组织独立重复模拟的目录、随机状态和资源分配

完整操作教程 · 参考固定 · 往返恢复验收

同一体系的独立重复模拟，需要分别记录初始状态与随机过程。把同一个TPR复制三份运行，或把一条轨迹分三段续算，都不能自动称为三个独立重复。本篇以GROMACS为例，从已验收的能量最小化结构开始，为两个重复分别生成速度，完成初始化阶段的独立任务组织。

运行边界：示例代码需按安装版本和真实项目输入复核；未提供真实生物数据，语法或小样本检查不代表科学结果和性能验收。

同一体系的独立重复模拟，需要分别记录初始状态与随机过程。把同一个TPR复制三份运行，或把一条轨迹分三段续算，都不能自动称为三个独立重复。本篇以GROMACS为例，从已验收的能量最小化结构开始，为两个重复分别生成速度，完成初始化阶段的独立任务组织。

先限定已验证的模拟协议

本例不是从零参数化体系。已有start.gro、topol.top及全部include文件，力场、质子化、溶剂、约束与积分设置均经过判断。base.mdp是已经验证的初始平衡模板，限定integrator=md，包含适用温度和能量、压缩坐标输出。两个重复要各自经过平衡和后续生产验收，不能直接将初始化段当作可比较生产结果。

层次	本例安排	不能混淆
共同材料	相同最小化结构与拓扑	记录完整预处理拓扑
初始速度	gen-vel=yes，不同gen-seed	续算时不重新生成速度
随机过程	分别设置ld-seed	具体是否使用取决于算法
资源	每项1个thread-MPI rank、2个OpenMP线程	两个同时任务会叠加CPU与内存

生成两个可追踪的初始化配置

以下在已有体系的项目根目录执行，不移动拓扑include路径。先将Python保存为prepare_replicas.py并运行。模板中速度与continuation相关项被明确替换，其他物理参数保持原协议；种子101和202仅为演示，不代表统计独立性的充分证明。

```
import re
from pathlib import Path
lines=Path("base.mdp").read_text().splitlines()
items=[];values={}
for line in lines:
    clean=line.split(";",1)[0].strip()
    if "=" in clean:
        key,value=[x.strip() for x in clean.split("=",1)]
        key=key.lower().replace("_","-")
        assert key not in values,"■■■■■■■"
        values[key]=value
        if key in {"gen-vel","gen-seed","ld-seed","continuation"}:continue
    items.append(line)
assert values.get("integrator")=="md"
root=Path("replicas-001");root.mkdir()
for name,seed in [("rep01",101),("rep02",202)]:
    d=root/name;d.mkdir()
    (d/"initial.mdp").write_text("\n".join(items)+
        f"\ncontinuation = no\ngen-vel = yes\ngen-seed = {seed}\nld-seed = {seed}\n")
```

```
set -euo pipefail
gmx --version > replicas-001/gromacs-version.txt 2>&1
for rep in rep01 rep02; do
  out="replicas-001/$rep"
  gmx grompp -f "$out/initial.mdp" -c start.gro -r start.gro -p topol.top \
    -o "$out/run.tpr" -po "$out/processed.mdp" -pp "$out/processed.top" \
    > "$out/grompp.log" 2>&1
```

```
  sha256sum start.gro "$out/initial.mdp" "$out/processed.top" "$out/run.tpr" \
    > "$out/inputs.sha256"
done
pids=()
for rep in rep01 rep02; do
  (
    out="replicas-001/$rep"
    /usr/bin/time -v -o "$out/run.time" gmx mdrun -s "$out/run.tpr" \
      -deffnm "$out/run" -ntmpi 1 -ntomp 2 -nb cpu > "$out/console.log" 2>&1
    for suffix in log cpt gro edr; do test -s "$out/run.$suffix"; done
  ) &
  pids+=("$!")
done
status=0
for pid in "${pids[@]}"; do wait "$pid" || status=$?; done
test "$status" -eq 0
```

确认不同初始状态与正常演化

核对两个processed.mdp中的种子、gen-vel和温度，保留TPR及展开后的processed.top。不同TPR的SHA只是追踪线索，必要时用gmx dump或轨迹读取检查实际初始速度；文件字节不同不能单独证明只改变了计划中的变量。grompp警告先解释，不通过自动增加maxwarn继续。

读取各自md.log、能量与检查点，确认阶段完成且温度、压力、约束及体系结构符合本阶段协议。两条轨迹后续应按相同规则舍弃平衡段、计算指标和评估不确定性；种子不同并不保证轨迹已充分采样或彼此统计无关。

先测CPU基线，再决定设备和并发

代码展示两个CPU任务并行，至少需要与其线程设置相符的获配额度并留出额外开销，不能在两核配额下直接照跑。MPI构建应按其启动方式另定，不机械套thread-MPI参数。GPU构建的非键计算卸载需先在相同体系验证，并为各任务明确分配可见设备；本例-nb cpu不会使用GPU进行非键计算。

先完成单项再测两项重叠，记录每条墙钟、RSS、实际GPU占用和整批吞吐。任务失败时其他子任务仍被wait回收，整批返回非零，日志留在独立目录；不能因其中一项成功就给两个重复都写通过。已完成段使用各自检查点续算，禁止跨重复交换checkpoint或重新生成速度后称为原轨迹延长。

每个重复记录自身的初始速度种子、随机算法设置、阶段起止步数和检查点路径。后续生产段若更换输出前缀，也要保持重复身份明确；在汇总表中用rep01与rep02作独立键，而不是根据轨迹文件排列顺序猜对应关系。

配套练习包

本PDF对应的ZIP包含本文代码块、主题练习模板或合成数据、README和SHA256校验文件。请从同一教程页面下载练习包；先核对校验值，再在独立目录复现。

参考资料

GROMACS速度与随机状态参数 · 来源资料

mdrun并行和设备接口 · 来源资料

GROMACS连续模拟与恢复边界 · 来源资料

操作建议：先在合成小样本上验证流程，再依据项目参考、标签合同、存储策略与获配资源验收真实数据。