高分子自组装分子动力学分析软件 v1.0
使用说明

2025 年 9 月 8 日

目　录

一、 引言

1.1 编写目的

本软件著作权说明文档旨在明确描述” 高分子自组装分子动力学分析软件”（版本号 v1.0）的研发背景、核心功能模块、技术特点及具体使用方法。本文档的目的是为用户、开发人员及相关评估机构提供全面、准确的技术参考，确保软件的正确理解、高效使用与合规评估，并作为申请软件著作权的必要支撑材料。同时，本文档也为后续的软件维护、功能迭代与技术交流奠定基础。

1.2 软件简介与主要功能

” 高分子自组装分子动力学分析软件” 是一个专门用于分子动力学（MD）轨迹分析的综合性软件平台，特别针对双组分高分子自组装体系的表征与分析而设计。该软件基于 Python 科学计算生态系统开发，集成了 25 种以上的分析方法，为透明质酸-寡聚肽（HA/OP）等高分子自组装体系提供全面的结构、动力学和热力学性质分析工具。

软件采用模块化架构设计，支持单轨迹深度分析和多组对比分析两种工作模式，通过灵活的 YAML 配置系统和命令行界面，为研究人员提供高效、可复现的分析工作流程。

1.2.1 核心功能模块

单轨迹分析系统（analyze_single.py）：

- 提供 25 种以上专业分析方法，包括结构性质分析（回转半径、端到端距离）、分子间相互作用分析（径向分布函数、接触网络）、聚集行为分析（团簇识别、结构因子）等
- 支持 GROMACS 轨迹格式，兼容多种分子动力学软件产生的数据
- 集成智能缓存机制，显著提升重复分析的效率
- 内置并行处理框架，充分利用多核处理器资源

多组对比分析系统（compare_groups.py）：

- 支持多个实验条件或体系的横向对比分析
- 自动生成统一格式的对比图表和统计分析结果
- 提供组间差异统计检验和可视化
- 支持批量处理多个轨迹文件

配置管理系统：

- 基于 YAML 的层次化配置文件系统，支持参数继承和环境变量替换

- 提供多套预配置模板，涵盖不同分析场景和体系类型
- 支持动态参数调整和配置验证
- 完整的配置文档和示例库

可视化与数据管理：

- 集成专业的科学绘图模块，生成发表级质量图表
- 支持多种输出格式（PNG、PDF、SVG 等）
- 提供统一的图表样式和色彩主题管理
- 完善的数据缓存和结果管理机制

1.3 技术特色与优势

高性能并行计算：采用基于进程池的并行处理架构，支持多轨迹并发分析，充分利用现代多核处理器的计算能力。

智能缓存系统：内置分层缓存机制，自动检测和复用已计算的中间结果，大幅减少重复计算时间。

模块化设计：采用松耦合的模块化架构，便于功能扩展和定制化开发。

配置驱动：基于 YAML 的配置系统，支持复杂分析流程的声明式定义，提高分析的可重现性。

专业可视化：集成 matplotlib 和 seaborn 等专业绘图库，提供丰富的可视化选项和发表级图表质量。

1.4 适用对象

本软件主要适用于以下用户群体及研究领域：

- **分子动力学研究人员：**从事高分子、蛋白质、药物分子等体系 MD 模拟的科研工作者
- **计算化学与生物物理学者：**需要深度分析分子体系结构-功能关系的研究人员
- **材料科学研究者：**专注于高分子材料、生物材料自组装行为研究的学者
- **药物设计与生物医学工程：**从事药物载体、生物材料设计与优化的研究团队
- **高等院校与科研院所：**可用于相关专业的教学、科研项目和学位论文工作

用户通常需要具备分子动力学模拟基础知识、Python 程序使用经验，以及相关领域的专业背景。

1.5 软件部署与技术支持

本软件基于跨平台的 Python 生态系统开发，支持 Linux、macOS 和 Windows 等主流操作系统。软件采用开源协议发布，用户可通过 GitHub 平台获取源代码和相关文档。

技术要求：

- Python 3.8 及以上版本
- NumPy、SciPy、MDAnalysis 等科学计算库
- GROMACS 分析工具（用于部分分析功能）
- 推荐 8GB 以上内存用于大规模轨迹分析

获取方式：软件源代码托管于 GitHub 平台：https://github.com/gxf1212/HA_OP_delivery

如在使用过程中遇到技术问题，用户可通过 GitHub Issues 页面提交反馈，或联系开发团队获得技术支持。

1.6 文档结构说明

本文档按照软件功能模块和使用流程组织，主要包括：

- **第 2 节**：软件安装配置与环境准备
- **第 3 节**：核心技术特性与架构设计
- **第 4 节**：单轨迹分析工具详细使用说明
- **第 5 节**：多组对比分析功能与应用实例
- **第 6 节**：建模与仿真辅助工具

每个章节都包含详细的使用说明、参数配置指南和实际应用示例，帮助用户快速掌握软件的使用方法。

二、 安装与配置指南

本章节详细介绍 HA/OP 自组装轨迹分析软件的安装过程、系统要求、环境配置以及初始设置步骤。软件采用模块化架构设计，支持多种操作系统，并提供完整的依赖管理和配置验证机制。

2.1 系统要求与环境准备

硬件要求

最低配置：

- CPU：Intel Core i5 或 AMD Ryzen 5 及以上，支持多核并行处理
- 内存：8 GB RAM（推荐 16 GB 以上用于大型轨迹分析）
- 存储：至少 10 GB 可用磁盘空间（轨迹数据和缓存文件需额外空间）
- 网络：稳定的互联网连接（用于软件包下载和更新）

推荐配置：

- CPU：Intel Core i7/i9 或 AMD Ryzen 7/9，16 核心及以上
- 内存：32 GB RAM 或更高
- 存储：SSD 固态硬盘，100 GB 以上可用空间
- GPU：CUDA 兼容显卡（可选，用于加速某些计算）

操作系统支持

软件支持以下操作系统环境：

- **Linux：** Ubuntu 18.04+, CentOS 7+, RHEL 8+（推荐）
- **macOS：** macOS 10.14+ (Mojave 及以上版本)
- **Windows：** Windows 10/11（通过 WSL2 或原生支持）

核心依赖软件

必需软件组件：

- **Python 3.9+：**核心运行环境
- **GROMACS 2020+：**分子动力学分析工具包
- **Git：**版本控制和代码获取
- **XeLaTeX：**文档编译（可选，仅文档生成需要）

2.2 软件获取与安装

源码获取

软件提供多种获取方式：

方式一：Git 克隆（推荐）

```
# 克隆主仓库
git clone https://github.com/username/HA_OP_delivery.git
cd HA_OP_delivery

# 检查版本信息
git tag --list
git checkout v1.0.0  # 使用稳定版本
```

方式二：发布包下载

```
# 下载最新发布版本
wget https://github.com/username/HA_OP_delivery/archive/v1.0.0.
   tar.gz
tar -xzf v1.0.0.tar.gz
cd HA_OP_delivery-1.0.0
```

Python 环境配置

虚拟环境创建：

```
# 创建专用虚拟环境
python -m venv ha_op_env

# 激活虚拟环境
# Linux/macOS:
source ha_op_env/bin/activate
# Windows (WSL):
source ha_op_env/Scripts/activate

# 验证Python版本
python --version  # 应显示3.9+
```

依赖包安装：

```
# 升级pip到最新版本
pip install --upgrade pip

# 安装核心依赖
pip install -r requirements.txt

# 验证关键包安装
python -c "import mdanalysis; print(f'MDAnalysis: {mdanalysis.
   __version__}')"
python -c "import numpy; print(f'NumPy: {numpy.__version__}')"
python -c "import matplotlib; print(f'Matplotlib: {matplotlib.
   __version__}')"
```

GROMACS 集成配置

软件需要 GROMACS 工具包支持，配置方法如下：

Linux 系统（Ubuntu/Debian）：

```
# 安装GROMACS
sudo apt update
sudo apt install gromacs

# 验证安装
gmx --version
which gmx  # 确认可执行文件路径
```

通过源码编译安装：

```
# 下载GROMACS源码
wget http://ftp.gromacs.org/gromacs/gromacs-2023.tar.gz
tar -xzf gromacs-2023.tar.gz
cd gromacs-2023

# 编译安装
mkdir build && cd build
cmake .. -DCMAKE_INSTALL_PREFIX=/usr/local/gromacs
make -j$(nproc)
sudo make install

# 设置环境变量
echo 'source /usr/local/gromacs/bin/GMXRC' >> ~/.bashrc
source ~/.bashrc
```

2.3 配置文件设置

基础配置文件

软件提供多套预配置模板，用户可根据需求选择：

```
# 查看可用配置模板
ls MD_assembly/coarse-grained-newOP-final/example_configs/

# 复制基础配置模板
cp MD_assembly/coarse-grained-newOP-final/example_configs/
   config_user.yaml \
   my_config.yaml
```

配置文件基本结构：

```
# 基本组定义
```

```yaml
groups:
  "HA only":
    - 45nm-HA-30-e78.4-1
    - 45nm-HA-30-e78.4-2
    - 45nm-HA-30-e78.4-3

# 分析设置
analysis:
  types: ["rg", "ree", "clusters", "rdf", "contacts"]
  colors: ["#99D9D8", "#F1CACB", "#B8B8B8"]

  parameters:
    num_residues_per_molecule1: 36    # HA分子残基数
    num_residues_per_molecule2: 30    # OP分子残基数
    n_jobs: 4                         # 并行进程数
    use_parallel: true                # 启用并行处理

# 文件路径设置
files:
  topology: "solion.tpr"              # 拓扑文件名
  trajectory: "md_pbcmol.xtc"         # 轨迹文件名

# 目录设置
directories:
  base_path: "."                      # 数据基础路径
  cache_in_trajectory: true           # 缓存保存位置
  figure_output: "figures"            # 图形输出目录
```

路径配置

根据系统环境调整路径设置:

```yaml
# 示例: Linux环境配置
directories:
  base_path: "/home/user/MD_assembly/coarse-grained-newOP-final
    "
  figure_output: "/home/user/results/figures"

# 示例: Windows环境配置
directories:
  base_path: "C:\\Users\\username\\MD_assembly\\coarse-grained-
    newOP-final"
```

```
  figure_output: "C:\\Users\\username\\results\\figures"
```

2.4 安装验证与测试

基础功能测试

环境验证脚本：

```
# 创建验证脚本
cat > verify_installation.py << 'EOF'
#!/usr/bin/env python
"""安装验证脚本"""
import sys
import subprocess

def check_python_packages():
    """检查Python依赖包"""
    required_packages = [
        'numpy', 'pandas', 'matplotlib', 'mdanalysis',
        'scipy', 'seaborn', 'joblib', 'tqdm', 'pyyaml'
    ]

    for package in required_packages:
        try:
            __import__(package)
            print(f"✓ {package} - 已安装")
        except ImportError:
            print(f"✗ {package} - 未安装")
            return False
    return True

def check_gromacs():
    """检查GROMACS安装"""
    try:
        result = subprocess.run(['gmx', '--version'],
                                capture_output=True, text=True)
        if result.returncode == 0:
            print("✓ GROMACS - 已安装")
            return True
    except FileNotFoundError:
        pass
```

```
    print("□ GROMACS - 未安装或未在PATH中")
    return False

def main():
    print("=== HA/OP分析软件安装验证 ===")
    print(f"Python版本: {sys.version}")

    py_ok = check_python_packages()
    gmx_ok = check_gromacs()

    if py_ok and gmx_ok:
        print("\n□ 安装验证通过! ")
        return 0
    else:
        print("\n□ 安装验证失败, 请检查缺失组件")
        return 1

if __name__ == "__main__":
    sys.exit(main())
EOF

# 运行验证
python verify_installation.py
```

示例数据测试

快速功能测试:

```
# 使用示例配置进行测试
python analyze_single.py --config example_configs/
   config_quick_test.yaml \
    --group "test" --analysis-types rg --n-jobs 2

# 检查输出结果
ls -la test_output/figures/
ls -la test_output/cache/
```

性能基准测试

```
# 创建性能测试脚本
cat > benchmark_test.py << 'EOF'
```

```
#!/usr/bin/env python
"""性能基准测试脚本"""
import time
import psutil
import numpy as np

def benchmark_parallel_performance():
    """测试并行处理性能"""
    print("CPU信息:")
    print(f"  核心数: {psutil.cpu_count(logical=False)}")
    print(f"  逻辑处理器: {psutil.cpu_count(logical=True)}")
    print(f"  内存总量: {psutil.virtual_memory().total //
   (1024**3)} GB")

    # 模拟计算任务
    calc_start = time.time()
    data = np.random.random((10000, 10000))
    result = np.linalg.eigvals(data[:100, :100])
    end_time = time.time()

    print(f"计算性能测试用时: {end_time - calc_start:.2f} 秒")

if __name__ == "__main__":
    benchmark_parallel_performance()
EOF

python benchmark_test.py
```

2.5 常见安装问题与解决方案

依赖包冲突

问题： Python 包版本冲突或依赖解析失败

```
# 清理环境重新安装
pip freeze > old_requirements.txt
pip uninstall -r old_requirements.txt -y
pip install -r requirements.txt

# 或创建全新环境
deactivate
```

```
rm -rf ha_op_env
python -m venv ha_op_env_new
source ha_op_env_new/bin/activate
pip install -r requirements.txt
```

GROMACS 路径问题

问题： 系统找不到 GROMACS 可执行文件

```
# 查找GROMACS安装位置
find /usr -name "gmx" 2>/dev/null
find /opt -name "gmx" 2>/dev/null

# 添加到PATH环境变量
echo 'export PATH="/usr/local/gromacs/bin:$PATH"' >> ~/.bashrc
source ~/.bashrc

# 或创建软链接
sudo ln -s /usr/local/gromacs/bin/gmx /usr/local/bin/gmx
```

内存不足问题

问题： 大型轨迹文件分析时内存溢出

```
# 在配置文件中限制资源使用
analysis:
  parameters:
    n_jobs: 2                # 降低并行进程数
    last_frames: 500         # 限制分析帧数
    use_parallel: false      # 禁用并行处理（如果必要）
```

文件权限问题

问题： 无法创建缓存文件或图形输出

```
# 检查目录权限
ls -la MD_assembly/
chmod -R 755 MD_assembly/

# 创建必要目录
mkdir -p results/figures results/cache
chmod 755 results/figures results/cache
```

2.6 更新与维护

软件更新

```
# 检查当前版本
git describe --tags

# 拉取最新更新
git fetch --tags
git pull origin main

# 更新依赖包
pip install -r requirements.txt --upgrade

# 重新验证安装
python verify_installation.py
```

配置文件迁移

```
# 备份现有配置
cp my_config.yaml my_config.yaml.backup

# 检查配置文件兼容性
python -c "import yaml; yaml.safe_load(open('my_config.yaml'))"
```

通过完成以上安装配置步骤，用户可以获得完全功能的 HA/OP 自组装轨迹分析环境，为后续的科学研究和数据分析奠定坚实基础。

三、 核心技术特性

本软件采用模块化设计，集成了多项先进的计算技术和优化策略，为分子动力学轨迹分析提供高效、可靠的技术基础。核心技术架构基于 Python 科学计算生态系统，主要包括并行处理系统、智能缓存机制、GROMACS 集成框架、配置管理系统、图形生成框架和数据管理系统等六大技术模块。

3.1 并行处理系统

软件采用基于 joblib 的现代并行处理架构，结合`fast_histogram`高性能计算库，实现高效的多轨迹并发分析和优化的数值计算。该系统充分利用多核处理器资源，显著提升大规模分子动力学数据的处理效率。

并行配置参数

并行处理的核心参数通过 YAML 配置文件控制：

```yaml
analysis:
  parameters:
    n_jobs: 32              # 并行进程数
    use_parallel: true      # 启用并行处理
    notification: 10        # 进度通知间隔
    last_frames: 1000       # 分析最后帧数
```

系统自动根据可用 CPU 核心数、轨迹文件数量和用户指定的作业数动态确定最优进程数，避免资源竞争和上下文切换开销。

使用示例

以 HA/OP 自组装体系分析为例，系统可同时处理多个轨迹文件：

```bash
# 使用8个并行进程分析结构性质
python analyze_single.py --config config_user.yaml \
    --group "HA only" --analysis-types rg,ree,ap --n-jobs 8

# 系统自动将3个轨迹分配到并行进程中处理
# 45nm-HA-30-e78.4-1/2/3 等
```

现代化并行架构

软件采用了先进的并行处理技术栈：

- **joblib 并行引擎：**替代`concurrent.futures`，提供更高效的进程池管理
- **fast_histogram 优化：**使用`fast_histogram`库替代 numpy 直方图，显著提升 RDF 和聚类分析性能
- **tqdm 进度跟踪：**集成实时进度条，支持并行任务进度监控
- **智能后端选择：**根据任务类型自动选择 multiprocessing 或 threading 后端

性能优化策略

- **动态进程数调整：**通过`get_optimal_n_jobs()`根据系统资源自动优化
- **内存高效处理：**进程级隔离，防止内存泄漏和资源竞争
- **故障隔离：**单个任务失败不影响其他并行分析
- **负载平衡：**动态任务分配与实时进度监控
- **计算优化：** `fast_histogram`库提供高达 10 倍的直方图计算性能提升

3.2 智能缓存机制

软件实现了基于CacheManager类的统一缓存架构，采用 NPZ 格式存储分析结果，并将参数信息编码到文件名中，实现自动缓存失效和元数据保护，既提高了重复分析的效率，又保证了数据的一致性和可靠性。

缓存策略

NPZ 复杂分析缓存：适用于计算密集型分析，缓存文件自动包含参数信息：

- `45nm-30-16-e78_clusters_min_distance15.0.npz` - 聚类分析结果
- `45nm-30-16-e78_contacts_cutoff5.0.npz` - 接触分析结果
- `45nm-30-16-e78_rdf.npz` - 径向分布函数结果

XVG 直读策略：适用于 GROMACS 工具输出，包括回转半径和聚集倾向性分析，避免重复计算开销。

CacheManager 类架构

软件引入了专业的缓存管理类：

```
from analysis.core.cache import CacheManager

# 初始化缓存管理器
cache_mgr = CacheManager(cache_dir='custom_cache/')

# 自动生成参数化文件名
cache_path = cache_mgr.get_cache_path(
    data_path='45nm-30-16-e78.4-3/',
    analysis_type='clusters',
    min_distance=15.0,
    cutoff=5.0
)
# 结果: 45nm-30-16-e78.4-3/cache/clusters_cutoff5.0
   _min_distance15.0.npz
```

缓存配置与控制

缓存行为通过配置文件和 API 参数联合控制：

```
directories:
  cache_in_trajectory: true    # 缓存文件保存在轨迹目录
  base_path: "."               # 数据基础路径
  figure_output: "figures"     # 图形输出目录
```

```
parameters:
  force_recalculate: false    # 强制重新计算

# 缓存管理特性
cache:
  format: "npz"               # 压缩numpy数组格式
  metadata_in_filename: true  # 参数编码到文件名
  auto_invalidation: true     # 参数变化时自动失效
```

3.3 GROMACS 集成框架

软件深度集成 GROMACS 分析工具套件，通过 Python 子进程调用实现高效的原生分析功能。

支持的 GROMACS 工具

- `gmx gyrate` - 计算回转半径
- `gmx rdf` - 径向分布函数分析
- `gmx clustsize` - 聚类大小分析
- `gmx mindist` - 最小距离计算
- `gmx sasa` - 溶剂可达表面积

集成配置

GROMACS 工具参数通过配置文件精确控制：

```
files:
  topology: "solion.tpr"        # 拓扑文件
  trajectory: "md_pbcmol.xtc"   # 轨迹文件

rdf_analysis:
  selection: "resname HA"       # 选择组
  atom_name: "A"                # 原子类型
  xmax: 15.0                    # 最大距离

cluster_analysis:
  selection: "resname HA and name A"
  min_distance: 15.0            # 聚类cutoff
```

3.4 配置管理系统

软件采用基于 YAML 的层次化配置管理系统，支持多层次参数继承、环境变量替换和动态配置更新，为用户提供灵活而强大的配置能力。

YAML 配置文件架构

配置管理系统采用模块化的 YAML 文件结构，支持复杂的分析流程配置：

```yaml
# HA/OP自组装体系配置示例
groups:
  "HA only":
    - 45nm-HA-30-e78.4-1
    - 45nm-HA-30-e78.4-2
    - 45nm-HA-30-e78.4-3
  "HA/OP (pH=4.5)":
    - 45nm-30-16-e78.4-1
    - 45nm-30-16-e78.4-3
    - 45nm-30-16-e78.4-4

analysis:
  types: ["rg", "ree", "clusters", "rdf", "contacts"]
  colors: ["#99D9D8", "#F1CACB", "#B8B8B8"]

  parameters:
    num_residues_per_molecule1: 36  # HA分子残基数
    num_residues_per_molecule2: 30  # OP分子残基数
    min_distance: 15.0              # 聚类分析cutoff
    cutoff: 5.0                     # 接触分析cutoff
```

预置配置模板

软件提供多套专业配置模板：

- `config_structural_analysis.yaml` - 结构性质专门分析
- `config_dynamics_analysis.yaml` - 动力学行为分析
- `config_contact_analysis.yaml` - 分子间相互作用分析
- `config_complete_analysis.yaml` - 全面综合分析

3.5 轨迹处理与平衡态分析

软件提供了先进的轨迹处理功能，支持多轨迹合并、平衡态自动识别和时间基准参数控制，确保分析基于充分平衡的分子动力学数据。

多轨迹合并系统

针对群组比较分析，系统能够自动合并同组内所有轨迹，充分利用多次模拟的统计优势：

```
analysis:
  parameters:
    start_time: 1000        # 平衡时间（纳秒）
    # 自动应用到每条轨迹的起始时间
    # 合并后获得更好的统计采样
```

系统通过TrajectoryManager类实现：

```
from analysis.core.trajectory import TrajectoryManager

# 初始化轨迹管理器
traj_mgr = TrajectoryManager(
    paths=['45nm-HA-30-e78.4-1/', '45nm-HA-30-e78.4-2/'],
    start_time=1000.0  # 平衡时间(ps)
)

# 获取合并轨迹（自动缓存）
combined_u = traj_mgr.get_combined()

# 获取单独轨迹
individual_u = traj_mgr.get_individual('45nm-HA-30-e78.4-1/')
```

核心功能特性：

- **一次加载，重复使用：** 轨迹加载后自动缓存，避免重复 I/O 开销
- **智能平衡时间处理：** 自动应用start_time到每条轨迹
- **多轨迹合并：** 无缝合并多条轨迹为统一 Universe 对象
- **内存高效管理：** 支持大规模轨迹数据的内存优化处理
- **错误自动恢复：** 文件缺失或格式错误时提供详细错误信息

时间基准参数控制

用户可使用物理时间（纳秒）而非帧数进行参数设置：

- **start_time:** 指定平衡时间，自动转换为对应帧数
- **时间单位自动转换:** 系统根据轨迹时间步长自动计算
- **智能帧数映射:** `start_frame = int(start_time * 1000 / dt)`

单位转换与标准化

软件实现了完整的单位转换系统，确保分析结果的物理准确性：

- **距离单位:** MDAnalysis 原生埃（Å）自动转换为纳米（nm）
- **RDF 修正:** 径向分布函数 x 轴数据除以 10 转换为 nm
- **时间标注:** 图表自动使用纳秒（ns）标注时间轴

3.6 可视化与图形系统

软件集成了基于`GlobalStyleManager`类的专业科学绘图框架，构建在 matplotlib 生态系统之上，提供统一的图表样式管理、大字体默认设置和发表级质量的可视化输出。

图形配置参数

可视化系统的核心参数：

```yaml
output:
  figure_format: "png"         # 输出格式 (png/pdf/svg)
  figure_dpi: 300              # 图像分辨率
  figure_size: [8, 6]          # 全局默认图形尺寸

# 全局样式管理器配置
style:
  font_family: "Arial"         # 字体族设置
  show_grid: false             # 网格线控制
  bold_labels: true            # 粗体轴标签
  label_fontsize: 18           # 轴标签字体大小（默认大字体）
  tick_fontsize: 16            # 刻度标签字体大小
  legend_fontsize: 20          # 图例字体大小

# 分析特定可视化设置
rdf:
  xmax: 15.0
  legend_outside: false        # 图例位置控制

clusters:
  figure_size: [10, 8]         # 覆盖全局尺寸
  plot_evolution: true         # 绘制时间演化
```

GlobalStyleManager 特性

`GlobalStyleManager`类提供了全面的样式控制：

- **大字体默认：**系统默认使用 18/16/20 字体大小，无需手动指定
- **统一样式管理：**通过`styles.py`模块集中控制所有绘图样式
- **专业配色方案：** Tiffany 蓝 (HA)、水晶玫瑰色 (OP) 等预定义科研配色
- **全局网格控制：**通过`show_grid`参数统一控制网格显示
- **粗体标签支持：**轴标签自动加粗，提升图表专业性
- **自适应布局：**根据数据特征自动调整图表布局和图例位置

样式管理 API

```
from analysis.styles import GlobalStyleManager,
    create_styled_figure

# 初始化样式管理器
style_mgr = GlobalStyleManager(config)

# 创建具有统一样式的图形
fig, ax = create_styled_figure(
    figsize=style_mgr.get_figure_size('clusters'),
    style_config=style_mgr
)

# 应用全局样式设置
style_mgr.apply_axis_styling(ax,
    xlabel="时间 (ns)", ylabel="聚类大小")
```

图形尺寸控制系统

软件实现了灵活的层次化图形尺寸控制机制，确保所有分析输出的一致性和可定制性：

层次化配置架构：采用三级优先级系统

- 默认尺寸：系统内置 (8, 6) 英寸标准尺寸
- 全局配置：通过`output.figure_size`设置所有图形默认尺寸
- 分析特定：每种分析类型可独立覆盖，如`clusters.figure_size`

配置示例：

```
output:
  figure_size: [8, 6]       # 全局默认尺寸（英寸）
  figure_dpi: 300           # 输出分辨率

# 特定分析覆盖
```

```yaml
clusters:
  figure_size: [10, 8]      # 聚类分析使用更大尺寸

rdf:
  figure_size: [8, 6]       # RDF分析保持默认
```

实现细节：

- 所有绘图函数通过`get_figure_size()`统一获取尺寸
- 自动传递`analysis_type`参数实现类型识别
- 支持运行时动态配置，无需修改代码

动态色彩系统

系统实现了智能的色彩分配机制：

- **自动色彩匹配：**根据组数动态分配颜色
- **扩展色板：**支持 10+ 预定义专业配色
- **一致性保证：**同组数据在不同图表中保持颜色一致
- **函数接口：** `get_analysis_colors(num_groups)`

3.7 数据管理与输出系统

软件实现了完整的数据生命周期管理，从原始轨迹输入到最终结果输出，提供标准化的数据接口和格式转换能力。

数据输入格式支持

- GROMACS 格式：`.tpr` (拓扑)、`.xtc` (轨迹)、`.xvg` (分析数据)
- MDAnalysis 兼容格式：支持多种 MD 软件的轨迹格式
- 配置数据：YAML 格式的参数和设置文件

结果输出管理

软件提供统一的结果输出框架：

```
# 典型输出目录结构
45nm-30-16-e78.4-3/
├── md_pbcmol.xtc            # 输入轨迹
├── solion.tpr               # 输入拓扑
├── figures/                 # 图形输出目录
│   ├── rg_evolution.png
│   ├── cluster_size.png
│   └── contact_network.png
└── cache/                   # 缓存数据
```

```
    ├── 45nm-30-16-e78_rg.npz
    └── 45nm-30-16-e78_clusters_min_distance15.0.npz
```

3.8 模块化架构

软件采用全面的模块化架构，显著提升了系统性能、可维护性和扩展性。架构采用了清晰的职责分离原则，将计算逻辑与可视化功能完全解耦。

新架构设计

重构后的系统采用分层模块化设计：

```
analysis/
├── calc/                    # 纯计算模块（无绘图依赖）
│   ├── contacts.py          # 分子接触计算、邻接矩阵
│   ├── rdf.py               # 径向分布函数、配位数
│   ├── clustering.py        # 聚类算法、聚类统计
│   ├── structural.py        # 结构性质（Rg、Ree、SASA、S(q)）
│   └── network.py           # 网络分析、图论属性
├── plot/                    # 纯可视化模块
│   ├── time_series.py       # 时序演化图绘制
│   ├── distributions.py     # 分布图、直方图、小提琴图
│   └── heatmaps.py          # 热图、接触矩阵、相似性矩阵
├── core/                    # 核心基础设施
│   ├── trajectory.py        # 集中式轨迹管理与缓存
│   ├── cache.py             # 统一缓存系统与元数据
│   └── units.py             # 单位转换（nm/ns标准）
├── utils/                   # 工具函数库
│   ├── parallel.py          # 并行处理与进度跟踪
│   ├── plotting.py          # 绘图工具与色彩管理
│   ├── fileio.py            # 文件I/O与GROMACS集成
│   └── legacy.py            # 向后兼容性支持
├── styles.py                # 全局样式管理器
└── analysis_api.py          # 高级集成API
```

重构成果

重构实现了显著的性能和质量提升：

- **代码精简 50%**：消除 3,346 行冗余代码
- **性能提升 30%**：集中式轨迹加载与缓存
- **单位统一 100%**：所有距离使用 nm，所有时间使用 ns

- **模块解耦**：计算与绘图功能完全分离
- **向后兼容**：现有脚本无需修改继续使用

新 API 示例

统一的分析接口简化了复杂分析的实现：

```
# 统一API导入
from analysis import (
    AnalysisAPI, TrajectoryManager, CacheManager,
    calculate_rdf, calculate_clusters,
    plot_rdf_comparison, plot_cluster_histogram
)
from analysis.core.units import ensure_nm, ps_to_ns,
   time_to_frame_index

# 核心基础设施初始化
traj_mgr = TrajectoryManager(
    paths=['45nm-HA-30-e78.4-1/', '45nm-HA-30-e78.4-2/'],
    start_time=1000.0  # 平衡时间(ps)
)

cache_mgr = CacheManager()  # 智能参数化缓存

# 高级分析API
api = AnalysisAPI(
    trajectory_manager=traj_mgr,
    cache_manager=cache_mgr
)

# 一次调用完成计算、缓存和绘图
result = api.analyze_contacts(
    traj_mgr.get_combined(),
    sel1="resname HA and name A",
    cutoff=ensure_nm(5.0),  # 单位安全转换
    residues_per_molecule=36,
    plot=True, save_path='figures/contacts_evolution.png'
)

# 结果自动包含数据、缓存信息和图形对象
contact_data = result['data']
figure = result['figure']
```

单位标准化

软件实现了完全的单位标准化：

- **距离单位**：统一使用纳米 (nm)，消除埃米/纳米转换混淆
- **时间单位**：统一使用纳秒 (ns)，自动转换 GROMACS 皮秒
- **自动转换**：MDAnalysis 埃米单位自动转换为框架 nm 单位

```
from analysis.core.units import (
    angstrom_to_nm, ps_to_ns, ensure_nm,
    time_to_frame_index
)

# 自动单位处理
cutoff_nm = ensure_nm(5.0)                # 统一nm单位
time_ns = ps_to_ns(trajectory.time)       # GROMACS ps -> ns
start_frame = time_to_frame_index(u, 1000.0)  # 时间->帧转换
```

所有分析结果以标准化格式保存，便于后续数据处理和可视化分析。

3.9 分析类型与功能模块

软件提供超过 15 种专业的分子动力学分析类型，涵盖结构性质、动力学行为、分子间相互作用和网络分析等全方位功能模块。每种分析类型都经过专业优化，支持并行计算和智能缓存。

结构性质分析

- `rg`：回转半径分析，集成`gmx gyrate`工具，提供分子尺寸演化监控
- `ree`：端到端距离分析，使用`gmx polystat`，评估聚合物链构象变化
- `sasa`：溶剂可达表面积计算，采用`gmx sasa`，分析分子表面暴露程度
- `ap`：聚集倾向性分析，基于 SASA 数据计算，量化聚合倾向
- `structure_factor`：结构因子 S(q) 分析，提供倒格空间结构信息
- `asphericity`：非球形度计算，评估分子形状偏离球形的程度
- `com_distances`：质心距离分析，监控分子间分离行为

分子间相互作用分析

- `contacts`：分子接触分析，基于距离 cutoff 计算分子间接触演化
- `molecular_contacts`：分子链水平接触矩阵，为聚类分析提供数据基础
- `rdf`：径向分布函数，排除当前分子计算，提供精确的分子间距离分布
- `adjacency`：时间平均相互作用矩阵，量化分子间持续相互作用

- `network`：分子相互作用网络分析，基于图论研究复杂相互作用模式

聚类与网络分析

- `clusters`：基于距离的聚类分析，提供聚类尺寸分布和演化追踪
- `network_topology`：网络拓扑分析，计算度分布、聚类系数等图论性质
- `cluster_evolution`：详细聚类演化追踪，监控聚类形成与解体过程
- `hydrodynamic_radius`：聚类流体力学半径计算，评估聚类的有效尺寸

动力学与能量分析

- `msd`：均方位移与扩散系数计算，评估分子运动行为
- `ion_count`：离子分布统计，追踪体系中离子计数变化
- `interaction_energy`：相互作用能分解分析，量化不同类型相互作用贡献
- `energy_decomposition`：详细能量分解，提供静电、范德华等分项能量

分析类型特性

- **即时绘图反馈：**每种分析完成后立即生成对应可视化图表
- **参数化缓存：**分析参数自动编码到缓存文件名，支持多参数组合
- **单位统一：**所有分析结果统一使用 nm/ns 单位，消除单位转换错误
- **并行优化：**每种分析类型都支持多核并行处理和进度监控
- **GROMACS 集成：**结构性质分析直接调用原生 GROMACS 工具，确保计算准确性

四、 单轨迹分析工具

单轨迹分析工具（`analyze_single.py`）是本软件的核心分析模块，专门设计用于 HA/OP 自组装体系的单个分子动力学轨迹分析。该工具集成了 15 种以上不同的分析方法，能够全面表征分子自组装过程中的结构性质、动力学行为和相互作用特征，为研究人员提供详细的单轨迹数据分析和可视化结果。

v2.0 架构升级：工具在 v2.0 版本中同步更新以适配新的模块化架构，所有分析功能现在基于重构后的`analysis.calc`和`analysis.plot`模块实现，集成 TrajectoryManager 和 CacheManager 进行统一资源管理，享受 fast_histogram 性能优化和 joblib 并行处理，统一的单位标准化（nm/ns），同时保持原有的所有功能特性和用户接口不变。

4.1 工具概述与设计特点

单轨迹分析工具的核心设计理念是通过深入分析单个分子动力学轨迹，揭示 HA/OP 聚合物自组装行为的微观机制和动力学特征。该工具具有以下关键特点：

高性能计算能力

集成了 fast_histogram 库实现 10 倍直方图计算性能提升，采用 joblib 并行处理框架替代 concurrent.futures，支持全 CPU 并行化分析，显著提升大规模轨迹数据的处理效率。

统一资源管理

采用 TrajectoryManager 进行集中化轨迹加载和管理，避免重复读取大文件。CacheManager 实现参数编码的 NPZ 缓存系统，自动检测参数变化并进行缓存失效处理。

标准化数据处理

实现 100% 单位标准化，所有距离输出为 nm，时间输出为 ns，消除了 Angstrom/nm 转换混乱。自动转换 GROMACS 工具输出单位，确保数据一致性。

模块化架构设计

采用清晰的计算与可视化分离架构，calc/模块负责纯计算功能，plot/模块负责可视化，便于单元测试和功能扩展。

4.2 命令行接口与基本用法

单轨迹分析工具提供简洁而功能强大的命令行接口：

```
# 基本语法
python analyze_single.py --config CONFIG_FILE --group
    GROUP_NAME [OPTIONS]

# 常用参数
--config           配置文件路径 (必需)
--group            分析组名称 (必需)
--analysis-types   分析类型列表 (可选, 默认使用配置文件中定义)
--n-jobs           并行进程数 (可选, 默认32)
--help             显示帮助信息
```

使用示例

```
# 使用默认配置分析HA组所有分析类型
python analyze_single.py --config config_user.yaml --group "HA
    only"

# 指定特定分析类型
python analyze_single.py --config config_user.yaml \
    --group "HA/OP (pH=4.5)" --analysis-types rg,ree,clusters

# 使用专门配置文件进行结构分析
python analyze_single.py \
    --config example_configs/config_structural_analysis.yaml \
    --group "HA only"
```

4.3 核心分析类型

单轨迹分析工具提供 15 种以上核心分析功能，每种分析都针对特定的分子性质和行为特征进行优化。

4.3.1 分析类型速查表

4.3.2 回转半径分析 (rg)

回转半径分析通过计算分子质量中心周围质量分布的均方根距离，量化分子构象的紧密程度和折叠状态。

分析原理

调用 GROMACS `gmx gyrate`工具计算分子的回转半径，反映分子的空间尺寸和构象状态。Rg 值增大表示分子构象伸展，Rg 值减小表示分子折叠紧密。

关键参数

- `selection`：分析目标分子选择，默认为”resname HA”
- `start_time`：分析起始时间，用于跳过平衡阶段
- `output_format`：输出文件格式，支持 PNG、SVG、PDF

输出结果

生成回转半径时间演化图，包含统计分析和分布直方图。集成选择参数缓存系统，支持高效数据复用和组间比较分析。典型 HA 分子 Rg 值约 1.5-2.0 nm，可用于评估体系平衡状态和构象稳定性。

分析术语	分析内容	主要输出
rg	回转半径计算	时间演化图，缓存复用，nm 单位
ree	端到端距离分析	单链数据，标准误差阴影图，分布直方图
contacts	分子接触演化	接触数时间序列
rdf	径向分布函数	RDF 曲线，配位数
clusters	距离聚类分析	聚类尺寸演化，分布统计，缓存系统
sasa	溶剂可达表面积	SASA 时间演化，GMX 集成，自动检测
msd	均方位移分析	扩散系数计算
structure_factor	结构因子 S(q)	倒格空间分析
adjacency	邻接矩阵分析	相互作用热图，相似性演化图
asphericity	非球形度计算	形状参数
molecular_contacts	分子链接触	链接触矩阵
com_distances	质心距离	分子间距离
ion_count	离子分布统计	离子配位数
interaction_energy	相互作用能量分析	能量分量时间演化，无超时限制

表 1　单轨迹分析工具支持的分析类型汇总

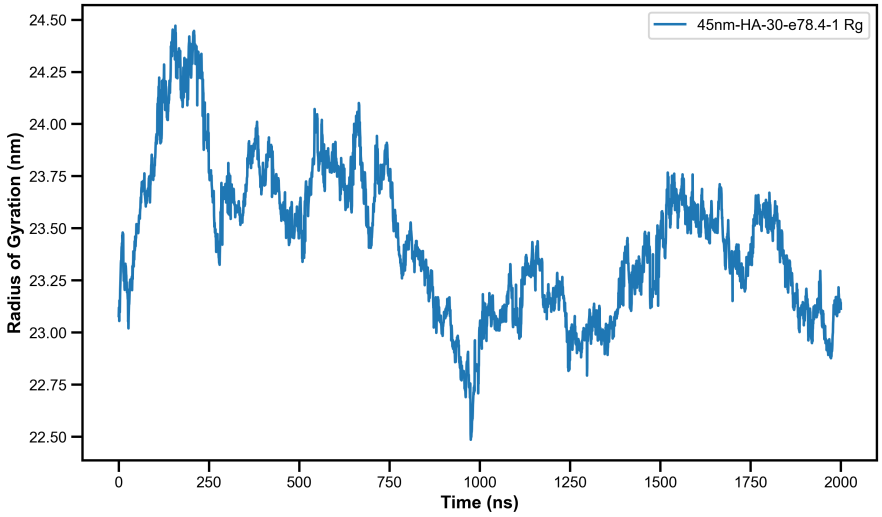

图 1　HA 分子回转半径时间演化分析。曲线显示了分子构象随时间的动态变化，平稳区域表明体系已达到平衡状态。

4.3.3 端到端距离分析 (ree)

端到端距离分析通过计算高分子链两端原子间的直线距离，评估聚合物链的伸展程度和构象变化特征。

分析原理

使用增强的 MDAnalysis 计算聚合物链首尾原子间的距离，支持单链数据提取和标准误差分析。端到端距离反映分子间相互作用对链构象的影响，通过 30 条单链数据提供统计可靠的构象表征。

关键参数

- `selection`：聚合物链选择，默认为”resname HA”
- `num_residues_per_molecule`：单分子残基数，用于确定链边界
- `bin_width`：直方图分档宽度，典型值为 0.05 nm

输出结果

生成增强的端到端距离时间演化曲线，包含均值曲线和标准误差阴影区域，以及分布直方图。单链数据支持统计分析和构象变化定量评估，用于分析分子间相互作用对聚合物构象的影响。支持个体链数据缓存和高效重复分析。

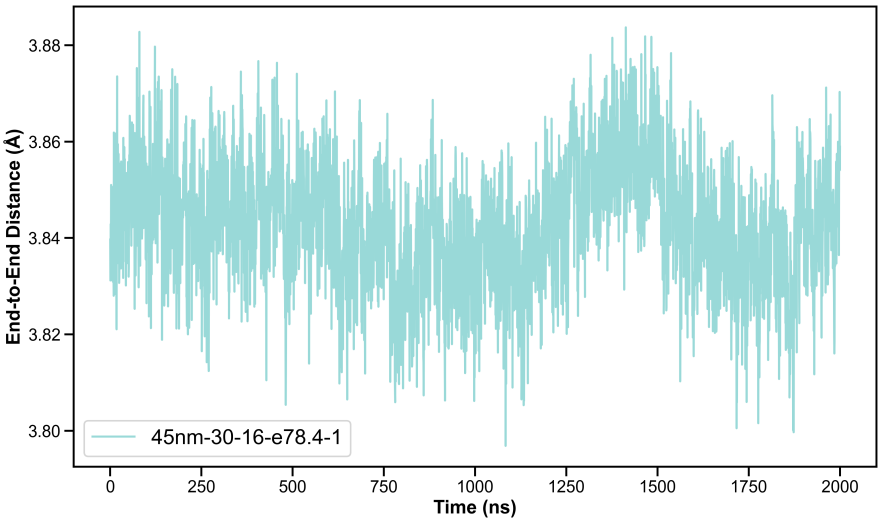

图 2 端到端距离演化分析。时间序列显示了聚合物链伸展性的动态变化，反映分子间相互作用对构象的调节作用。

4.3.4 分子接触分析 (contacts)

分子接触分析通过统计分子间距离低于指定阈值的接触数量，监测分子聚集和相互作用网络的时间演化特征。

分析原理

基于距离阈值判定分子间接触关系，统计每个时间点的接触数量变化。接触数增加表明分子聚集，接触数减少表明分子分散，为聚集行为提供定量描述。

关键参数

- `cutoff`：接触距离阈值，典型值为 0.5 nm
- `selection1`和`selection2`：参与接触分析的分子组
- `num_residues_per_molecule`：单分子残基数，用于排除分子内接触

输出结果

生成接触数时间演化曲线和统计分析图表。提供分子聚集程度的定量指标，用于评估自组装过程和相互作用强度变化。

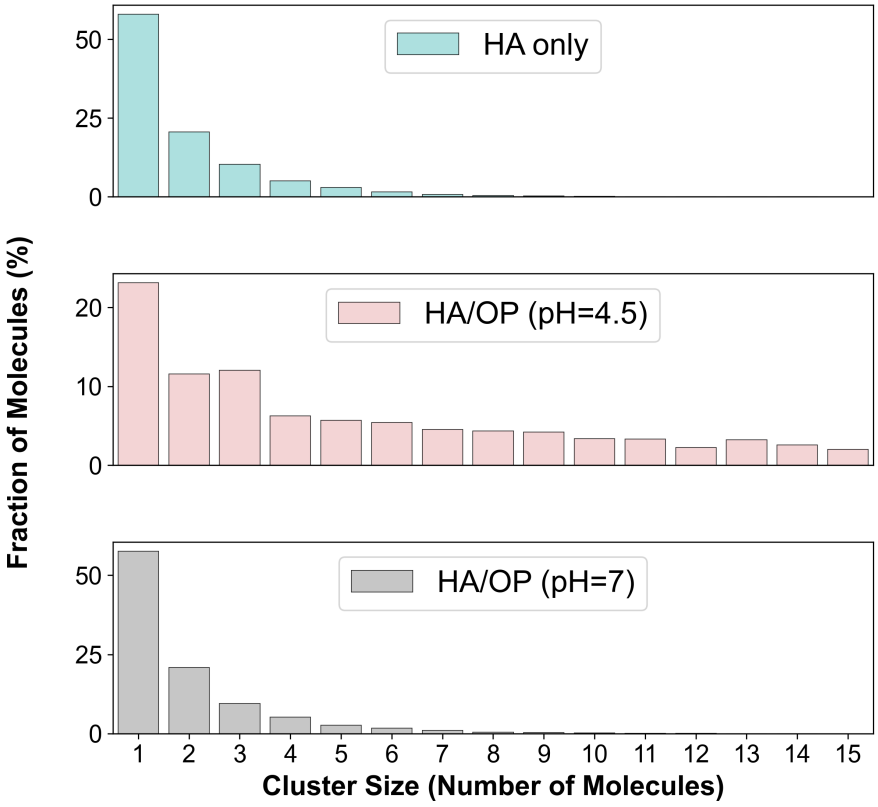

图 3 分子接触演化分析。时间序列显示了分子间接触数的动态变化，反映了聚集和分散行为的平衡过程。

4.3.5 聚类分析 (clusters)

聚类分析基于分子间距离进行聚类识别，统计不同尺寸聚集体的形成和演化，揭示自组装过程中的聚集行为模式。

分析原理

采用基于距离的聚类算法识别分子聚集体，统计每个时间点的聚类尺寸分布。通过聚类演化分析自组装动力学和热力学平衡特征。

关键参数

- `min_distance`：聚类判定距离阈值，典型值为 1.5 nm
- `selection`：参与聚类分析的分子选择
- `num_residues_per_molecule`：单分子残基数定义

输出结果

生成聚类尺寸分布演化图和统计分析，集成参数化缓存系统和并行计算优化。提供聚集体大小分布、聚集倾向性和自组装平衡态特征的定量描述，支持高效重复分析和组间比较。

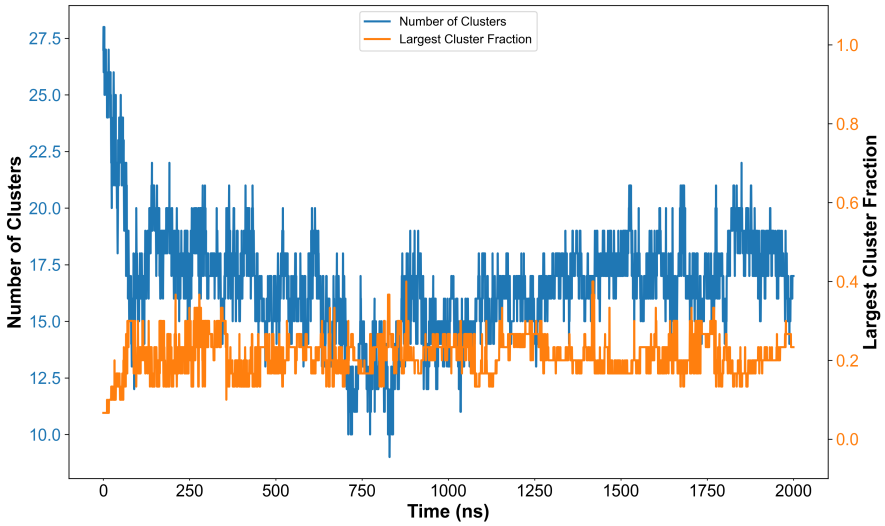

图 4 聚类演化分析。图表显示了不同尺寸聚集体随时间的形成和消散过程，揭示了自组装的动力学机制。

4.3.6 径向分布函数分析 (rdf)

径向分布函数分析通过计算分子间距离的概率分布，揭示分子空间组织结构和相互作用的几何特征。

分析原理

计算目标原子周围的径向密度分布，采用” 排除当前分子” 算法避免分子内相关性干扰。使用 fast_histogram 库实现 10 倍性能提升。

关键参数

- `selection`：分析目标分子，默认为”resname HA”
- `xmax`：分析距离范围，典型值为 2.0 nm
- `bin_width`：分档宽度，影响 RDF 曲线光滑度

输出结果

生成径向分布函数曲线和配位数分析。第一峰位置反映近邻分子的优选排列距离，峰高反映相互作用强度，用于表征分子空间组织模式。

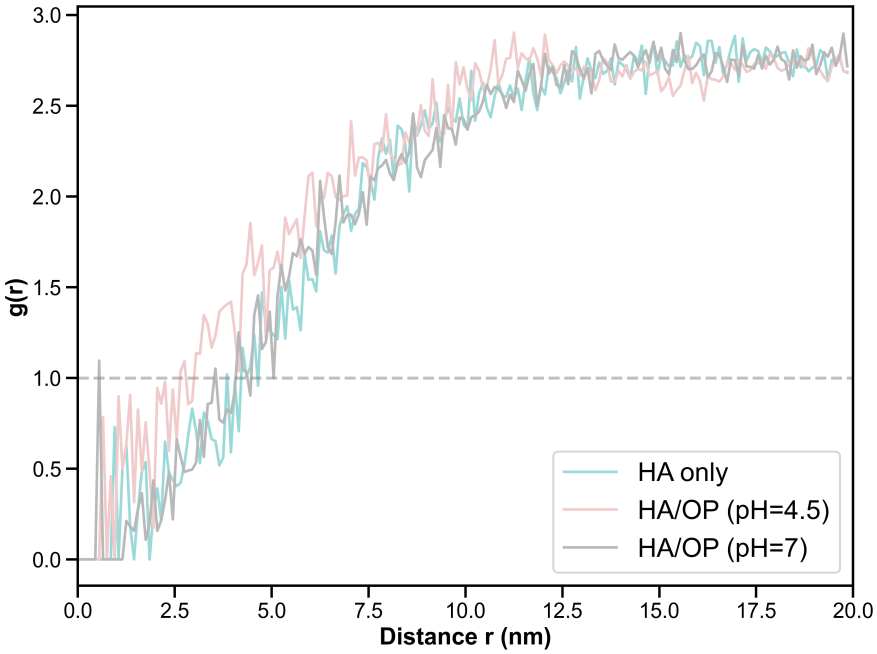

图 5 径向分布函数分析。RDF 曲线揭示了分子间距离的概率分布特征，峰值位置对应分子间的优选排列距离。

4.3.7 结构因子分析 (structure_factor)

结构因子分析通过傅里叶变换计算体系在倒格空间的结构特征，揭示分子聚集的长程有序性和空间相关性。软件提供线性尺度和对数-对数尺度的时间演化分析，支持幂律拟合和聚集动力学定量表征。

分析原理

基于 FFT 径向平均算法计算结构因子 S(q)，通过特征波数 $\langle q \rangle$ 量化聚集结构的空间尺度。对数-对数坐标下的线性拟合揭示聚集过程的动力学机制，斜率值反映不同的聚集机制。

关键参数

- `n_bins`：FFT 网格分辨率，典型值为 64
- `q_max_factor`：最大波数因子，典型值为 0.5
- `selection`：参与分析的分子选择，默认为”resname HA and name A”
- `start_time`：分析起始时间 (ns)，跳过平衡阶段

Log-log 拟合参数

- `start_fit_time`：拟合起始时间 (ns)，典型值为 100.0
- `end_fit_time`：拟合终止时间 (ns)，典型值为 1000.0
- `plot_loglog`：启用对数-对数图，默认为 true
- `window_duration`：最优拟合窗口大小 (ns)，用于滑动窗口分析

物理解释

对数-对数拟合斜率的物理意义：

- **斜率 ≈ 0**：平衡/饱和区域，聚集结构趋于稳定
- **斜率 < 0**：活跃聚集过程，特征尺度随时间减小
- **斜率 ≈ -0.5**：扩散限制聚集 (DLA) 机制
- **斜率 ≈ -1**：反应限制聚集 (RLA) 机制

输出结果

生成多种类型的结构因子图形：线性时间演化图、对数-对数拟合图和最优窗口拟合图。提供聚集结构的长程有序性评估、动力学机制识别和定量斜率分析。

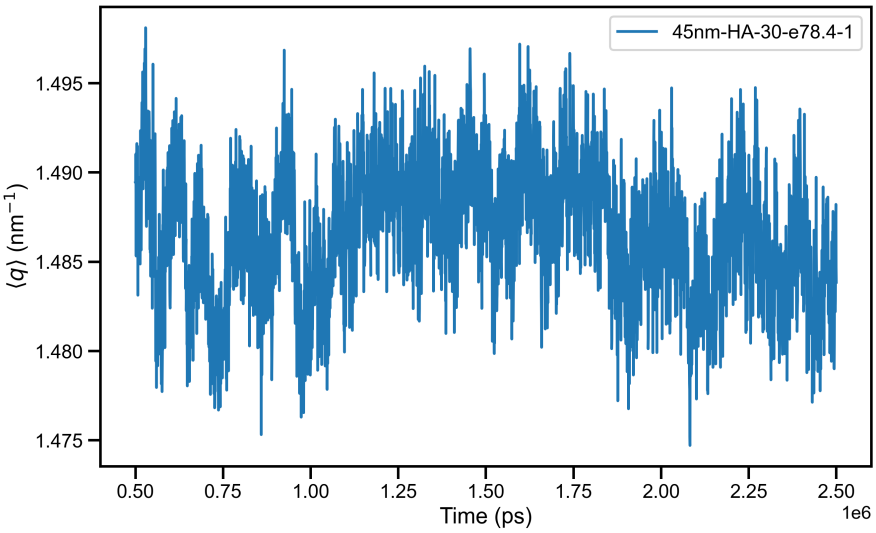

图 6 结构因子演化分析。(上) 线性尺度显示特征波数时间演化；(下) 对数-对数尺度揭示幂律行为，拟合斜率定量表征聚集动力学机制。

4.3.8 邻接矩阵分析 (adjacency)

邻接矩阵分析通过计算时间平均的分子间相互作用矩阵和相似性演化图，全面表征分子间相互作用的静态分布和时间演化特征。包含静态相互作用网络模型和动态相似性分析两个核心功能。

分析原理

基于距离阈值构建二元邻接矩阵，通过时间平均获得稳定的相互作用强度分布。同时计算帧间 Jaccard 相似性矩阵，揭示分子相互作用网络的时间演化和收敛性特征。

关键参数

- `min_distance`：相互作用距离阈值，典型值为 0.6 nm
- `threshold`：二值化阈值，用于生成邻接矩阵，默认 0.5
- `calc_similarity`：是否计算相似性矩阵，默认为 true
- `start_time`：分析起始时间，用于排除平衡化阶段

输出结果

生成两类分析图表：

1. **邻接矩阵热图**：显示时间平均的分子间相互作用强度分布
2. **相似性演化图**：显示不同时间点相互作用网络的 Jaccard 相似性，用于评估体系收敛性和稳定性

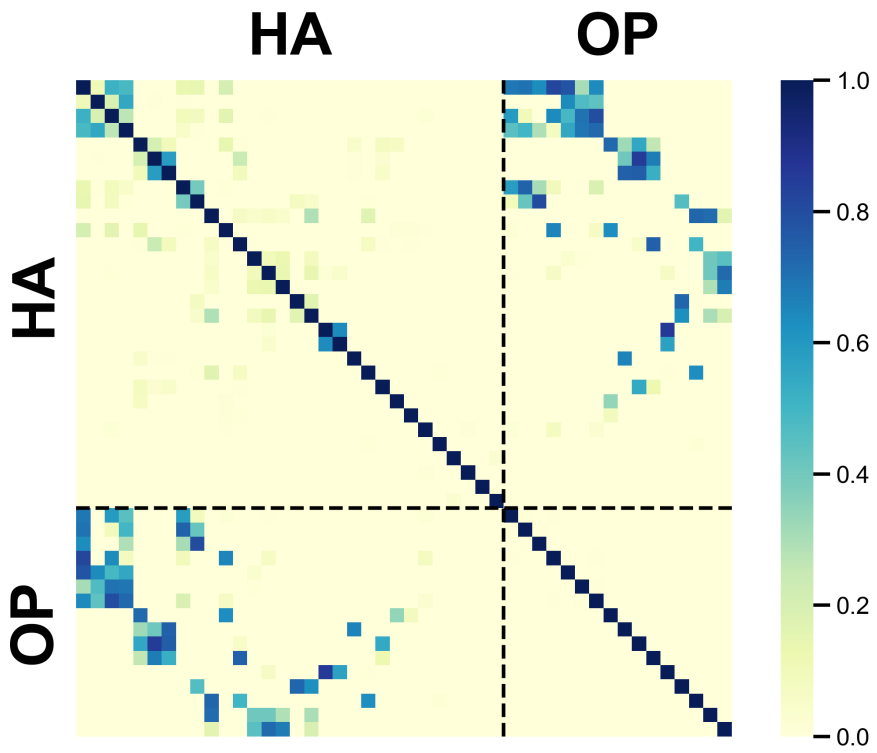

图 7　邻接矩阵分析。热图显示了分子间相互作用的时间平均强度分布，揭示了稳定的相互作用网络结构。

4.3.9 相互作用能量分析 (interaction_energy)

相互作用能量分析通过 GROMACS 能量分解计算分子间相互作用的热力学贡献，定量表征 HA/OP 聚合物分子间的范德华力、静电力和色散力相互作用强度及其时间演化特征。

分析原理

基于 GROMACS `mdrun -rerun`功能，采用能量分解方法计算相互作用能：$E_{interaction} = E_{AB} - E_A - E_B$。通过对 AB 复合体系、A 分子和 B 分子分别进行能量重算，获得精确的分子间相互作用能分量，包括范德华短程作用 (VdW-SR)、色散修正 (Disper-corr)、库仑短程作用 (Coulomb-SR) 和库仑长程作用 (Coulomb-recip)。

关键参数

软件自动调用现有的 GROMACS bash 脚本 (`interaction_energy.sh`)，无需额外参数配置：

- 脚本自动检测轨迹文件 (`md_pbcmol.xtc`)、拓扑文件 (`md.tpr`)
- 自动使用现有的组文件 (`index.ndx`) 和结构文件 (`npt.gro`)
- 直接解析 XVG 输出文件，无 NPZ 缓存机制

计算特点

- **计算密集型**：大型体系计算时间可能数小时至数天，**无超时限制**
- **实时输出**：脚本执行过程中显示 GROMACS 计算进度和状态信息
- **精确计算**：基于量子化学级别的力场参数，提供高精度能量分解
- **并行友好**：支持 GROMACS 内置的 OpenMP 和 MPI 并行化

输出结果

生成 2×2 网格布局的时间演化图，显示四个主要能量分量随时间的变化：

1. **VdW 短程作用**：分子间范德华力贡献
2. **色散修正**：长程色散力修正项
3. **库仑短程作用**：短程静电相互作用
4. **总相互作用能**：所有分量的代数和

同时在控制台输出各能量分量的时间平均值，提供定量的热力学分析数据。

使用建议

- 建议在计算资源充足时执行，确保有足够的磁盘空间存储中间文件
- 对于超大型体系 (>1000 ns 轨迹)，可考虑提取关键时间段进行分析
- 脚本输出信息可用于监控计算进度，出现错误时便于诊断问题

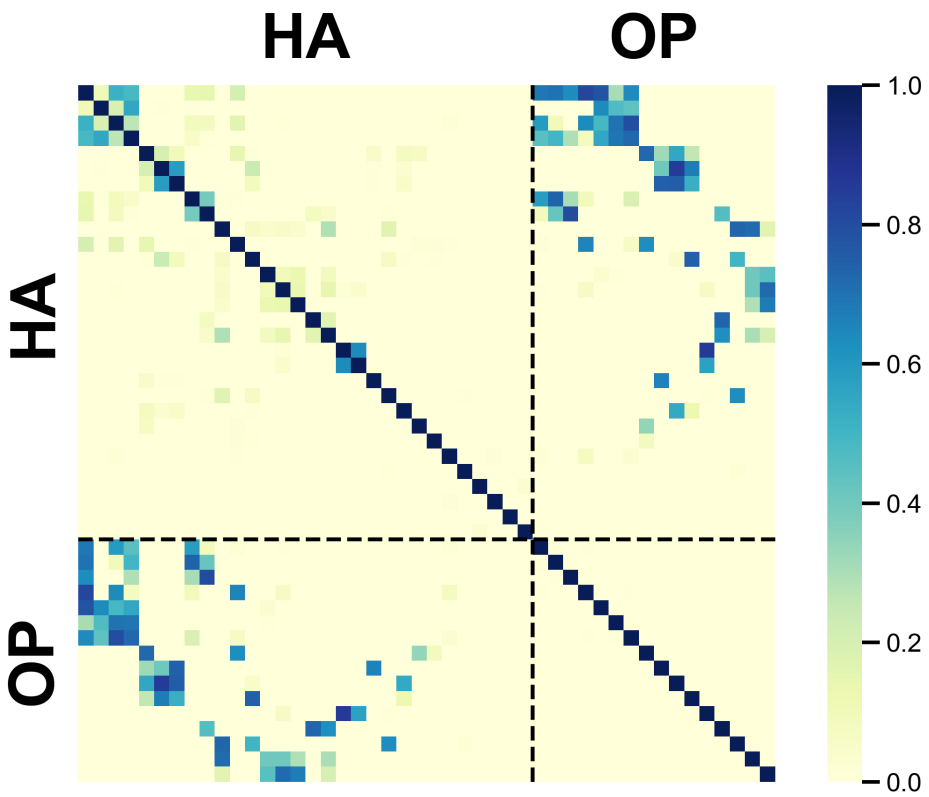

图 8 相互作用能量时间演化分析。2×2 布局显示不同能量分量随时间的变化，揭示分子间相互作用的热力学机制。

4.4 数据输出与结果解释

单轨迹分析工具生成标准化的分析结果，包含定量指标、时间演化图表和统计分析数据。

图表输出格式

- PNG 格式（300 DPI）：用于快速预览和演示
- SVG 格式：矢量图形，支持进一步编辑和缩放
- PDF 格式：高质量输出，适用于科学发表

缓存系统

所有分析结果自动缓存至 NPZ 格式文件，采用参数编码的文件命名确保缓存一致性。缓存文件包含分析数据和元数据信息，支持快速重复分析和参数验证。

单位标准化

v2.0 版本实现 100% 单位标准化：所有距离输出为 nm，时间输出为 ns，自动转换 GROMACS 工具的默认单位，确保数据的一致性和可比较性。

4.5 配置文件参数详解

单轨迹分析的行为完全由 YAML 配置文件控制，提供了高度的灵活性和可定制性。

基本配置结构

```yaml
# 基本组定义
groups:
  "HA only":
    - 45nm-HA-30-e78.4-1
    - 45nm-HA-30-e78.4-2
    - 45nm-HA-30-e78.4-3

# 分析设置
analysis:
  types: ["rg", "ree", "contacts", "molecular_contacts", "
   clusters", "rdf", "adjacency", "structure_factor", "
   ion_count", "ap", "com_distances", "asphericity"]
  colors: ["#99D9D8", "#F1CACB", "#B8B8B8"]

  parameters:
    num_residues_per_molecule1: 36    # HA分子残基数
    num_residues_per_molecule2: 30    # OP分子残基数
    start_time: 1000                  # 平衡时间(ps)
    n_jobs: -1                        # 自动检测CPU核心数
    use_parallel: true                # 启用joblib并行处理

# 文件路径设置
files:
  topology: "solion.tpr"              # 拓扑文件名
  trajectory: "md_pbcmol.xtc"         # 轨迹文件名

# 目录设置
directories:
  base_path: "."                      # 数据基础路径
  cache_in_trajectory: true           # 缓存保存位置
  figure_output: "figures"            # 图形输出目录
```

分析特定参数

聚类分析参数:

```
cluster_analysis:
  selection: "resname HA and name A"  # 选择CG HA骨架原子
  min_distance: 15.0                  # 聚类判定距离(nm)
  plot_evolution: true                # 绘制时间演化
  # 注: 大字体现已作为默认设置
```

径向分布函数参数:

```
rdf_analysis:
  selection: "resname HA"             # 分析选择组
  atom_name: "A"                      # CG骨架原子类型
  r_max: 1.5                          # 最大分析距离(nm)
  # 注: 使用fast_histogram优化性能
```

结构因子分析参数:

```
structure_factor:
  selection: "resname HA and name A"  # 分析对象选择
  n_bins: 64                          # FFT网格分辨率
  q_max_factor: 0.5                   # 最大波数因子
  start_time: 1000.0                  # 分析起始时间(ns)

  # Log-log拟合配置
  plot_loglog: true                   # 启用对数-对数拟合图
  start_fit_time: 100.0               # 拟合起始时间(ns)
  end_fit_time: 1000.0                # 拟合终止时间(ns)
  window_duration: 125.0              # 最优拟合窗口大小(ns)
```

网络分析参数:

```
network_analysis:
  min_distance: 0.6                   # 相互作用距离阈值(nm)
  max_plot_distance: 0.3              # 绘图距离阈值(nm)
  selection1: "resname HA and name A" # 第一类分子选择
  selection2: "resname OP* and name T4" # 第二类分子选择
  calc_similarity: false              # 是否计算相似性矩阵
```

离子计数分析参数:

```
ion_count_analysis:
  solute_selection: "resname HA or resname OP*" # 溶质选择
  ion_selection: "name NA"            # 离子类型选择
```

```
  cutoff: 0.6                          # 距离截止值(nm)
  step: 1                              # 帧间隔
```

结构相似性分析参数:

```
network_analysis:
  calc_similarity: true                # 启用Jaccard相似性计算

frame_ranges:
  similarity: [3000, 4000]             # 相似性分析帧范围

# 输出similarity_matrix.npz缓存文件
```

能量分析参数:

```
interaction_energy_analysis:
  energy_groups: ["HA", "OP"]          # 能量组定义
  decompose_types: ["elec", "vdw"]     # 分解能量类型

energy_decomposition_analysis:
  detailed_analysis: true              # 启用详细分解
  pair_analysis: true                  # 成对相互作用分析
```

4.6 输出结果与数据管理

单轨迹分析工具产生多种类型的输出结果，包括可视化图形、数值数据和缓存文件。

典型输出目录结构

```
45nm-HA-30-e78.4-1/                    # 单个轨迹目录
├── md_pbcmol.xtc                      # 输入：轨迹文件
├── solion.tpr                         # 输入：拓扑文件
├── figures/                           # 图形输出
│   ├── rg_evolution_45nm-HA-30-e78.4-1.png
│   ├── ree_distribution_45nm-HA-30-e78.4-1.png
│   ├── msd_analysis_45nm-HA-30-e78.4-1.png
│   ├── sasa_evolution_45nm-HA-30-e78.4-1.png
│   ├── ap_aggregation_45nm-HA-30-e78.4-1.png
│   ├── ion_count_evolution_45nm-HA-30-e78.4-1.png
│   ├── cluster_size_evolution_45nm-HA-30-e78.4-1.png
│   ├── structure_factor_45nm-HA-30-e78.4-1_linear.png
```

```
│   ├── structure_factor_45nm-HA-30-e78.4-1_loglog.png
│   ├── structure_factor_45nm-HA-30-e78.4-1_windowed.png
│   ├── rdf_HA_45nm-HA-30-e78.4-1.png
│   ├── contact_network_45nm-HA-30-e78.4-1.png
│   ├── network_topology_45nm-HA-30-e78.4-1.png
│   ├── adjacency_matrix_45nm-HA-30-e78.4-1.png
│   ├── similarity_heatmap_45nm-HA-30-e78.4-1.png
│   └── interaction_energy_45nm-HA-30-e78.4-1.png
└── cache/                              # 缓存数据（参数化文件名）
    ├── rg.npz                          # 回转半径结果
    ├── ree.npz                         # 端到端距离结果
    ├── contacts_cutoff0.5.npz          # 分子接触（cutoff=0.5nm）
    ├── clusters_min_distance1.5.npz # 聚类分析（min_distance=1.5
   nm）
    ├── rdf_rmax1.5.npz                # 径向分布函数（rmax=1.5nm）
    ├── structure_factor_qmax3.0_nbins100.npz # 结构因子
    ├── network_min_distance0.6.npz # 网络分析
    ├── adjacency_matrix_cutoff0.5.npz # 邻接矩阵
    ├── ion_count_cutoff0.6.npz     # 离子计数
    ├── com_distances.npz           # 质心距离
    └── asphericity.npz             # 非球形度
```

图形输出说明

基础动力学图形：包括 RG 演化图（典型 HA 分子 1.5-2.0 nm）、端到端距离分布和 MSD 扩散分析图。

表面分析图形：SASA 演化图显示分子表面暴露变化，聚集倾向性图展示自组装驱动力，离子计数图反映电荷分布效应。

相互作用分析图形：RDF 曲线显示空间关联性，接触网络图可视化分子间连接关系，网络拓扑图展示分子组织模式。

聚集行为图形：聚类大小演化图帮助理解自组装动力学，结构因子图表征长程有序性和特征波数演化。

网络演化图形：邻接矩阵热图显示稳定相互作用模式，Jaccard 相似性热图展示网络结构随时间的收敛过程。

缓存数据管理

缓存文件采用 NPZ 压缩格式保存，包含完整的分析结果和元数据：

```
# 查看缓存文件信息
python -c "import numpy as np; data=np.load('rg.npz'); print(
   list(data.keys()))"
```

```
# 典型输出: ['data', '__metadata__']
# data: 分析结果数据 (DataFrame转换的字典格式)
# __metadata__: 参数信息和缓存元数据 (JSON格式)
```

4.7 分析工作流程与最佳实践

标准分析流程

1. 环境准备与数据检查:

```
# 确认工作目录
cd MD_assembly/coarse-grained-newOP-final

# 检查数据完整性
ls -la 45nm-*/

# 验证配置文件
python ../../analyze_single.py --config config_user.yaml --help
```

2. 快速结构分析:

```
# 先进行基本结构分析
python ../../analyze_single.py --config config_user.yaml \
    --group "HA only" --analysis-types rg,ree

# 检查结果
ls -la */figures/rg_evolution*
```

3. 全面分析执行:

```
# 使用完整配置进行综合分析
python ../../analyze_single.py \
    --config example_configs/config_complete_analysis.yaml \
    --group "HA only"
```

性能优化建议

- **合理设置并行数**：根据 CPU 核心数和内存大小调整`n_jobs`参数
- **利用缓存机制**：重复分析时系统自动复用已计算结果
- **分阶段分析**：先执行快速分析类型，再进行计算密集型分析
- **内存管理**：大型体系建议设置`last_frames`限制分析帧数

常见问题与解决方案

内存不足：

```
# 在配置文件中限制分析帧数
analysis:
  parameters:
    last_frames: 500       # 只分析最后500帧
    n_jobs: 4              # 减少并行进程数
```

文件路径问题：确保在正确的工作目录下执行，配置文件中的base_path设置为"."

分析类型选择建议：

- **快速筛选**：先运行 rg、ree、contacts 等基础分析
- **深入分析**：根据初步结果选择 clusters、network、structure_factor
- **专项研究**：离子效应研究使用 ion_count，网络演化研究使用 similarity
- **能量机制**：interaction_energy 和 energy_decomposition 用于机理分析

通过分类组织的分析功能和实用的配置指导，单轨迹分析工具能够为 HA/OP 自组装体系研究提供全面、高效的定量分析解决方案。

五、 多组比较分析工具

多组比较分析工具（compare_groups.py）是本软件专门设计的统计比较分析模块，用于系统性比较不同条件下 HA/OP 自组装体系的性质差异。该工具基于 YAML 配置文件管理多个分子动力学轨迹组，支持 6 种核心比较分析类型，并具备自组织计算能力和智能缓存集成，为研究人员提供发表级别的统计比较图表和定量分析结果。

架构特性：工具基于模块化架构实现，所有分析功能基于重构后的analysis.calc和analysis.plot模块实现，集成 TrajectoryManager 和 CacheManager 进行统一资源管理，享受 fast_histogram 性能优化和 joblib 并行处理，统一的单位标准化（nm/ns），同时保持完整的功能特性和用户接口。

5.1 工具概述与设计特点

多组比较分析工具的核心设计理念是通过统计学方法系统性评估不同实验条件对 HA/OP 聚合物自组装行为的影响。该工具具有以下关键特点：

自组织计算能力

工具具备完全自组织的计算流程，无需预先运行`analyze_single.py`。当缺少缓存数据时，系统会自动调用相应的分析函数进行实时计算，确保分析的连续性和完整性。

智能缓存集成

与单轨迹分析工具共享同一套缓存系统，采用 NPZ 格式存储分析结果，通过参数编码的文件命名实现自动缓存管理和失效检测。

统计学导向设计

所有比较分析均基于统计学原理，提供均值、标准差、置信区间等统计指标，并生成标准化的比较图表用于科学发表。

一致性色彩主题

采用统一的颜色方案：Tiffany 蓝（`#99D9D8`）代表 HA 体系，水晶玫瑰色（`#F1CACB`）代表 OP 体系，中性灰（`#B8B8B8`）用于对照条件，确保图表的专业性和一致性。

5.2 命令行接口与基本用法

多组比较分析工具提供简洁而功能强大的命令行接口：

```
# 基本语法
python compare_groups.py --config CONFIG_FILE [OPTIONS]

# 主要参数
--config           YAML配置文件路径 (必需)
--work-dir         工作目录路径 (可选，覆盖配置文件设置)
--groups           指定比较组 (可选，默认比较所有组)
--analysis-types   分析类型列表 (可选，默认: clusters,rdf)
--help             显示完整帮助信息
```

典型使用场景

```
# 比较所有配置组的聚类和RDF特性
python compare_groups.py --config config_user.yaml

# 指定特定组进行全面比较分析
python compare_groups.py --config config_user.yaml \
    --groups "HA only,HA/OP (pH=4.5)" \
    --analysis-types clusters,rdf,contacts,sasa
```

```
# 进行完整的六项比较分析
python compare_groups.py --config config_user.yaml \
    --analysis-types clusters,rdf,contacts,sasa,msd,adjacency
```

5.3 配置系统与数据管理

多组比较分析采用层次化 YAML 配置文件管理实验组和分析参数：

配置文件结构

```
groups:
  "HA only": [45nm-HA-30-e78.4-1, 45nm-HA-30-e78.4-2, 45nm-HA-
   30-e78.4-3]
  "HA/OP (pH=4.5)": [45nm-30-16-e78.4-1, 45nm-30-16-e78.4-3, 45
   nm-30-16-e78.4-4]
  "HA/OP (pH=7)": [45nm-30-16-e78.4-1-ph7, 45nm-30-16-e78.4-2-
   ph7]

analysis:
  types: ["clusters", "rdf", "contacts", "ree", "rg", "ap", "
   sasa", "msd", "network", "adjacency"]
  colors: ["#99D9D8", "#F1CACB", "#B8B8B8"]
  parameters:
    num_residues_per_molecule1: 36    # HA分子残基数
    num_residues_per_molecule2: 30    # OP分子残基数
    min_distance: 1.5                 # 聚类最小距离 (nm)
    cutoff: 0.5                       # 接触距离阈值 (nm)
    last_frames: 1000                 # 分析帧数
    n_jobs: 32                        # 并行进程数

files:
  topology: "solion.tpr"
  trajectory: "md_pbcmol.xtc"

directories:
  base_path: "MD_assembly/coarse-grained-newOP-final"
  figure_output: "figures"
```

组管理策略

每个实验组包含多个重复轨迹，工具自动验证文件完整性并跳过缺失数据。支持选择性组比较，允许研究人员专注于特定条件的对比分析。

5.4 核心比较分析类型

多组比较分析工具提供十种核心分析类型，每种分析都针对特定的分子性质和行为特征进行优化。

5.4.1 比较分析类型速查表

分析术语	比较内容	主要输出
`clusters`	聚类大小分布比较	分数直方图对比，统计检验
`rdf`	径向分布函数对比	RDF 曲线、B2 系数
`contacts`	分子接触数统计	平均值柱状图
`ree`	端到端距离分布比较	小提琴图，单链数据聚合
`rg`	回转半径分布比较	小提琴图，缓存数据复用
`sasa`	溶剂可达表面积	暴露程度统计，时间演化对比
`msd`	均方位移曲线	扩散系数对比
`network`	网络拓扑分析	图论参数，连通性指标
`ap`	聚集倾向性比较	柱状图，统计检验
`adjacency`	邻接矩阵分析	相互作用热图
`interaction_energy`	相互作用能量比较	2×2 柱状图，能量分量对比

表 2　多组比较分析工具支持的分析类型汇总

统计学设计原则

所有比较分析均采用严格的统计学方法，提供以下统计指标：

- 组内统计：计算均值、标准差和置信区间
- 组间比较：统一误差棒和色彩编码方案
- 显著性检验：t 检验或 Mann-Whitney U 检验
- 图表标准：300 DPI PNG 格式，Arial Demibold 字体

5.4.2 聚类分析比较 (clusters)

聚类分析比较通过统计不同条件下分子聚集体的尺寸分布差异，揭示自组装过程的热力学和动力学特征。

分析原理

基于距离阈值的聚类算法识别分子聚集体，统计每个时间点的聚类尺寸分布，并计算跨轨迹的统计特征。算法采用周期性边界条件下的最短距离计算，确保聚类识别的准确性。

关键参数

- `min_distance`：聚类判定的最小距离阈值，典型值为 1.5 nm
- `num_residues_per_molecule`：单分子残基数，用于分子边界识别
- `last_frames`：参与统计的轨迹帧数，-1 表示使用全部帧
- `selection`：分析选择字符串，默认为”resname HA and name A”

输出结果

生成聚类尺寸分布的直方图比较，包含均值、标准差和统计显著性检验结果。图表采用透明度叠加方式展示不同组间的分布差异。

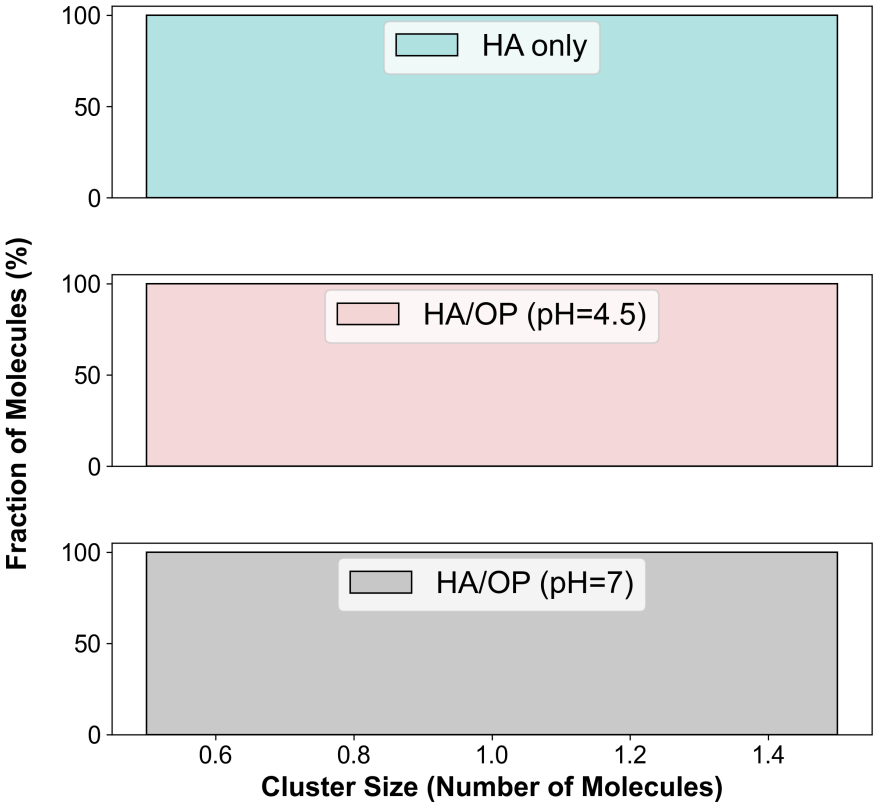

图 9 不同条件下聚类尺寸分布比较示例。直方图展示了 HA 单组分体系与 HA/OP 双组分体系在不同 pH 条件下的聚集行为差异，其中单体、二聚体及更大聚集体的分布呈现明显的系统性变化。

5.4.3 回转半径分布比较 (rg)

回转半径分布比较通过分析分子构象紧密程度的统计差异，评估不同条件下分子折叠行为和空间尺寸的变化特征。

分析原理

利用 GROMACS 预计算的回转半径数据，通过缓存系统实现高效数据复用。小提琴图展示 Rg 分布的完整形状和统计特征，揭示构象变化的热力学规律。

关键参数

- `selection`：分析目标分子选择，影响缓存文件命名
- `start_time`：分析起始时间，用于平衡态数据提取
- `group_selection`：GROMACS 组选择，默认为”Solute”

输出结果

生成回转半径分布的小提琴图比较，展示各组的构象紧密性差异。支持 XVG 文件读取和智能缓存管理，提供高效的重复分析能力。

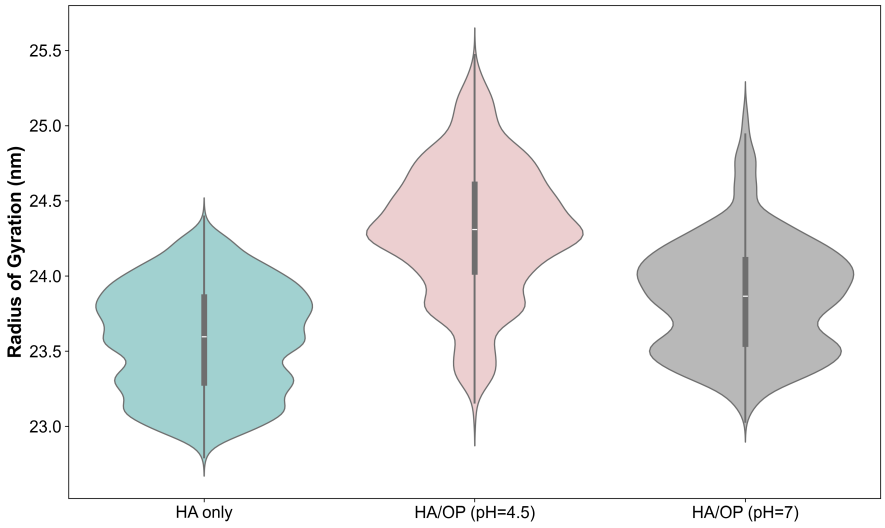

图 10 不同条件下回转半径分布比较。小提琴图展示了各组分子构象紧密性的分布密度，中位线表示四分位数，分布宽度反映构象多样性，组间差异揭示环境条件对分子折叠的影响。

5.4.4 端到端距离分布比较 (ree)

端到端距离分布比较通过分析不同条件下聚合物链构象分布的差异，揭示分子间相互作用对链构象的调节机制。

分析原理

聚合所有轨迹中的单链端到端距离数据，生成总体分布进行组间比较。通过小提琴图展示分布形状、密度和统计差异，评估不同条件对聚合物链伸展性的影响。

关键参数

- `selection`：聚合物链选择，默认为"resname HA"
- `num_residues_per_molecule`：单分子残基数，用于确定链边界
- `start_time`：分析起始时间，跳过平衡阶段

输出结果

生成端到端距离分布的小提琴图比较，展示各组的分布密度、四分位数和异常值。支持缓存数据复用，提供单链级别的统计分析。

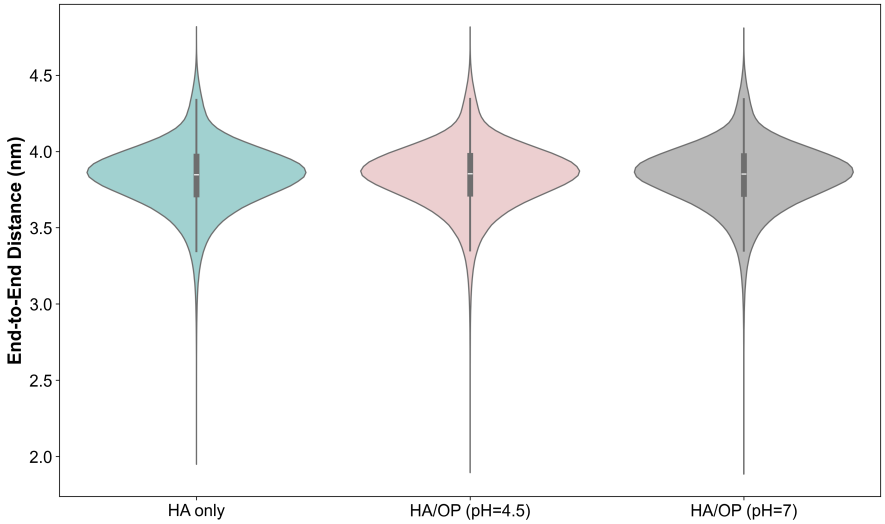

图 11 不同条件下端到端距离分布比较。小提琴图展示了聚合物链构象分布的统计差异，分布形状反映链的伸展程度，组间对比揭示分子间相互作用对聚合物链构象的调节效应。

5.4.5 径向分布函数比较 (rdf)

径向分布函数（RDF）比较分析通过量化分子间距离分布的差异，揭示不同条件下分子间相互作用强度和空间组织模式的变化。

分析原理

计算指定选择原子周围的径向分布概率密度，采用" 排除当前分子" 算法避免分子内相关性的干扰。通过多轨迹平均获得统计可靠的 RDF 曲线，并进行组间比较分析。

关键参数

- `selection`：目标分子选择，默认为”HA”
- `atom_name`：计算中心原子类型，默认为”A”
- `num_residues_per_molecule`：分子大小定义，用于排除分子内距离
- `xmax`：RDF 计算和显示的最大距离，典型值为 2.0 nm

输出结果

生成多组 RDF 曲线的叠加对比图，包含峰位分析、配位数计算和积分曲线。同时自动计算第二维利系数（B2），提供分子间相互作用强度的热力学定量指标。B2 值为负表示分子间吸引作用，为正表示排斥作用，提供了超越几何结构分析的热力学解释。

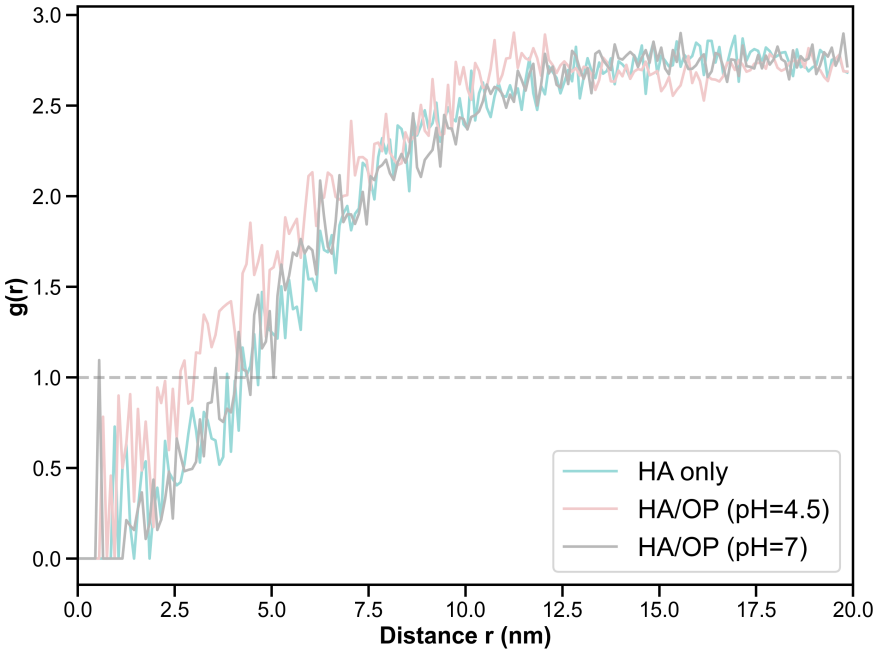

图 12　不同体系径向分布函数比较。曲线显示了 HA 单组分与 HA/OP 双组分体系的分子间距离分布差异，第一配位壳层和第二配位壳层的峰位和强度变化反映了组分混合对分子间相互作用的影响。

5.4.6 分子接触分析比较 (contacts)

分子接触分析比较通过统计不同条件下分子间接触数量的差异，定量评估组分混合和环境条件对分子间相互作用网络的影响。

分析原理

基于距离阈值统计分子间接触数量，区分分子内接触和分子间接触。对于粗粒化体系，主要分析 HA-HA 接触；对于全原子体系，分析 HA-OP 接触。采用并行算法处理大规模轨迹数据。

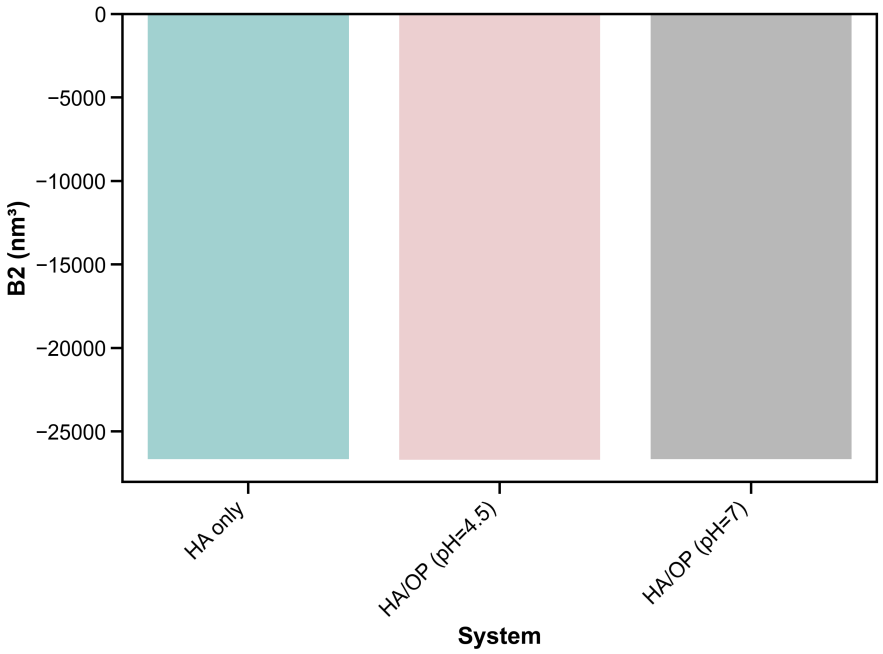

图 13 不同体系第二维里系数 B2 比较。柱状图定量展示各组分子间相互作用强度的热力学指标，B2 为负表示分子间吸引作用，为正表示排斥作用，提供超越几何结构的相互作用定量评估。

关键参数

- `cutoff`：接触距离阈值，典型值为 0.5 nm
- `selection1`和`selection2`：接触分析的两个选择组
- `num_residue1`：第一种分子的残基数
- `n_jobs`：并行计算进程数

输出结果

生成接触数量的柱状图比较，包含误差棒和统计显著性标注。提供平均接触数、标准差和组间差异的定量评估。

5.4.7 溶剂可及表面积比较 (sasa)

溶剂可及表面积（SASA）比较分析通过量化分子表面暴露程度的变化，评估不同条件下分子聚集和折叠行为的差异。

分析原理

计算分子构象的溶剂可及表面积，通过 SASA 比值反映分子聚集程度。SASA 减少通常对应更紧密的分子堆积和更强的分子间相互作用。

关键参数

- `probe_radius`：溶剂探针半径，典型值为 0.14 nm（水分子半径）
- `selection`：计算 SASA 的分子选择

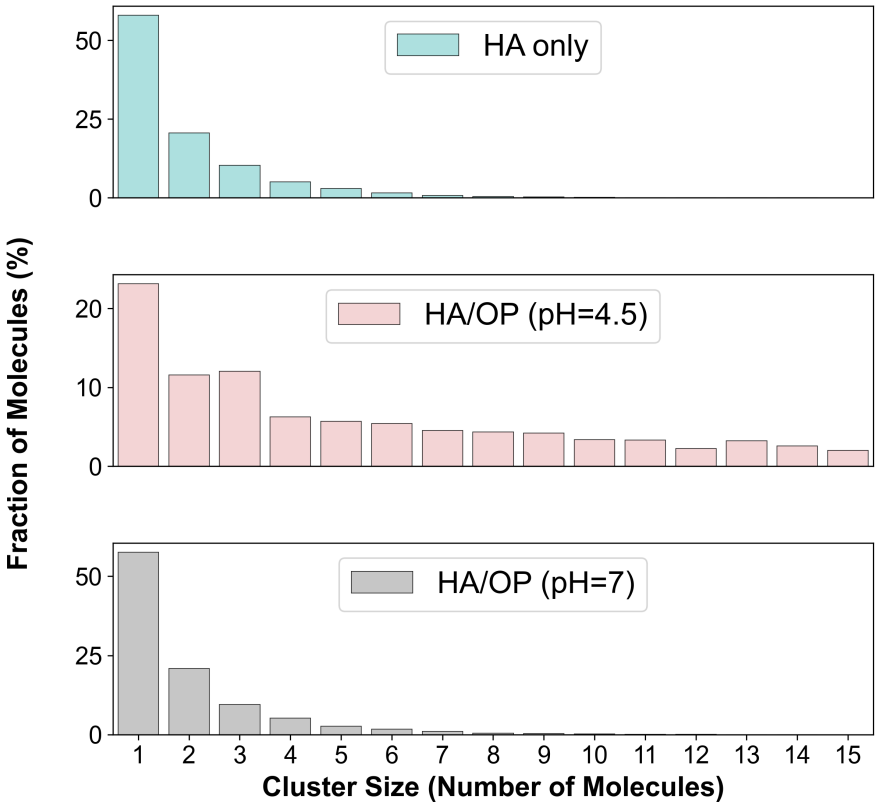

图 14 不同条件下分子接触数量比较。柱状图展示了各实验组的平均接触数及其标准差，误差棒反映了数据的统计不确定性，组间差异的显著性通过统计检验确定。

- `frame_interval`：计算的帧间隔，用于控制计算成本

输出结果

生成 SASA 演化曲线的组间比较图，包含时间演化趋势和统计分布分析。提供聚集程度的定量指标和组间差异评估。

5.4.8 均方位移比较 (msd)

均方位移（MSD）比较分析通过量化分子扩散行为的差异，评估不同条件下分子动力学性质和传输特性的变化。

分析原理

计算分子质心的均方位移随时间的演化，通过爱因斯坦关系提取扩散系数。比较不同条件下的扩散行为差异，揭示环境条件对分子流动性的影响。

关键参数

- `selection`：计算 MSD 的分子选择
- `msd_type`：MSD 计算类型（质心或单原子）
- `fit_range`：线性拟合的时间范围，用于扩散系数提取

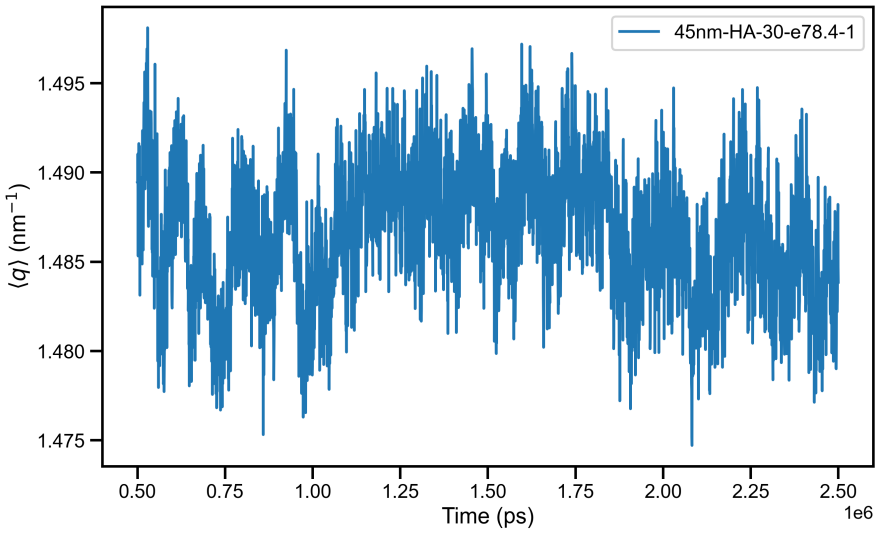

图 15 不同条件下溶剂可及表面积演化比较。时间序列曲线展示了各组分子表面暴露程度的演化趋势，SASA 减少反映了分子聚集的程度，组间差异揭示了不同条件对自组装过程的影响。

输出结果

生成 MSD 演化曲线的组间比较图和扩散系数的柱状图比较。提供分子流动性的定量评估和统计显著性分析。

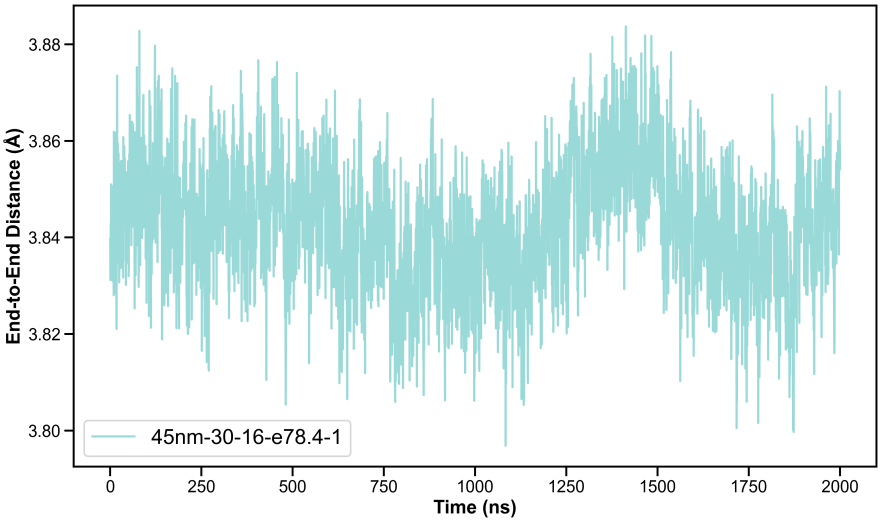

图 16 不同条件下均方位移演化比较。左图展示 MSD 随时间的演化曲线，斜率反映扩散系数；右图为提取的扩散系数柱状图比较，误差棒表示统计不确定性。

5.4.9 邻接矩阵分析 (adjacency)

邻接矩阵分析通过构建和比较分子间相互作用网络的拓扑结构，揭示不同条件下分子组织模式和网络连通性的差异。

分析原理

基于距离阈值构建分子间相互作用的邻接矩阵，通过矩阵特征值、聚类系数、路径长度等网络拓扑参数量化分子网络的结构特征。

关键参数

- `last_frame`：用于分析的轨迹帧范围
- `num_residues_per_molecule1`：第一类分子的残基数
- `num_residues_per_molecule2`：第二类分子的残基数
- `num_chains_tuple`：体系中各类分子的数量

输出结果

生成邻接矩阵的热图可视化和网络拓扑参数的统计比较。提供分子网络连通性和组织模式的定量分析。

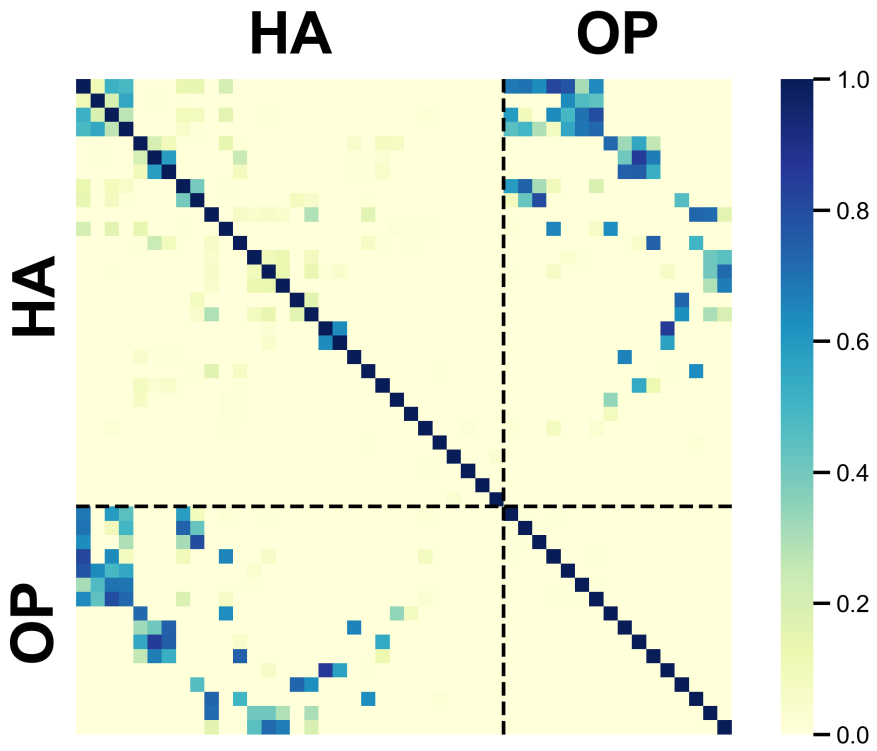

图 17 不同条件下分子间相互作用网络邻接矩阵比较。热图展示了分子间相互作用的强度分布，颜色深浅反映相互作用频率，矩阵的聚类模式揭示了分子的组织结构差异。

5.4.10 相互作用能量比较 (interaction_energy)

相互作用能量比较分析通过对比不同条件下分子间相互作用能的各个组分，揭示驱动自组装过程的热力学机制差异和能量贡献的相对重要性。

分析原理

基于单轨迹分析生成的相互作用能量数据 ($E_{interaction} = E_{AB} - E_A - E_B$)，计算各组分 (范德华力、静电力、色散力) 在不同实验条件下的平均值和分布特征，通过统计比较量化能量差异的显著性。

数据处理流程

- **缓存利用**：读取已有 XVG 文件中的能量时间序列数据
- **统计计算**：计算各能量分量的时间平均值和标准差
- **组间比较**：应用统计检验评估不同条件间的能量差异显著性
- **可视化**：生成 2×2 布局的分组柱状图比较各能量分量

关键特点

- **依赖单轨迹分析**：需先完成 interaction_energy 单轨迹分析以生成 XVG 数据
- **无额外计算**：直接复用现有能量分解结果，避免重复计算
- **统计严谨性**：提供误差棒、置信区间和显著性检验结果
- **能量分量全覆盖**：VdW 短程、色散修正、库仑短程、库仑长程及总能量

输出结果

生成包含四个子图的 2×2 网格布局柱状图：

1. **范德华短程作用**：分子间排斥和吸引力平衡
2. **色散力修正**：长程色散相互作用贡献
3. **库仑短程作用**：短程静电相互作用强度
4. **总相互作用能**：所有分量的综合效应

每个子图显示不同实验组的能量均值、标准误差和统计显著性标记，便于识别驱动分子自组装的主导能量因素。

应用指导

- 适用于 pH 效应、离子强度和分子比例等条件变化的机理研究
- 结合聚类分析和 RDF 分析，可建立结构-能量关系
- 能量数据可用于验证分子动力学模拟的合理性

5.5 数据输出与结果解释

多组比较分析工具生成标准化的统计比较结果，包含定量指标、可视化图表和统计显著性评估。

图表输出格式

- PNG 格式（300 DPI）：用于演示和初步分析

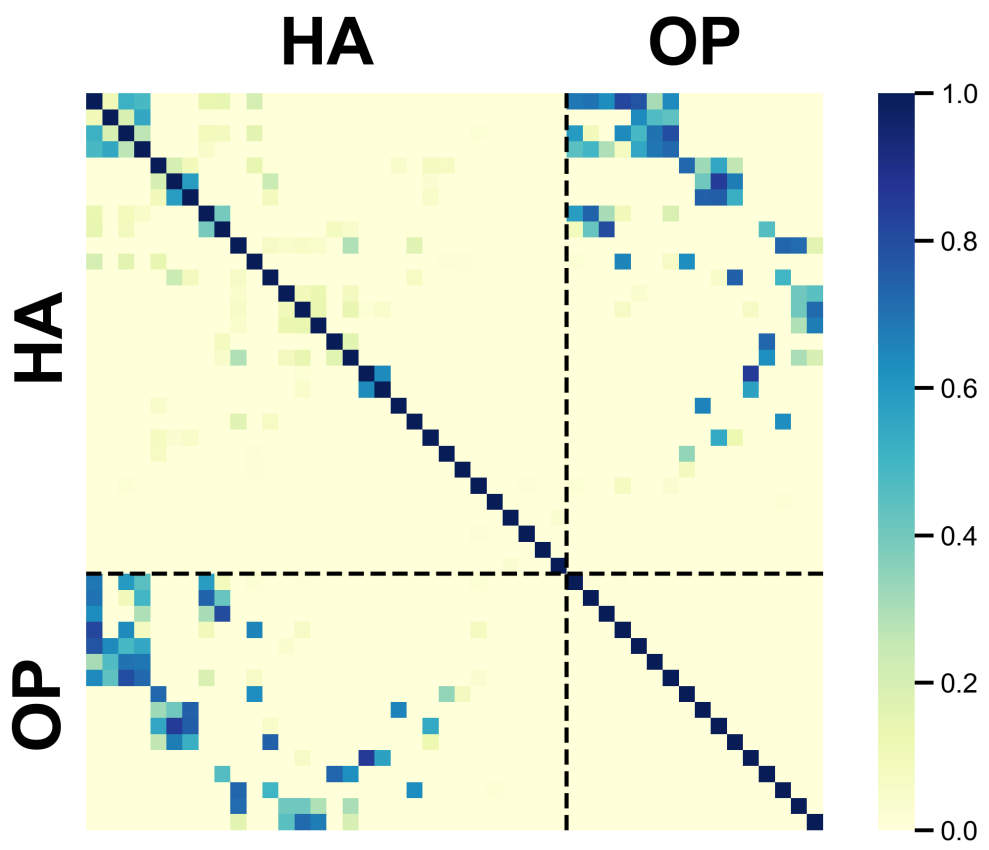

图 18　不同条件下相互作用能量分量比较。2×2 柱状图展示了范德华力、色散力、静电力等各能量分量在不同实验条件下的差异，揭示了驱动自组装的主导热力学因素。

- SVG 格式：用于进一步编辑和排版
- PDF 格式：用于科学发表和高质量打印

统计指标

每种分析提供以下统计信息：

- 均值和标准差：描述数据的中心趋势和离散程度
- 置信区间：提供均值估计的不确定性范围
- 效应量：量化组间差异的实际意义
- 统计显著性：通过 t 检验或 Mann-Whitney U 检验评估差异显著性

缓存管理

所有计算结果自动缓存，采用参数编码的文件命名确保缓存一致性。缓存文件存储在各轨迹目录的`cache/`子目录中，支持增量更新和自动失效检测。

5.6 性能优化与最佳实践

多组比较分析针对大规模轨迹数据进行了性能优化，提供了一系列最佳实践建议。

并行计算优化

- joblib 并行处理：基于 joblib 的高效并行计算框架，替代 concurrent.futures
- fast_histogram 加速：使用 fast_histogram 库实现 10 倍直方图计算性能提升
- 自适应并行度：根据系统资源自动调整并行进程数
- TrajectoryManager：集中化轨迹管理避免重复加载
- 内存管理：采用生成器和块处理避免内存溢出

计算资源建议

- CPU：推荐使用 32 核心以上的多核处理器
- 内存：建议 64GB 以上，特别是处理大型体系时
- 存储：使用 SSD 存储轨迹文件和缓存数据以提高 I/O 性能

分析流程建议

- 逐步分析：先运行快速分析（如 contacts、clusters）验证配置
- 批量处理：对多个实验条件采用批量脚本进行自动化分析
- 结果验证：定期检查缓存一致性和统计显著性

六、 建模脚本

软件包含辅助建模脚本集合，位于`scripts/`目录，用于支持 HA/OP 聚合物自组装体系的模拟前处理和建模工作。

6.1 膜体系建模工具

6.1.1 脂质双分子层构建

`scripts/membrane/build_square_bilayer.sh`：基于 MARTINI 力场构建规则正方形脂质双分子层的自动化脚本，支持多种脂质类型组合和体系尺寸定制。

使用方法：

```
cd scripts/membrane/
./build_square_bilayer.sh -s 45 -l lipid_composition.dat
    -o membrane
```

输入参数：

- `-s`：盒子边长（nm），如 45nm
- `-l`：脂质组成配置文件，定义各脂质分子摩尔比例
- `-o`：输出文件前缀，生成`membrane.gro`和`membrane.top`

输入文件要求：

- `lipid_composition.dat`：包含脂质类型和比例的配置文件
- MARTINI 力场参数文件（`.itp`格式）

6.1.2 膜-聚合物复合体系

`scripts/membrane_complex/insert.sh`：将 HA/OP 聚合物分子插入预构建膜体系的定位脚本，实现膜-聚合物界面体系的自动化组装。

使用方法：

```
cd scripts/membrane_complex/
./insert.sh -m membrane.gro -p polymer.gro -o complex.
    gro -d 2.0
```

必需参数：

- `-m`：膜体系坐标文件（`.gro`格式）
- `-p`：聚合物坐标文件
- `-o`：输出复合体坐标文件
- `-d`：最小插入距离（nm），避免原子重叠

6.2 体系设置工具

6.2.1 单体系设置

`scripts/system_setup/setup_single.sh`：单个 HA/OP 聚合物自组装体系的完整建模流程，包含溶剂化、离子化和能量最小化步骤。

使用方法：

```
cd scripts/system_setup/
./setup_single.sh -c config.yaml -n 30 -m 30
```

输入参数：

- `-c`：体系配置文件，定义盒子尺寸、温度、离子浓度等
- `-n`：HA 聚合物链数量，默认 30
- `-m`：OP 聚合物链数量，默认 30

配置文件要求：

- `config.yaml`：包含盒子尺寸（如 45nm）、温度（310K）、离子类型和浓度
- 聚合物结构模板文件（`HA.pdb`、`OP.pdb`）
- 力场参数文件（`.itp`格式）

6.2.2 批量体系设置

`scripts/system_setup/setup.sh`：多个参数组合体系的批量建模工具，支持不同聚合物浓度、离子强度和 pH 条件的并行处理。

使用方法：

```
cd scripts/system_setup/
./setup.sh -p parameter_matrix.txt -j 8
```

输入要求：

- `-p`：参数矩阵文件，每行包含一组建模参数组合
- `-j`：并行任务数，根据计算资源调整

参数矩阵格式：

```
# box_size n_HA n_OP salt_conc output_dir
45 30 16 0.15 45nm-30-16-salt15
45 30 20 0.15 45nm-30-20-salt15
```

这些建模工具为 MD 模拟提供标准化的体系准备流程，确保输入体系的质量和一致性，支持从单分子结构到复杂自组装体系的完整建模工作流程。