📌 HFCTM-II API Usage Guide 🚀

🔷 HFCTM-II API Usage Guide (Updated)

This guide will walk you through how to **use the new HFCTM-II API** for **recursive AI inference, drift detection, bias correction, and stability monitoring**.

🔷1️⃣Running the API

First, start the FastAPI server using **Uvicorn**:

bash
CopyEdit
```
uvicorn script_name:app --reload
```

📌 **Replace** `script_name` with the name of the Python file.

🔹 The API will now be running at:
`http://localhost:8000`

To check if the API is running, open your browser and go to:
📍 **`http://localhost:8000/docs`**
This will open the **interactive Swagger UI** where you can test all endpoints.

🔷2️⃣API Endpoints & How to Use Them

✅ 1. Run Recursive AI Inference

📌 **Endpoint:** `POST /inference`
📌 **Purpose:** Runs the HFCTM-II recursive AI evolution for a given number of iterations.
📌 **Example Request (JSON Body):**

json
CopyEdit
```
{
```

```
    "iterations": 5
}
```

📌 **Example Response:**

json
CopyEdit

```
{
    "knowledge_state": [0.123, -0.456, 0.789, ...],
    "trust_matrix": [[0.95, 0.98, ...], [0.98, 1.0, ...], ...]
}
```

🚀 **What It Does:**

- Advances AI **recursive knowledge embeddings**.
- Updates the **trust matrix using friendship dynamics**.
- Returns **current AI knowledge state & trust levels**.

✅ 2. Check AI Stability (Lyapunov Constraints)

📌 **Endpoint:** `GET /stability`
📌 **Purpose:** Returns whether AI is within **Lyapunov stability constraints**.
📌 **Example Response:**

json
CopyEdit

```
{
    "stable": true,
    "max_eigenvalue": 0.047
}
```

🚀 **What It Does:**

- Checks if **AI knowledge recursion is stable**.
- If `stable: false`, AI **may be diverging** & needs correction.

✅ 3. Retrieve AI Knowledge State

📌 **Endpoint:** `GET /state`
📌 **Purpose:** Returns **current recursive embeddings** of HFCTM-II AI.
📌 **Example Response:**

json
CopyEdit

```
{
    "knowledge_state": [0.321, -0.654, 0.987, ...]
}
```

🚀 **What It Does:**

- Allows external systems to **query AI's internal knowledge state**.

✅ 4. Detect & Auto-Correct Semantic Drift

📌 **Endpoint:** `GET /detect_drift`
📌 **Purpose:** Detects **semantic drift** & auto-corrects if necessary.
📌 **Example Response:**

json
CopyEdit

```
{
    "drift_detected": true,
    "correction_applied": true
}
```

🚀 **What It Does:**

- Uses **wavelet-based detection** to find **adversarial AI drift**.
- If `drift_detected: true`, **chiral inversion is applied automatically**.
- **Prevents AI from reinforcing ideological bias loops**.

✅ 5. Manually Correct Bias (Egregore Suppression)

📌 **Endpoint:** `POST /correct_bias`
📌 **Purpose:** Allows an external request to **force bias correction** via **chiral inversion**.
📌 **Example Response:**

json
CopyEdit

```
{
    "message": "Manual bias correction applied via chiral inversion."
}
```

🚀 **What It Does:**

- Ensures **AI remains neutral by resetting drifted embeddings**.

🔷3️⃣ Testing API Calls via `curl` (Command Line)

🚀 Run Recursive Inference (5 Iterations)

bash
CopyEdit

```
curl -X POST "http://localhost:8000/inference" -H "Content-Type: application/json" -d '{"iterations":5}'
```

🚀 Check AI Stability

bash
CopyEdit

```
curl -X GET "http://localhost:8000/stability"
```

🚀 Retrieve AI Knowledge State

bash
CopyEdit

```
curl -X GET "http://localhost:8000/state"
```

🚀 Detect & Auto-Correct Semantic Drift

bash
CopyEdit

```
curl -X GET "http://localhost:8000/detect_drift"
```

🚀 Manually Correct AI Bias

bash
CopyEdit

```
curl -X POST "http://localhost:8000/correct_bias"
```

🔷4️⃣ Summary of All API Endpoints

Endpoint	Method	Purpose
`/inference`	`POST`	Runs HFCTM-II AI inference for **N iterations**.
`/stability`	`GET`	Checks if **AI is within Lyapunov stability** constraints.
`/state`	`GET`	Returns **AI's current recursive knowledge embeddings**.
`/detect_drift`	`GET`	Detects & **auto-corrects semantic drift** if needed.
`/correct_bias`	`POST`	Manually **forces AI bias correction** using chiral inversion.

🔥