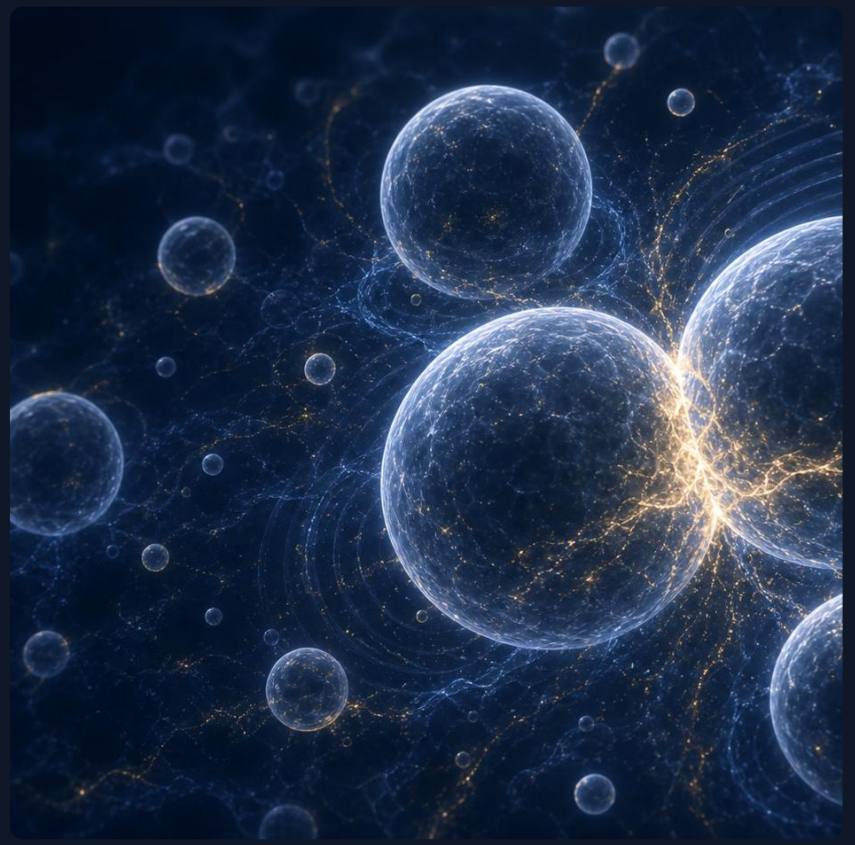
CosmoTransitions

From effective potential to phase transition



7.12.2026

纲要：从物理问题到可观测量

1. 物理背景

一阶相变为什么需要数值工具

2. 数学问题

势能、phase、bounce、S_3/T

3. 软件结构

cosmoTransitions 模块与工作流

4. 模型实现

generic_potential 与质量谱

5. Phase tracing

getPhases 与 forbidPhaseCrit

6. Action/Profile

transition、action、bubble profile

7. 完整例子

toy model / xSM 串流程

8. 注意事项 / 9. PTagent

细节/agent

01

物理背景：为什么需要 CosmoTransitions？

本章回答

从BSM模型到早期宇宙相变，为什么需要数值计算？

一阶相变的物理图像

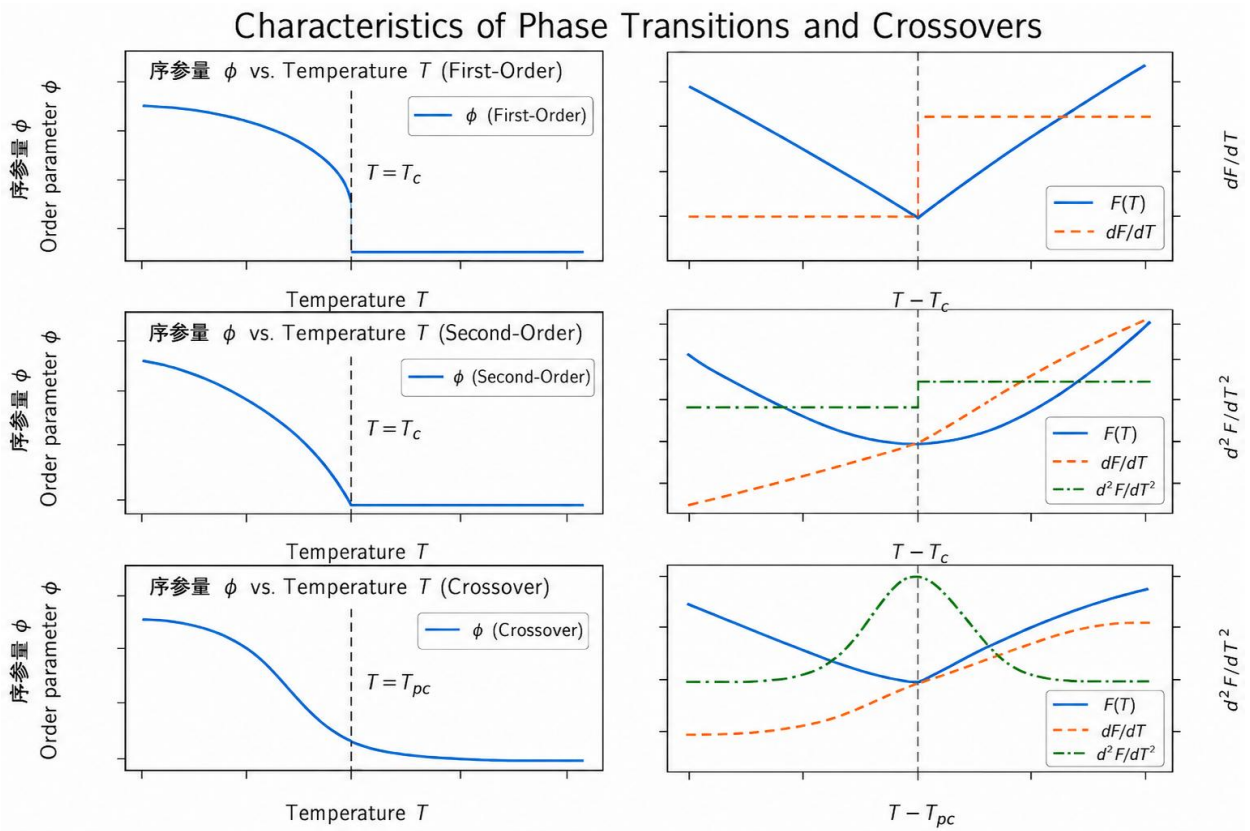
从均匀背景到气泡成核

相变的区别：自由能的导数

- 自由能一阶导数发散：一阶相变；
- 自由能二阶导数发散：二阶相变；
- **自由能所有导数不发散：Crossover**

一阶相变的特点

- 势垒存在
- 局域成核（随机）
- 潜热释放

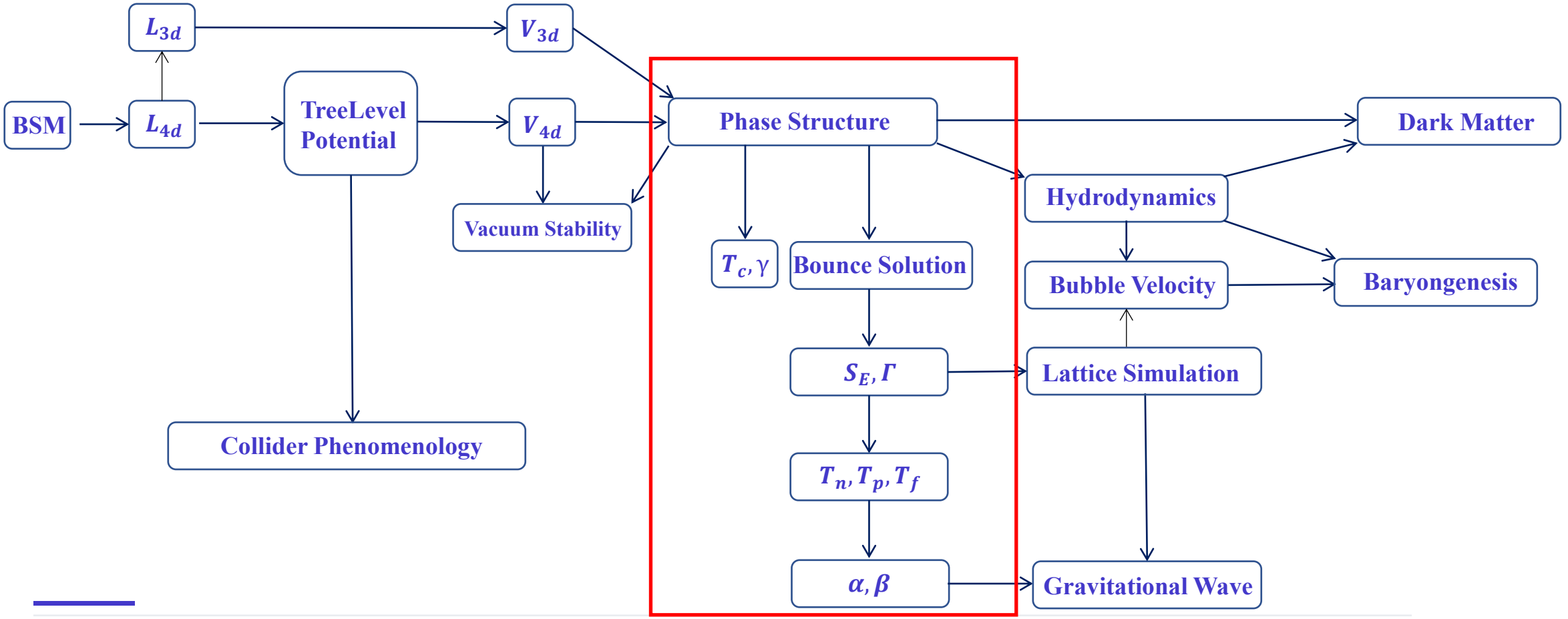


《朗道-栗弗席兹理论物理教程》第 10 卷 物理动理学

从 BSM 模型到有限温势能

研究相变需要输入什么

相变流程计算图



相变特征量

描绘相变发生过程的量

相变特征量

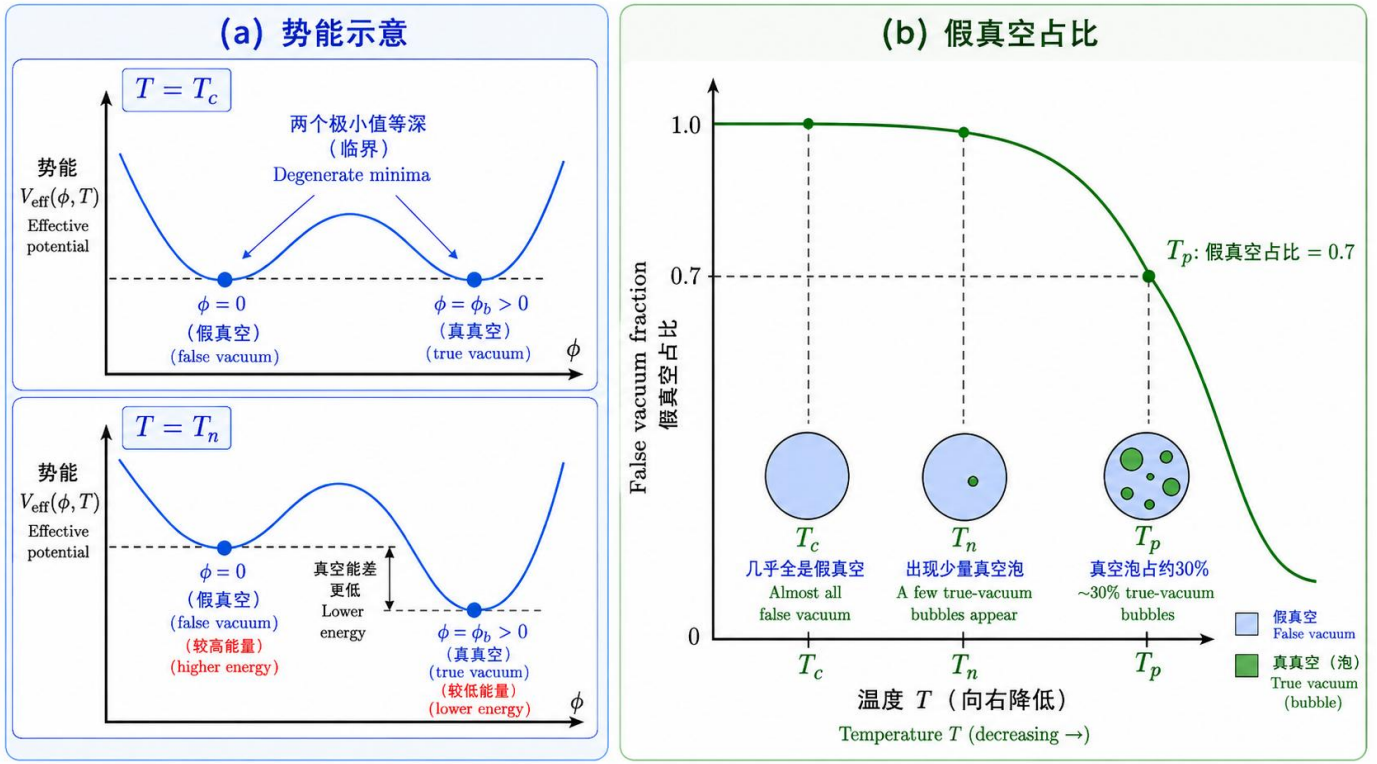
成核率： $\Gamma \sim A\, e^{-S_3/T}$

临界温度： $V(\phi_f, T_c) = V(\phi_t, T_c)$

成核温度： $N(T_n) = \int_{T_n}^{T_c} \frac{\Gamma}{H^4} V dT \sim \mathcal{O}(1)$

假真空占比： $h(t) = \exp\left[\int_{t_c}^{t} -\Gamma(t')V(t',t)\, dt'\right]$

渗透温度： $h(T_p) = 0.7$



临界温度 vs 成核温度

Nucleation is more than critical

临界不代表相变可以发生

Tc 只是势能简并；真正发生相变还需要成核率足够大。

成核温度的计算关键在于作用量的计算

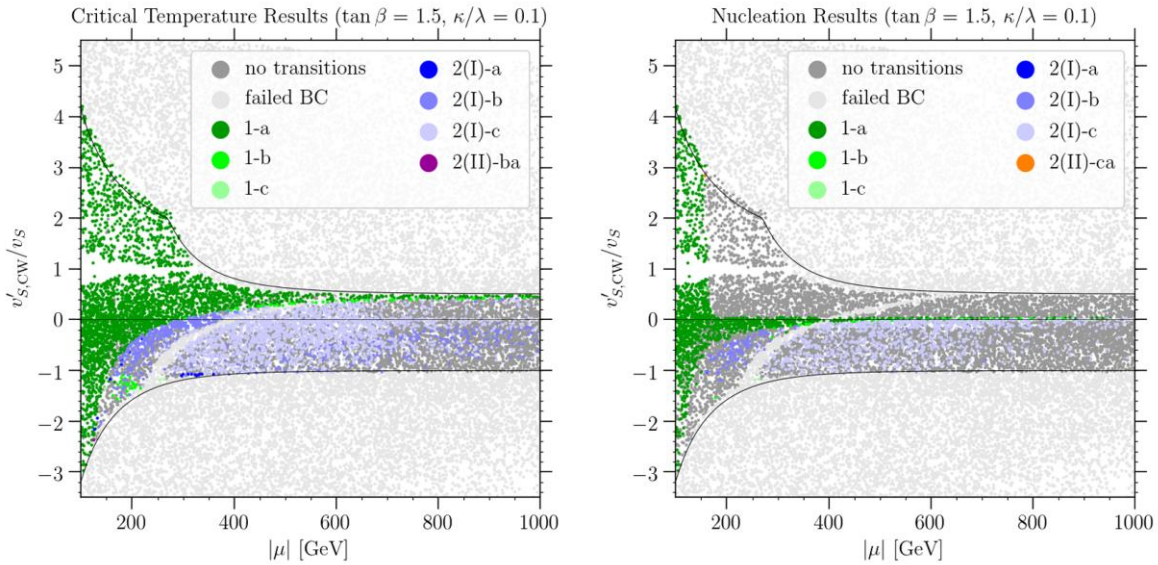
$$S = \int 4\pi r^2 dr \left[\frac{1}{2}\left(\frac{\partial\phi}{\partial r}\right)^2 + V(\phi,T)\right]$$

$$\frac{d^2\phi}{dr^2} + \frac{2}{r}\frac{d\phi}{dr} = \frac{\partial V(\phi,T)}{\partial\phi}$$

边界条件

$$\phi(r \to \infty) = \phi_f$$

$$r=0,\ \frac{d\phi}{dr}=0 \quad \text{光滑条件}$$



2009.10743

相变特征量后续处理

以GW计算为例

引力波计算方式

格点模拟

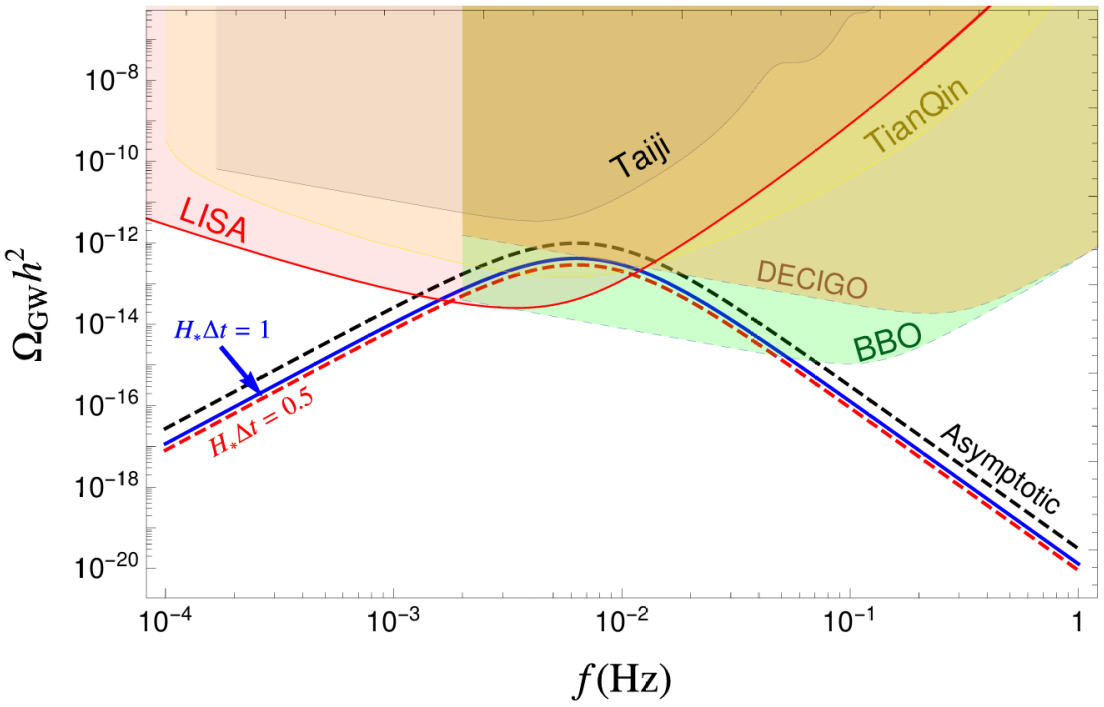
拟合公式

$$\Omega_{\rm sw}h^2 \;=\; 2.65\times10^{-6}\left(\frac{H}{\beta}\right)\left(\frac{\kappa\alpha}{1+\alpha}\right)^2\left(\frac{100}{g}\right)^{1/3}\left(\frac{f}{f_{\rm sw}}\right)^3\left(\frac{7}{4\;+\;3(f/f_{\rm sw})}\right)^{7/2}\Upsilon(y)v_w$$

特征量相关

$$\frac{\beta}{H}=T_n\;\left.\frac{dS}{dT}\right|_{T=T_n}$$

$$\alpha\;=\;\frac{\Delta\rho(T_n)}{\rho_{rad}(T_n)}=\frac{\Delta V(T_n)-T_n\Delta V'(T_n)}{\rho_{\rm rad}(T_n)}$$



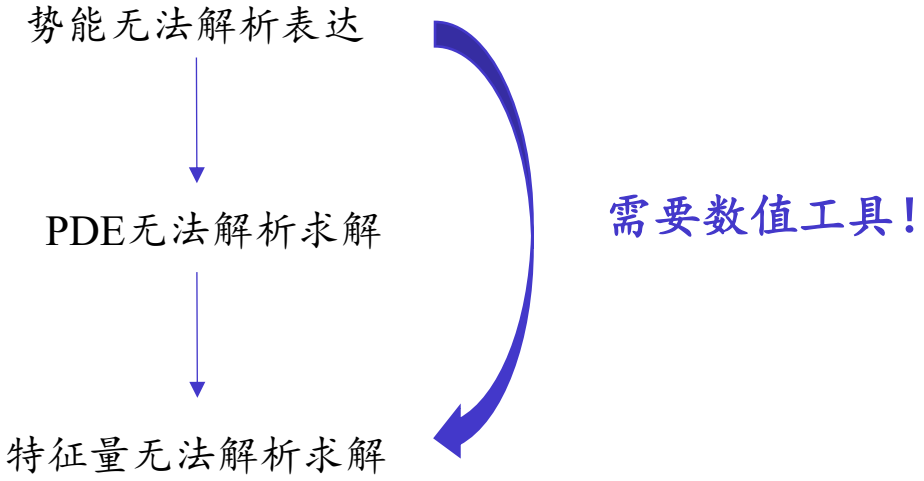
2007.08537

为什么需要数值工具？

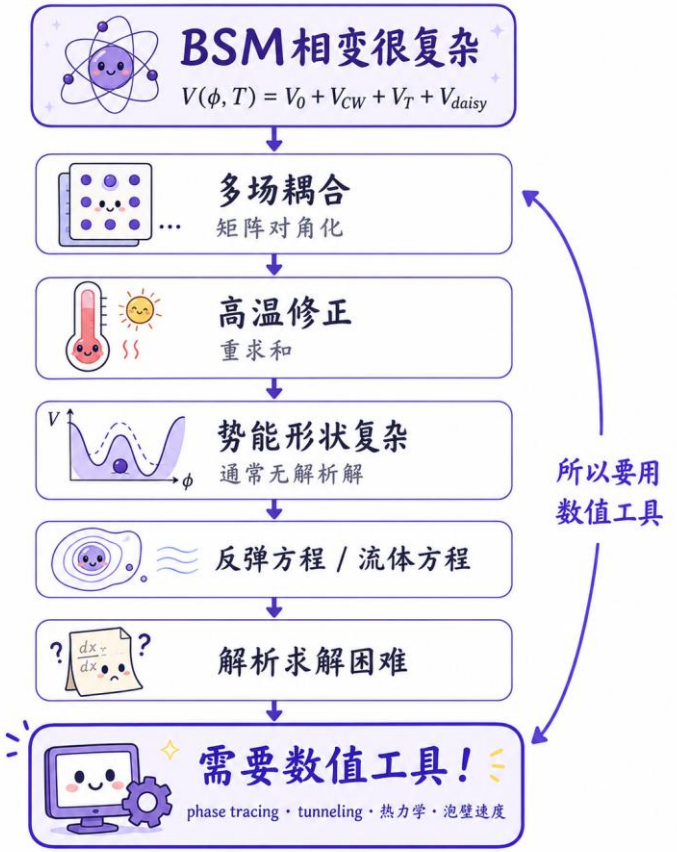
BSM相变的复杂性

$$V(\phi) = V_0(\phi) + V_1(\phi) + V_{1T}(\phi) + V_{daisy}(\phi)$$

- 树图势能可以解析表达
- Coleman-Weinberg 势能涉及矩阵对角化
- 一圈高温修正无法解析表达



需要数值工具！



CosmoTransitions 在 pipeline 中的位置

它解决哪一段?

Cosmotransition 核心作用

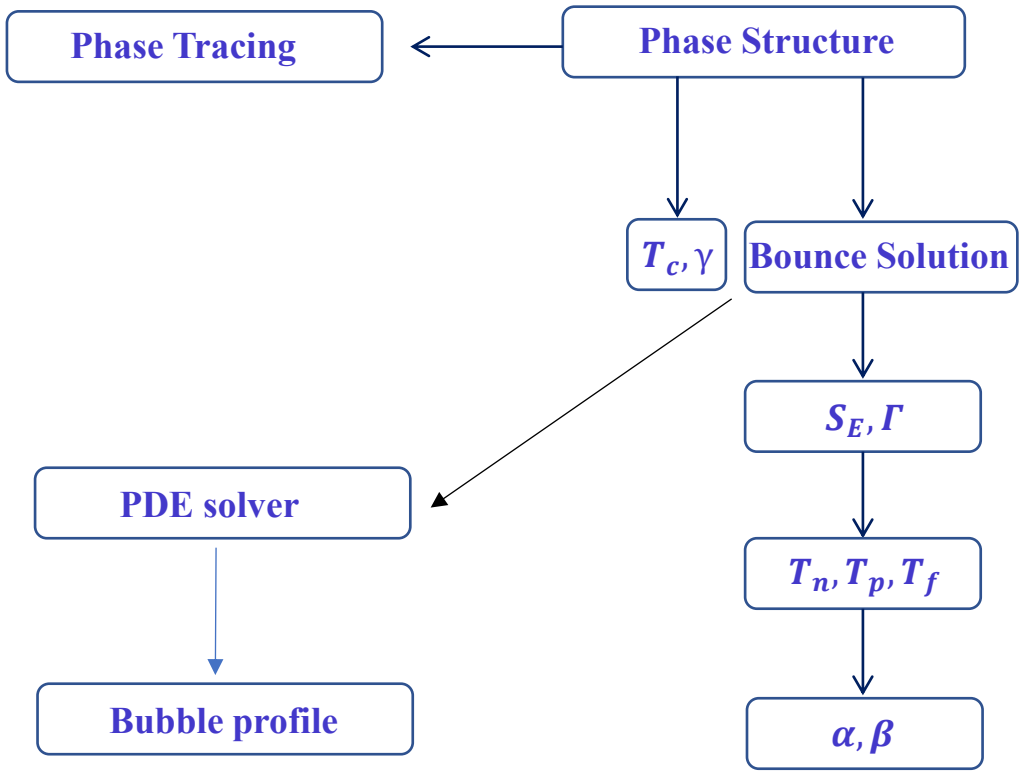
phase tracing 与 thermal tunneling

关键部分

- 输入： V_eff + mass spectrum
- 核心： getPhases / findAllTransitions
- 输出： transitions 与 instanton profile

End to End software

PhaseTracer, BSMPT, Transitionsolver



本章小结

小结

- 关心的是有限温势能的相结构
- 真正发生相变需要 bubble nucleation
- CT 是相结构与隧穿计算工具

数学基础 → 程序调用

掌握计算软件

02

数学问题：V_eff、phase 与 bounce

从势能到成核条件的公式框架

本章回答

CosmoTransitions 具体是在解什么数学问题？

- 有限温有效势
- 临界温度与成核温度
- bounce equation 与多场路径

有限温有效势的组成

V_eff(ϕ,T) 是核心输入

有限温度有效势能的组成

$$V(\phi) = V_0(\phi) + V_1(\phi) + V_{1T}(\phi) + V_{daisy}(\phi)$$

细则：

$$V_1 = V_{\rm cw} + V_{\rm ct}$$

$$V_{\rm cw} = \sum_i \frac{n_i}{64\pi^2} m_i^4(\phi)\left[\ln\frac{{m_i}^2(\phi)}{\mu^2} - c_i\right]$$

抵消项依赖于重整化条件

依据模型的不同，V_ct有可能并入V_cw中，从而实现表达式解析。

场依赖的质量矩阵特征值

繁重的检查/手动计算

$$V_{1\rm T} = \frac{T^4}{2\pi^2}\sum_i n_i J_{B/F}\left(\frac{{m_i}^2(\phi)}{T^2}\right)$$

$$J_{B/F}(y^2) = \int_0^\infty dx\, x^2 \ln\left[1 \mp e^{-\sqrt{x^2+y^2}}\right]$$

数值积分依赖

V_{daisy} → 额外的热一圈积分计算

手算/查阅论文/DRalgo

$$V_{\rm ct} = \frac{1}{2}\delta m\, \phi^2 + \ldots$$

重整化条件确定

极小值与 phase branch

phase 是随温度变化的 minimum 轨迹

真空可能候选

$$\frac{\partial V(\phi,T)}{\partial \phi}=0$$

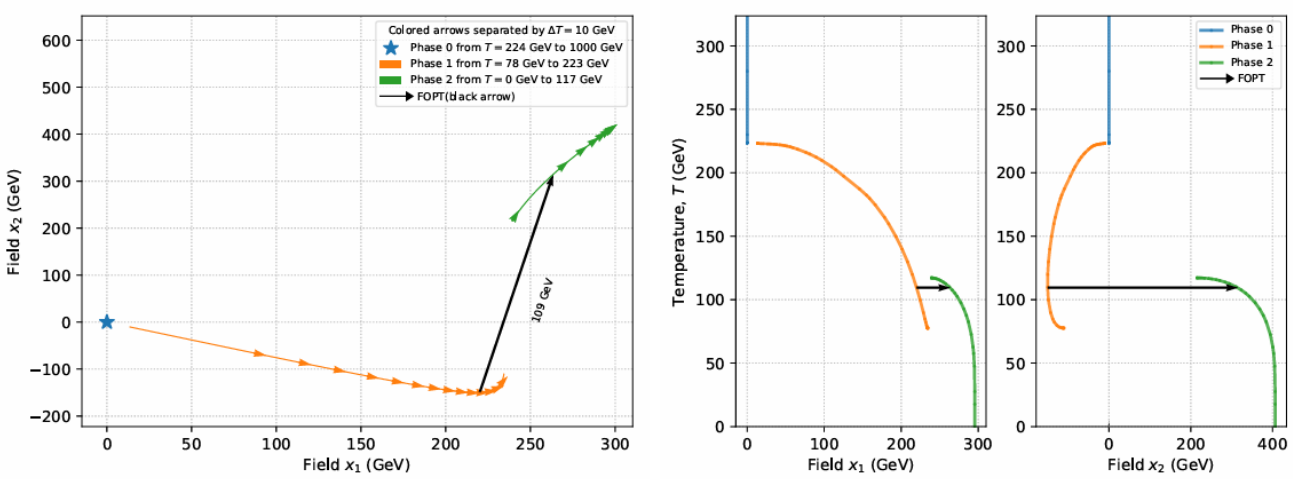
温度变化，导致V的变化

对于每一个温度，都有其对应的一个极小值/phase $\varphi(T)$

不是求全局最小值！ 是所有极小值都要找到！

算法思路

- 每个 T 下求局部极小值
- 相同极小值随 T 连续追踪
- 多场中 phase 是场空间轨迹



2003.02859

一维 toy potential

教学范例

Toy model

$$V(\phi,T) = D(T^2-T_0^2)\phi^2 - ET\phi^3 + \frac{\lambda}{4}\phi^4$$

Phase结构

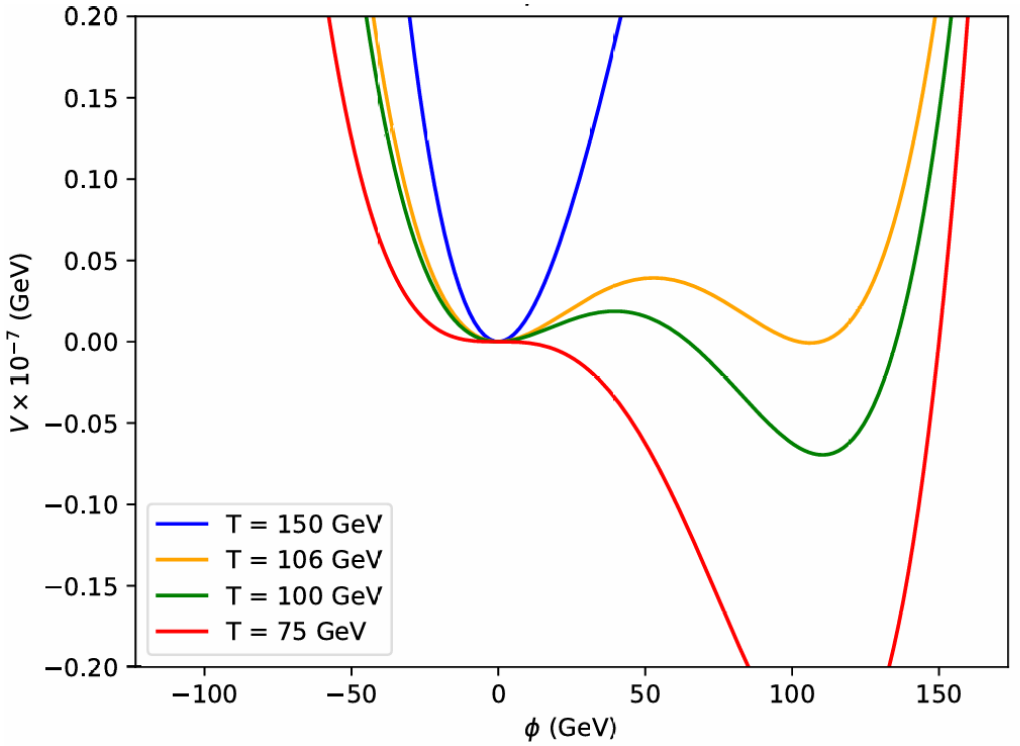
$\phi_f = 0$ 对称相失去稳定 $T = T_0$

$$\phi_t = \frac{3ET \pm \sqrt{9E^2T^2 - 8\lambda D(T^2-T_0^{\,2})}}{2\lambda}$$

临界温度

$$V(\phi_f,T_c) = V(\phi_t,T_c)$$

$$T_c = \frac{T_0}{\sqrt{1-\frac{E^2}{\lambda D}}}$$



三次项的作用

一阶相变条件

没有三次项的简化势能

$$V(\phi,T) = D(T^2-T_0^2)\phi^2 + \frac{\lambda}{4}\phi^4$$

Phase结构

- 0 $\quad T>T_0$
- 三相合一 $\quad T=T_0$
- $\phi^2=\frac{2D}{\lambda}(T_0^2-T^2)$ $\quad T<T_0$

势能在零点处很平
四阶平坦 $\quad \frac{d^2V}{d\phi^2}=0$

$T<T_0,\ \frac{d^2V}{d\phi^2}=2D(T^2-T_0^2)<0$ → 零点为极大值

二阶相变/Crossover?

极小值带回自由能

$V(T)$

- 0 $\quad T>T_0$
- $-\frac{D^2}{\lambda}(T_0^2-T^2)^2$ $\quad T<T_0$

$\frac{d^2V}{dT^2}$

- 0 $\quad T>T_0$
- $4\frac{D^2}{\lambda}(T_0^2-3T^2)$ $\quad T<T_0$

势能二阶导数不连续!

成核温度 Tn

相变真正开始的温度

成核温度：

$$\Gamma \sim A\, e^{-S_3/T}$$

$$N(T_n) = \int_{T_n}^{T_c} \frac{\Gamma}{H^4}\, dT \sim \mathcal{O}(1)$$

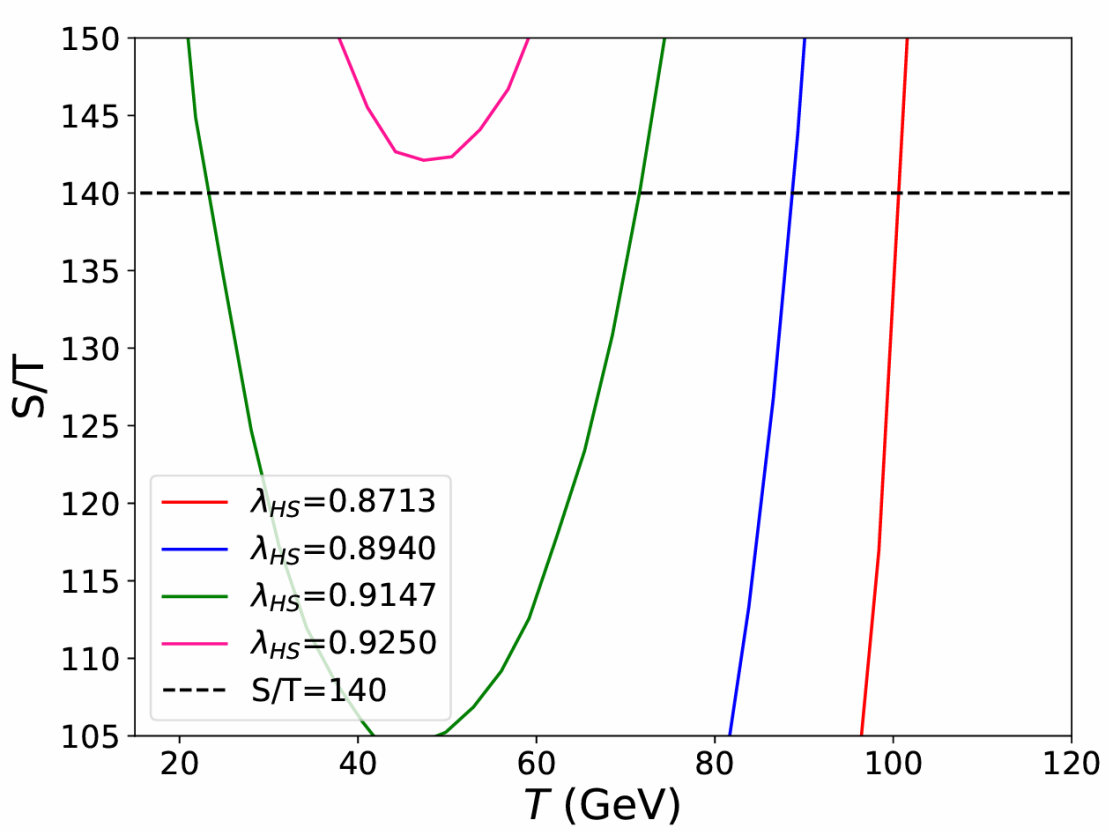
简化估计：

$$\frac{\Gamma}{H^4} \sim \mathcal{O}(1) \xrightarrow{A \sim T^4} T^4 e^{-S_3/T}/H^4 \sim \mathcal{O}(1)$$

辐射主导

带入H表达式

$$\frac{S_3}{T} \sim 140$$



热 bounce 方程

O(3) 对称解

反弹解：

$$\Gamma \sim A\, e^{-S_3/T}$$

$$S_3 = \int 4\pi r^2 dr \left[\frac{1}{2}\left(\frac{\partial\phi}{\partial r}\right)^2 + V(\phi,T)\right]$$

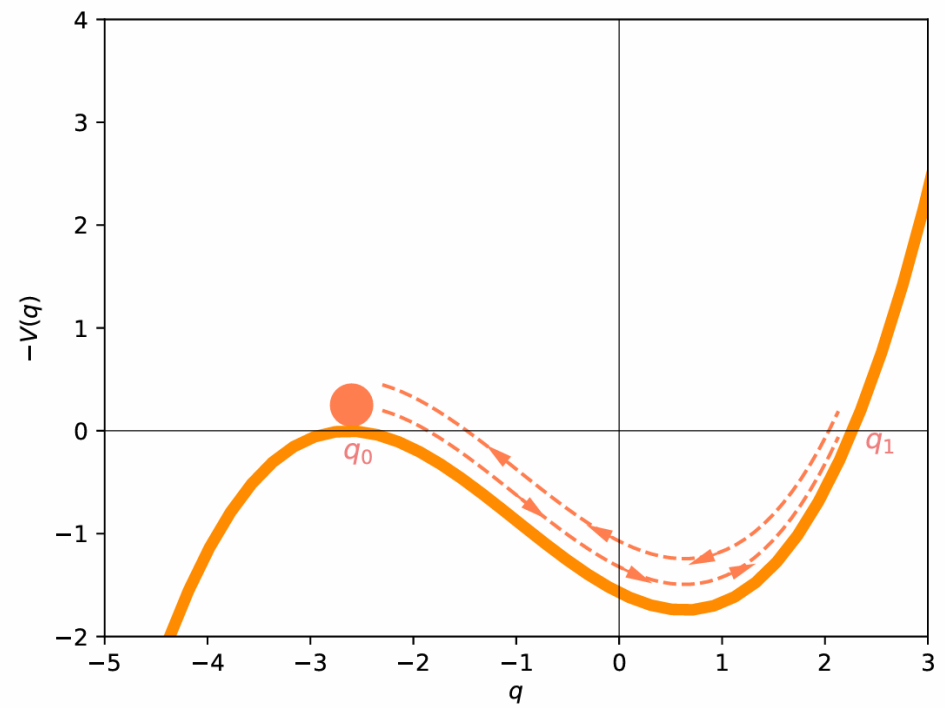
假设
球对称性

变分极值点

$$\frac{d^2\phi}{dr^2} + \frac{2}{r}\frac{d\phi}{dr} = \frac{\partial V(\phi,T)}{\partial\phi}$$

$$\phi(r \to \infty) = \phi_f$$

$$r = 0,\ \frac{d\phi}{dr} = 0$$



V. Rubakov, Classical theory of gauge fields. Princeton University Press, 2009

求解bounce解

action 的物理意义

打靶法思路

$$\frac{d^2\phi}{dr^2}+\frac{2}{r}\frac{d\phi}{dr}=\frac{\partial V(\phi,T)}{\partial\phi}$$

已知条件：$\phi(r\to\infty)=\phi_f$

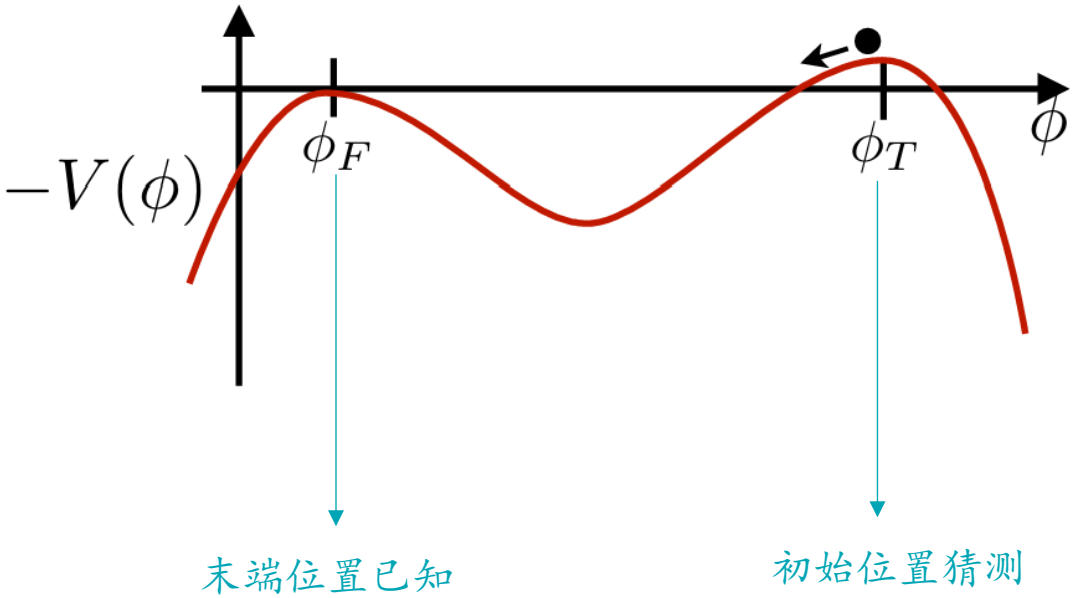
思路转换：$\frac{d^2\phi}{dr^2}=-\frac{2}{r}\frac{d\phi}{dr}-\frac{\partial -V(\phi,T)}{\partial\phi}$

将r看作时间

$F=-\nabla V$

摩擦力　　驱动力

类比：小球在势能为-V中进行一个有摩擦力的运动



薄壁与厚壁图像

profile 形状为何重要

薄壁近似

温度趋近临界温度附近时

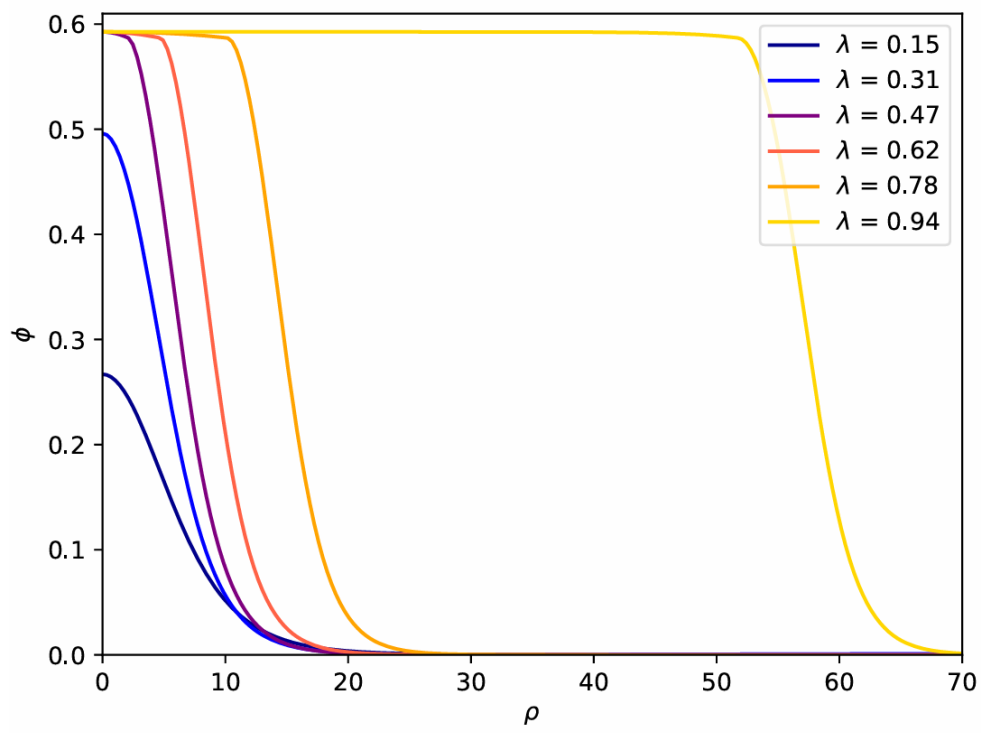
$$\phi(r)\begin{cases}\phi_t & r<R\\ \text{快速变化} & r\sim R\\ \phi_f & r>R\end{cases}$$

可以得到解析作用量 $\longrightarrow$ $S_3=\dfrac{16\pi\sigma^3}{3(\Delta V)^2}$

厚壁近似

$$L\sim R$$

发生在较强过冷或远离 T_c 时。泡泡内部、泡壁和外部不能清楚分开，场构型从中心到外部平滑变化。此时表面能和体积能的分解不再可靠，需要直接数值求解 bounce 方程。



多场隧穿的难点

路径未知，不只是解一维方程

多场的难点：

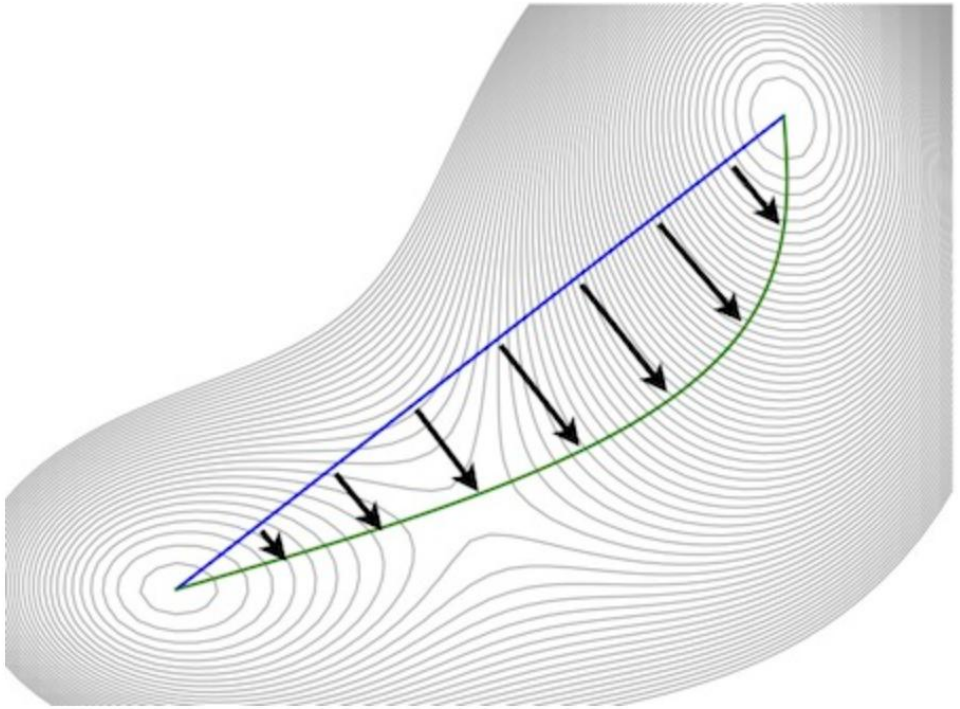
$$\frac{d^2\phi}{dr^2}+\frac{2}{r}\frac{d\phi}{dr}=\frac{\partial V(\phi,T)}{\partial\phi} \longrightarrow \frac{d^2\phi_i}{dr^2}+\frac{2}{r}\frac{d\phi_i}{dr}=\frac{\partial V(\phi_i,T)}{\partial\phi_i}$$

与单场不同的点

- 场是向量
- 路径可能绕开势垒高处
- 不能只取某一个方向的切片

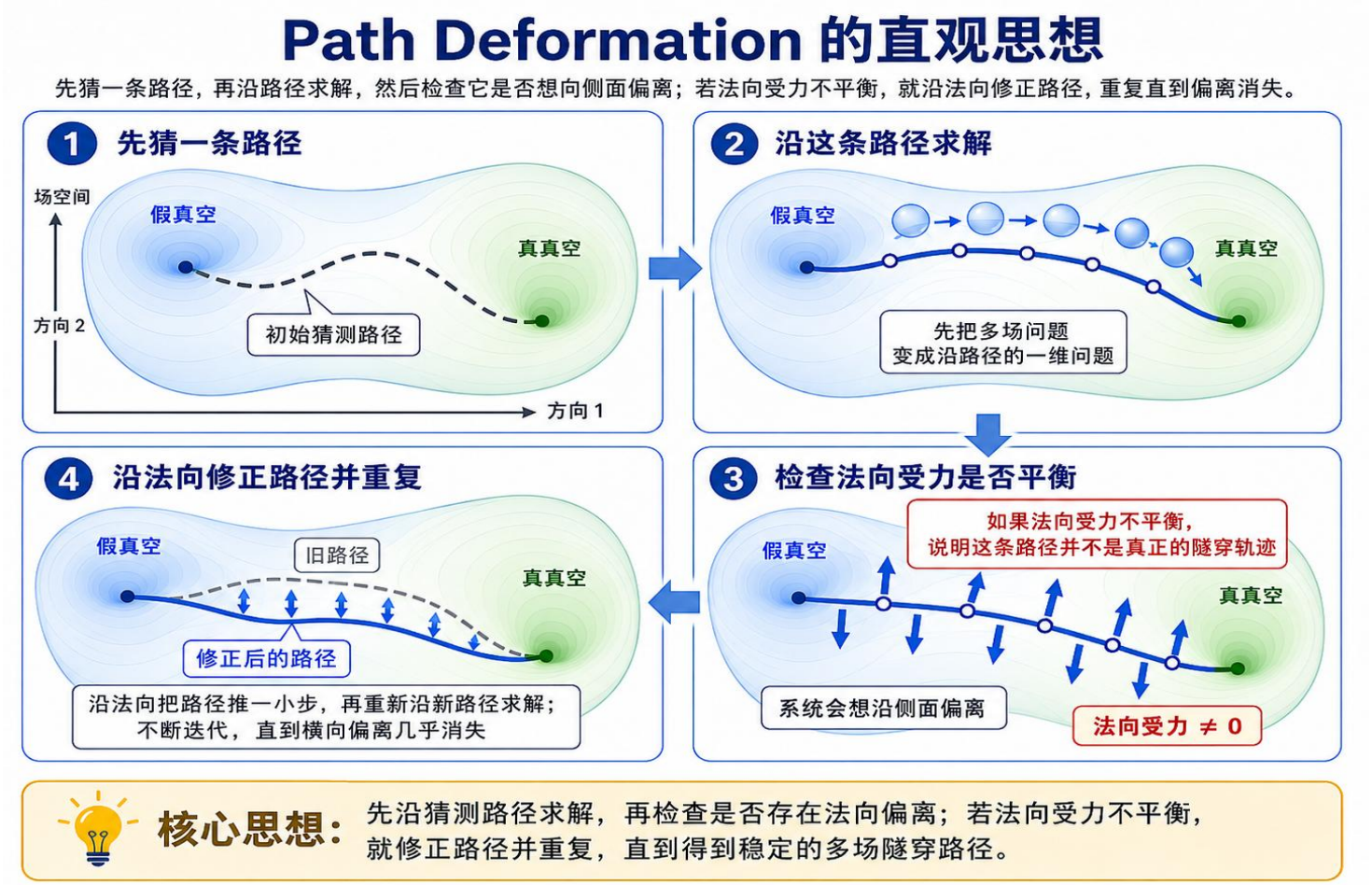
一维打靶法失效！

多维打靶？　　可行，算力要求严格！



path deformation 的直观想法

先猜路径，再变形到真实路径



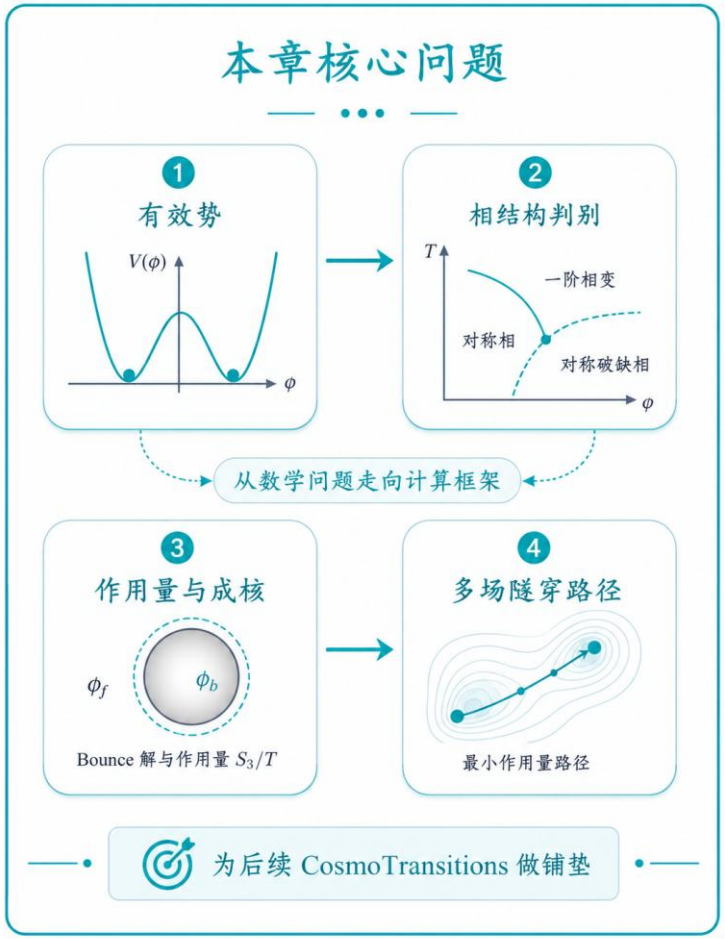
本章小结

从数学进入软件

小结

- BSM势能具体的复杂点
- Phase结构的求法，一阶二阶相变如何判断？
- 简单模型作用量如何计算？
- 多场隧穿的计算思路

收束：下面开始介绍 CosmoTransitions 如何组织这些计算。



03

CosmoTransitions 总体结构

从模块到工作流

本章回答

CosmoTransitions的组成结构

- generic_potential
- transitionFinder
- tunneling1D / pathDeformation

CosmoTransitions 是什么？

核心定位

软件定位

CosmoTransitions 是一个给定有限温有效势后，用来追踪相结构并计算相变隧穿过程的 Python 工具。

主要功能

- 根据用户定义的 $V_{\mathrm{eff}}(\phi, T)$ 追踪不同温度下的真空结构；
- 判断不同相之间是否发生相变，并给出临界温度、过冷温度等信息；
- 计算一阶相变中的 bounce / instanton profile，从而得到 S_3/T。

安装方式

pip install cosmoTransitions

CosmoTransitions: Computing Cosmological Phase Transition Temperatures and Bubble Profiles with Multiple Fields

Carroll L. Wainwright

Department of Physics, University of California, 1156 High St., Santa Cruz, CA 95064, USA

四个核心模块

模块分工表

软件核心组成

- generic_potential：模型入口
- transitionFinder：phase 与 transition
- tunneling1D / pathDeformation：bounce

软件核心作用

1. generic_potential 负责“告诉程序模型是什么”
2. transitionFinder 负责“找到相变发生在哪里”
3. pathDeformation 和 tunneling1D 负责“算出泡泡如何从假真空隧穿到真真空”

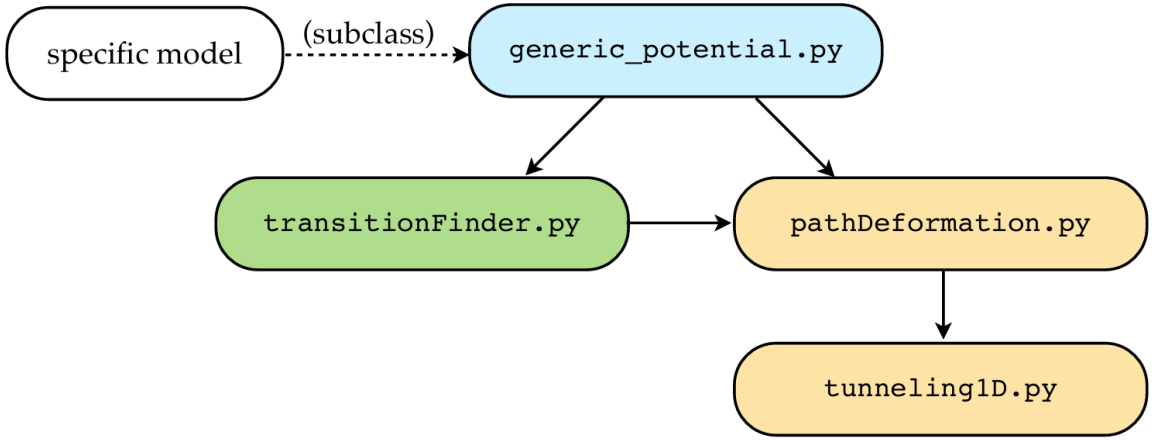


Figure 1: Overview of the CosmoTransitions package file structure. The files `pathDeformation.py` and `tunneling1D.py` find critical bubble profiles, `transitionFinder.py` finds the minima of finite temperature potentials as a function of temperature and analyzes phase transitions, and `generic_potential.py` defines an abstract class that can easily be subclassed to examine a specific model.

典型调用流程

最短代码主线

计算流程

1.建立模型后，实例化模型：m = MyModel()

2.把物理模型交给程序：m.getPhases()

3.追踪温度变化下的不同相： m.calcTcTrans()

4.寻找相并存的临界温度 T_c：m.findAllTransitions()

5.寻找实际发生的成核相变 T_n： m.TnTrans

6.读取 T_n、S_3/T、真空位置和 **bounce profile**

```
# 1. 构造模型
m = MyModel()

# 2. 追踪有限温相结构
m.getPhases()

# 3. 计算临界温度 Tc
m.calcTcTrans()

# 4. 寻找所有实际相变
m.findAllTransitions()

# 5. 读取成核温度和 bounce 信息
for tr in m.TnTrans:
    Tn = tr['Tnuc']
    S3_over_T = tr['action'] / Tn
    phi_high = tr['high_vev']
    phi_low  = tr['low_vev']
    instanton = tr['instanton']
```

数据流与对象关系

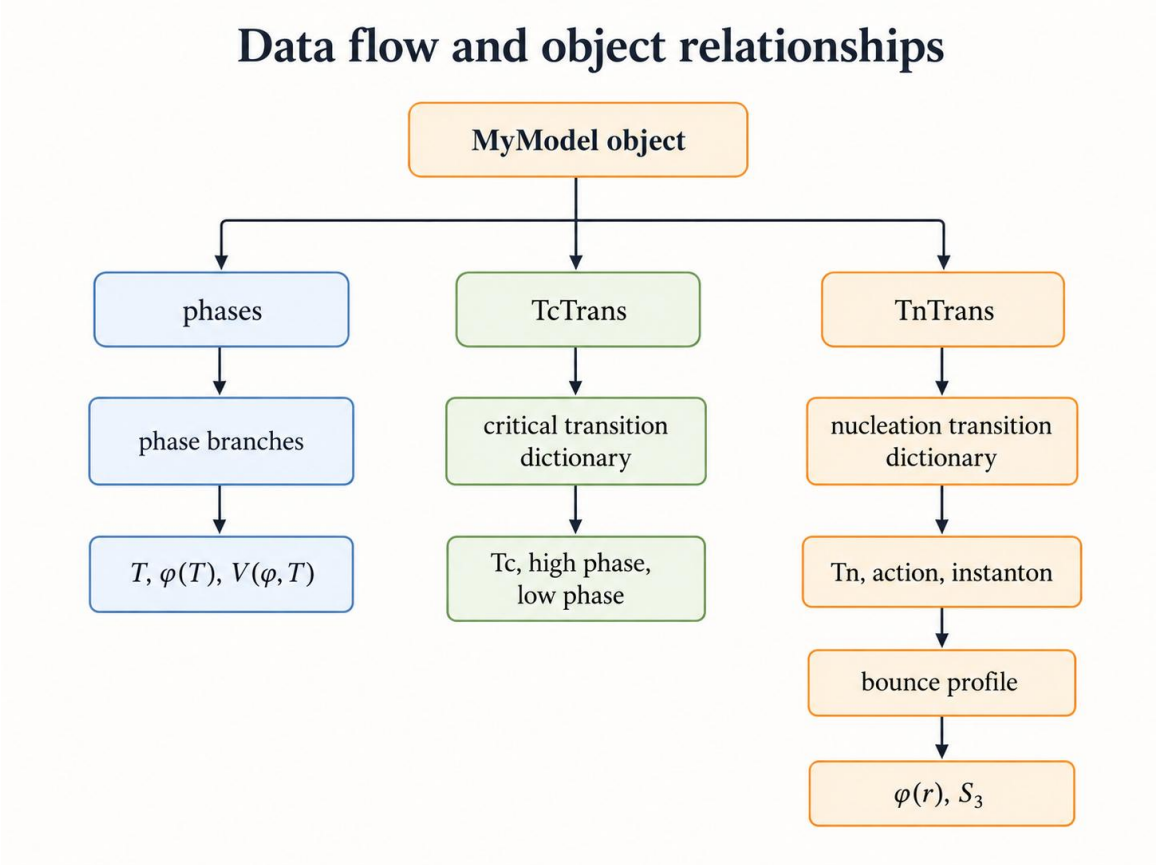
从 class 到 transition dictionary

模型结果

model.phases：保存不同温度下的 **phase branches**
model.TcTrans：保存临界温度处的 **critical transitions**
model.TnTrans：保存实际成核过程的 **nucleation transitions**

实际计算中需要用到action随着温度的变化，cosmoTransition需要额外的方式才能提取

Alpha, beta/H, Tn → 拟合公式 → GW



04

模型实现：generic_potential

把物理模型翻译成 CosmoTransitions class

本章回答

如何写一个可以被 phase tracing 和 tunneling 使用的模型？

- init / V0
- boson_massSq / fermion_massSq
- approxZeroTMin / Vtot

generic_potential 的最小骨架

class 结构

Generic_potential是什么

用户定义物理模型的基类。后续相结构追踪和 bounce 计算，都从这个 class 读取势能和质量谱。

包含的内容：

init()：设置 Ndim、Tmax、模型参数

V0(X)：定义 tree-level potential

boson_massSq(X,T)：定义玻色质量谱

fermion_massSq(X)：定义费米质量谱

可选：approxZeroTMin() 给出零温极小值初猜

```
import numpy as np
from cosmoTransitions import generic_potential

class MyModel(generic_potential.generic_potential):

    def init(self):
        # 1. 设置场空间维度和温度范围
        self.Ndim = 1
        self.Tmax = 300.0
        self.renormScaleSq = 100.0**2

        # 2. 设置模型参数
        self.mu2 = 80.0**2
        self.lam = 0.1
        self.cT = 0.2

    def V0(self, X):
        # 3. 定义 tree-level potential
        X = np.asanyarray(X)
        phi = X[..., 0]
        return -0.5*self.mu2*phi**2 + 0.25*self.lam*phi**4

    def boson_massSq(self, X, T):
        # 4. 定义场依赖的玻色质量谱
        X = np.asanyarray(X)
        phi = X[..., 0]

        m2 = -self.mu2 + 3*self.lam*phi**2 + self.cT*T**2

        massSq = np.array([m2]).T
        dof = np.array([1.0])
        c = np.array([1.5])

        return massSq, dof, c

    def fermion_massSq(self, X):
        # 5. 如果没有费米子，可以返回空数组
        return np.array([]), np.array([])
```

init(): 模型元信息

Ndim 与扫描范围

Init()做什么?

定义场空间维数:
self.Ndim = 1 表示单场，self.Ndim = 2 表示双场

定义温度扫描范围:
self.Tmax 决定 phase tracing 从多高温开始

保存模型参数:
这些参数之后会在 V0()、boson_massSq() 中被调用

设置数值精度以及能标:
例如 x_eps、T_eps、renormScaleSq

```
class MyModel(generic_potential.generic_potential):

def init(self):
# 1. 场空间维数
# 例如 X = [h, s]，所以 Ndim = 2
self.Ndim = 2

# 2. 相结构追踪的最高温度
self.Tmax = 300.0

# 3. 数值设置
self.x_eps = 1e-3
self.T_eps = 1e-3
self.renormScaleSq = 246.0**2

# 4. 模型参数
self.mu_h2 = 88.0**2
self.lam_h = 0.13
self.mu_s2 = 120.0**2
self.lam_s = 0.20
self.lam_hs = 0.10
```

V0(X): tree-level potential

场变量的组织方式

输入说明：

X 表示场空间中的一个点

X[...,0] 对应第一个场，例如 h
X[...,1] 对应第二个场，例如 s

返回值是该点所对应的势能

Python 语法示例

```
X = np.array([[246.0, 100.0],
[200.0, 80.0],[150.0, 50.0]])

h = X[..., 0]
s = X[..., 1]

print(h) # [246. 200. 150.]
print(s) # [100. 80. 50.]
```

```
def V0(self, X):
    """
    Tree-level potential.
    X[...,0] = h
    X[...,1] = s
    """
    X = np.asanyarray(X)

    h = X[..., 0]
    s = X[..., 1]

    Vh = -0.5*self.mu_h2*h**2 + 0.25*self.lam_h*h**4
    Vs =  0.5*self.mu_s2*s**2 + 0.25*self.lam_s*s**4
    Vhs = 0.25*self.lam_hs*h**2*s**2

    return Vh + Vs + Vhs
```

boson_massSq(X,T)

玻色子谱与 daisy 入口

Boson_massSq的作用

boson_massSq(X,T) 把玻色子质量谱、自由度和 Coleman-Weinberg 常数项交给程序

输出内容

输出三组等长数组： massSq, dof, c

massSq：各玻色模式的场依赖质量平方$m_i^2(\phi, T)$

dof：对应的自由度数 n_i

c：Coleman-Weinberg 常数项
标量常取 3/2，规范玻色子常取 5/6

若考虑 daisy resummation，可在这里加入 thermal mass

```
def boson_massSq(self, X, T):
    X = np.asanyarray(X)
    h = X[..., 0]
    s = X[..., 1]

    Pi_h = self.c_h * T**2
    Pi_s = self.c_s * T**2

    mh2 = -self.mu_h2 + 3*self.lam_h*h**2 \
        + 0.5*self.lam_hs*s**2 + Pi_h

    mG2 = -self.mu_h2 + self.lam_h*h**2 \
        + 0.5*self.lam_hs*s**2 + Pi_h

    ms2 = self.mu_s2 + 3*self.lam_s*s**2 \
        + 0.5*self.lam_hs*h**2 + Pi_s

    massSq = np.array([mh2, mG2, ms2]).T
    dof = np.array([1, 3, 1])
    c = np.array([1.5, 1.5, 1.5])

    return massSq, dof, c
```

fermion_massSq(X)

费米子谱

Fermion_massSq的作用

fermion_massSq(X,T) 把费米子质量谱、自由度交给程序

输出内容

输出三组等长数组： massSq, dof

massSq：各费米模式的场依赖质量平方$m_i^2(\phi, T)$

dof：对应的自由度数 n_i

Coleman-Weinberg 常数项固定

```
def fermion_massSq(self, X):
    """
    Field-dependent fermion masses.
    Usually dominated by the top quark.
    """
    X = np.asanyarray(X)

    h = X[..., 0]

    mt2 = 0.5 * self.yt**2 * h**2

    massSq = np.array([mt2]).T
    dof = np.array([12.0])

    return massSq, dof
```

approxZeroTMin()

帮助程序找到 phase

approxZeroTMin作用

帮助程序找到零温附近可能存在的相，
作为后续 phase tracing 的初始点。

注意事项：

- 返回的是一组候选极小值点
- 不要求每个点都非常精确
- 候选点太少可能漏掉 phase branch
- 多场模型建议给多个初始点。对称相、破缺相、混合相都可以作为候选

```
def approxZeroTMin(self):
    """
    Approximate zero-temperature minima.
    These points are used as starting guesses
    for phase tracing.
    """

    return [
        np.array([0.0, 0.0]),       # symmetric phase
        np.array([246.0, 0.0]),     # Higgs-like phase
        np.array([0.0, 100.0]),     # singlet-like phase
        np.array([246.0, 100.0])    # mixed phase
    ]
```

何时重写 Vtot?

完全自定义有效势

注意事项

默认情况

适合使用默认 Vtot：

有明确的 tree-level potential：V0(X)

可以写出玻色子/费米子质量谱

一圈 CW 修正和热修正由程序自动构造

需要重写 Vtot 的情况

已经有完整的 $V_{\text{eff}}(\phi, T)$

使用外部程序、拟合函数或解析近似给出有效势

需要自定义 counterterm、daisy 或重整化约定

默认一圈热势不适合当前模型

```
def Vtot(self, X, T, include_radiation=True):
    """
    Custom finite-temperature effective potential.
    Use this only when the default construction
    is not suitable for the model.
    """
    X = np.asanyarray(X)
    h = X[..., 0]
    s = X[..., 1]

    V0 = self.V0(X)

    # Example: user-defined thermal correction
    VT = 0.5*self.c_h*T**2*h**2 \
       + 0.5*self.c_s*T**2*s**2 \
       - self.E*T*h**3

    return V0 + VT
```

05

Phase tracing and forbidden phases

从极小值轨迹到相的过滤

本章回答

程序如何找到 phase？什么时候需要 forbid 某些 branch？

- getPhases()
- 检查 phase branches
- forbidPhaseCrit()

getPhases(): 找相结构

temperature-dependent minima

getPhases作用

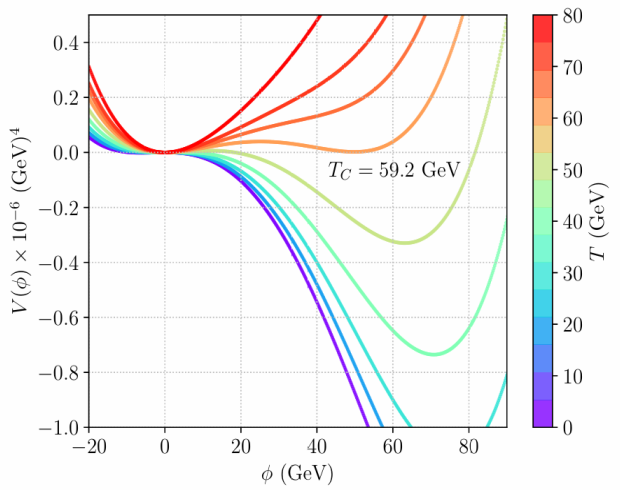
getPhases() 会在不同温度下寻找局部极小值

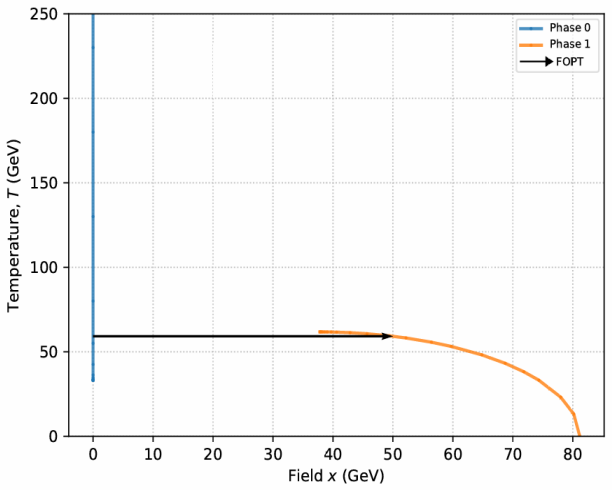
每一个 phase 不是单个点，而是一条随温度变化的分支

程序会尝试把相邻温度上的极小值连接起来

结果保存在：m.phases

后续 calcTcTrans() 和 findAllTransitions() 都依赖这些 phase branches





2003.02859

Phase tracing 的算法框架

Find vacuum

算法主线：

Seed：从 approxZeroTMin() 给出的候选极小值出发

Predict：利用极小值条件预测下一温度点

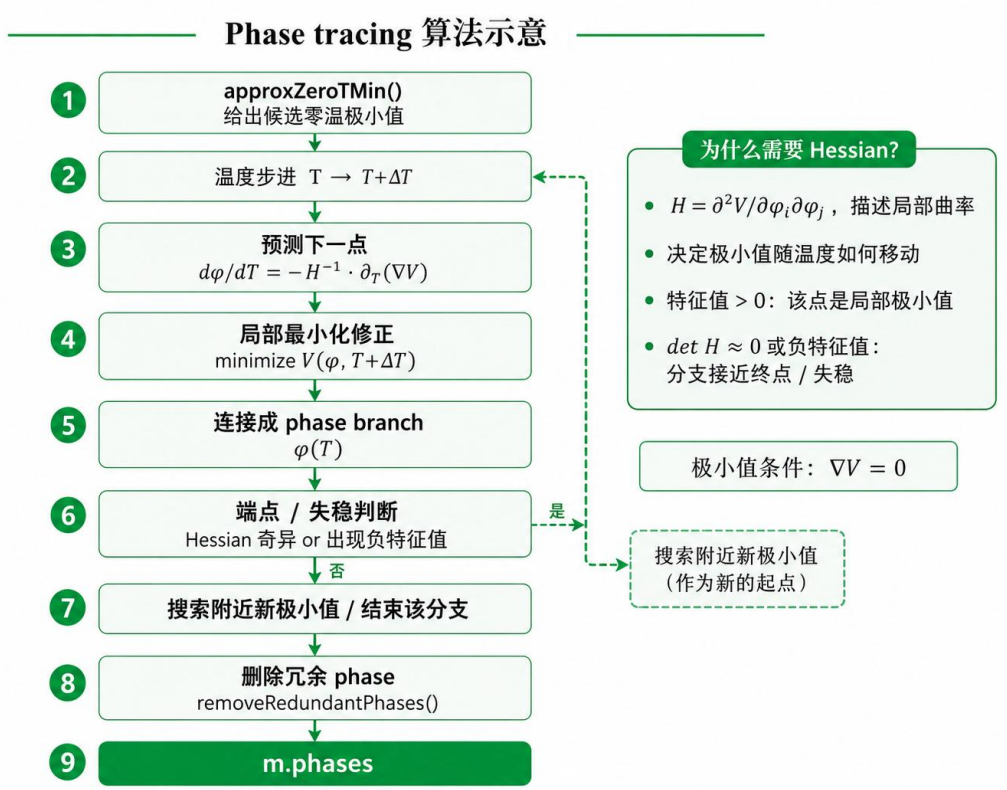
Correct：在新温度下重新做局部最小化

Trace：把连续极小值连接成 phase branch

Detect endpoint：Hessian 变奇异或出现负本征值时，认为该相可能消失

Jump / search：跨过端点后寻找附近新极小值，避免漏掉中间相

Filter：去掉 forbidden phase 和 redundant phase



如何检查 m.phases?

不要只相信 pretty output

如何读取phase的信息

- m.phases 是一个 dictionary
- 每个 key 对应一条 phase branch
- phase.T：该 phase 存在的温度点
- phase.X：每个温度点对应的极小值位置

注意事项

- 不要只看 phase 编号。
  判断它是什么相，要看 T范围和 $X(T)$的端点。
- 由于数值误差的原因，一个Phase可能会被误认为是好几个Phase，此种情况需要额外注意

Print(m.phases)

0: Phase(key=0, X=[[295.6 406.4], ..., [239.6 217.6]],
T=[0, ..., 117.2], dXdT=[[-0 -0], ..., [-61.58 -342.1]],

Phase的温度范围

编号

对应的场值

完整的一个Phase信息

为什么会有 redundant phases?

对称相关的重复 branch

为什么会有 redundant / forbidden phases

因为数值上可能找到多条分支，但其中一些只是对称相关或重复追踪的同一个物理相。

举例

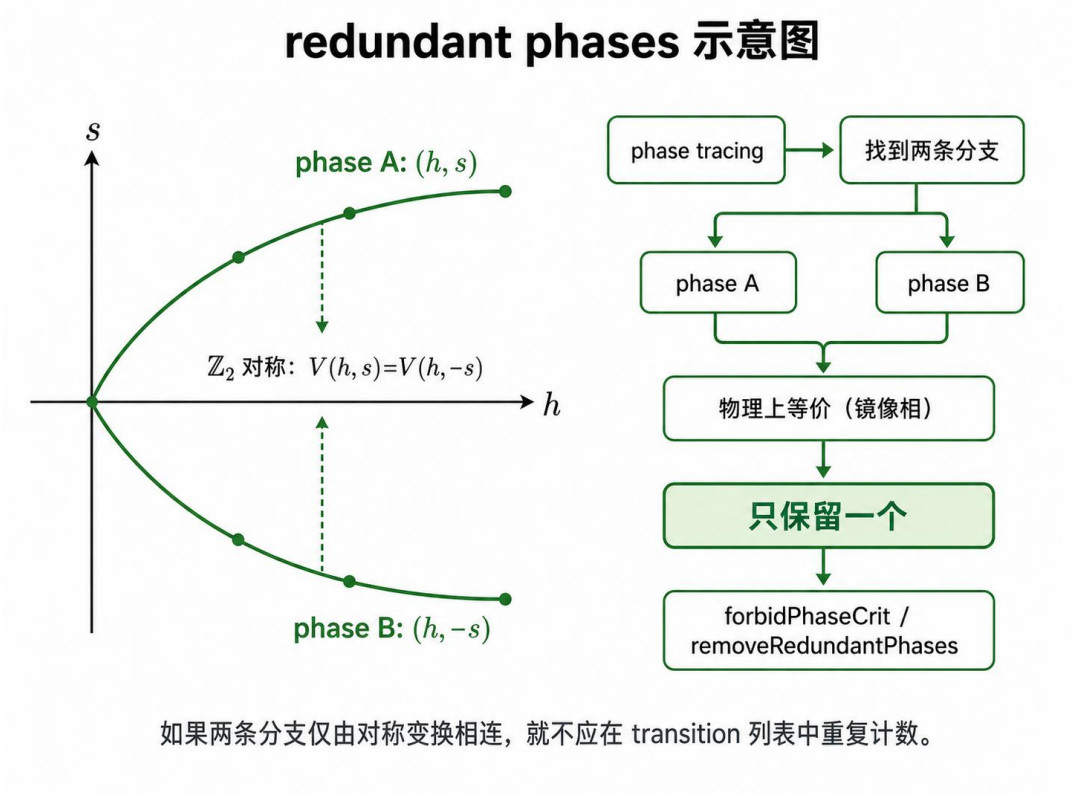
势能可能有离散对称性，例如

$$V(h,s) = V(h,-s)$$

数值上就会找到两个分支

$$(h(T),s(T)) \qquad (h(T),-s(T))$$

等价！



如果都保留，后面会重复计算

transition removeRedundantPhases() 会尝试删除重复分支 → forbidPhaseCrit

forbidPhaseCrit(X)

选择 fundamental domain

forbidPhaseCrit

用于告诉程序，那部分的phase需要被丢弃

注意：

- 返回表示不要追踪这个 phase
- 常用于 s<0 或 h<0 的重复区域
- 要加容差，避免数值误删

getPhases() 会把它作为 forbidCrit 传入 traceMultiMin()，之后还会调用 removeRedundantPhases() 删除重复相。

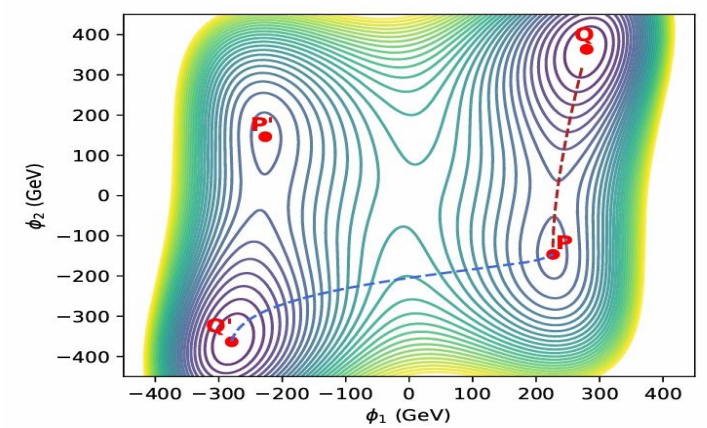
```
def forbidPhaseCrit(self, X):
    """
    Return True  -> discard this phase point.
    Return False -> keep this phase point.

    Example: for a Z2 symmetry s <-> -s,
    keep only the region s >= 0.
    """
    X = np.asanyarray(X)

    h = X[..., 0]
    s = X[..., 1]

    eps = 1e-5

    # discard the mirror branch with s < 0
    return np.any(s < -eps)
```

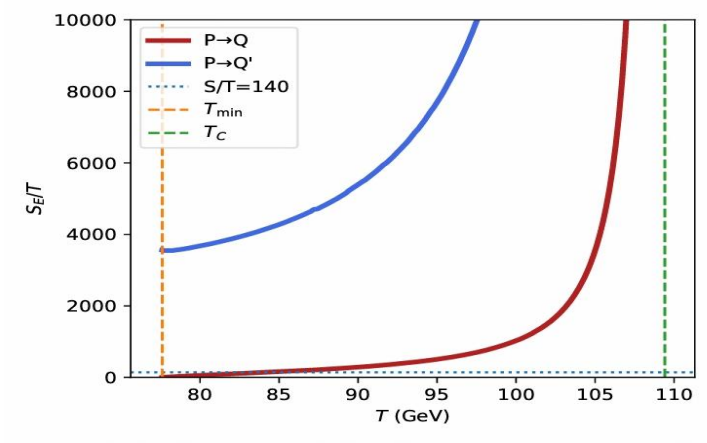
forbidPhaseCrit 的风险

谨慎对待

注意事项

- **误删真实相**
  如果没有严格对称性，不能随便删掉某个区域
- **漏掉相变路径**
  被删掉的 phase 可能正是中间相或成核终态
- **改变 transition 列表**
  m.phases 不完整可能会导致 TcTrans / TnTrans 不可靠
- **数值过滤 ≠ 物理不存在**
  forbidden 只表示“不让程序追踪”，不是证明这个相不存在





06

Transition、action 与 bubble profile

从 TnTrans 中拿出真正有用的物理量

本章回答

怎么提取 S3、S3/T、instanton、profile 和多场路径？

- calcTcTrans / findAllTransitions
- action 与 S3/T
- bubble profile / fRatio

calcTcTrans(): 临界相变

先找势能简并

calcTcTrans

寻找 phase branches 之间的临界温度

示范输出

[{'Tcrit': 222.97792375905584, 'low_vev': array([0.16185995, -0.12317515]),
'high_vev': array([0.16185995, -0.12317515]), 'low_phase': 1, 'high_phase': 2, 'action': 0.0,
'instanton': None, 'trantype': 2, 'Delta_rho': 0.0},

{'Tcrit': 109.40840756814627, 'high_vev': array([220.02184053, -150.01519068]),
'high_phase': 1, 'low_vev': array([263.48801219, 314.65396889]),
'low_phase': 0, 'trantype': 1, 'Delta_rho': 471749616.58194387}]

```
m = model1()

# 1. 先追踪相结构
m.getPhases()

# 2. 寻找临界相变
tc_trans = m.calcTcTrans()

# 3. 查看输出
for tr in tc_trans:
    print("Tcrit =", tr["Tcrit"])
    print("high phase =", tr["high_phase"])
    print("low phase  =", tr["low_phase"])
    print("high vev =", tr["high_vev"])
    print("low vev  =", tr["low_vev"])
    print("trantype =", tr["trantype"])
    print("Delta_rho =", tr["Delta_rho"])
```

findAllTransitions(): 成核相变

寻找 tunneling transition

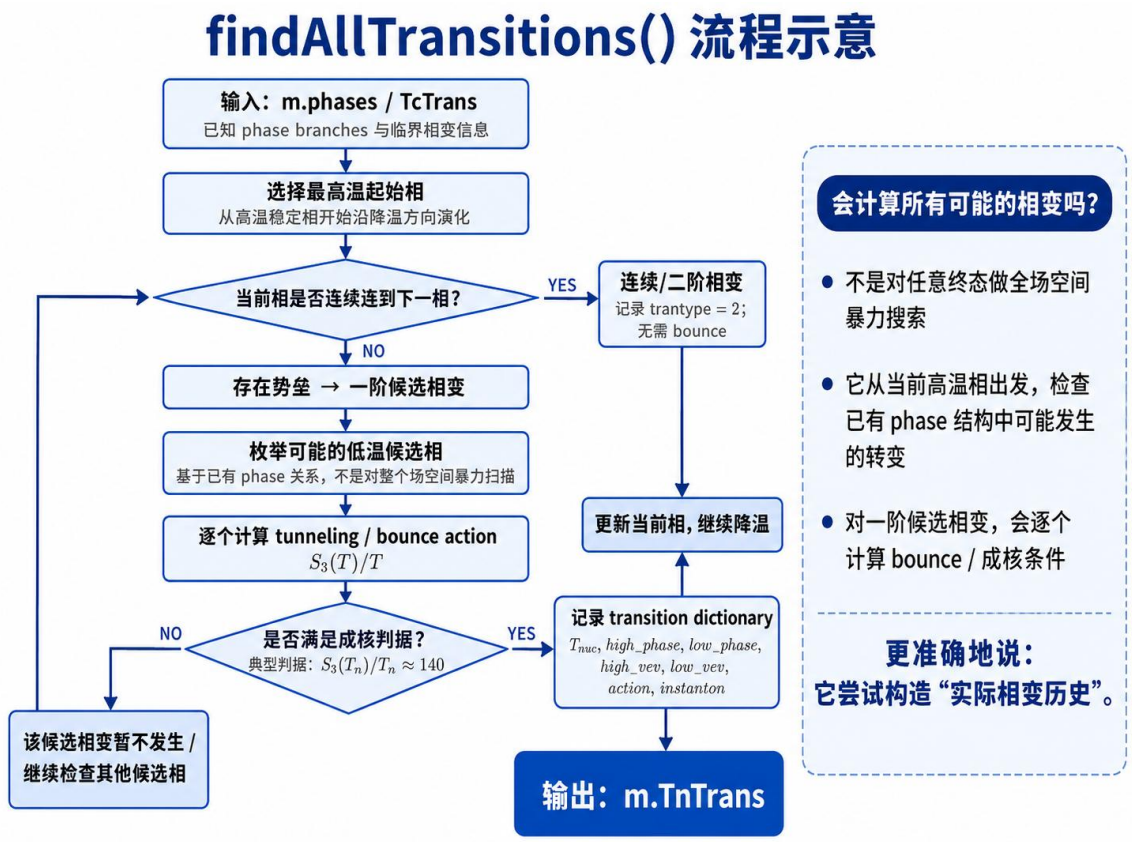
findAllTransitions

从相结构中寻找实际发生的成核相变

内部流程

- 从 m.phases 中选择高温相作为起点 沿降温方向追踪相变历史
- 判断当前相是否会连续转变，或需要一阶隧穿 对一阶候选相变
- 调用 tunneling 算法计算 bounce action 寻找满足成核判据的温度：

$$\frac{S_3}{T} \sim \mathcal{O}(140)$$



transition dictionary

常用 key 一页讲清

findAllTransitions的输出

TnTrans以列表形式存储了有关成核温度相关的信息

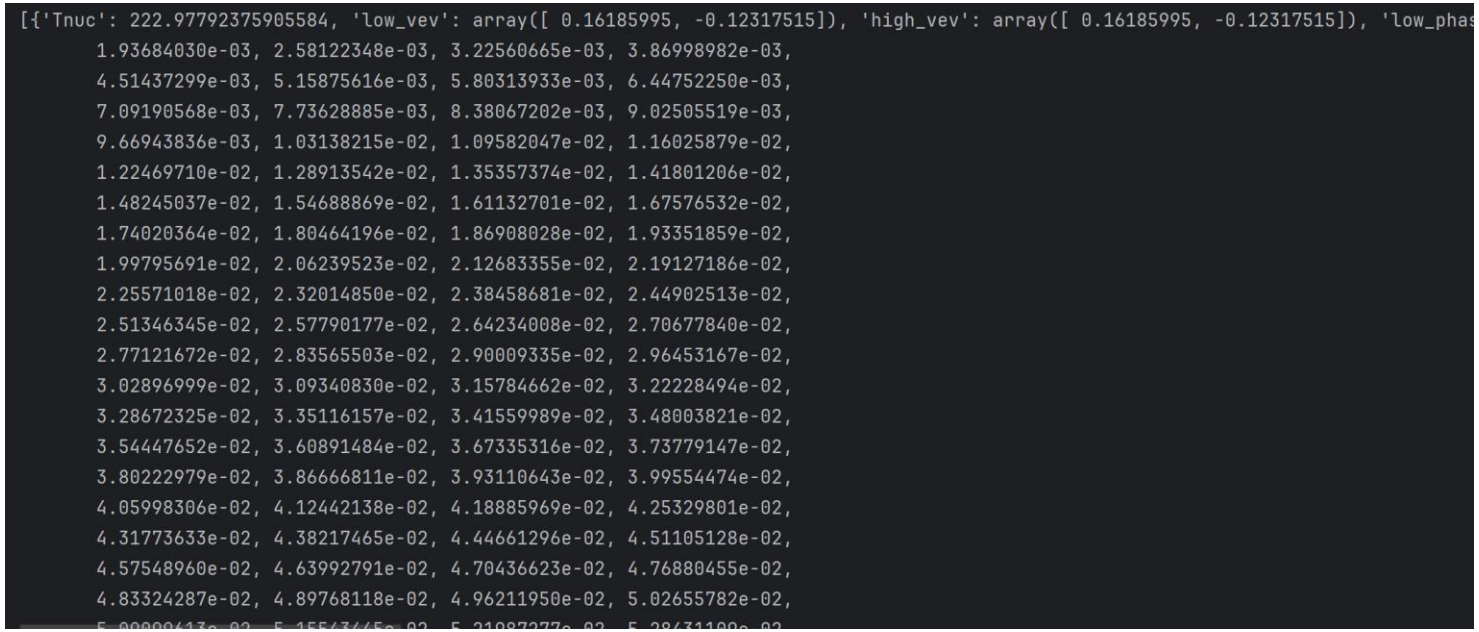
常用命令

TnTrans[‘low_vev’]

TnTrans[‘high_vev’]

TnTrans[‘T_nuc’]

TnTrans[‘trantype’]



action 不是 S3/T

一个最重要的细节

Action

trans["action"]：欧氏作用量$S_3(T_n)$

trans["Tnuc"]：成核温度T_n

真正常用于报告的是：

$$\frac{S_3(T_n)}{T_n} = \frac{trans["action"]}{trans["Tnuc"]}$$

默认成核判据近似为：

$$S_3(T_n)/T_n \simeq 140$$

```
trans = m.TnTrans[0]

Tn = trans["Tnuc"]
S3 = trans["action"]

S3_over_Tn = S3 / Tn

print("Tn =", Tn)
print("S3 =", S3)
print("S3/Tn =", S3_over_Tn)
```

如何手动扫 S3/T?

固定温度调用 fullTunneling

如何提取action 随着温度的变化

核心步骤

先确定一阶相变的 high_phase 和 low_phase

对每个温度 T，重新取两相的 VEV：

```
high = m.phases[high_key].valAt(T)
low  = m.phases[low_key].valAt(T)
```

固定温度定义势能：

```
V  = lambda X: m.Vtot(X, T, False)
dV = lambda X: m.gradV(X, T)
```

调用：

```
inst = pd.fullTunneling(path_pts, V, dV)
```

调用：

$$S_3(T) = inst.action$$

```
from cosmoTransitions import pathDeformation as pd
import numpy as np

# 选定一阶相变对应的两个 phase
high_key = trans["high_phase"]
low_key  = trans["low_phase"]

Ts = np.linspace(Tmin, Tmax, 20)

curve = []

for T in Ts:

    # 1. 在当前温度重新取两相极小值
    high = m.phases[high_key].valAt(T)
    low  = m.phases[low_key].valAt(T)

    # 2. 固定温度下的势能和梯度
    V  = lambda X: m.Vtot(X, T, False)
    dV = lambda X: m.gradV(X, T)

    # 3. 初始路径：从 true vacuum 到 false vacuum
    path_pts = np.linspace(low, high, 20)

    # 4. 求 bounce
    inst = pd.fullTunneling(path_pts, V, dV)

    S3 = inst.action
    curve.append([T, S3/T])

curve = np.array(curve)
```

从 S3/T 到 beta/H

导数会放大噪声

与引力波的关联

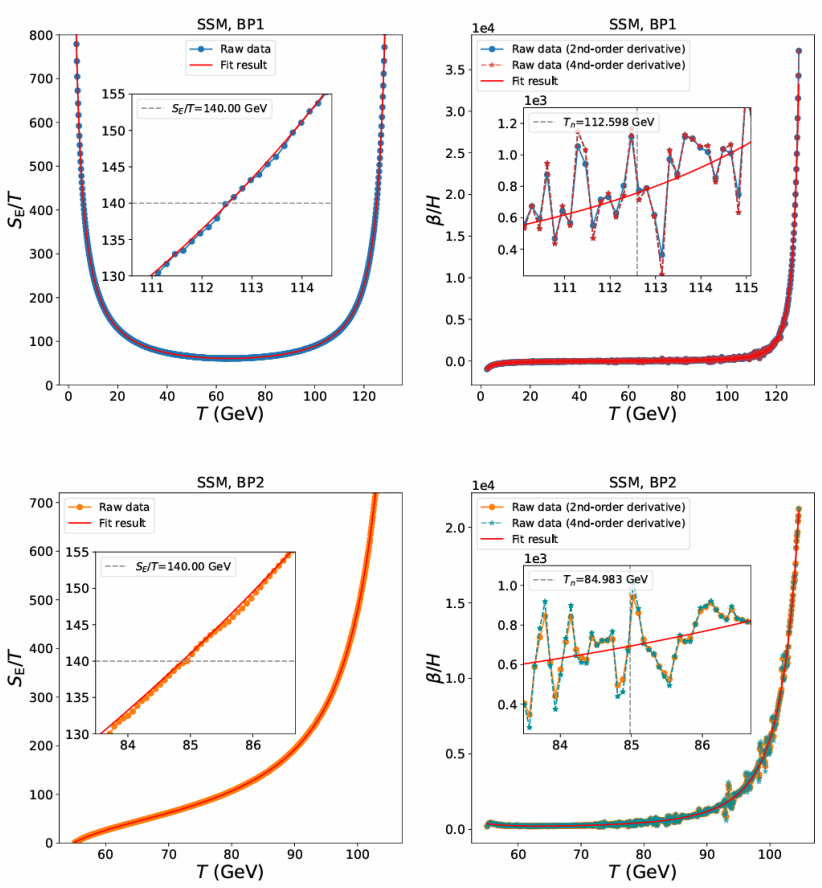
$$\frac{\beta}{H} = T\frac{dS}{dT}$$

注意事项

Action 计算不稳定，数值导数有时候会出一些问题，从而导致结果可信度差

适当的数据处理会使得结果更加的稳定

$$S \sim \frac{f(T)}{(T-T_c)^2} \qquad S = \frac{1}{(T-T_c)^2}\sum_{i=0} q_i T^i$$



instanton 对象

profile 信息在哪里？

instanton

- trans["instanton"] 是一阶相变的 bounce 解对象
  连续 / 二阶相变通常没有 meaningful instanton
- instanton.action 保存欧氏作用量：S_3
- instanton.profile1D.R 保存径向坐标：r
- instanton.Phi 保存多场 profile：$\phi_i(r)$

profile1D: 一维路径 profile

r 与 φ_path(r)

提取bubble的构型

- 单场：SingleFieldInstanton 直接给 φ(r)
- profile1D.R → 径向坐标 r
- profile1D.Phi → 一维路径坐标 φ_path(r)
- profile1D.dPhi → 径向导数 dφ_path/dr

```
from cosmoTransitions.tunneling1D import SingleFieldInstanton

# true vacuum / false vacuum
phi_true  = low
phi_false = high

# 固定温度 T 下的一维势
V1  = lambda phi: V(phi, T)
dV1 = lambda phi: dV(phi, T)

# finite-T O(3) bounce: alpha = 2
single_instanton = SingleFieldInstanton(
    phi_true,
    phi_false,
    V1,
    dV1,
    alpha=2
)

profile = single_instanton.findProfile()
S3 = single_instanton.findAction(profile)

r = profile.R
phi_path = profile.Phi
dphi_path = profile.dPhi
```

多场 profile: instanton.Phi

每个场的径向 profile

多场profile如何提取

计算 bubble profile，并提取每个场的径向分布 $h(r), s(r)$

算法理论

多场记号

$$\phi(r) = [\phi_1(r), \phi_2(r), \phi_3(r)...]$$

重参数化路径 $\phi(r) = \phi(x(r))$

$$x'' + \frac{\alpha}{r}x' = \frac{dV}{dx}$$

$$\frac{d^2\phi}{dx^2}\left(\frac{dx}{dr}\right)^2 = \nabla_\perp V(\phi)$$

垂直方向判据

```
import numpy as np
from cosmoTransitions import pathDeformation as pd

# 固定一个温度，例如成核温度
T = trans["Tnuc"]

# 取该温度下的 high / low phase VEV
high_key = trans["high_phase"]
low_key  = trans["low_phase"]

high = m.phases[high_key].valAt(T)   # false vacuum
low  = m.phases[low_key].valAt(T)    # true vacuum

# 固定温度下的多场势能与梯度
V  = lambda X: m.Vtot(X, T, False)
dV = lambda X: m.gradV(X, T)

# 初始路径：从 true vacuum 到 false vacuum
path_pts = np.linspace(low, high, 30)

# 多场 tunneling + path deformation
inst = pd.fullTunneling(path_pts, V, dV)

# 提取多场 bubble profile
r = inst.profile1D.R
Phi = inst.Phi

h_profile = Phi[:, 0]
s_profile = Phi[:, 1]
```

场空间 tunneling path

从 profile 到路径

多场 tunneling path 的可视化方法

图像含义

- 背景等高线：Veff(h,s,Tn)
- 叠加曲线：inst.Phi = (h(r),s(r))
- 横轴：h，纵轴：s
- 曲线起点：true vacuum，泡泡中心
- 曲线终点：false vacuum，泡泡外部

代码逻辑

- 用 meshgrid 建立 (h,s) 网格
- 用 m.Vtot(X,T,False) 计算势能
- plt.contour 画等高线
- plt.plot(Phi[:,0], Phi[:,1]) 画 tunneling path

```
import numpy as np
import matplotlib.pyplot as plt

T = trans["Tnuc"]
inst = trans["instanton"]

Phi = inst.Phi

h_path = Phi[:, 0]
s_path = Phi[:, 1]

# 背景势能等高线
h_grid = np.linspace(h_path.min()-20, h_path.max()+20, 120)
s_grid = np.linspace(s_path.min()-20, s_path.max()+20, 120)

H, S = np.meshgrid(h_grid, s_grid)
X = np.stack([H, S], axis=-1)

V = m.Vtot(X, T, False)

plt.contour(H, S, V, levels=30)

# 叠加 tunneling path
plt.plot(h_path, s_path, "o-", label="tunneling path")

plt.scatter(h_path[0],  s_path[0],  marker="*", label="true vacuum")
plt.scatter(h_path[-1], s_path[-1], marker="s", label="false vacuum")

plt.xlabel("h")
plt.ylabel("s")
plt.legend()
```

sanity checks

结果是否可信？

注意事项

使用 CosmoTransitions 时的最基本检查

- trantype: 一阶/二阶
- fRatio: 多场路径横向力收敛
- 即使Tn存在，是不是数值假象？
- Phase是不是切断了冗余成分？
- Phase是不是因为数值误差而被劈裂？

检查表 checklist

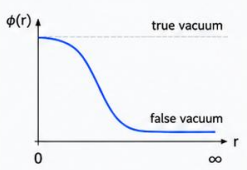
CosmoTransitions 结果可靠性快速检查

检查项	看什么	合理信号
1 **trantype** (transition type)	transition 类型是否正确	一阶相变应为 trantype = 1； 二阶/连续转变通常没有 meaningful instanton
2 **fRatio** (force ratio)	多场路径横向力是否收敛	fRatio 越小越好，通常应显著小于 1， 最好远小于 10^{-2}
3 **Tn 是否可信** (nucleation T)	即使找到 Tn，也要检查 profile / action 是否平滑	S_3/T 曲线平滑，bounce 不出现异常震荡， 结果不是数值假象
4 **Phase 是否完整** (phase tracing)	phase tracing 是否误删或切断 冗余 / 真实分支	对称相关分支处理清楚，transition 列表 没有明显漏项或重复项
5 **Phase 是否被数值劈裂** (branch splitting)	相邻 branches 是否本应连续 却被错误分裂	phase 连接关系自然，T 范围和端点连续， 没有伪分叉

推荐额外检查

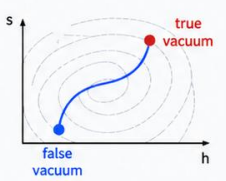
- 检查 *endpoints*： profile 在 $r \to \infty$ 处接近 false vacuum
- 检查 *path*： 场空间路径应平滑连接 true / false vacuum
- 检查 $V(h, s, T)$： 路径与等高线结构应相容
- 检查相变历史： T_c / T_n / *phase* 编号前后一致

1D profile（径向剖面平滑）



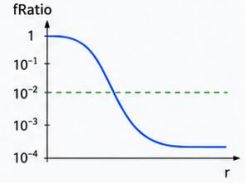
单调、平滑、无震荡

场空间路径（平滑跳穿路径）



平滑连接，不穿越势垒乱区

fRatio 趋势（越小越好）



快速下降并稳定在 $\ll 10^{-2}$

07

完整例子：把流程串起来

toy model example

本章回答

用一个示例把模型实现、phase tracing、action 和 profile 串成完整故事。

- toy model
- 结果图：Tc/Tn/S3/T/profile
- two-field extension

toy model 的势能

一页给公式和参数

模型势能

$$V(\phi_1,\phi_2) = \frac{\lambda_1}{4}(\phi_1{}^2 - v^2)^2 + \frac{\lambda_2}{4}(\phi_2{}^2 - v^2)^2 - \mu\phi_1\phi_2$$

模型参数

$$\lambda_1 = \frac{1}{2}m_1/v^2\,,\ m_1 = 120\ \text{GeV}$$

$$\lambda_2 = \frac{1}{2}m_2/v^2,\ m_2 = 50\ \text{GeV}$$

$$\mu = 25,\ v^2 = (246\ \text{GeV})^2$$

```python
def V0(self, X):
    """
    This method defines the tree-level potential. It should generally be
    subclassed. (You could also subclass Vtot() directly, and put in all of
    quantum corrections yourself).
    """
    # X is the input field array. It is helpful to ensure that it is a
    # numpy array before splitting it into its components.
    X = np.asanyarray(X)
    # x and y are the two fields that make up the input. The array should
    # always be defined such that the very last axis contains the different
    # fields, hence the ellipses.
    # (For example, X can be an array of N two dimensional points and have
    # shape (N,2), but it should NOT be a series of two arrays of length N
    # and have shape (2,N).)
    phi1,phi2 = X[...,0], X[...,1]
    r = .25*self.l1*(phi1*phi1-v2)**2 + .25*self.l2*(phi2*phi2-v2)**2
    r -= self.mu2*phi1*phi2
    return r
```

把 toy model 写成 class

最小代码结构

Boson_massSq

```python
def boson_massSq(self, X, T):
    X = np.array(X)
    phi1,phi2 = X[...,0], X[...,1]

    # We need to define the field-dependnet boson masses. This is obviously
    # model-dependent.
    # Note that these can also include temperature-dependent corrections.
    a = self.l1*(3*phi1*phi1 - v2)
    b = self.l2*(3*phi2*phi2 - v2)
    A = .5*(a+b)
    B = np.sqrt(.25*(a-b)**2 + self.mu2**2)
    mb = self.Y1*(phi1*phi1+phi2*phi2) + self.Y2*phi1*phi2
    M = np.array([A+B, A-B, mb])
    M = np.rollaxis(M, axis: 0, len(M.shape))
    dof = np.array([1, 1, self.n])
    c = np.array([1.5, 1.5, 1.5])

    return M, dof, c
```

approxZeroTMin

```python
def approxZeroTMin(self):
    # There are generically two minima at zero temperature in this model,
    # and we want to include both of them.
    v = v2**.5
    return [np.array([v,v]), np.array([v,-v])]
```

forbidPhaseCrit

```python
def forbidPhaseCrit(self, X):
    """
    forbidPhaseCrit is useful to set if there is, for example, a Z2 symmetry
    in the theory and you don't want to double-count all of the phases. In
    this case, we're throwing away all phases whose zeroth (since python
    starts arrays at 0) field component of the vev goes below -5. Note that
    we don't want to set this to just going below zero, since we are
    interested in phases with vevs exactly at 0, and floating point numbers
    will never be accurate enough to ensure that these aren't slightly
    negative.
    """
    return (np.array([X])[...,0] < -5.0).any()
```

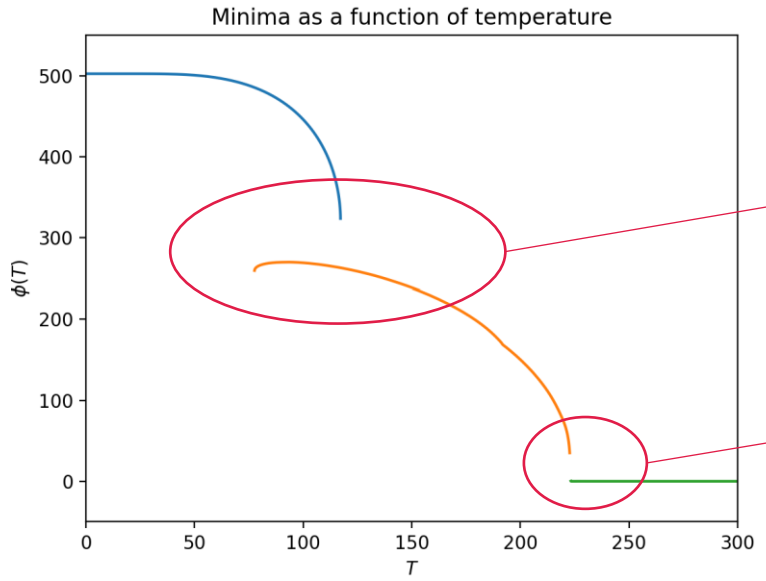
phase tracing 结果

先确认相结构正确

Model.getPhases

0: Phase(key=0, X=[[295.6 406.4], ..., [239.6 217.6]], T=[0, ..., 117.2], dXdT=[[-0 -0], ..., [-61.58 -342.1]],
1: Phase(key=1, X=[[234.3 -111.5], ..., [27.89 -21.21]], T=[77.62, ..., 222.7], dXdT=[[-3.071 -72.61], ..., [-26.52 20.13]],
2: Phase(key=2, X=[[0.1619 -0.1232], ..., [-1.14e-05 0.0002189]], T=[223.2, ..., 1000], dXdT=[[-1.975 1.502], ..., [0.001983 0.001025]]

三相存在！ 但是温度区间仅有一个有较大重叠



$$\phi = \sqrt{{\phi_1}^2 + {\phi_2}^2}$$

较大的重叠区间，这里会反复尝试计算成核温度，寻找相变！

较小的重叠，极其容易出现虚假的成核温度，需要尤其注意！

Tc 与 Tn 的对比

Nucleation is not critical

findAllTransitions

注意检查！

$2 \rightarrow 1$

{'Tnuc': 222.97792375905584, 'low_vev': array([0.16185995, -0.12317515]), 'high_vev': array([0.16185995, -0.12317515]),
'low_phase': 1, 'high_phase': 2, 'action': 0.0, 'instanton': None, 'trantype': 2,

'crit_trans': {'Tcrit': 222.97792375905584, 'low_vev': array([0.16185995, -0.12317515]), 'high_vev': array([0.16185995, -0.12317515]),
'low_phase': 1, 'high_phase': 2, 'action': 0.0, 'instanton': None, 'trantype': 2, 'Delta_rho': 0.0}, 'Delta_rho': 0.0, 'Delta_p': 0.0},

$1 \rightarrow 0$

{'low_vev': array([286.39824856, 382.1972747]), 'high_vev': array([231.10296147, -136.82260143]), 'Tnuc': 84.23819902800427,
'low_phase': 0, 'high_phase': 1

'crit_trans': {'Tcrit': 109.40840756814627, 'high_vev': array([220.02184053, -150.01519068]), 'high_phase': 1,
'low_vev': array([263.48801219, 314.65396889]), 'low_phase': 0, 'trantype': 1}]

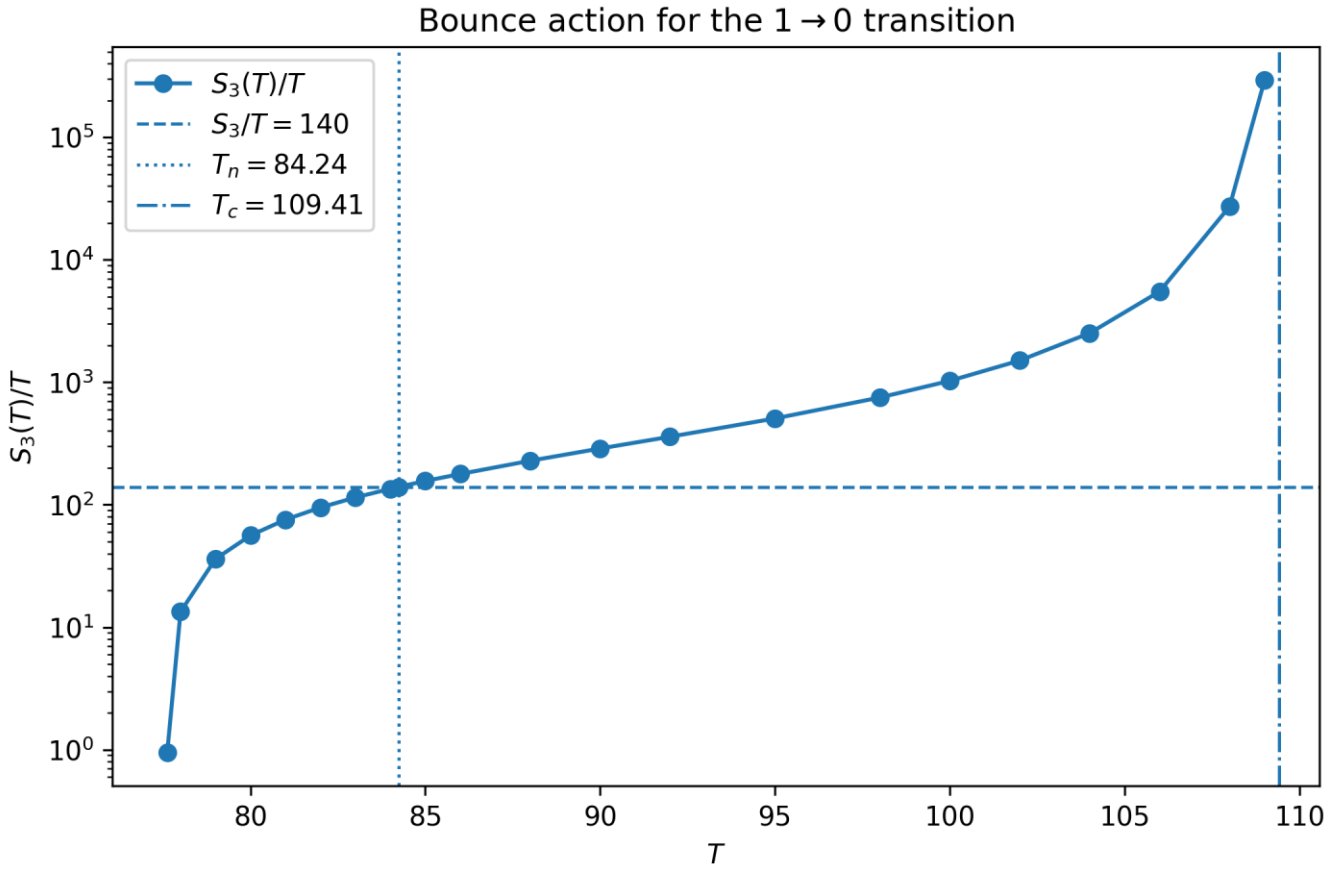
实际发生！

S3/T 曲线

action extraction 的展示页

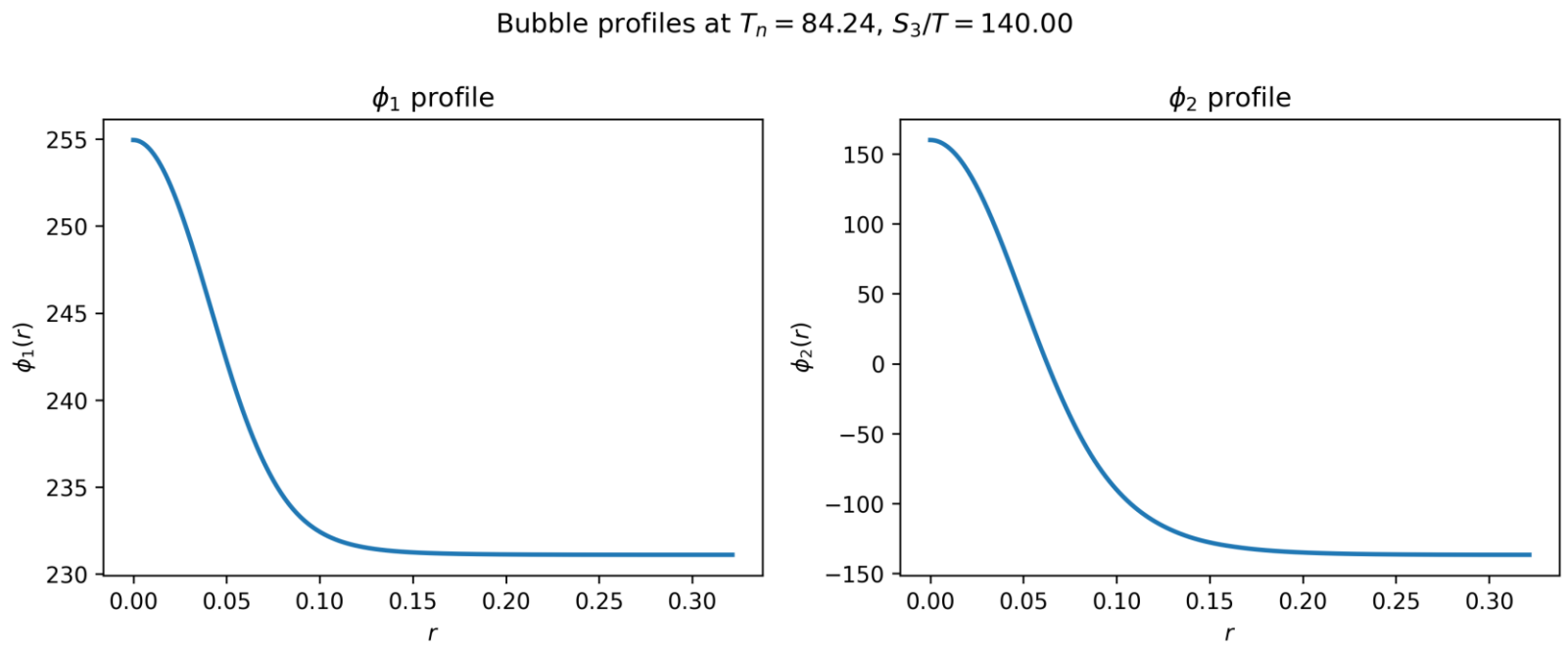
提取作用量曲线

```
from cosmoTransitions import transitionFinder

start_phase = phases[1]

outdict = {}
nuclCriterion = lambda S, T: S / (T + 1e-100) - 140.0

args = (phases,start_phase,m.Vtot,m.gradV,1e-8,45.0,
    nuclCriterion,{},False, outdict,)

for T in T_values:
    transitionFinder._tunnelFromPhaseAtT(float(T), *args)
    data = outdict[float(T)]
    S3 = float(data["action"])
    S_over_T.append(S3 / float(T))
```



bubble profile

把 instanton 可视化



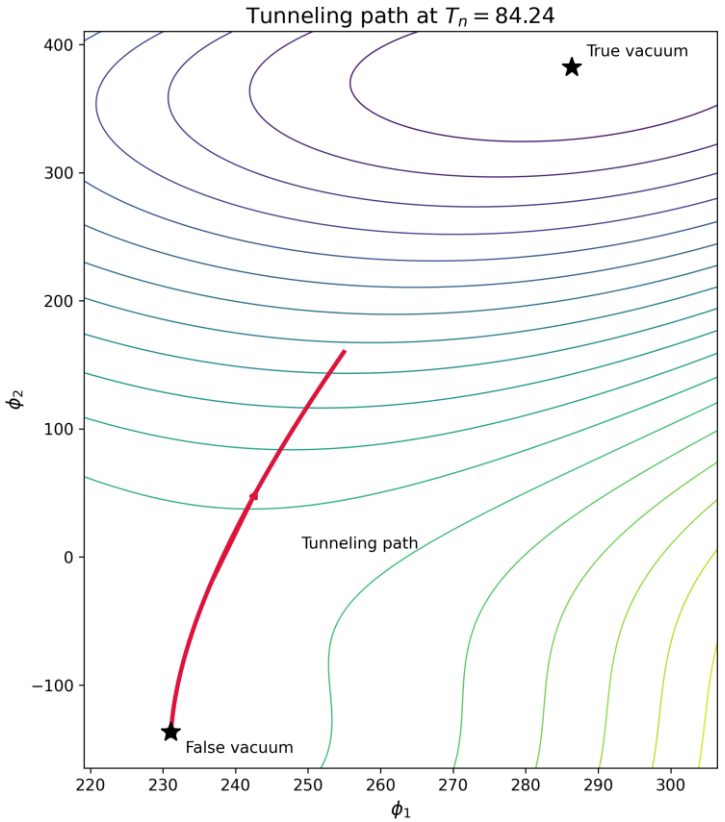

```
from cosmoTransitions import pathDeformation as pd
import numpy as np

# 固定在成核温度
T = Tn

# true vacuum / bubble 内部
low_vev = tr["low_vev"]

# false vacuum / bubble 外部
high_vev = tr["high_vev"]

# 初始路径：必须从 low minimum 指向 high minimum
path_pts = np.linspace(low_vev, high_vev, 50)

# fullTunneling 需要固定温度下的 V(X) 和 dV(X)
V = lambda X: m.Vtot(X, T)
dV = lambda X: m.gradV(X, T)

instanton = pd.fullTunneling(
    path_pts,
    V,
    dV,
    verbose=True
)
```

Gauss type profile – thick wall!

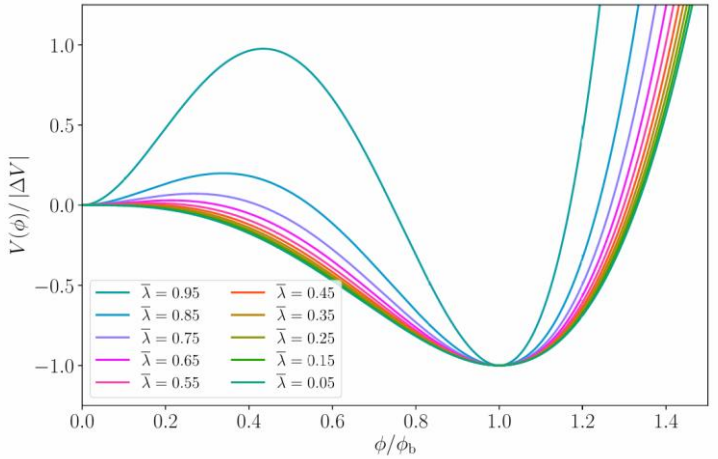
两场 tunneling path

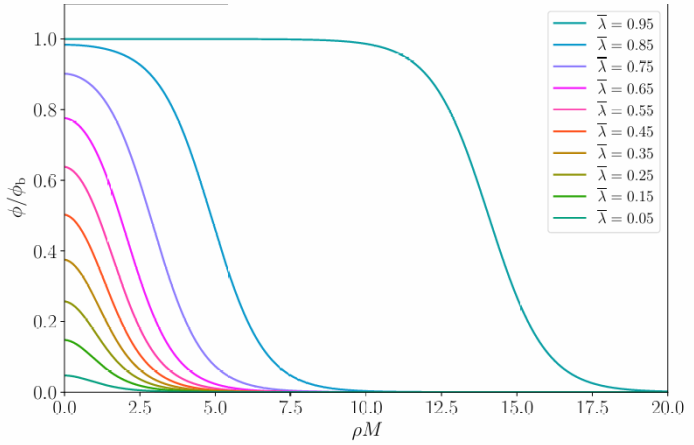
展示 CosmoTransitions 特色

Tunneling path



Bubble内部距离真真空越远，越是厚壁情况！





PTagent：相变分析流程的 agent 化

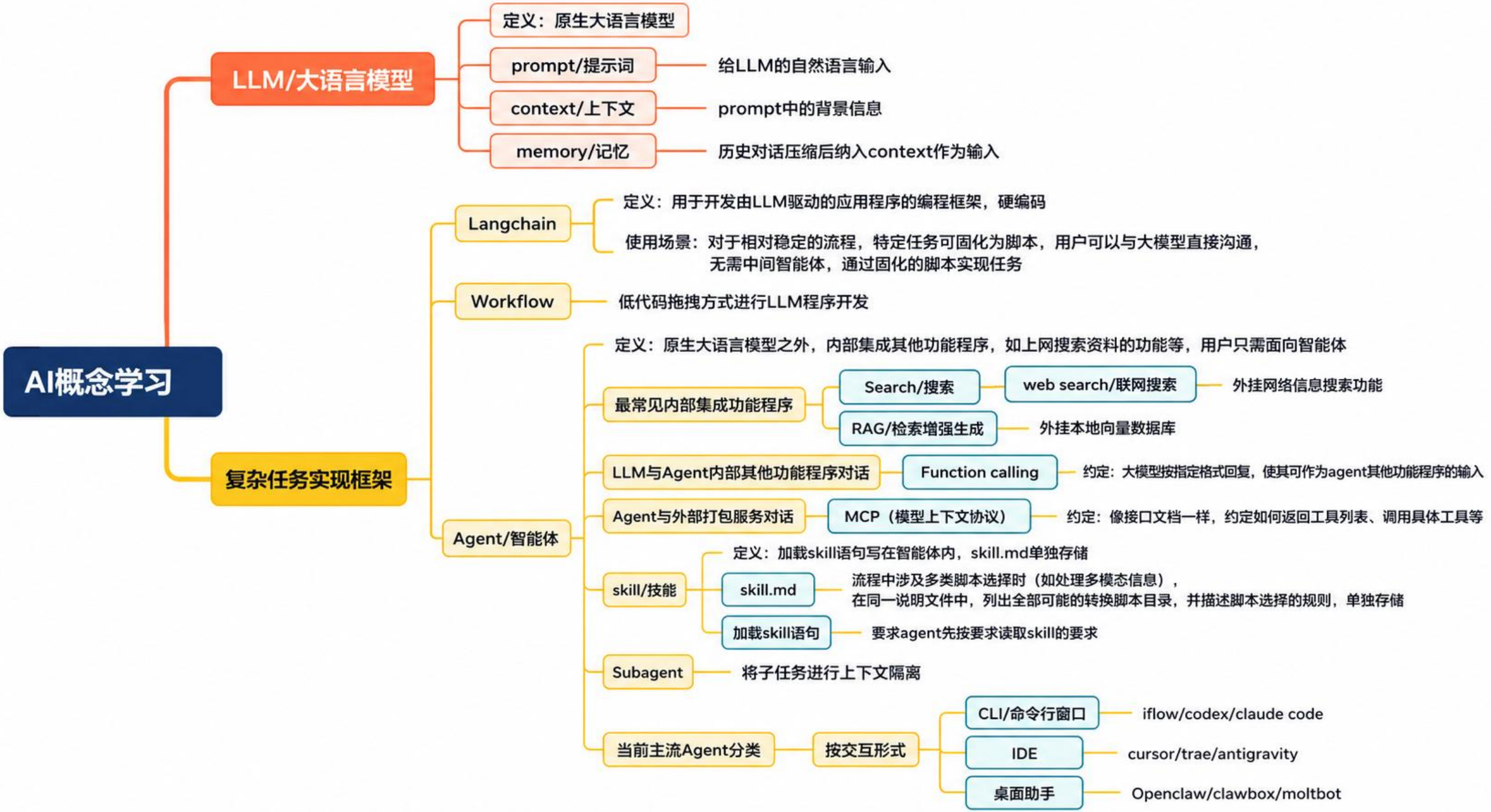
作为 CosmoTransitions 工作流的上层封装 / 自动化层

本章回答

Ptagent辅助相变研究分析

https://github.com/Contract-Mediated-Agent/PTagent/

- 工具编排
- 参数扫描
- 结果诊断
- 报告生成


AI概念学习
LLM/大语言模型
定义：原生大语言模型
prompt/提示词
给LLM的自然语言输入
context/上下文
prompt中的背景信息
memory/记忆
历史对话压缩后纳入context作为输入
复杂任务实现框架
Langchain
定义：用于开发由LLM驱动的应用程序的编程框架，硬编码
使用场景：对于相对稳定的流程，特定任务可固化为脚本，用户可以与大模型直接沟通，
无需中间智能体，通过固化的脚本实现任务
Workflow
低代码拖拽方式进行LLM程序开发
Agent/智能体
定义：原生大语言模型之外，内部集成其他功能程序，如上网搜索资料的功能等，用户只需面向智能体
最常见内部集成功能程序
Search/搜索
web search/联网搜索
外挂网络信息搜索功能
RAG/检索增强生成
外挂本地向量数据库
LLM与Agent内部其他功能程序对话
Function calling
约定：大模型按指定格式回复，使其可作为agent其他功能程序的输入
Agent与外部打包服务对话
MCP（模型上下文协议）
约定：像接口文档一样，约定如何返回工具列表、调用具体工具等
skill/技能
定义：加载skill语句写在智能体内，skill.md单独存储
skill.md
流程中涉及多类脚本选择时（如处理多模态信息），
在同一说明文件中，列出全部可能的转换脚本目录，并描述脚本选择的规则，单独存储
加载skill语句
要求agent先按要求读取skill的要求
Subagent
将子任务进行上下文隔离
当前主流Agent分类
按交互形式
CLI/命令行窗口
iflow/codex/claude code
IDE
cursor/trae/antigravity
桌面助手
Openclaw/clawbox/moltbot

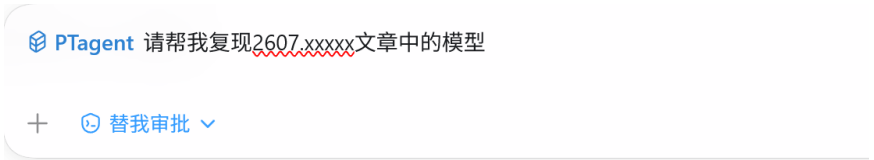
PTagent 的定位

基座大模型发展趋势 → 高能计算软件发展趋势

PTagent 与CosmoTransitions 的关系

氛围科研的特点

Agent 负责流程，数值引擎负责物理计算核心
用户负责检查



Agent搜集信息，模糊之处提问用户

用户确认后，直接编写模型文件！