Bounce solution from gradient flow equation

Ryosuke Sato

Deutsches Elektronen-Synchrotron (DESY)

Phys.Rev.D101(2020)016012, arXiv:1907.02417
arXiv:1908.10868
https://github.com/rsato64/SimpleBounce

Take home message

1. **SimpleBounce : fast code to calculate bounce solution & action for false vacuum decay.**

2. **The code is available at GitHub**

   (https://github.com/rsato64/simplebounce)

3. **Please use it!**

Code is available at GitHub

https://github.com/rsato64/simplebounce

If you are familiar with git, just type

> git clone https://github.com/rsato64/SimpleBounce.git

Fast

model	$\mathcal{S}_E$ SB	$\mathcal{S}_E$ CT	Time[s] SB	Time[s] CT
#1 in Tab. 1 of [8]	52.4	52.6	0.04	0.05
#2 in Tab. 1 of [8]	20.8	21.1	0.05	0.35
#3 in Tab. 1 of [8]	22.0	22.0	0.06	0.17
#4 in Tab. 1 of [8]	55.8	56.1	0.08	0.31
#5 in Tab. 1 of [8]	16.4	16.4	0.09	0.26
#6 in Tab. 1 of [8]	24.5	24.5	0.10	0.25
#7 in Tab. 1 of [8]	36.7	36.7	0.12	0.22
#8 in Tab. 1 of [8]	46.0	46.1	0.13	0.23
Eq. 40 of [19]	1.08×10^3	1.09×10^3	0.03	0.10
Eq. 41 of [19]	6.62	6.65	0.04	0.06
Eq. 42 of [19]	1.75×10^3	1.77×10^3	0.35	0.53
Eq. 43 of [19]	4.46	4.50	0.05	0.18

Table 1: The comparison with `CosmoTransitions`. For `CosmoTransitions`, we take `fRatioConv` as 0.02. The bounce action $\mathcal{S}_E$ is calculated for $d = 3$. The runtimes are measured by Thinkpad X250 with Ubuntu 16.04, whose CPU is Intel®Core™i7-5600U (2.60 GHz) and compiler is GCC version 5.4.0. We take `-O3` as the optimization option.

For models with 1-8 scalar field(s),

- Bounce action with O(0.1) % accuracy can be obtained within O(0.1) s.
- Faster than other public codes.

Easy

sample1.cc (~/SimpleBounce) - gedit

File Edit View Search Tools Documents Help

Open Save

```
#include<iostream>
#include"simplebounce.h"
using namespace std;
using namespace simplebounce;

class MyModel : public GenericModel{
  public:
        MyModel(){
                // number of scalar field(s)
                setNphi(1);
        }
        // potential for scalar field(s)
        double vpot (const double* phi) const{
                return phi[0]*phi[0]/2. - phi[0]*phi[0]*phi[0]/3.;
        }
        // first derivative(s) of potential
        void calcDvdphi(const double *phi, double *dvdphi) const{
                dvdphi[0] = phi[0] - phi[0]*phi[0];
        }
};

int main() {

        BounceCalculator bounce;
        bounce.verboseOn(); // verbose mode
        bounce.setRmax(1.); // phi(rmax) = phi(False vacuum)
        bounce.setDimension(4); // number of space dimension
        bounce.setN(100); // number of grid
        MyModel model;
        bounce.setModel(&model);

        double phiTV[1] = {10.}; // a point at which V<0
        double phiFV[1] = {0.}; // false vacuum
        bounce.setVacuum(phiTV, phiFV);


        // calcualte the bounce solution
        bounce.solve();

        // show the results
        bounce.printBounce();

        // show the Euclidean action
        cout << "S_E = " << bounce.action() << endl;

        return 0;
}
```

C++ Tab Width: 8 Ln 1, Col 1 INS

Number of scaler field

$V(\phi)$

$\partial V/\partial \phi_i$

Dimension

A point around $\phi_{true\ vac.}$

$\phi_{false\ vac.}$

Easy

sample1.cc (~/SimpleBounce) - gedit

File Edit View Search Tools Documents Help

Open

```
#include<iostream>
#include"simplebounce.h"
using namespace std;
using namespace simplebounce;

class MyModel : public GenericModel{
  public:
        MyModel(){
                // number of scalar field(s)
                setNphi(1);
        }
        // potential for scalar field(s)
        double vpot (const double* phi) const{
                return phi[0]*phi[0]/2. - phi[0]*phi[0]*phi[0]/3.;
        }
        // first derivative(s) of potential
        void calcDvdphi(const double *phi, double *dvdphi) const{
                dvdphi[0] = phi[0] - phi[0]*phi[0];
        }
};

int main() {

        BounceCalculator bounce;
        bounce.verboseOn(); // verbose mode
        bounce.setRmax(1.); // phi(rmax) = phi(False vacuum)
        bounce.setDimension(4); // number of space dimension
        bounce.setN(100); // number of grid
        MyModel model;
        bounce.setModel(&model);

        double phiTV[1] = {10.}; // a point at which V<0
        double phiFV[1] = {0.}; // false vacuum
        bounce.setVacuum(phiTV, phiFV);


        // calcualte the bounce solution
        bounce.solve();

        // show the results
        bounce.printBounce();

        // show the Euclidean action
        cout << "S_E = " << bounce.action() << endl;

        return 0;
}
```

C++ Tab Width: 8

Num

∂V

Dime

A po

ϕ_{fals}

rsato@rsato-ThinkPad-X250: ~/SimpleBounce

File Edit View Search Terminal Help

```
# r      phi[0]  RHS of EOM[0]
0       8.73836 -0.00666804
0.0533292       8.71432 -0.00661343
0.106658        8.64265 -0.00645129
0.159988        8.52505 -0.006188
0.213317        8.36421 -0.00583337
0.266646        8.16369 -0.00540022
0.319975        7.92776 -0.00490351
0.373304        7.66123 -0.00435945
0.426633        7.36924 -0.00378458
0.479963        7.05704 -0.00319494
0.533292        6.72983 -0.00260538
0.586621        6.39261 -0.00202903
0.63995 6.05004 -0.00147695
0.693279        5.70635 -0.000957962
0.746609        5.3653  -0.00047865
0.799938        5.03009 -4.34475e-05
0.853267        4.70341 0.000345153
0.906596        4.38745 0.00068634
0.959925        4.08388 0.000980698
1.01325 3.79396 0.00122992
1.06658 3.51852 0.00143651
1.11991 3.25808 0.00160356
1.17324 3.01284 0.00173452
1.22657 2.78277 0.001833
1.2799  2.56762 0.00190266
1.33323 2.36702 0.00194707
1.38656 2.18046 0.00196961
1.43989 2.00733 0.00197349
1.49322 1.84698 0.00196161
1.54655 1.69872 0.00193665
1.59988 1.56184 0.00190096
1.6532  1.43562 0.00185667
1.70653 1.31936 0.00180562
1.75986 1.21238 0.0017494
1.81319 1.114   0.0016894
1.86652 1.02359 0.0016268
1.91985 0.940556        0.00156258
1.97318 0.864319        0.00149757
2.02651 0.794348        0.00143247
2.07984 0.730141        0.00136784
2.13317 0.671234        0.00130412
2.1865  0.617194        0.00124169
```

How does it work?

Coleman, Glaser, Martin (1977)
RS (2019)

The bounce solution

$$S_E[\phi] = T[\phi] + V[\phi]$$

$$T[\phi] = \int d^d x \frac{1}{2}(\nabla\phi)^2$$

$$V[\phi] = \int d^d x\, U(\phi)$$

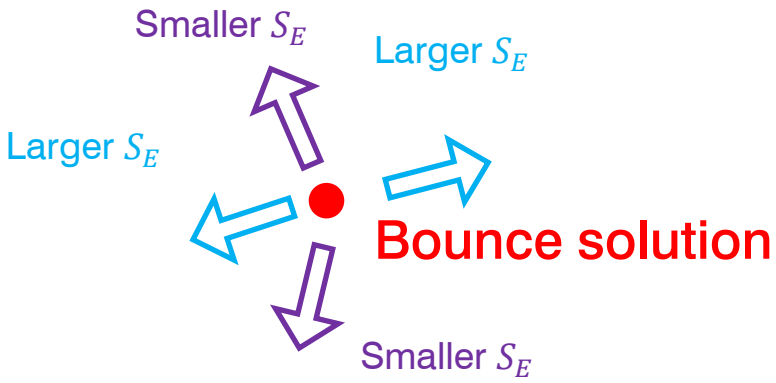
$$-\nabla^2\phi + \frac{\partial U}{\partial \phi} = 0$$

The bounce

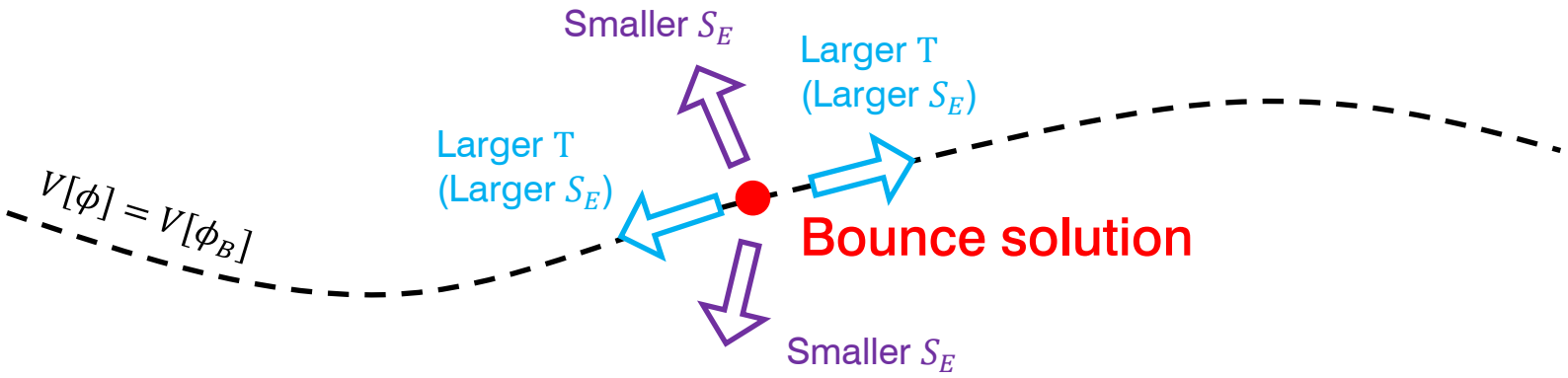
- Is a non-trival solution of EOM
- has the least $S_E[\phi]$ among solutions of EOM.
- is a saddle point of $S_E[\phi]$ under small perturbation

$$S_E[\phi] = S_E[\phi_B] + \frac{1}{2}\int d^d x\, \delta\phi\, M\, \delta\phi + \cdots$$

M has one and only one negative eigenvalue
[Coleman (1988)]



Minimization of S_E does not work.



Minimization of S_E does not work.

The bounce solution minimizes $T[\phi]$ in $V[\phi] = V[\phi_B]$ plane.

[Coleman, Glaser, Martin (1977)]

$T[\phi] = \int d^d x \frac{1}{2} (\nabla\phi)^2$

$V[\phi] = \int d^d x \, U(\phi)$

Reduced problem

- **Freezing unstable direction by hand**

  The unstable direction can be frozen by imposing constraint on $V[\phi]$

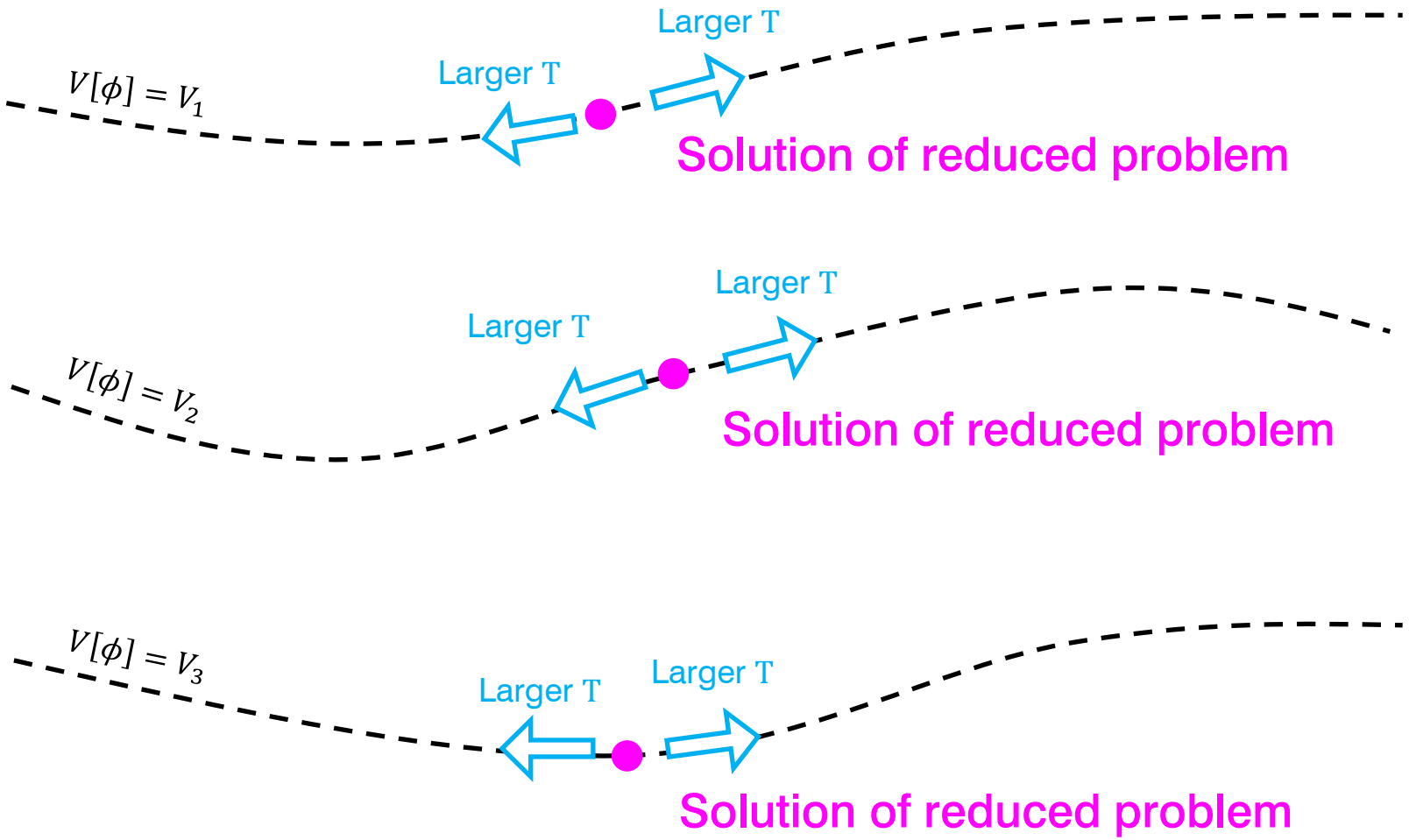
  → Solving the bounce sol. ⇔ minimizing $T[\phi]$

- **Reduced problem** [Coleman, Glaser, Martin (1977)]

  Minimize kinetic energy $T[\phi]$ while fixing potential energy $V[\phi] = V_0 < 0$

  The existence of the solution is guaranteed in wide class of models.

  [Coleman, Glaser, Martin (1977)]
  [Brezis, Lieb (1984)]



We do not know value of $V[\phi_B]$ a priori.
Which one is the bounce??

Scale transformation & reduced problem

[Coleman, Glaser, Martin (1977)]

All of the solutions of the reduced problem is connected by scale transformation.

Let ϕ_{V_1} be the solution of reduced problem with $V[\phi] = V_1$

$V[\phi] = V_1$ ϕ_{V_1}

$V[\phi] = V_2$ ϕ_{V_2}

Scale transformation & reduced problem

[Coleman, Glaser, Martin (1977)]

All of the solutions of the reduced problem is connected by scale transformation.

Let ϕ_{V_1} be the solution of reduced problem with $V[\phi] = V_1$

$V[\phi] = V_1$

ϕ_{V_1}

Scale transformation

$V[\phi] = V_2$

$\tilde{\phi}_2(r) \equiv \phi_{V_1}(\lambda r)$

$\tilde{\phi}_1(r) \equiv \phi_{V_2}(\lambda^{-1} r)$

Scale transformation

ϕ_{V_2}

$$\lambda \equiv \left(\frac{V_2}{V_1}\right)^{-1/d}$$

Scale transformation & reduced problem

[Coleman, Glaser, Martin (1977)]

All of the solutions of the reduced problem is connected by scale transformation.

Let ϕ_{V_1} be the solution of reduced problem with $V[\phi] = V_1$

$V[\phi] = V_1$

ϕ_{V_1}

Scale transformation

$V[\phi] = V_2$

$\tilde{\phi}_2(r) \equiv \phi_{V_1}(\lambda r)$

$\tilde{\phi}_1(r) \equiv \phi_{V_2}(\lambda^{-1} r)$

Scale transformation

ϕ_{V_2}

$$\lambda \equiv \left(\frac{V_2}{V_1}\right)^{-1/d}$$

$$T[\phi_{V_1}] \leq T[\tilde{\phi}_1]$$

$$T[\phi_{V_2}] \leq T[\tilde{\phi}_2]$$

Scale transformation & reduced problem

[Coleman, Glaser, Martin (1977)]

All of the solutions of the reduced problem is connected by scale transformation.

Let ϕ_{V_1} be the solution of reduced problem with $V[\phi] = V_1$

$V[\phi] = V_1$ $\quad \phi_{V_1}$ $\quad \tilde{\phi}_1(r) \equiv \phi_{V_2}(\lambda^{-1} r)$

Scale transformation $\quad$ Scale transformation

$V[\phi] = V_2$ $\quad \tilde{\phi}_2(r) \equiv \phi_{V_1}(\lambda r)$ $\quad \phi_{V_2}$

$$\lambda \equiv \left(\frac{V_2}{V_1}\right)^{-1/d}$$

$$T[\phi_{V_1}] \leq T[\tilde{\phi}_1]$$

$$\lambda^{-d+2} T[\tilde{\phi}_1] = T[\phi_{V_2}] \leq T[\tilde{\phi}_2] = \lambda^{-d+2} T[\phi_{V_1}] \quad \Rightarrow \quad T[\tilde{\phi}_1] \leq T[\phi_{V_1}]$$

Scale transformation & reduced problem

[Coleman, Glaser, Martin (1977)]

All of the solutions of the reduced problem is connected by scale transformation.

Let ϕ_{V_1} be the solution of reduced problem with $V[\phi] = V_1$

$V[\phi] = V_1$ $\quad \phi_{V_1}$ $\quad \tilde{\phi}_1(r) \equiv \phi_{V_2}(\lambda^{-1} r)$

Scale transformation $\quad$ Scale transformation

$V[\phi] = V_2$ $\quad \tilde{\phi}_2(r) \equiv \phi_{V_1}(\lambda r)$ $\quad \phi_{V_2}$

$$\lambda \equiv \left(\frac{V_2}{V_1}\right)^{-1/d}$$

$$T[\phi_{V_1}] \leq T[\tilde{\phi}_1]$$

$$\lambda^{-d+2} T[\tilde{\phi}_1] = T[\phi_{V_2}] \leq T[\tilde{\phi}_2] = \lambda^{-d+2} T[\phi_{V_1}] \quad \Rightarrow \quad T[\tilde{\phi}_1] \leq T[\phi_{V_1}]$$

$$T[\phi_{V_1}] = T[\tilde{\phi}_1] \qquad T[\phi_{V_2}] = T[\tilde{\phi}_2]$$

$\tilde{\phi}_1$ and $\tilde{\phi}_2$ are also solution of the reduce problem.

Scale transformation & reduced problem

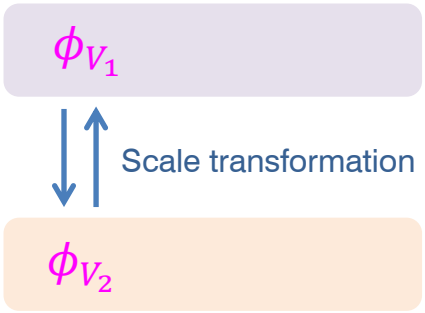
[Coleman, Glaser, Martin (1977)]

All of the solutions of the reduced problem is connected by scale transformation.

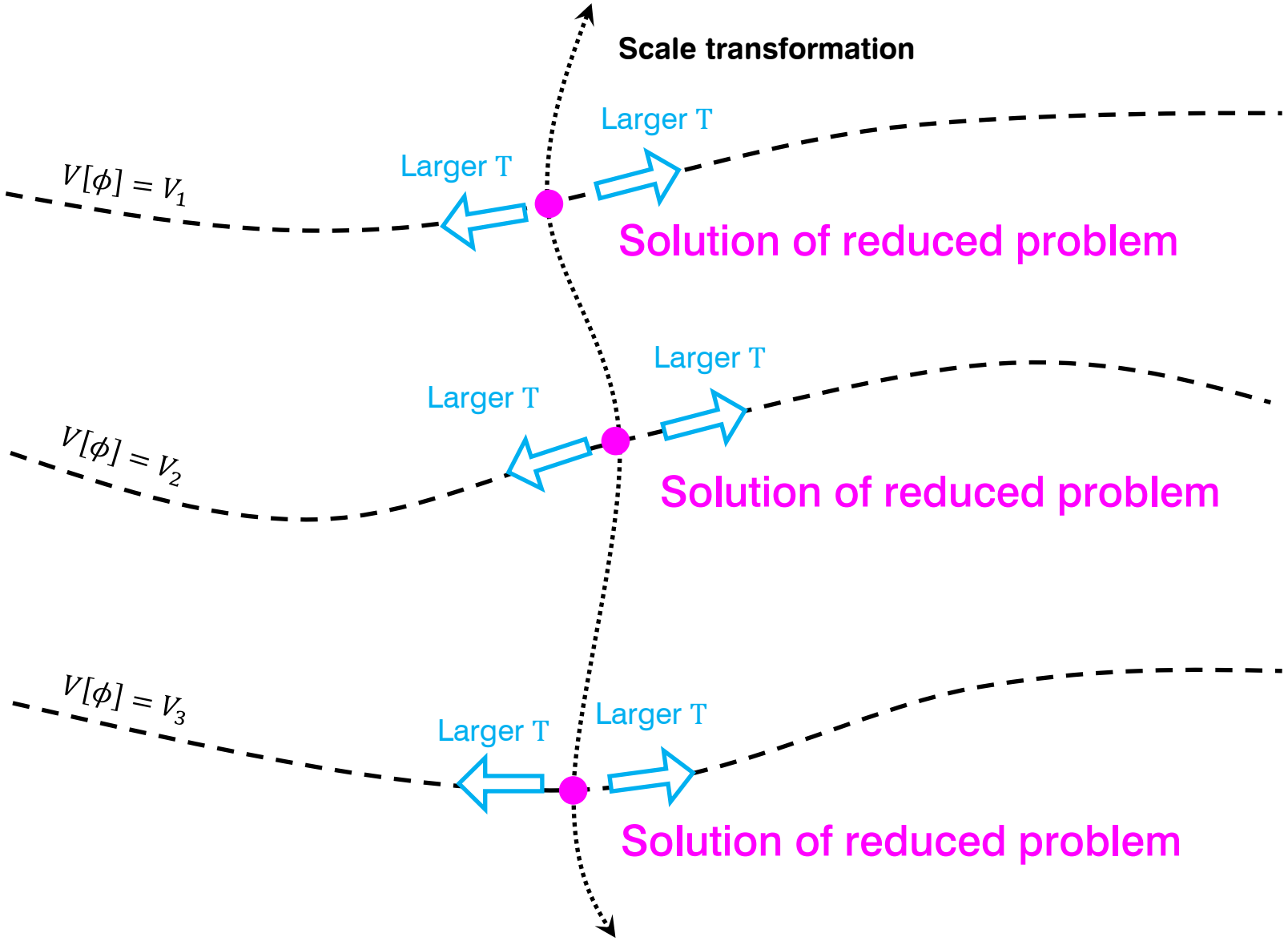
Let ϕ_{V_1} be the solution of reduced problem with $V[\phi] = V_1$

$V[\phi] = V_1$

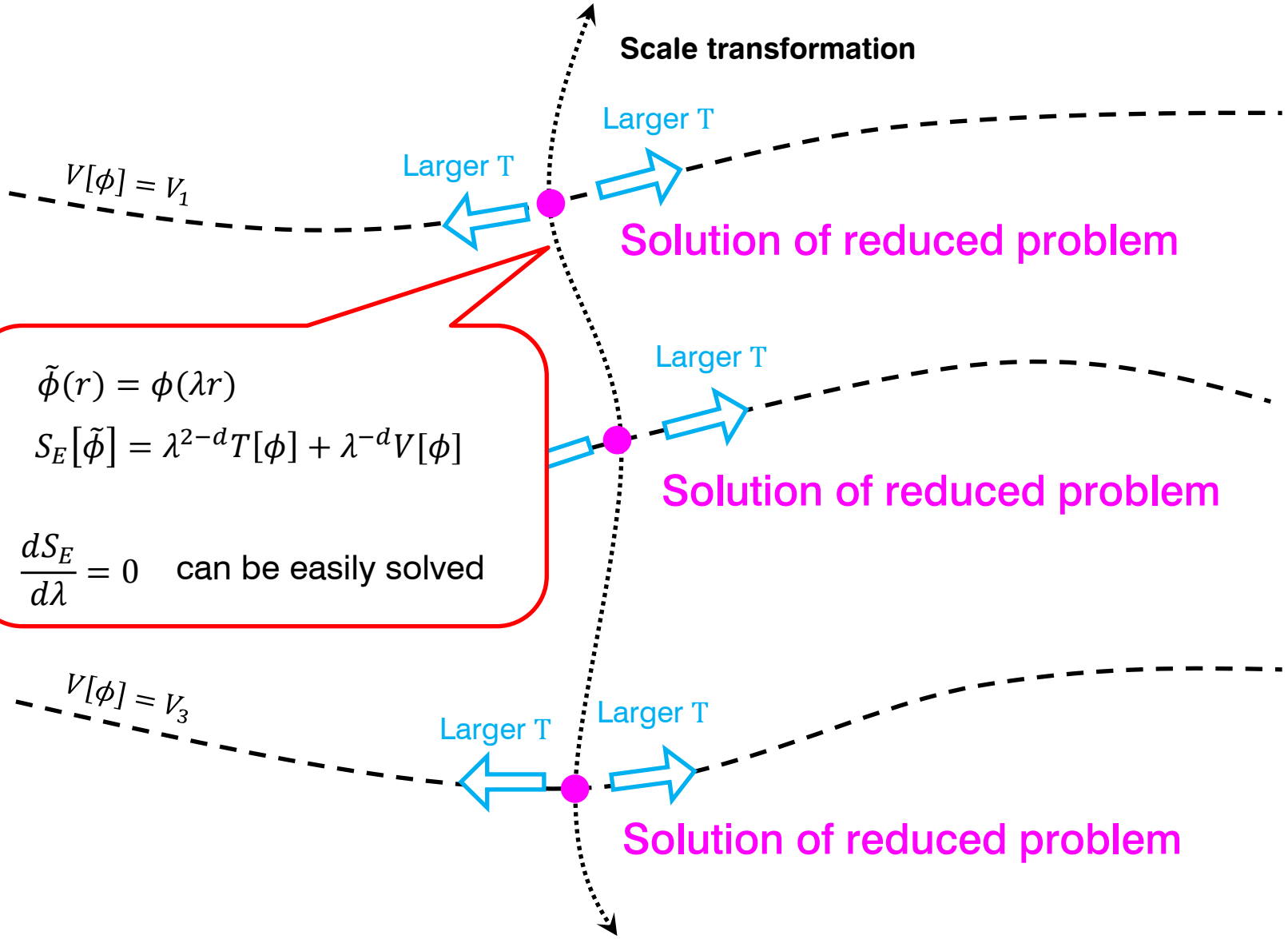
$V[\phi] = V_2$



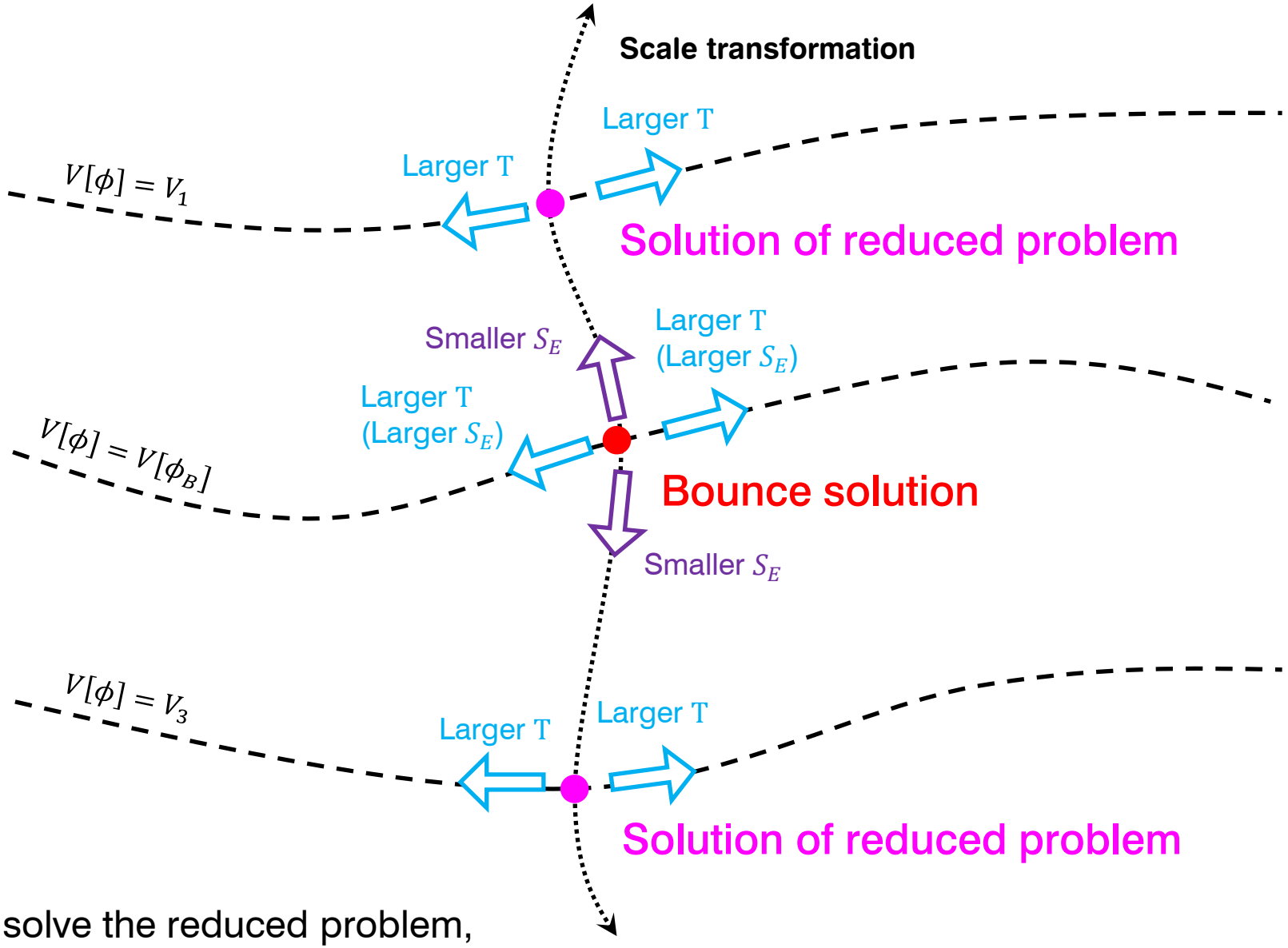
ϕ_{V_1} and ϕ_{V_2} are connected by scale transformation!



All of magenta points are connected by scale transformation.



All of magenta points are connected by scale transformation.



If we solve the reduced problem,
we can easily get the bounce solution.

Solving reduced problem from gradient flow

- **Reduced problem**

  Minimize kinetic energy $T[\phi]$ while fixing potential energy $V[\phi] = V_0 < 0$

- **Gradient flow equation** [See also Chigusa, Moroi, Shoji (2019)]

  What I want : $\frac{\partial}{\partial\tau}\varphi(r,\tau) = F(r,\varphi)$ such that $\frac{\partial}{\partial\tau}T[\varphi] \leq 0,$ $\frac{\partial}{\partial\tau}V[\varphi] = 0$

An answer : $\frac{\partial}{\partial\tau}\varphi(r,\tau) = \nabla^2\varphi - \lambda[\varphi]\frac{\partial V}{\partial\phi},$ $\lambda[\varphi] = \frac{\int_0^\infty dr\, r^{d-1}\frac{\partial V}{\partial\phi}\nabla^2\phi}{\int_0^\infty dr\; r^{d-1}\left(\frac{\partial V}{\partial\phi}\right)^2}$

Algorithm

1. Take initial condition $\varphi(r, \tau = 0) = \phi(r)$ such that $V[\phi] < 0$

2. Solve $$\frac{\partial}{\partial \tau}\varphi(r,\tau) = \nabla^2\varphi - \lambda[\varphi]\frac{\partial V}{\partial \phi}, \qquad \lambda[\varphi] = \frac{\int_0^\infty dr\, r^{d-1}\frac{\partial V}{\partial \phi}\nabla^2\phi}{\int_0^\infty dr\ r^{d-1}\left(\frac{\partial V}{\partial \phi}\right)^2}$$

3. $\phi_B(r) = \varphi(\lambda r, \tau = \infty)$ λ can be obtained from $\frac{d}{d\lambda}S[\varphi(\lambda r, \tau = \infty)] = 0$

Take home message

1. **SimpleBounce : fast code to calculate bounce solution & action for false vacuum decay.**

2. **The code is available at GitHub**

   (https://github.com/rsato64/simplebounce)

3. **Please use it!**

Backup

Unstable direction & scale transf.

Let us define $\phi_\lambda(r) \equiv \phi_B(\lambda r)$

$$S[\phi_\lambda] = \lambda^{2-d} T[\phi_B] + \lambda^{-d} V[\phi_B]$$

$$\left.\frac{dS}{d\lambda}\right|_{\lambda=1} = (2-d)T[\phi_B] - dV[\phi_B] = 0$$

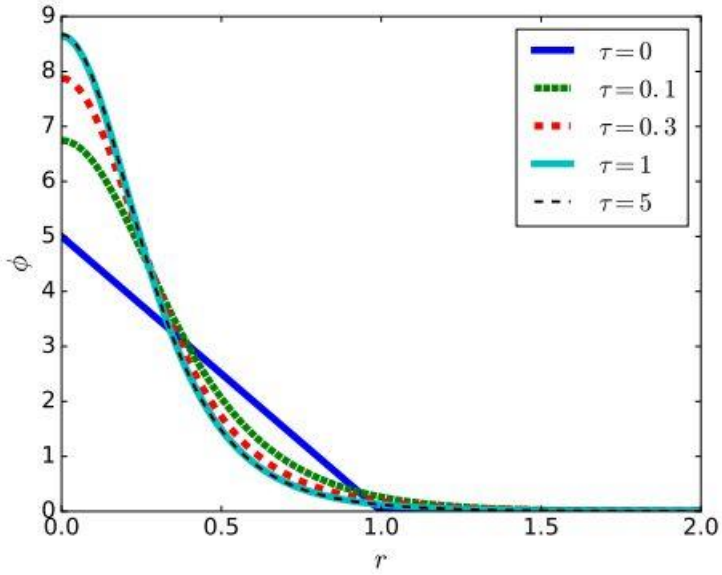
$$\left.\frac{d^2S}{d\lambda^2}\right|_{\lambda=1} = 2(2-d)T[\phi_B] < 0$$

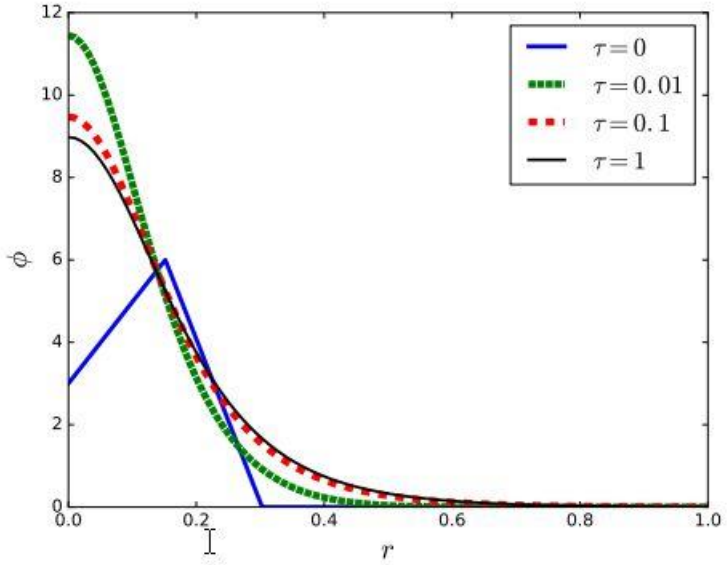
unstable!

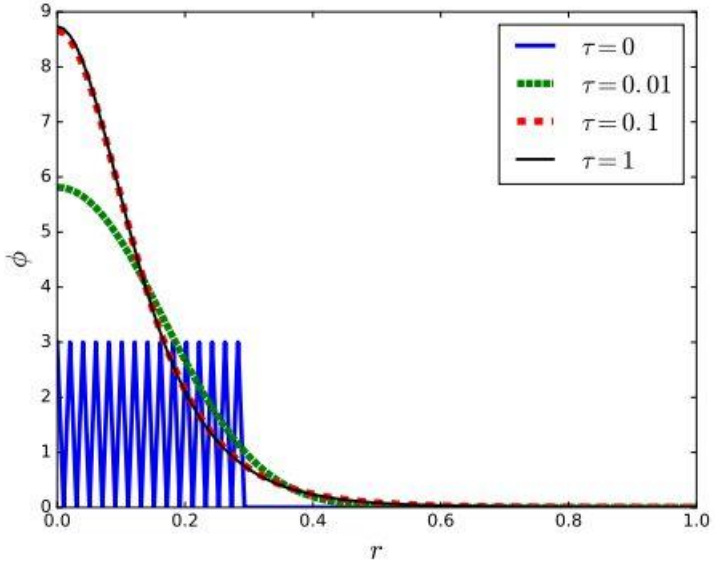
$$T[\phi] = \int d^d x \frac{1}{2}(\nabla\phi)^2$$

$$V[\phi] = \int d^d x V(\phi)$$

Stablity against initial condition







How to get gradient flow equation

- **Reduced problem**

  Minimize kinetic energy $T[\phi]$ while fixing potential energy $V[\phi] = V_0 < 0$

- **Gradient flow equation**

  What I want : $\frac{\partial}{\partial\tau}\varphi(r,\tau) = F(r,\varphi)$ such that $\frac{\partial}{\partial\tau}T[\varphi] \leq 0,$ $\frac{\partial}{\partial\tau}V[\varphi] = 0$

  Let's try this! $\frac{\partial}{\partial\tau}\varphi(r,\tau) = \nabla^2\varphi - \lambda[\varphi]\frac{\partial V}{\partial\phi}$

  $\frac{\partial}{\partial\tau}V[\varphi] = 0$ ⇨ $\int d^dx \left[\frac{\partial V}{\partial\phi}\nabla^2\varphi - \lambda[\varphi]\left(\frac{\partial V}{\partial\phi}\right)^2\right] = 0$

How to get gradient flow equation

- **Reduced problem**

  Minimize kinetic energy $T[\phi]$ while fixing potential energy $V[\phi] = V_0 < 0$

- **Gradient flow equation**

  What I want : $\frac{\partial}{\partial\tau}\varphi(r,\tau) = F(r,\varphi)$ such that $\frac{\partial}{\partial\tau}T[\varphi] \le 0,$ $\frac{\partial}{\partial\tau}V[\varphi] = 0$

  Let's try this! $\frac{\partial}{\partial\tau}\varphi(r,\tau) = \nabla^2\varphi - \lambda[\varphi]\frac{\partial V}{\partial\phi}$

  $\frac{\partial}{\partial\tau}V[\varphi] = 0$ ⇨ $\lambda[\varphi] = \frac{\int_0^\infty dr\, r^{d-1}\frac{\partial V}{\partial\phi}\nabla^2\phi}{\int_0^\infty dr\; r^{d-1}\left(\frac{\partial V}{\partial\phi}\right)^2}$

  We can also verify $\frac{\partial}{\partial\tau}T[\varphi] \le 0$

Bounce is a solution of reduced problem

If the bounce is NOT solution of the reduced problem, there exists ϕ such that

$$T[\phi] < T[\phi_B], \qquad V[\phi] = V[\phi_B]$$

$$-\nabla^2\phi + \lambda\frac{\partial V}{\partial \phi} = 0 \qquad (d-2)T[\phi] + \lambda dV[\phi] = 0$$

$$(d-2)T[\phi_B] + dV[\phi_B] = 0 \qquad \lambda < 1$$

$$(d-2)T[\phi] + dV[\phi] < 0$$

$$\tilde{\phi}(x) = \phi(\lambda^{-1/2}x)$$

$$T\left[\tilde{\phi}\right] = \lambda^{(d-2)/2}T[\phi] < T[\phi_B] \qquad \text{inconsisent}$$

The bounce IS a solution of the reduced problem

Coleman-Glaser-Martin theorem A

- ϕ_{V_0} can be scale transformed to a solution of EOM ϕ_B

ϕ_{V_0} stationalizes $T[\phi] + \lambda(V[\phi] - V_0)$ ⇨

$$-\nabla^2\phi_{V_0} + \lambda\frac{\partial V}{\partial\phi}\Big|_{\phi=\phi_{V_0}} = 0$$

$$(d-2)T[\phi_{V_0}] + d\lambda V[\phi_{V_0}] = 0 \longrightarrow \lambda > 0$$

Let us define $\phi_B(r) \equiv \phi_{V_0}(\lambda^{-1/2}r)$ ⇨ $-\nabla^2\phi_B + \frac{\partial V}{\partial\phi}\Big|_{\phi=\phi_B} = 0$

- ϕ_B has the least action among the solutions of EOM.

ϕ_B satisfies EOM → $(d-2)T[\phi_B] + dV[\phi_B] = 0$ →

Let $\tilde{\phi}$ be a solution of EOM. → $(d-2)T[\tilde{\phi}] + dV[\tilde{\phi}] = 0$

$\phi_B(r) = \phi_{V_0}(\lambda^{-1/2}r)$ → $\lambda^{d/2-1}(d-2)T[\phi_{V[\tilde{\phi}]}] + \lambda^{d/2}dV[\tilde{\phi}] = 0$

$T[\phi_{V[\tilde{\phi}]}] \le T[\tilde{\phi}]$ $\lambda^{d/2-1}(d-2)T[\tilde{\phi}] + \lambda^{d/2}dV[\tilde{\phi}] \ge 0$

$\lambda < 1$

$T[\phi_B] = \lambda^{d/2-1}T[\phi_{V[\tilde{\phi}]}] \le T[\phi_{V[\tilde{\phi}]}] \le T[\tilde{\phi}]$

$S_E[\phi_B] \le S_E[\tilde{\phi}]$ because $S_E = \frac{2}{d}T$