mit einer prüfbaren Spezifikation, der Softwarearchitektur und Hinweisen für Betrieb und Weiterentwicklung.

Adressaten sind Lehrende, astronomisch interessierte Anwender und Entwickler. Die Anwendung verarbeitet bereits gemessene Positionen. Die Auswertung von Bilddateien und die astrometrische Vermessung erfolgen außerhalb der Orbitwerkstatt.

Dokumentbasis und Verbindlichkeit

Grundlage sind der Quellcode der Version 0.14.0, die mitgelieferte Hilfe, README.md, PORTABEL_ANLEITUNG.md und PRUEFBERICHT.md. Die Anforderungen sind aus dem vorhandenen Stand rekonstruiert. Sie bilden keine nachträgliche Abnahme und keine Zusage astronomischer Genauigkeit.

MUSS kennzeichnet die spezifizierte Basisfunktion. SOLLTE kennzeichnet einen ausdrücklich offenen Qualitätsvorschlag. Eine Implementierung im Quellcode und ein bestandener Test sind getrennte Aussagen. Historische Prüfergebnisse werden nicht als neue Prüfung ausgegeben.

Dokumentumfang

Enthalten sind 7 Epics, 14 User Stories, 8 Use Cases, 40 funktionale Anforderungen und 12 nichtfunktionale Anforderungen sowie eine Zuordnung zwischen diesen Ebenen. Kapitel 15 benennt Grenzen und offene Punkte.

Änderungsstand

Fassung	Bezug	Änderung
1.0	Orbitwerkstatt 0.14.0	Erste konsolidierte Softwaredokumentation einschließlich Bahn am Himmel und portabler Vorbereitung.

1 Orientierung und Systemgrenze

1.1 Kapitelübersicht

- 2 Anforderungsmethodik
- 3 Epics
- 4 User Stories
- 5 Use Cases
- 6 Funktionale Anforderungen
- 7 Nichtfunktionale Anforderungen
- 8 Architektur und Laufdaten
- 9 Datenformate und Zeitbezug
- 10 Mathematische Modelle
- 11 Installation und Betrieb
- 12 Bedienung und Lehrszenario
- 13 Fehlerbehandlung
- 14 Prüfungen und Rückverfolgbarkeit
- 15 Grenzen und Weiterentwicklung
- 16 Begriffe und Quellen

1.2 Rollen und externe Systeme

Anwender importieren Messungen, berechnen Bahnen und vergleichen Ergebnisse. Lehrende verwenden dieselben Funktionen für Demonstrationen; es gibt keine technische Rollen- oder Rechteverwaltung. Betreiber richten Python, Find_Orb und Standortdateien ein. Entwickler pflegen Module und Regressionstests.

Der Browser stellt die lokale Streamlit-Oberfläche dar. Find_Orb ist der externe Rechenkern. MPC-Dateien liefern Referenzelemente und Standortdefinitionen. Astropy liefert Zeitumrechnungen und näherungsweise Planetenpositionen. Tycho ist ein vorgelagertes Mess- und Exportwerkzeug.

1.3 Nicht Bestandteil

Nicht enthalten sind FITS-Bildvermessung, automatische Teleskopsteuerung, NEOCP-Import, automatische Aktualisierung der MPC-Kataloge, Benutzerkonten, ein Mehrbenutzerdienst und ein MP4-Export. Das frühere FITS-Auswahlskript ist ein separates Werkzeug. Ergebnis-ZIPs können archiviert, aber nicht über eine Wiederherstellungsfunktion eingelesen werden.

2 Anforderungsmethodik

2.1 SOPHIST Satzschablonen

Die funktionalen Anforderungen verwenden den FunktionsMASTeR von SOPHIST [Q1]. Selbsttätige Aktivität: „[Bedingung] muss die Orbitwerkstatt <Objekt> <Vollverb>.“ Benutzerinteraktion: „Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, <Objekt> <Vollverb>.“ Schnittstelle: „Die Orbitwerkstatt muss fähig sein, <Objekt> <Vollverb>.“

Bedingungen werden mit „Falls“ für einen Zustand, „Sobald“ für ein Ereignis oder „Solange“ für eine Dauer eingeleitet. Jede Detailanforderung beschreibt eine wesentliche Verpflichtung; zusätzliche Prüfkriterien stehen getrennt.

Für Qualitätsmerkmale wird der EigenschaftsMASTeR verwendet [Q2]: „[Bedingung] muss <Eigenschaft> des <Betrachtungsgegenstands> <Vergleichsausdruck> <Wert> sein.“ Auch ein eindeutig bestimmter qualitativer Wert ist möglich, etwa eine festgelegte Dateikodierung.

2.2 Anwendung auf Epics und Stories

Epics beschreiben größere fachliche Fähigkeiten. Jede Epic erhält eine SOPHIST-Leitanforderung. User Stories behalten die übliche Form „Als ... möchte ich ...“, damit ...“ und erhalten zusätzlich eine SOPHIST-Fassung. Use Cases strukturieren Interaktionen; ihre verpflichtenden Systemschritte stehen ebenfalls im SOPHIST-Stil. Diese Darstellungsformen ergänzen einander und sind keine eigenständigen SOPHIST-Satzschablonen.

2.3 Kennungen und Status

Kennung	Bedeutung
EP-01 bis EP-07	Epic und fachliche Fähigkeit
US-01 bis US-14	User Story mit Akzeptanzkriterium
UC-01 bis UC-08	Use Case
FA-01 bis FA-40	Funktionale Basisanforderung
NFA-01 bis NFA-12	Nichtfunktionale Anforderung

Status B bedeutet im aktuellen Quellcode abgebildete Basis. Status O bedeutet offenes, noch nicht abgenommenes Ziel. B allein bedeutet nicht auf allen Betriebssystemen geprüft. Nachweise nennen Module oder vorhandene Tests; eine formale Vollabnahme aller Anforderungen liegt nicht vor.

2.4 Regeln für die Abnahme

Für jeden Test sind Version, Eingabedatei, Standort, Zeitskala und erwartetes Ergebnis festzuhalten. Bei numerischen Ergebnissen gehört eine Toleranz dazu. Eine geringe Abweichung von Referenzparametern ist nur bei passendem Objekt, Zentrum, Bezugssystem und Zeitbezug fachlich aussagekräftig.

3 Epics

EP-01 Beobachtungen vorbereiten

Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, astrometrische Beobachtungen eines Objekts für eine Bahnberechnung aufzubereiten.

Zuordnung: US-01, US-02 · FA-01 bis FA-09

EP-02 Beobachtungsbögen untersuchen

Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, benannte Bahnberechnungen für ausgewählte Beobachtungstage auszuführen.

Zuordnung: US-03, US-04 · FA-10 bis FA-18

EP-03 Ergebnisse beurteilen

Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, berechnete Bahnelemente und Messabweichungen zu untersuchen.

Zuordnung: US-05, US-06 · FA-19 bis FA-21

EP-04 MPC Referenzen vergleichen

Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, eigene Bahnelemente mit einer ausgewählten MPC-Referenz zu vergleichen.

Zuordnung: US-07, US-08 · FA-22 bis FA-28

EP-05 Bewegung im Sonnensystem zeigen

Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, mehrere berechnete Bahnen in einer gemeinsamen Animation darzustellen.

Zuordnung: US-09, US-10 · FA-29 bis FA-33

EP-06 Bewegung am Himmel zeigen

Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, vorhergesagte Himmelspositionen für einen Beobachtungsort darzustellen.

Zuordnung: US-11, US-12 · FA-34 bis FA-38

EP-07 Ergebnisse bereitstellen und betreiben

Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, die Anwendung lokal mit dokumentierten Ergebnissen zu verwenden.

Zuordnung: US-13, US-14 · FA-39, FA-40; NFA-01 bis NFA-12

4 User Stories

US-01 Beobachtungen importieren

Als Beobachter möchte ich meine exportierten Positionen laden, damit ich ohne erneutes Abtippen rechnen kann.

SOPHIST: Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, Beobachtungsdateien zu importieren.

Akzeptanz: Eine gültige CSV mit drei verschiedenen Zeiten wird als Messreihe übernommen. Bezug: FA-01 bis FA-04.

US-02 Messungen kontrollieren

Als Beobachter möchte ich Zeit und Koordinaten prüfen, damit ich Eingabefehler vor dem Fit erkenne.

SOPHIST: Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, normalisierte Messungen mit UTC und JD anzuzeigen.

Akzeptanz: UTC und abgeleitetes JD stehen nebeneinander; unzulässige Werte blockieren den Lauf. Bezug: FA-05 bis FA-09.

US-03 Tage kombinieren

Als Lehrender möchte ich Beobachtungstage auswählen, damit ich den Einfluss der Bogenlänge zeigen kann.

SOPHIST: Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, Beobachtungstage für einen Fit auszuwählen.

Akzeptanz: Ein abgewählter Tag wird nicht an Find_Orb übergeben. Bezug: FA-10, FA-11.

US-04 Läufe benennen

Als Lehrender möchte ich mehrere Ergebnisse unter eigenen Namen halten, damit ich Versuchsanordnungen wiedererkenne.

SOPHIST: Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, Berechnungsläufe in der Sitzung zu benennen.

Akzeptanz: Zwei nacheinander berechnete Kombinationen sind unter unterscheidbaren Namen auswählbar. Bezug: FA-12 bis FA-18.

US-05 Fit beurteilen

Als Beobachter möchte ich Elemente und Residuen ansehen, damit ich auffällige Messungen erkenne.

SOPHIST: Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, Fit-Ergebnisse und Residuen abzurufen.

Akzeptanz: Die Ergebnisansicht zeigt die vom Rechenkern gelieferten Daten. Bezug: FA-19, FA-20.

4 User Stories Fortsetzung

US-06 Vorhersage prüfen

Als Lehrender möchte ich ausgelassene Beobachtungen vergleichen, damit ich Anpassung und Kontrolle unterscheiden kann.

SOPHIST: Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, geeignete Kontrollmessungen mit Vorhersagen zu vergleichen.

Akzeptanz: Bei passendem Standort und Zeitraster werden Winkelabstände der Kontrollpositionen gezeigt. Bezug: FA-21.

US-07 Referenz finden

Als Beobachter möchte ich mein Objekt in einer MPC-Datei suchen, damit ich Referenzwerte nicht abschreiben muss.

SOPHIST: Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, eine MPC-Referenz anhand des Objektnamens zu suchen.

Akzeptanz: Ein eindeutiger Treffer wird übernommen; mehrere Treffer bleiben auswählbar. Bezug: FA-22 bis FA-25.

US-08 Parameter vergleichen

Als Lehrender möchte ich eigene und offizielle Elemente gegenüberstellen, damit ich Abweichungen erläutern kann.

SOPHIST: Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, Referenzparameter numerisch zu vergleichen.

Akzeptanz: M wird bei ungleichen Epochen nicht numerisch verglichen. Bezug: FA-26 bis FA-28.

US-09 Mehrere Bahnen animieren

Als Lehrender möchte ich benannte Lösungen gleichzeitig zeigen, damit ich unterschiedliche Fits visuell vergleichen kann.

SOPHIST: Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, mehrere eigene Bahnen gemeinsam zu animieren.

Akzeptanz: Alle gewählten Marker verwenden denselben Zeitpunkt. Bezug: FA-29 bis FA-31.

US-10 Referenzbahn ergänzen

Als Lehrender möchte ich die MPC-Bahn einmal hinzufügen, damit die gemeinsame Referenz erkennbar bleibt.

SOPHIST: Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, eine gemeinsame MPC-Bahn in der Animation anzuzeigen.

Akzeptanz: Eine hyperbolische Referenz mit q , e , Orientierung und T_p benötigt kein M . Bezug: FA-32, FA-33.

4 User Stories Fortsetzung

US-11 Himmelsbahn zeigen

Als Beobachter möchte ich RA und Deklination über die Zeit darstellen, damit ich die erwartete Bewegung von meinem Standort sehe.

SOPHIST: Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, eine standortbezogene Himmelsbahn zu erstellen.

Akzeptanz: Der Export enthält UTC, JD, RA, Deklination und Entfernung. Bezug: FA-34 bis FA-36.

US-12 Himmelsbahnen vergleichen

Als Lehrender möchte ich Messungen und Referenz nebeneinander sehen, damit ich Modellunterschiede erläutern kann.

SOPHIST: Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, eigene Himmelsbahnen mit einer Referenz zu vergleichen.

Akzeptanz: Messpunkte werden nach Ort und Zeit gefiltert; Modellabstände sind als solche bezeichnet. Bezug: FA-37, FA-38.

US-13 Ergebnisse mitnehmen

Als Lehrender möchte ich Ergebnisdateien herunterladen, damit ich sie archivieren oder im Unterricht zeigen kann.

SOPHIST: Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, Ergebnisse und Animationen zu exportieren.

Akzeptanz: Das HTML enthält Plotly und benötigt für seine Darstellung kein CDN. Bezug: FA-39; NFA-06.

US-14 Lokal betreiben

Als Betreiber möchte ich die Anwendung lokal starten, damit ich Beobachtungen auf meinem Rechner bearbeiten kann.

SOPHIST: Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, eine lokale Bedienoberfläche mit Hilfe zu verwenden.

Akzeptanz: Der reguläre Start verwendet orbit_env; die portable Variante besitzt eine eigene Laufzeit. Bezug: FA-40; NFA-01 bis NFA-12.

5 Use Cases

UC-01 Messreihe laden

Akteur: Anwender

Vorbedingung: Dateien mit Messungen eines Objekts liegen vor.

Auslöser: Importformat und Dateien sind ausgewählt.

1. Der Anwender wählt CSV, Tycho oder MPC-Beobachtungen.
2. Sobald Dateien ausgewählt sind, muss die Orbitwerkstatt deren Beobachtungen normalisieren.
3. Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, die normalisierten Messungen zu prüfen.

Nachbedingung: Eine geprüfte Messreihe steht für die Auswahl bereit.

Alternativen und Fehler: Ungültige Werte oder gemischte Objektidentitäten verhindern die Übernahme.

Unbekannte Standorte dürfen nur durch bewussten Ausschluss entfernt werden.

Bezug: FA-01 bis FA-09

UC-02 Tage auswählen und Bahn berechnen

Akteur: Anwender

Vorbedingung: Gültige Messungen und funktionsfähiges Find_Orb sind vorhanden.

Auslöser: Der Anwender startet die Berechnung.

1. Der Anwender wählt Tage, Teilmenge, Name, Standort und Epoche.
2. Die Orbitwerkstatt muss die ausgewählten Beobachtungen an Find_Orb übergeben.
3. Sobald Find_Orb erfolgreich beendet ist, muss die Orbitwerkstatt das Ergebnis der aktuellen Sitzung hinzufügen.

Nachbedingung: Ein benannter Lauf und ein eigener Laufordner existieren.

Alternativen und Fehler: Weniger als drei verschiedene Zeiten, fehlende Engine-Ressourcen oder ein Timeout verhindern ein gültiges Ergebnis. Die Fehlermeldung bleibt sichtbar.

Bezug: FA-10 bis FA-18

5 Use Cases Fortsetzung

UC-03 Ergebnisse und Kontrolle untersuchen

Akteur: Anwender

Vorbedingung: Ein erfolgreicher Lauf ist ausgewählt.

Auslöser: Der Anwender öffnet die Ergebnisreiter.

1. Die Orbitwerkstatt muss die berechneten Bahnelemente anzeigen.
2. Die Orbitwerkstatt muss die verfügbaren Residuen anzeigen.
3. Falls geeignete Kontrollpositionen vorliegen, muss die Orbitwerkstatt deren Winkelabstände zu den Vorhersagen berechnen.

Nachbedingung: Die Güte des Fits kann anhand der verfügbaren Daten beurteilt werden.

Alternativen und Fehler: Fehlende oder nicht passende Kontrollzeiten erlauben keine unabhängige Kontrollaussage. Kleines RMS bestätigt keine eindeutige Kurzbahn-Lösung.

Bezug: FA-19 bis FA-21

UC-04 MPC Referenz laden und vergleichen

Akteur: Anwender

Vorbedingung: Ein eigener Lauf und ein lokaler oder hochgeladener Katalog liegen vor.

Auslöser: Der Anwender öffnet Referenzvergleich.

1. Der Anwender sucht das Objekt und wählt bei Bedarf eine Lösung.
2. Die Orbitwerkstatt muss die übernommenen MPC-Parameter in die Referenztabelle eintragen.
3. Der Anwender bestätigt Objekt und gemeinsames Bezugssystem.
4. Falls die Vergleichsvoraussetzungen erfüllt sind, muss die Orbitwerkstatt Parameterdifferenzen berechnen.

Nachbedingung: Der Vergleich ist im Laufordner mit Herkunftsinformation gespeichert.

Alternativen und Fehler: Fehlende Epoche ist fachlich zu ergänzen, nicht mit Tp gleichzusetzen. Unbekanntes Zentrum oder System sperrt den Vergleich. Unterschiedliche Epochen sperren M.

Bezug: FA-22 bis FA-28

5 Use Cases Fortsetzung

UC-05 Gemeinsame Sonnenbahn animieren

Akteur: Lehrender

Vorbedingung: Eigene Läufe desselben Objekts und optional eine gültige Referenz sind vorhanden.

Auslöser: Der Anwender erstellt eine Animation.

1. Der Anwender wählt Läufe, Planeten und Zeitraum.
2. Die Orbitwerkstatt muss die ausgewählten Bahnen auf einem gemeinsamen Zeitraster fortschreiben.
3. Falls eine gültige MPC-Referenz ausgewählt ist, muss die Orbitwerkstatt diese einmal in die Animation aufnehmen.

Nachbedingung: Eine synchronisierte Animation steht bereit.

Alternativen und Fehler: Mehr als zehn Jahre oder unvollständige Referenzparameter verhindern die Erstellung. Hyperbeln und Parabeln verwenden Tp statt M.

Bezug: FA-29 bis FA-33

UC-06 Bahn am Himmel erstellen

Akteur: Anwender

Vorbedingung: Ein eigener Lauf und ein gültiger Beobachtungsort liegen vor.

Auslöser: Der Anwender öffnet Bahn am Himmel.

1. Der Anwender wählt Läufe, Ort, Zeitraum und Schrittweite.
2. Die Orbitwerkstatt muss standortbezogene Himmelspositionen mit Lichtlaufzeitkorrektur berechnen.
3. Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, passende Messungen einzublenden.
4. Falls eine Referenz ausgewählt ist, muss die Orbitwerkstatt die Winkelabstände zwischen den Modellbahnen anzeigen.

Nachbedingung: Himmelsbahn und exportierbare Positionstabelle sind vorhanden.

Alternativen und Fehler: Unbekannter oder bewegter Standort wird zurückgewiesen. Nicht passende Messorte werden nicht eingeblendet. Projektionsferne Punkte sind nicht sichtbar.

Bezug: FA-34 bis FA-38

5 Use Cases Fortsetzung

UC-07 Ergebnisse exportieren

Akteur: Anwender

Vorbedingung: Ein Ergebnis oder eine erzeugte Animation liegt vor.

Auslöser: Der Anwender betätigt einen Download.

1. Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, das Ergebnis als ZIP herunterzuladen.
2. Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, erzeugte Animationen als eigenständiges HTML herunterzuladen.

Nachbedingung: Die angeforderte Datei kann außerhalb der Sitzung archiviert werden.

Alternativen und Fehler: Ein ZIP ist ein Archiv, kein über die Oberfläche wiederherstellbarer Sitzungsstand. Ein MP4-Export ist nicht vorhanden.

Bezug: FA-39

UC-08 Anwendung starten und Hilfe nutzen

Akteur: Betreiber und Anwender

Vorbedingung: Python und Engine sind eingerichtet oder ein vorbereitetes portables Paket liegt vor.

Auslöser: Das passende Startskript wird ausgeführt.

1. Das Startskript startet die lokale Streamlit-Anwendung.
2. Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, die integrierte Hilfe ohne gültige Messungen aufzurufen.

Nachbedingung: Die lokale Oberfläche ist erreichbar.

Alternativen und Fehler: Fehlende Engine-Dateien werden vor einer Berechnung gemeldet. Portable Erstellung benötigt zunächst Internet; der Betrieb auf Windows ist gesondert zu prüfen.

Bezug: FA-40; NFA-01 bis NFA-12

6 Funktionale Anforderungen

FA-01

Die Orbitwerkstatt muss fähig sein, Beobachtungen aus einer Standard-CSV zu importieren.

Prüfkriterium: Pflichtfelder UTC und Koordinaten; gültiges Beispiel aus Kapitel 9.

Status B · Nachweis im Code: `core.py`

FA-02

Die Orbitwerkstatt muss fähig sein, Beobachtungen aus dem unterstützten Tycho-CSV-Format zu importieren.

Prüfkriterium: Acht Felder ohne Kopfzeile; Datum und JD auf Konsistenz prüfen.

Status B · Nachweis im Code: `tycho_import.py`

FA-03

Die Orbitwerkstatt muss fähig sein, unterstützte MPC1992-Beobachtungen zu importieren.

Prüfkriterium: 80 Zeichen; Typen C, B, P und V/v; zugehörige Standortzeile erhalten.

Status B · Nachweis im Code: `mpc_import.py`

FA-04

Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, mehrere Dateien desselben Importformats für ein Objekt zu laden.

Prüfkriterium: Abweichende Objektidentitäten zurückweisen; Quelldatei behalten.

Status B · Nachweis im Code: `multi_observations.py`

FA-05

Falls mehrere importierte Messungen dieselbe kanonische UTC und denselben Standort besitzen, muss die Orbitwerkstatt den ersten Eintrag beibehalten.

Prüfkriterium: Importreihenfolge maßgeblich; RGB-Mehrfachmessungen nicht mitteln.

Status B · Nachweis im Code: `multi_observations.py`

FA-06

Die Orbitwerkstatt muss ungültige normalisierte Messwerte zurückweisen.

Prüfkriterium: RA 0 bis kleiner 360 Grad; Dec -90 bis +90 Grad; endliche Werte; sigma positiv.

Status B · Nachweis im Code: `core.py`

6 Funktionale Anforderungen Fortsetzung

FA-07

Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, CSV- und Tycho-Messungen vor der Berechnung zu bearbeiten.

Prüfkriterium: Native MPC-Datensätze bleiben schreibgeschützt; Änderungen erneut prüfen.

Status B · Nachweis im Code: measurement_view.py

FA-08

Die Orbitwerkstatt muss das aus UTC abgeleitete Julianische Datum neben der Beobachtungszeit anzeigen.

Prüfkriterium: JD.UTC ist Anzeigegröße und keine unabhängige editierbare Zeitquelle.

Status B · Nachweis im Code: measurement_view.py

FA-09

Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, Messungen mit unbekannten Standortcodes auszuschließen.

Prüfkriterium: Nur nach bewusster Auswahl; verbleibende Messungen erneut validieren.

Status B · Nachweis im Code: app.py

FA-10

Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, Beobachtungstage über Checkboxes für einen Lauf auszuwählen.

Prüfkriterium: Tageszuordnung nach UTC; Abwahl entfernt die Messungen aus der Auswahl.

Status B · Nachweis im Code: app.py

FA-11

Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, aus der Auswahl alle oder drei zeitlich verteilte Beobachtungen zu verwenden.

Prüfkriterium: Dreierauswahl: Anfang, Mitte und Ende der gewählten Menge.

Status B · Nachweis im Code: app.py

FA-12

Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, eine Berechnungsepoche in JD TT festzulegen.

Prüfkriterium: Automatische Epoche aus Messzeiten oder manuelle Eingabe; zulässiger Bereich 1600–2200.

Status B · Nachweis im Code: epoch_view.py

6 Funktionale Anforderungen Fortsetzung

FA-13

Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, den Beobachtungsort als MPC-Code oder eigene WGS84-Koordinaten festzulegen.

Prüfkriterium: Eigene Koordinaten nicht für den nativen MPC-Import anbieten.

Status B · Nachweis im Code: app.py, core.py

FA-14

Falls weniger als drei verschiedene Beobachtungszeiten vorliegen, muss die Orbitwerkstatt den Berechnungsstart verhindern.

Prüfkriterium: Auch drei Zeilen mit identischen Zeiten reichen nicht aus.

Status B · Nachweis im Code: core.py

FA-15

Die Orbitwerkstatt muss für jeden Rechenlauf einen getrennten Arbeitsordner anlegen.

Prüfkriterium: Eingaben, Konfiguration, Protokoll und Ausgaben sind laufbezogen.

Status B · Nachweis im Code: core.py

FA-16

Die Orbitwerkstatt muss fähig sein, die ausgewählten Beobachtungen an Find_Orb zu übergeben.

Prüfkriterium: Normalisierte Messungen als ADES PSV; native MPC-Datensätze unverändert im OBS-Format.

Status B · Nachweis im Code: core.py

FA-17

Sobald eine Berechnung erfolgreich abgeschlossen ist, muss die Orbitwerkstatt das Ergebnis unter einem Laufnamen in der Sitzung bereitstellen.

Prüfkriterium: Automatischer Name vorhanden; eigener Name möglich.

Status B · Nachweis im Code: app.py

FA-18

Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, gespeicherte Sitzungsnamen zu ändern.

Prüfkriterium: Lauf bleibt unter geändertem Anzeigenamen auswählbar.

Status B · Nachweis im Code: app.py

6 Funktionale Anforderungen Fortsetzung

FA-19

Die Orbitwerkstatt muss die verfügbaren Bahnelemente des ausgewählten Laufs anzeigen.

Prüfkriterium: Darstellung basiert auf geparsten Find_Orb-Ausgaben.

Status B · Nachweis im Code: app.py, core.py

FA-20

Die Orbitwerkstatt muss die verfügbaren Messabweichungen des ausgewählten Laufs anzeigen.

Prüfkriterium: Verwendete und ausgeschlossene Messungen fachlich prüfen.

Status B · Nachweis im Code: core.py, app.py

FA-21

Falls Standort und Zeitraster zu ausgelassenen Kontrollmessungen passen, muss die Orbitwerkstatt deren Winkelabstände zur Ephemeride berechnen.

Prüfkriterium: Zeitzuordnung mit 0,1 Sekunden Toleranz; keine allgemeine Kontrolle für eigenen WGS84-Ort.

Status B · Nachweis im Code: core.py

FA-22

Die Orbitwerkstatt muss fähig sein, CometEls- und AllCometEls-Dateien als Referenzkatalog einzulesen.

Prüfkriterium: Festspalten beachten; fehlende Epochen und historische Daten nicht erfinden.

Status B · Nachweis im Code: comet_elements.py

FA-23

Die Orbitwerkstatt muss fähig sein, NEA- und MPCORB-Dateien als Referenzkatalog einzulesen.

Prüfkriterium: Gepackte Epoche dekodieren; elliptische Elemente und Einheiten prüfen.

Status B · Nachweis im Code: minor_planet_elements.py

FA-24

Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, einen lokalen oder hochgeladenen Referenzkatalog auszuwählen.

Prüfkriterium: Kataloge neben app.py; Upload hat Vorrang; keine automatische Internetabfrage.

Status B · Nachweis im Code: comet_import_view.py

6 Funktionale Anforderungen Fortsetzung

FA-25

Falls die Objektsuche genau einen Treffer liefert, muss die Orbitwerkstatt dessen Referenzparameter übernehmen.

Prüfkriterium: Exakte Kennung, Teilstring und Ähnlichkeit unterstützen; Mehrfachtreffer auswählbar.

Status B · Nachweis im Code: comet_import_view.py

FA-26

Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, Referenzparameter manuell einzugeben.

Prüfkriterium: Mindestens ein vergleichbarer Parameter und Referenzepoche erforderlich.

Status B · Nachweis im Code: reference_view.py

FA-27

Falls Objekt, Zentrum, Bezugssystem und Parameter gültig bestätigt sind, muss die Orbitwerkstatt die zulässigen Parameterdifferenzen berechnen.

Prüfkriterium: Sun und J2000 erforderlich; Winkel zyklisch; M nur elliptisch bei gleicher Epoche.

Status B · Nachweis im Code: reference_compare.py

FA-28

Sobald ein Referenzvergleich erstellt ist, muss die Orbitwerkstatt dessen Angaben im Laufordner speichern.

Prüfkriterium: referenzvergleich.json mit Werten und Importherkunft, soweit verfügbar.

Status B · Nachweis im Code: reference_view.py

FA-29

Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, mehrere eigene Läufe desselben Objekts für eine Animation auszuwählen.

Prüfkriterium: Auswahl nach Namen; gemeinsames UTC-Raster.

Status B · Nachweis im Code: multi_animation.py

FA-30

Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, den Animationszeitraum vorwärts oder rückwärts festzulegen.

Prüfkriterium: Maximal 3652,5 Tage; in der Sonnenanimation 20 bis 720 Frames.

Status B · Nachweis im Code: animation_view.py

6 Funktionale Anforderungen Fortsetzung

FA-31

Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, die acht Planeten in der Sonnenanimation einzublenden.

Prüfkriterium: Merkur bis Neptun; Planetenpositionen aus Astropy-Builtin-Ephemeriden.

Status B · Nachweis im Code: orbit_animation.py

FA-32

Die Orbitwerkstatt muss eine ausgewählte gemeinsame MPC-Referenzbahn genau einmal in der Mehrlaufanimation darstellen.

Prüfkriterium: Bei mehreren gespeicherten Referenzen eine gemeinsame auswählen.

Status B · Nachweis im Code: multi_animation.py

FA-33

Die Orbitwerkstatt muss fähig sein, elliptische, parabolische und hyperbolische Referenzbahnen für die Animation zu initialisieren.

Prüfkriterium: Ellipse aus M und Epoche; offene Bahn aus q, e, Orientierung und Tp.

Status B · Nachweis im Code: reference_animation.py

FA-34

Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, eine Bahn am Himmel für einen angegebenen Beobachtungsort zu erzeugen.

Prüfkriterium: MPC-Code oder WGS84; 500 geozentrisch; bewegte Codes 247/250 nicht unterstützt.

Status B · Nachweis im Code: sky_view.py, sky_track.py

FA-35

Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, Zeitraum und Schrittweite der Himmelsbahn festzulegen.

Prüfkriterium: Vorwärts oder rückwärts; maximal zehn Jahre und 720 Zeitpunkte.

Status B · Nachweis im Code: sky_track.py

FA-36

Die Orbitwerkstatt muss für die Himmelsbahn die geometrische Lichtlaufzeit iterativ berücksichtigen.

Prüfkriterium: Beobachter zum Empfangszeitpunkt, Objekt zum Emissionszeitpunkt.

Status B · Nachweis im Code: sky_track.py

6 Funktionale Anforderungen Fortsetzung

FA-37

Die Orbitwerkstatt muss eingblendete Messungen nach Beobachtungsort und Darstellungszeitraum filtern.

Prüfkriterium: Identische Kombination aus Zeit, RA und Dec nur einmal; Farbe nach UTC-Tag.

Status B · Nachweis im Code: sky_view.py

FA-38

Falls eine Referenzhimmelsbahn ausgewählt ist, muss die Orbitwerkstatt die sphärischen Winkelabstände der eigenen Modellbahnen zur Referenz anzeigen.

Prüfkriterium: Einheit Bogensekunden; nicht mit Fit-Residuen gleichsetzen.

Status B · Nachweis im Code: sky_track.py, sky_view.py

FA-39

Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, verfügbare Ergebnisse in den vorgesehenen Dateiformaten herunterzuladen.

Prüfkriterium: Ergebnis-ZIP, Animations-HTML sowie Himmelsbahn-CSV; kein MP4.

Status B · Nachweis im Code: core.py, animation_view.py, sky_view.py

FA-40

Die Orbitwerkstatt muss dem Anwender die Möglichkeit bieten, die integrierte Hilfe ohne gültige Messreihe aufzurufen.

Prüfkriterium: Hilfe zu Bedienung, Formaten, Software und Architektur.

Status B · Nachweis im Code: help_view.py

7 Nichtfunktionale Anforderungen

NFA-01

Die Bindeadresse des regulären lokalen Starts muss 127.0.0.1 sein.

Status B · Prüfung: Startskripte prüfen; kein öffentlich erreichbarer Dienst zugesagt.

Nachweis: Startskripte

NFA-02

Die Python-Mindestversion der regulären Installation muss 3.11 sein.

Status B · Prüfung: Installation und Import der Abhängigkeiten auf unterstütztem Zielsystem prüfen.

Nachweis: README.md, requirements.txt

NFA-03

Die Isolation der Find_Orb-Konfiguration muss laufbezogen sein.

Status B · Prüfung: Zwei Läufe dürfen keine übernommenen Fit-Caches aus einem früheren Arbeitsordner verwenden.

Nachweis: core.py und Ressourcentests

NFA-04

Die Standard-Zeitbegrenzung eines Find_Orb-Aufrufs muss 180 Sekunden sein.

Status B · Prüfung: Hängenden Prozess simulieren; Timeout statt endloser Wartezeit.

Nachweis: core.py

NFA-05

Die Eingabevalidierung numerischer Messwerte muss nichtendliche Werte ausschließen.

Status B · Prüfung: NaN und unendliche RA, Dec oder sigma zurückweisen.

Nachweis: core.py und Validierungstests

NFA-06

Die externe JavaScript-Abhängigkeit einer exportierten Animations-HTML muss für Plotly gleich null sein.

Status B · Prüfung: HTML enthält Bibliothek; Darstellung ohne CDN. Offline-Browserprüfung als Betriebsabnahme.

Nachweis: HTML-Exportfunktionen

7 Nichtfunktionale Anforderungen Fortsetzung

NFA-07

Die Herkunftsdocumentation eines Find_Orb-Laufs muss SHA-256-Prüfsummen für Eingabe und Engine enthalten.

Status B · Prüfung: manifest.json auf entsprechende Werte prüfen; keine Behauptung vollständig deterministischer Ergebnisse.

Nachweis: core.py

NFA-08

Die Umgebungsabhängigkeit eines vorbereiteten portablen Starts muss vom installierten System-Python unabhängig sein.

Status B · Prüfung: Paket besitzt runtime; native Windows-Abnahme noch offen.

Nachweis: portable/launch_portable.py

NFA-09

Die Pfadbindung der portablen Anwendung muss relativ zum Paketverzeichnis sein.

Status B · Prüfung: Paket auf anderes Laufwerk verschieben; Engine, Daten und Start erneut prüfen.

Nachweis: portable/ und Paketprüfung

NFA-10

Die Antwortzeit der Messvorschau sollte bei 1000 CSV-Zeilen höchstens zwei Sekunden sein.

Status O · Prüfung: Vorschlag: lokales SSD-System mit vier CPU-Kernen und 16 GB RAM; warmer Prozess, zehn Läufe, 95. Perzentil; Engine ausgenommen.

Nachweis: Noch keine Messung oder Freigabe

NFA-11

Die Abweichung der berechneten Himmelspositionen sollte innerhalb einer vorab fachlich festgelegten Referenztoleranz sein.

Status O · Prüfung: Noch nicht abnahmefähig: Toleranz, unabhängige Quelle, Objektklasse, Zeitraum und Modellkonventionen müssen festgelegt werden.

Nachweis: Offene Qualitätsanforderung zur Präzisierung

NFA-12

Die Testabdeckung der unterstützten Windows-Betriebsarten sollte beide Varianten regulär und portabel umfassen.

Status O · Prüfung: Je ein dokumentierter Start-, Import-, Fit-, Export- und Offline-Test auf realem Windows; portable Variante auch nach Laufwerkswechsel.

Nachweis: Offene Systemabnahme

8 Architektur und Laufdaten

8.1 Schichten und Verantwortlichkeiten

Schicht	Module	Verantwortung
Oberfläche	app.py, *_view.py	Eingaben, Sitzungszustand, Ergebnisreiter, Downloads
Import und Prüfung	core.py, tycho_import.py, mpc_import.py, multi_observations.py	Formate normalisieren, Auswahl und Objektzuordnung prüfen
Engine-Anbindung	core.py	Isolierter Find_Orb-Aufruf, Protokoll und Ergebnisparser
Referenzen	comet_elements.py, minor_planet_elements.py, reference_compare.py	Katalogsuche, Elemente, Vergleichsregeln
Sonnenanimation	orbit_animation.py, multi_animation.py, reference_animation.py	Zweikörperfortschreibung, Planeten und gemeinsame Frames
Himmelsdarstellung	sky_track.py, sky_view.py	Beobachter, Lichtlaufzeit, Projektion und Winkelabstände
Betrieb	Startskripte, portable/	Umgebung, Abhängigkeiten, Start und Paketaufbau

8.2 Verarbeitungskette

Dateiimport → Normalisierung und Dublettenbehandlung → Messprüfung → Auswahl von Tagen und Teilmenge → isolierter Find_Orb-Lauf → Ergebnisparser → benannter Sitzungslauf → Tabellen, Vergleich und Visualisierung → Export. Referenzkataloge speisen den Parametervergleich und die optionalen Referenzbahnen; sie ersetzen nicht die eigenen Messungen.

8.3 Zustände und Persistenz

Die Oberfläche hält mehrere Ergebnisse in `st.session_state.results`. Diese Auswahl lebt in der aktuellen Sitzung. Dateien verbleiben in `runs`, werden bei einem Neustart jedoch nicht automatisch als Sitzung rekonstruiert. Der tabellarische Vergleich eigener Läufe ist an denselben Datenstand gebunden; die Animation kann über die Objektidentität auch Läufe verschiedener Datenstände zusammenführen.

8.4 Fehlergrenzen

Datei- und Wertefehler werden vor dem Fit behandelt. Find_Orb läuft als Unterprozess ohne Shell-Auswertung. Timeout und Prozessausgaben werden separat erfasst. Für die Visualisierung wird der berechnete Zustand übernommen; es erfolgt kein erneuter Fit. Eine erfolgreiche Animation belegt daher nicht die Richtigkeit der Bahn.

8 Architektur und Laufdaten Fortsetzung

8.5 Dateien eines Laufs

Datei oder Ordner	Inhalt
input.csv	Normalisierte, an den Lauf gebundene Messungen
observations.psv oder observations.obs	ADES-Eingabe beziehungsweise ursprüngliche MPC-Datensätze
config/	Isolierte Kopie erforderlicher Find_Orb-Ressourcen
elements.json, elements.txt	Bahnelemente und gegebenenfalls Zustandsvektor
ephemeri.json, ephemeris.txt	Vom Rechenkern erzeugte Vorhersagen
run.log	Ausführung und Diagnoseinformationen
manifest.json	Aufruf, Standort, Epoche und Prüfsummen
referenzvergleich.json	Optionaler gespeicherter Referenzvergleich
Ergebnis-ZIP	Archiv einschließlich result.json und aufbereiteter Ergebnisdateien

8.6 Fachliches Datenmodell

Eine Beobachtung besitzt Empfangszeit UTC, RA, Deklination, Standort und gegebenenfalls Messunsicherheit und Herkunft. Ein Messbestand enthält Beobachtungen eines Objekts. Ein Lauf bindet eine Auswahl, die Rechenkonfiguration und die daraus entstandene Lösung. Eine Referenz bindet Elemente an Objekt, Epoche, Rahmen und Herkunft. Eine Animation ist eine aus ausgewählten Lösungen abgeleitete Darstellung auf einem gemeinsamen Zeitraster.

Der Anzeigename ist kein mathematischer Identifikator. Die Objektzuordnung entscheidet über zulässige gemeinsame Animationen. Datum und Name allein dürfen nicht als Beweis gleicher physikalischer Lösungen verwendet werden.

8.7 Austausch und Wiederholbarkeit

Ein Ergebnisarchiv ermöglicht die Prüfung von Eingabe und Rechenkonfiguration. Die Python-Abhängigkeiten besitzen Versionsbereiche statt einer vollständig fixierten Lockdatei. Eine spätere Neuinstallation muss daher nicht exakt dieselbe Laufzeit ergeben. Das portable Paket dokumentiert zusätzlich Paketversionen und Prüfsummen. Für wissenschaftliche Wiederholung sollten Engine, Eingaben, Standortdatei und Umgebung gemeinsam archiviert werden.

9 Datenformate und Zeitbezug

9.1 Standard CSV

Kodierung UTF-8; Trennzeichen Komma oder Semikolon; Kopfzeile erforderlich. Dezimalzahlen verwenden einen Punkt. utc bezeichnet UTC, vorzugsweise die Belichtungsmitte. Zeitzonenoffsets und eine automatische Umrechnung lokaler Uhrzeit sind nicht vorgesehen.

Spalte	Pflicht und Format	Beispiel
utc	Pflicht; Datum mit Uhrzeit, optional Z	2026-01-01T08:00:00Z
ra_deg	RA in Grad; $0 \leq \text{RA} < 360$	20.500975772
dec_deg	Deklination in Grad; $-90 \leq \text{Dec} \leq 90$	31.468878558
site	Optional; dreistelliger MPC-Code	568
sigma_arcsec	Optional; positive Unsicherheit in Bogensekunden	0.3
ra_hms, dec_dms	Alternative gemeinsam statt Gradspalten	01 22 00.234 / +31 28 08.72

Fehlende optionale Standort- und Unsicherheitswerte werden aus den Einstellungen ergänzt. Führende Nullen von Standortcodes müssen erhalten bleiben. Nach der Normalisierung sind mindestens drei verschiedene Beobachtungszeiten nötig.

9.2 Beispiel

utc,ra_deg,dec_deg,site,sigma_arcsec

2026-01-01T08:00:00Z,20.500975772,31.468878558,568,0.3

2026-01-02T08:00:00Z,21.154287516,31.230390091,568,0.3

2026-01-03T08:00:00Z,21.820199139,30.994273697,568,0.3

Die Beispiele stammen aus dem synthetischen Eros-Datensatz und sind keine eigenen Messungen. Drei Positionen sind eine technische Mindestvoraussetzung, kein Nachweis einer eindeutigen oder belastbaren Bahn.

9.3 Tycho und native MPC Beobachtungen

Der unterstützte Tycho-Export enthält acht Felder ohne Kopfzeile: Objektkennung, MPC-Datum, JD, RA als Stundenwinkel, Deklination, Helligkeit, Band und Standortcode. Datum und JD werden gegengeprüft. Bei Mehrfacheinträgen, etwa getrennten RGB-Auswertungen, bleibt der erste passende Datensatz erhalten.

MPC1992 verwendet feste Zeichenpositionen, nicht freie durch Leerzeichen getrennte Spalten. Unterstützte Beobachtungsarten sind C, B, P und V/v. Ein V/v-Paar bleibt zusammen. Radar- oder andere nicht unterstützte Sonderformate werden zurückgewiesen. Originalzeilen bleiben für Find_Orb erhalten.

9 Datenformate und Zeitbezug Fortsetzung

9.4 MPC Referenzdateien

CometEls.txt und AllCometEls.txt enthalten Kometenelemente; NEA.txt und MPCORB.DAT enthalten Kleinplanetenelemente. Es sind Referenzkataloge, keine gemessenen RA/Dec-Reihen. Die Dateien können lokal neben app.py liegen oder im Referenzvergleich hochgeladen werden. Komprimierte Dateien müssen vorher entpackt werden. Lokale Kataloge sind bis 600 MB vorgesehen; der Upload unterliegt zusätzlich der Streamlit-Konfiguration.

Für große Kleinplanetenkataloge wird zuerst gesucht und anschließend nur die Kandidatenmenge numerisch eingelesen. Es werden höchstens 200 Kandidaten angeboten. Exakte Kennungen und Namensvarianten werden vor Teilstring- und Ähnlichkeitstreffern bevorzugt.

Element	Bedeutung	Einheit
a / q	Große Halbachse / Periheldistanz	AE
e	Exzentrizität	dimensionslos
i / Ω / ω	Neigung / Knotenlänge / Perihelargument	Grad
M	Mittlere Anomalie zur Epoche	Grad
n / P	Mittlere Bewegung / Umlaufzeit	Grad pro Tag / Tage
Epoche / Tp	Gültigkeitszeit der Elemente / Periheldurchgang	Intern JD TT

9.5 UTC, TT und Epoche

UTC bezeichnet die Beobachtungszeit. TT ist die gleichmäßig laufende Zeitskala für die Elementepoche. JD ist eine Datumsdarstellung und für sich genommen keine Zeitskala: JD UTC und JD TT sind zu unterscheiden. Die Messwerttabelle zeigt JD UTC; die Rechenepoche verwendet JD TT.

Die Referenzepoche ist der Zeitpunkt, auf den sich der Elementsatz bezieht. Tp ist der Zeitpunkt des Periheldurchgangs und darf eine fehlende Referenzepoche nicht stillschweigend ersetzen. In historischen Kometenkatalogen werden Kalenderdaten vor 1583 beibehalten, ohne eine ungeklärte Kalenderumrechnung vorzunehmen.

Gepackte Kleinplanetenepochen werden dekodiert; K2669 entspricht 9. Juni 2026 TT. Bei unterschiedlicher Epoche werden viele Elementdifferenzen nur orientierend gezeigt. M wird nur für elliptische Bahnen bei gleicher Epoche verglichen. Der Parametervergleich propagiert die Elemente nicht auf eine gemeinsame Epoche.

10 Mathematische Modelle

10.1 Aus Winkeln wird eine Bahn

RA α und Deklination δ legen einen Sichtstrahl fest. Sein Einheitsvektor lautet $u = (\cos \delta \cos \alpha, \cos \delta \sin \alpha, \sin \delta)$. Der heliozentrische Objektort erfüllt näherungsweise $r = R + pu$, wobei R den Beobachterort relativ zur Sonne und p die zunächst unbekannte Entfernung bezeichnet. Mehrere Zeitpunkte und das Bewegungsgesetz schränken die möglichen Entfernungen und Geschwindigkeiten ein.

Find_Orb bestimmt eine zu den Messungen passende Lösung. Die Anwendung übergibt die Beobachtungen und wertet die Ergebnisse aus; sie implementiert keine eigenständige vollständige Gaußsche Bahnbestimmung. Standort, Zeit, Messunsicherheit und Beobachtungsbogen wirken unmittelbar auf die Lösung.

10.2 Sonnenanimation

Die Anzeige übernimmt einen heliozentrischen Zustandsvektor und integriert das Zweikörperproblem: $r'' = -\mu r / |r|^3$. Verwendet wird DOP853 aus SciPy. Positionen sind in AE und Zeiten in Tagen. Der Zustandsvektor wird vom äquatorialen J2000-Rahmen in den ekliptikalen Darstellungsrahmen gedreht.

Für eine Ellipse gilt $q = a(1-e)$, $n = \sqrt{\mu/a^3}$ und $M(t) = M_0 + n(t-t_0)$, mit konsistenten Winkel- und Zeiteinheiten. Eine offene Referenzbahn wird am Perihel aus q , e und der Geschwindigkeit $v_p = \sqrt{\mu(1+e)/q}$ initialisiert. Daher benötigt eine Hyperbel kein elliptisches M .

Die Planetenpositionen stammen aus Astropys eingebauten Ephemeriden. Die Objektanimation enthält keine gravitativen Planetenstörungen, auch wenn Find_Orb diese beim Fit berücksichtigt. Langfristige Unterschiede zwischen Animation und Rechenkern sind deshalb möglich.

10.3 Himmelsbahn

Die Himmelsdarstellung berechnet die Richtung vom Beobachter zum Objekt. Für die Lichtlaufzeit gilt $t_e = t_r - \Delta/c$. Objekt und Sonne werden zum Emissionszeitpunkt, der Beobachter zum Empfangszeitpunkt ausgewertet. Die Richtung wird in RA und Deklination umgerechnet. Die Anzeige verwendet eine azimutale äquidistante Projektion mit Osten links und Norden oben.

Die Koordinaten sind näherungsweise astrometrische J2000/ICRS-Richtungen. Refraktion, Aberration, Lichtablenkung und gravitative Störungen der fortgeschriebenen Objektbahn sind nicht enthalten. Die Kurve ersetzt keine präzise Teleskop-Ephemeride. Der Vergleich unter der Grafik zeigt sphärische Winkelabstände zwischen Modellen, nicht die Residuen des Fits.

11 Installation und Betrieb

11.1 Reguläre Installation

Das Paket in ein beschreibbares Verzeichnis entpacken. Python ab Version 3.11 bereitstellen. Unter Windows `start_windows.bat` ausführen; unter Unix das entsprechende Shellskript verwenden. Der reguläre Start legt bei Bedarf `orbit_env` an und installiert die Abhängigkeiten aus `requirements.txt`. Er kann beim Start erneut auf die Paketquellen zugreifen.

`Find_Orb` separat installieren beziehungsweise das mitgelieferte Windows-Installationsskript verwenden. In der Seitenleiste die ausführbare Datei und den Konfigurationsordner festlegen. Erforderliche Begleitdateien umfassen unter anderem `environ.def`, `ObsCodes.htm` und `efindorb.txt`. Der Ordner mit diesen Engine-Dateien ist mit „engine“ gemeint; es handelt sich nicht um die Python-Umgebung.

Nach dem Start die lokale Browseradresse öffnen. Die regulären Skripte binden den Server an 127.0.0.1. Für die Verarbeitung sind Schreibrechte im Anwendungs- und Laufbereich erforderlich. Standortdateien und optionale JPL-DE-Dateien werden lokal verwendet; die Anwendung lädt keine fehlende Präzisionsephemeride automatisch nach.

11.2 Portable Vorbereitung

`Portable_erstellen.bat` startet den Paketaufbau unter Windows. Dabei werden eine eigene Python-Laufzeit, Python-Pakete und die benötigte `Find_Orb`-Installation zusammengestellt. Der Aufbau benötigt zunächst Internet. Das Ergebnis liegt unter `portable_output` als Verzeichnis und ZIP vor. Vorhandene Sitzungen, persönliche Einstellungen und `orbit_env` werden nicht einfach übernommen.

Im vorbereiteten Paket startet die mitgelieferte Batchdatei die eigene Laufzeit über `launch_portable.py`. Der Start installiert keine Pakete nach. Pfade werden aus dem Paketverzeichnis abgeleitet; ein anderes Laufwerk soll dadurch möglich sein. Der Launcher prüft Ressourcen, Schreibbarkeit und verfügbare lokale Ports im Bereich 8501–8550.

11.3 Betriebsabnahme

Die portable Erstellung und Startlogik sind vorbereitet. Ein vollständig abgenommener Windows-/USB-Betrieb ist für diesen Dokumentationsstand nicht nachgewiesen. Vor einer Lehrveranstaltung auf dem tatsächlichen Zielrechner testen: Start, Import, `Find_Orb`-Fit, Referenzvergleich, beide Animationen und Downloads. Danach Offline-Start und Laufwerkswechsel prüfen.

Ergebnisse und Einstellungen vor dem Entfernen des Datenträgers sichern und laufende Prozesse beenden. Browser und Betriebssystem können außerhalb des USB-Sticks Daten zwischenspeichern; „portabel“ bedeutet hier keine garantierte Spurenfreiheit.

12 Bedienung und Lehrszenario

12.1 Vom Import zum Vergleich

1. Engine und Standort in der Seitenleiste einrichten. Das zur Datei passende Importformat wählen und Messdateien eines Objekts laden.
2. In „Messungen ansehen und bearbeiten“ UTC, JD und Koordinaten prüfen. Unbekannte Standortcodes klären oder deren Messungen bewusst ausschließen.
3. Beobachtungstage über Checkboxen wählen. Für einen ersten Lauf einen Tag, später mehrere Tage auswählen. Name und Epoche festlegen, dann berechnen.
4. Elemente, Residuen und gegebenenfalls Kontrollvorhersagen prüfen. Weitere Tageskombinationen nacheinander berechnen und eindeutig benennen.
5. Im Ergebnisreiter „Referenzvergleich“ die MPC-Referenzdatei importieren. Lokalen Katalog wählen oder Datei hochladen, Objekt suchen und bei mehreren Treffern die Lösung auswählen.
6. Übernommene Parameter und Referenzepoche prüfen. Objekt und gemeinsames Bezugssystem bestätigen. Erst dann den Vergleich erstellen.

12.2 Gemeinsame Animation

Unter der Bahndarstellung die gewünschten Laufnamen auswählen. Einen gemeinsamen MPC-Vergleich als Referenz ergänzen und die gewünschten Planeten aktivieren. Start und Ende als Datum oder Datum mit Uhrzeit eingeben. Auch ein rückwärts laufender Zeitraum, etwa 2026-01-01 bis 2023-01-01, ist möglich. Maximal zehn Jahre sind zulässig.

Ist die MPC-Referenz nicht auswählbar, prüfen, ob ein gültiger gespeicherter Vergleich vorliegt. Für eine Ellipse sind M und Epoche nötig; für eine Parabel oder Hyperbel q, e, Orientierung und Tp. Nach einem Importwechsel muss gegebenenfalls die passende Referenz neu übernommen werden.

12.3 Didaktisches Beispiel mit 220P

Aus den eigenen Aufnahmen bereits vermessene Positionen von mehreren UTC-Tagen laden. Zunächst „220P ein Tag“, danach „220P drei Tage“ und schließlich „220P alle Tage“ berechnen. Gleiche Einstellungen und nachvollziehbare Messauswahl verwenden. Alle Lösungen gemeinsam mit genau einer MPC-Referenz animieren.

Die zentrale Lernfrage lautet: Wie verändert zusätzliche zeitliche Information die Bahn? Ein kleineres RMS ist nicht automatisch eine bessere langfristige Vorhersage. Eine sehr kurze Messspanne kann viele plausible Bahnen zulassen. Die Zahl der Messpunkte ersetzt keine ausreichend lange Zeitbasis.

12 Bedienung und Lehrscenario Fortsetzung

12.4 Bahn am Himmel

Im Reiter „Bahn am Himmel“ eigene Läufe und optional eine MPC-Referenz auswählen. MPC-Beobachtungsort oder WGS84-Koordinaten festlegen. Start, Ende und positive Schrittweite eingeben. Die Anzeige begrenzt die Anzahl auf höchstens 720 Zeitpunkte einschließlich Endpunkt; bei zu feinem Raster muss die Schrittweite erhöht werden.

Die Karte zeigt die projizierten Bahnen und synchrone Zeitmarker. RA und Deklination stehen in den Detailinformationen und im CSV-Export. Die Diagrammachsen sind projizierte Winkelkoordinaten, keine direkt ablesbaren RA-Stunden. Es gibt keinen Sternkatalog-Hintergrund.

Eigene Messpunkte erscheinen nur für den passenden Beobachtungsort und Zeitraum. Tagesfarben helfen, mehrere Beobachtungsnächte zu unterscheiden. Sehr weit vom Projektionszentrum entfernte Punkte über 175 Grad werden ausgelassen; unterbrochene Kurven werden nicht künstlich verbunden.

12.5 Exporte verwenden

Das Ergebnis-ZIP archiviert den Lauf mit Eingaben, Metadaten und verfügbaren Engine-Ausgaben. HTML-Animationen sind eigenständig und enthalten Plotly. Die Himmelsbahn kann zusätzlich als CSV mit Standort, Lauf, UTC, JD_UTC, RA_Grad, Dec_Grad, Entfernung_AE und Abstand_Referenz_Bogensekunden heruntergeladen werden.

HTML ist eine interaktive Darstellung im Browser, keine Videodatei. Ein erneutes Öffnen eines Ergebnis-ZIPs stellt die Auswahl der Streamlit-Sitzung nicht wieder her. Für Demonstrationen daher entweder die laufende Sitzung beibehalten oder die exportierten HTML-Dateien vorbereiten.

12.6 Empfehlung für eine Lehrveranstaltung

Drei benannte Lösungen, eine ausgewählte Referenz und kurze Animationszeiträume vorab erstellen. Die CSV-Beispiele und HTML-Exporte lokal bereithalten. Den Unterschied zwischen Messung, Anpassung, Referenz und vereinfachter Visualisierung ausdrücklich erläutern. Erst nach diesem Vergleich sollte eine Diskussion über Prognosequalität oder Ausreißer folgen.

13 Fehlerbehandlung

Meldung oder Beobachtung	Ursache und Vorgehen
Unbekannte Standortcodes	Lokale Standortdatei enthält den Code nicht. Datei prüfen oder betroffene Messungen bewusst ausschließen. Nicht irgendeinen anderen Standort einsetzen.
Error opening efindorb.txt	Engine-Konfiguration unvollständig oder falscher Pfad. Vollständige Begleitdateien bereitstellen.
Weniger als drei Zeiten	Auswahl oder Dublettenbereinigung lässt zu wenige unabhängige Zeitpunkte übrig. Weitere Messungen wählen.
Sun / J2000 nicht erkannt	Zentrum oder Bezugssystem des eigenen Laufs ist unbekannt oder nicht passend. Laufdaten prüfen; Referenzimport allein behebt dies nicht.
Objekt und Bezugssystem bestätigen	Bestätigung im Referenzformular fehlt. Objektidentität und Rahmen fachlich prüfen, anschließend bestätigen.
Mindestens einen Referenzparameter eingeben	Import noch nicht übernommen oder Tabelle leer. Treffer wählen beziehungsweise Werte eingeben.
Referenzepoche fehlt	Epoche des Elementsatzes fehlt. Aus Quelle ergänzen; Tp ist kein Ersatz.
MPC-Referenzbahn deaktiviert	Gespeicherte Referenz fehlt oder Zustand nicht bestimmbar. Für offene Bahn Tp und q prüfen; kein elliptisches M erzwingen.
Messpunkte fehlen am Himmel	Ort oder Zeitfenster passt nicht; Projektion kann weit entfernte Punkte ausschließen. Filter prüfen.
Läufe nach Neustart verschwunden	Auswahlliste gehört zur Sitzung. Laufdateien bleiben separat erhalten; keine automatische Wiederherstellung.
Berechnung abgebrochen	Engine-Protokoll und Ressourcen prüfen; Timeout standardmäßig 180 Sekunden. Kein erfolgreiches Ergebnis aus Teildateien ableiten.

13.1 Diagnoseinformationen

Für eine reproduzierbare Fehlermeldung Version, Importformat, anonymisierten Beispieldatensatz, gewählte Tage, Standort, Epoche und betroffene Bedienaktion notieren. run.log und manifest.json erlauben die technische Zuordnung. Ein Screenshot allein ersetzt keine reproduzierbare Eingabe.

14 Prüfungen und Rückverfolgbarkeit

14.1 Vorliegende Nachweise

Für Version 0.14.0 wurden in der Entwicklungsumgebung 57 automatisierte Tests als bestanden dokumentiert. Die neueren Prüfungen betreffen unter anderem Zeitraster, RA-Übergang, Polnähe, analytische Kreisbahn mit Lichtlaufzeit, Standorte, Messfilterung und Exporte. Streamlit-AppTests prüfen ausgewählte Bedienpfade, nicht sämtliche visuellen Interaktionen.

Der historische Linux-Integrationstest aus PRUEFBERICHT.md verwendet synthetische Eros-Positionen: zwölf Positionen für den Fit und drei ausgelassene Kontrollpositionen. Die Winkelabstände betrugen 0,0245, 0,0474 und 0,0751 Bogensekunden. Das Ergebnis gilt nur für diesen Datensatz und diese Engine-Konfiguration.

Für die neue Himmelsbahn liegt kein unabhängiger Präzisionsvergleich mit einer vollständigen externen Ephemeride vor. Ein nativer Windows-/USB-Test und eine vollständige visuelle Browserprüfung sind nicht nachgewiesen. Für diese Dokumentation wurde keine neue Find_Orb-Bahnberechnung als Abnahme durchgeführt.

14.2 Traceability Matrix

Epic	Stories	Use Cases	Anforderungen
EP-01	US-01, US-02	UC-01	FA-01–09
EP-02	US-03, US-04	UC-02	FA-10–18
EP-03	US-05, US-06	UC-03	FA-19–21
EP-04	US-07, US-08	UC-04	FA-22–28
EP-05	US-09, US-10	UC-05	FA-29–33
EP-06	US-11, US-12	UC-06	FA-34–38
EP-07	US-13, US-14	UC-07, UC-08	FA-39–40; NFA-01–12

14.3 Ergänzende Abnahmeszenarien

Vor Freigabe für einen konkreten Kurs mindestens einen vollständigen Durchlauf pro Importfamilie dokumentieren. Je einen elliptischen, parabolischen und hyperbolischen Referenzfall prüfen. Die Auswahl mehrerer Tage muss anhand der tatsächlichen Engine-Eingabe nachvollziehbar sein. Bei Zeitvergleichen UTC und TT ausdrücklich festhalten.

Die Matrix sichert die fachliche Zuordnung, nicht die vollständige Testabdeckung. Für eine formale Abnahme ist zusätzlich je FA/NFA eine Testkennung mit Eingabe, erwartetem Ergebnis, Toleranz, Umgebung und Prüfdatum zu führen.

15 Grenzen und Weiterentwicklung

15.1 Fachliche Grenzen

Kurzbeobachtungen können stark unsichere oder mehrdeutige Bahnen liefern. Viele Messungen innerhalb von zehn Minuten lösen das Entfernungsproblem nicht zwangsläufig. Systematische Messfehler, Kometenkoma und nichtgravitative Effekte können die Anpassung beeinflussen. Der Prototyp liefert keine vollständige Untersuchung aller zulässigen Kurzbahnlösungen.

Numerische Elementdifferenzen bei unterschiedlichen Epochen sind nur orientierend. Die Visualisierung verwendet ein vereinfachtes Bewegungsmodell und enthält keine Unsicherheitsregionen. Für die Himmelsbahn werden weder Horizontsichtbarkeit noch Tag/Nacht oder atmosphärische Refraktion geprüft.

15.2 Technische Grenzen

Sitzungen sind nicht dauerhaft über die Oberfläche wiederherstellbar. Der Vergleich eigener Läufe und die Mehrlaufanimation haben unterschiedliche Gruppierungsregeln. Referenzdateien werden nicht automatisch aktualisiert. Python-Abhängigkeiten sind nicht vollständig festgeschrieben. Die portable Lösung ist vorbereitet, aber noch nicht auf realem Windows vollständig abgenommen.

15.3 Priorisierte Weiterentwicklung

Priorität	Vorschlag	Abnahmepunkt
1	Windows und USB systematisch prüfen	NFA-12; regulär, portabel, offline und Laufwerkswechsel
1	Himmelsbahn unabhängig validieren	NFA-11 präzisieren; Referenzkonventionen und Toleranz festlegen
2	Läufe wiederherstellen	Archivimport mit Versionsprüfung und eindeutiger Objektzuordnung
2	Gemeinsame Vergleichsepoche	Elemente fachlich konsistent fortschreiben, statt nur Differenzen zu bilden
2	Reproduzierbare Umgebung	Getestete Abhängigkeiten fixieren und Engine-Version archivieren
3	Messbare Bedienleistung	NFA-10 unter festgelegter Last messen
3	Unsicherheit vermitteln	Mehrere zulässige Lösungen oder Unsicherheitsdarstellung fachlich entwickeln

Diese Punkte sind Entwicklungsvorschläge und keine vorhandenen Funktionen. Eine Erweiterung des Himmelsmodells um präzise Ephemeriden oder scheinbare Koordinaten benötigt eine eigene fachliche Spezifikation und unabhängige Tests.

16 Begriffe und Quellen

16.1 Glossar

Begriff	Bedeutung
Astrometrie	Messung von Himmelspositionen
RA und Dec	Rektaszension und Deklination; zwei Winkel einer Himmelsrichtung
AE	Astronomische Einheit als Längeneinheit
UTC und TT	Beobachtungszeitskala und terrestrische Zeitskala
JD	Fortlaufende Tageszahl; Zeitskala zusätzlich angeben
Epoche	Bezugszeitpunkt eines Zustands oder Elementsatzes
Tp	Zeitpunkt des Periheldurchgangs
Residuum	Abweichung einer Messung vom angepassten Modell
MPC	Minor Planet Center; Kataloge und Beobachtungsformate
Find_Orb	Externer Rechenkern für Bahnbestimmung und Ephemeriden
Heliozentrisch / topozentrisch	Auf die Sonne / einen Beobachtungsort bezogen
Zweikörpermodell	Bewegungsmodell aus Objekt und zentraler Sonne

16.2 Quellen und Nachweisbasis

Q1 SOPHIST: Anforderungen richtig formulieren – die Satzschablone MASTeR.

<https://blog.sophist.de/2026/08/26/anforderungen-richtig-formulieren-die-satzschablone-master-mit-beispielen/>

Q2 SOPHIST: Der EigenschaftsMASTER – Eine Anforderungsschablone für nicht-funktionale Anforderungen.

<https://blog.sophist.de/2020/07/13/der-eigenschaftsmaster-eine-anforderungsschablone-fuer-nicht-funktionale-anforderungen/>

Q3 Projektbasis: Orbitwerkstatt 0.14.0; Python-Module, tests/, README.md, PRUEFBERICHT.md, PORTABEL_ANLEITUNG.md und help/. Implementierungsangaben dieses Dokuments beziehen sich auf diesen Stand.

Q4 Weiterführende Formatdokumentation des MPC:

<https://www.minorplanetcenter.net/iau/info/MPOrbitFormat.html> sowie

<https://www.minorplanetcenter.net/iau/MPCORB/CometEls.txt> und

<https://www.minorplanetcenter.net/iau/MPCORB/AllCometEls.txt>.

Q5 Rechenkern und Hintergrund: https://www.projectpluto.com/find_orb.htm. Die konkrete ausführbare Engine und ihre Konfiguration sind im jeweiligen Laufmanifest nachzuweisen.

SOPHIST ist die Quelle der verwendeten Satzschablonen. Die hier formulierten Anforderungen, Stories und Use Cases sind anwendungsspezifische Ableitungen aus dem vorliegenden Projekt und keine von SOPHIST geprüfte Spezifikation.