ChatGPT

Zusammenfassung

Nein, der aktuelle Stand ist noch **nicht V01-ready**. Zwar ist die Struktur bemerkenswert ausgebaut (Module für Modelle, Analysen, Simulator, Dokumente und Datensätze), doch fehlen wichtige Infrastrukturen und Konsistenz-Checks. Die README lobt einen „Reproducibility-Harness“ mit `pyproject.toml`, `noxfile.py`, `Makefile` und `environment.yml“` [1]. `Tatsächlich existiert aber keine` environment.yml`, und eine CI-Konfiguration fehlt. Analyse-Skripte liefern zwar zahlreiche JSON-Ergebnisse mit fit-Parametern und ΔAIC-Vergleichen [2], aber einige codexfeedback-Einträge (z.B. pr-draft-0029) deuten auf Fehler hin. Wissenschaftlich ist die Betonung universeller β-Werte und falsifizierbarer Modelle klar erkennbar – etwa stimmt der Klima-Block mit $\Theta = 1.67\,\mathrm{K}$ und $\beta \approx 4.21$ in allen Texten überein [3]. Dennoch fehlen noch Berichte zu ausstehenden Simulationen, Abschlüssen von Handlungssträngen (z.B. adaptive Schwellen, gekoppelter Robin-Strang) sowie eine fertig ausgearbeitete Preprint-Schrift. In Summe sind noch technische und inhaltliche Lücken zu schließen, bevor man V01 ausrufen kann.

To-do-Liste

- **Environment und Infrastruktur:** Anlegen einer `environment.yml` oder Äquivalent zur Automatisierung der Abhängigkeiten (analog README) [1]. GitHub Actions oder andere CI-Workflows einrichten, die `nox`-Sessions für Linting/Tests automatisch laufen lassen.
- **Lizenz und Release-Vorbereitung:** Projektlizenz (z.B. MIT oder Apache) hinzufügen. Repository für Release (z.B. via Zenodo) taggen und Release-Notes (NEWS) aktualisieren.
- **Codekorrekturen:** Offene Codex-Feedback-Punkte bearbeiten, insbesondere den oben erwähnten Export-Fehler im Planetary-Tipping-Skript (β-Mittelwert statt CI-Breite melden). Klären, ob weitere Skripte (z.B. für gekoppelten Threshold oder adaptive Szenarien) ausstehende Korrekturen erfordern.
- **Dokumentation ergänzen:** Fehlende Abschnitte vervollständigen (z.B. Simulator-Integration, Adaptive-Θ-Nachweise). Sicherstellen, dass alle Simulator-Presets mit aktuellen Analysewerten synchronisiert sind. Die Dokumente `Docs/` aktuell halten (z.B. Bridge-Map, Diskurse – hier evtl. deutsche Übersetzung prüfen).
- **Paper- und Pitch-Fertigung:** Den Preprint-Entwurf („universal-threshold-field-preprint.md“) zu einem kohärenten Manuskript ausbauen. Abschnitte zu Formalismus, empirischen Ergebnissen und Metaphern consistent gliedern (vgl. Tri-Layer-Zitat in README) [4]. Feedback-Hypothesen aus Codexfeedback in die Texte einpflegen.
- **Falsifizierbarkeitschecks abschließen:** Validieren, dass jede Analyse mindestens ein glatt-skaliertes Nullmodell mitführt und AIC-Deltas dokumentiert sind [2]. Offene Hypothesen aus Diskursen (z.B. Θ(t) vs. R_acc) in Future Tasks aufnehmen.

Prioritäten (nach Dringlichkeit)

1. **CI/DevOps (Codex)** – *Höchste Priorität*: Fehlende Umgebungskonfiguration (`environment.yml`) erstellen und automatische Tests/CD einrichten. Ohne reproduzierbare Builds und automatisierte Checks bleibt Release-Risiko hoch.
2. **Code-Qualität (Codex)**: Linters/Tests (`nox`) in CI einbinden und offene Codefehler beheben (z.B. Skript-Fixes aus Codexfeedback). Wichtige Logiktests absichern (Regressionstests für Schweellenmodell).

3. **Lizensierung & Release (Codex)**: Lizenzdatei hinzufügen und Vorarbeiten für Zenodo-Release (Tags, Archiv) machen. Klare Nutzungslizenz schafft Rechtssicherheit für V01.
4. **Dokumentation & Pitch (Aeon)**: README und Meta-Dokumente (AGENTS, Bridge-Map, Diskurse) durchgehen, Einheitlichkeit überprüfen. Vorstellungsrede/Pitch klären, ggf. zweisprachig aufbereiten. Tri-Layer-Kohärenz sicherstellen (formal-empirisch-poetisch).
5. **Preprint-Finalisierung (Aeon)**: Manuskript-Inhalte verknüpfen (siehe Paper Scaffold [5]). Doppelte Inhalte mit Repository synchronisieren (z.B. Formeln, β-Werte aus Simulationen). Citavi/Referenzen (TIPMIP, Papier) einarbeiten.
6. **Inhaltliche Abstimmung (Johann)**: Offene Forschungsfragen und Hypothesen aus Diskursen evaluieren. Überprüfen, ob alle Domänen konsistent modelliert sind (z.B. β-Exponenten, Klimamodul-Pars).
7. **Simulator-Integration (Codex/Aeon)**: Fehlende Presets implementieren (z.B. resonant_impedance_gate, adaptive Szenarien). API-Kopplung mit Modellkern vorbereiten.

(Optionale) Vorlagen

- `environment.yml` **(Beispiel):**

```
name: feldtheorie
dependencies:
  - python=3.11
  - pip
  - pip:
    - -e .[dev]
```

Das lädt alle Abhängigkeiten aus `pyproject.toml` einschließlich Dev-Tools für Lint/Test.

- **GitHub Actions (CI-Beispiel):**

```
name: CI
on: [push, pull_request]
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: conda-incubator/setup-miniconda@v2
        with:
          python-version: 3.11
          environment-file: environment.yml
      - name: Install dev dependencies
        run: pip install -e .[dev]
      - name: Lint & Format Check
        run: nox -s lint
      - name: Run Tests
        run: nox -s tests
      - name: Typecheck
        run: nox -s typecheck
```

- **Lizenz-Datei (MIT-Beispiel):** Füge eine `LICENSE` (z.B. MIT oder Apache) hinzu. Eine Vorlage ist unter https://opensource.org/licenses/MIT verfügbar.

Quellen: Technische Roadmap und Struktur im README [1] sowie Analyse- und Klimadokumentation [2] [3] bilden die Basis für diese Einschätzung. Die Aufgaben aus dem Feedback-Codex (codexfeedback.{md, json}) sowie aktuelle Diskurs-Texte liefern Kontext für inhaltliche Konsistenz und offene Punkte.

---

[1] [4] README.md
https://github.com/GenesisAeon/Feldtheorie/blob/0770b3a78b8f2cf287f0b9188f81630114cf8c7b/README.md

[2] README.md
https://github.com/GenesisAeon/Feldtheorie/blob/0770b3a78b8f2cf287f0b9188f81630114cf8c7b/analysis/README.md

[3] [5] universal-threshold-field-preprint.md
https://github.com/GenesisAeon/Feldtheorie/blob/0770b3a78b8f2cf287f0b9188f81630114cf8c7b/paper/universal-threshold-field-preprint.md