Project Development Guide: Mixed-Feedback Model of Collective Emotion

1. 项目概述 (Project Overview)

本项目旨在实现一个基于统计物理的混合反馈模型（Mixed-Feedback Model），用于研究集体情绪中的相变、极化及临界慢化现象。项目分为理论计算（Mean-Field Theory）与网络模拟（Agent-Based Simulation）两大部分。

核心目标：

1. **理论求解**：基于 Ginzburg-Landau (GL) 方程，解析计算相变临界点 r_c 及稳态势能。
2. **数值模拟**：利用 Euler-Maruyama 方法求解随机微分方程 (SDE)，验证概率分布。
3. **网络仿真**：在复杂网络（BA/ER）上运行主体模型，验证拓扑结构下的理论预测。

2. 技术栈与目录结构 (Tech Stack & Structure)

Tech Stack: Python 3.9+ **Libraries:** `numpy`, `scipy`, `matplotlib`, `networkx`, `pandas`, `seaborn`

Directory Structure:

```
project_root/
├── src/
│   ├── __init__.py
│   ├── theory.py        # 理论核心：rc计算, alpha映射, 势能函数
│   ├── sde_solver.py    # SDE 数值解法 (Algorithm 1)
│   ├── network_sim.py   # 网络 ABM 模拟 (Algorithm 2)
│   └── utils.py         # 绘图与辅助工具
├── notebooks/           # 探索性分析与论文绘图
│   ├── 01_potential_landscape.ipynb  # 势能阱演化
│   ├── 02_bifurcation_diagram.ipynb  # 分岔图 (理论 vs 模拟)
│   └── 03_critical_slowing_down.ipynb # 临界慢化验证
├── tests/               # 单元测试
└── DEVELOPMENT.md       # 本文档
```

3. 开发计划 (Development Plan)

Phase 1: 理论核心模块 (Theoretical Core)

目标：实现 Method Part 2 中的所有解析公式，特别是临界点 r_c 和 GL 参数 α, u 的计算。

任务清单 (Tasks):

- [ ] **实现微观响应函数的导数计算** `calculate_chi(phi, theta, k_avg)`。
  - *关键点*：使用 `scipy.stats.binom` 计算边界概率密度。
- [ ] **实现临界点计算** `calculate_rc(n_m, n_w, chi)`。
  - *公式*：$r_c = 1 + \frac{n_w}{n_m}\frac{2-\chi}{2+\chi}$。
- [ ] **实现宏观系数映射** `get_gl_params(r, rc)`。
  - *逻辑*：$\alpha = r_c - r, u = 1.0$ (或其他拟合常数)。

- [ ] **实现有效势能函数** `potential_energy(q, alpha, u)`。
  - *公式*: $V(q) = \frac{1}{2}\alpha q^2 + \frac{1}{4}uq^4$。

验收标准 (Review Criteria):

- 当 θ, ϕ 对称且 $\chi > 2$ 时，函数应返回 $r_c < 1$。
- 当 $\alpha > 0$ 时，势能曲线应为单底（U型）；当 $\alpha < 0$ 时，应为双底（W型）。

Phase 2: SDE 数值求解器 (Stochastic Dynamics)

目标：利用 Langevin 动力学验证理论分布，并展示相变过程。

任务清单 (Tasks):

- [ ] **实现 Euler-Maruyama 迭代步。**
  - *方程*: $q_{t+1} = q_t + (-\alpha q_t - uq_t^3)dt + \sigma\sqrt{dt}\xi_t$。
- [ ] **开发 SDE 模拟主循环** `run_sde_simulation(...)`。
  - 支持并行跑多条轨迹 (Ensemble)。
- [ ] **实现稳态分布解析解** `theoretical_pdf(q, alpha, u, sigma)`。
  - *公式*: $P(q) \propto \exp(-V(q)/D)$。

验收标准 (Review Criteria):

- **分布验证**：运行 SDE 得到的 q 值直方图，必须能完美覆盖在解析解 $P(q)$ 的曲线上。
- **分岔验证**：扫描 $r \in [0, 1]$，绘制 q_{steady} vs r，应观测到 Pitchfork 分岔。

Phase 3: 网络主体模拟 (Network Simulation)

目标：实现 Method Part 3，引入网络拓扑和局部交互，验证理论的鲁棒性。

任务清单 (Tasks):

- [ ] **网络初始化**：集成 `networkx` 生成 BA 或 ER 网络。
- [ ] **实现局部感知逻辑 (Local Perception)。**
  - *输入*：全局 r, 邻居状态, 个人阈值 (ϕ_i, θ_i)。
  - *输出*：局部风险 p_i。
- [ ] **实现状态更新逻辑 (Stochastic Decision)。**
  - 基于二项分布规则或阈值规则更新 $S_i \in \{H, M, L\}$。
- [ ] **实现宏观统计器。**
  - 每个时间步计算 $Q(t)$ 和 $A(t)$。
- [ ] **封装模拟器类** `NetworkAgentModel`。

验收标准 (Review Criteria):

- **自洽性检查**：在 $\beta = 0$ (无邻居耦合) 且参数对称时，网络模拟的相变点应与 Phase 1 计算的 r_c 高度吻合。
- **不对称性检查**：当设置不对称阈值时，模拟应能自发演化出偏离 0.5 的 p^*，这是理论推导无法直接给出的。

Phase 4: 临界慢化与高级分析 (Critical Phenomena)

目标：计算自相关函数，验证 Critical Slowing Down (CSD)。

任务清单 (Tasks):

- [ ] **实现自相关函数计算** `calculate_autocorrelation(time_series, lag)`。
- [ ] **数据管线**：在 r 逼近 r_c 的过程中，记录 $q(t)$ 序列。
- [ ] **拟合弛豫时间**：$\tau \propto 1/|r - r_c|$。

验收标准 (Review Criteria):

- 随着 $r \to r_c$，自相关系数（Lag-1 Autocorrelation）应显著上升并趋近于 1。

4. 编码规范 (Coding Standards)

1. **参数解耦**：所有物理参数（n_m, n_w, θ, ϕ）必须在 config 字典或类属性中定义，严禁在计算逻辑中写死硬编码（Hard-coding）。
2. **向量化计算**：在 SDE 和理论计算中，尽量使用 NumPy 的向量化操作，避免 `for` 循环。
3. **随机数种子**：所有模拟函数必须接受 `seed` 参数，保证结果可复现 (Reproducibility)。
4. **文档字符串**：关键函数（特别是 `calculate_chi` 和 `step`）必须包含 Physics Docstring，说明对应论文中的哪个公式。

5. 快速启动 (Quick Start for Agent)

Agent 指令： "请首先阅读 Phase 1 的任务。在 `src/theory.py` 中实现 `calculate_chi` 和 `calculate_rc` 函数。请注意，计算 `chi` 时需要使用二项分布在边界处的概率质量近似。完成后，请编写一个简单的测试脚本，验证当 χ 很大时，r_c 是否小于 1。"

```
### 如何使用这个文档？

1. **复制**：将上面的内容保存为 `DEVELOPMENT.md` 放在你的项目根目录。
2. **喂给 Agent**：当你开始写代码时，直接把这个文件发给你的 AI 助手（Cursor, Claude, ChatGPT 等
3. **按阶段执行**：
    * 你可以说：“基于 DEVELOPMENT.md 的 Phase 1，请帮我生成 `src/theory.py` 的代码。”
    * 写完后说：“请根据 Phase 1 的验收标准，帮我写一个测试脚本来验证 $r_c$ 的推导是否正确。”

这样，Agent 就不会乱写，而是严格遵循你的物理逻辑进行工程实现。
```