False Vacuum Decay

Exact Computation of the Classical Bounce

A thesis submitted to The Department of Mathematics and Natural Sciences, BRAC University, in partial fulfilment of the requirements for the degree of Bachelor of Science in Physics

Submitted by

M M Tarique Hasan

Physics

Department of Mathematics and Natural Sciences

BRAC University

January 2016

DECLARATION

I do hereby declare that the thesis titled "False Vacuum Decay - Exact Computation of the Classical Bounce" is submitted to the Department of Mathematics and Natural Sciences of BRAC University in partial fulfilment of the requirements for the degree of Bachelor of Science in Physics. This research is the work of my own and has not been submitted elsewhere for award of any other degree or diploma. Every work that has been used as reference for this work has been cited properly.

M M Tarique Hasan

ID: 13211002

Candidate

Certified

(Professor Arshad Momen)

Supervisor

Chairperson

Department of Theoretical Physics

University of Dhaka, Dhaka

ACKNOWLEDGEMENTS

Firstly, I would like to express my sincere gratitude to my advisor Professor Arshad Momen for his continuous support of my thesis and for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my BSc thesis.

Besides my advisor, I would like to thank the Chairperson of the Department of Mathematics and Natural Sciences, Professor A. A. Ziauddin Ahmad, for his unwavering support and constant guidance, which has helped me to navigate not only the long, arduous journey of a BSc degree in Physics, but also to learn and grow as a scientist and as a person. I am also thankful to him for encouraging the use of correct grammar and consistent notation in my writings and for carefully reading and commenting on the revisions of this manuscript.

I would also like to thank Professor Mahbub Majumdar for being such a kind and wonderful mentor and for helping me to develop my background in theoretical high-energy physics with his courses in quantum field theory, general relativity and group theory.

I would like to thank Rafid Mahbub and Shadman Salam, who as good friends, were always willing to help and give their best suggestions. It would have been a difficult two years without them.

Last but not the least, I would like to thank my parents for supporting me spiritually throughout writing this thesis and my life in general.

ABSTRACT

This thesis reviews the exact numerical computation of the classical bounce solution in the false vacuum decay for a quartic scalar field potential which is offset by an external source. The decay rate in a semiclassical analysis consists of a leading exponential contribution due to the classical action evaluated on the classical bounce solution as well as a prefactor of functional determinants due to quantum fluctuations about the classical bounce solution. The equation of motion of the classical bounce solution is obtained using the Euler-Lagrange equation and is numerically solved using the shooting method and the finite-difference method. The appendix reviews a computational technique for evaluating the functional determinants of one-dimensional operators via the Gel'fand-Yaglom theorem.

Contents

List of Figures

Chapter 1

Introduction

1.1 A Qualitative Preview of False Vacuum Decay

False vacuum decay is the transition of a scalar field ϕ from a local minimum ϕ_- of the potential energy $U(\phi)$ towards another local minimum ϕ_+ of lower potential energy. This is shown in Figure 1.1 [5, 6]. The local minima at $\phi = \phi_-$ and $\phi = \phi_+$ of the potential $U(\phi)$ are such that $\phi_- < \phi_+$ and $U(\phi_-) > U(\phi_+)$. Therefore, the decay occurs from the scalar field ϕ_- to the scalar field ϕ_+. ϕ_- and ϕ_+ are, therefore, called the false vacuum and the true vacuum, respectively.

During the decay, the scalar field transitions across a potential barrier. Therefore, false vacuum decay is classically forbidden and the process is mediated by quantum tunnelling. The physical mechanism of such a process is the first-order phase transition in bubble

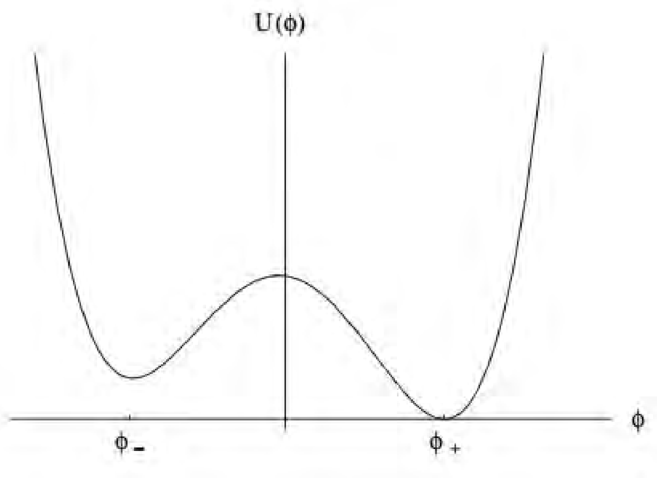

FIGURE 1.1: A scalar field potential $U(\phi)$ with its local minima at $\phi = \phi_-$ and $\phi = \phi_+$.

nucleation - the decay proceeds by the nucleation of expanding bubbles of true vacuum within the metastable false vacuum [5, 6, 12, 13].

The decay rate per unit volume per unit time of the scalar field has been calculated in the semiclassical approximation and consists of, among other factors, a dominant exponential contribution due to the action evaluated on the classical bounce solution.[1] Therefore, the quantitative form of the classical bounce solution for a given scalar field potential is required to compute a certain part of the decay rate per unit volume per unit time of the associated scalar field. The goal of the thesis is to derive an equation of motion of the classical bounce solution (using the least action principle) for a particular quantitative form of the scalar field potential in Figure 1.1 and to illustrate and compare numerical computational methods which can be used to solve for the classical bounce solution.

1.2 Physical Application to the Phenomenological Theory of Cosmic Inflation

The theory of false vacuum decay was used to propose the phenomenological model of cosmic inflation [10]. The theory posits an exponential expansion of space that lasted from 10^{-36} seconds after the Big Bang to sometime between 10^{-33} and 10^{-32} seconds. The hypothetical scalar field thought to be responsible for inflation is called the inflaton field. The key idea of inflation is the following: as the early universe cooled, it was trapped in a false vacuum with a high energy density (it was supercooled), which is much like a cosmological constant. The universe could only decay out of this metastable vacuum through the process of bubble nucleation via quantum tunneling. Therefore, bubbles of true vacuum spontaneously formed in the sea of false vacuum and rapidly began expanding at the speed of light, thereby causing inflation.

1.3 Road-Map of the Thesis

This section presents an overview of the chapters in the thesis.

Chapter 1 introduces the problem of false vacuum decay. In section 1.1, the key idea of quantum tunnelling via bubble nucleation of a scalar field from the false vacuum to the true vacuum is illustrated. In section 1.2, one particular phenomenon - the cosmic inflation - in which the theory of false vacuum decay has been used is then discussed.

[1]The classical bounce solution is the scalar field configuration that results in the extremum of the action.

Chapter 2 shows that the decay rate per unit volume per unit time consists of a dominant exponential contribution due to the action evaluated on the classical bounce solution and a prefactor of functional determinants due to quantum fluctuations about the classical bounce solution. Firstly, the decay rate is derived for a system with finitely many degrees of freedom in sections 2.2 to 2.5. The expression is then generalised to field theory in section 2.6.The calculation of the decay rate for a system with finitely many degrees of freedom in sections 2.2 to 2.5 is further divided into the calculation of the exponential contribution via WKB tunnelling in sections 2.2 and 2.3 and the calculation of the prefactor of functional determinants in sections 2.4 and 2.5.

Chapters 3 and 4 and Appendix A focus exclusively on the computation of the classical bounce solution for a specific quantitative form of the scalar field potential shown in Figure 1.1. Appendix B focuses exclusively on the calculation of the pre-factor of functional determinants for the specific quantitative form of the scalar field potential chosen in Chapter 3.

Chapter 3 derives the equation of motion of the classical bounce solution for a specific quantitative form of the scalar potential shown in Figure 1.1. In section 3.1, the Euclidean classical action is derived from the action in Minkowski space via Wick rotation. In section 3.2, a specific quantitative form of the scalar field potential is chosen. In section 3.3, the equation of motion of the classical bounce solution is derived for the chosen scalar field potential. In section 3.4, as an aside, an approximation technique - the thin-wall limit - used to analytically compute the classical bounce is reviewed.

Chapter 4 numerically computes, using two alternative techniques, the classical bounce solution from its equation of motion. In section 4.1, the shooting method is used to compute the solution. The algorithm and C/C++ implementation of the algorithm are discussed. In section 4.2, the finite-difference method is used to compute the solution. The algorithm and Mathematica implementation of the algorithm are discussed. In section 4.3, the results are discussed and the two numerical methods are compared with each other.

Appendix A presents the source code for the C/C++ implementation of the shooting method and the Mathematica implementation of the finite-difference method for the numerical computation of the classical bounce solution from its equation of motion.

Appendix B reviews the procedure to compute the ratio of determinants of one-dimensional operators - the so-called Gel'fand Yaglom theorem. This is only the first step in the calculation of the pre-factor of functional determinants for the specific quantitative form of the scalar field potential chosen in Chapter 3. Subsequent steps in the calculation are discussed in [9].

Chapter 2

Euclidean Solutions

2.1 Introduction

The crux of this chapter is Equation 2.43, which defines the decay width of false vacuum. This formula is derived using a semiclassical approximation - solutions of classical field equations are used to obtain properties of corresponding quantum field theories. The solutions obtained in this limit are called Euclidean solutions, and in the particular case of false vacuum decay, these are called bounce solutions. Even though the full power of this method becomes apparent in quantum field theories, it is first introduced in the context of single-particle quantum mechanics for pedagogical reasons. In section 2.2, the WKB tunnelling formula for quantum mechanics with a single degree of freedom is presented. In section 2.3, the WKB tunnelling formula is extended to the quantum mechanics of many degrees of freedom. In section 2.4, the instanton solution for tunnelling in a symmetric double-well potential is obtained. In section 2.5, the classical bounce solution for metastable decay is obtained. Finally, section 2.6 extends the results of this chapter to field theory [6, 14].

2.2 WKB Tunnelling in One-Dimensional Quantum Mechanics

The dynamics of a particle is governed by the Hamiltonian

$$H = \frac{p^2}{2m} + V(q), \tag{2.1}$$

where the potential profile of $V(q)$ is shown in Figure 2.1. If such a particle with an energy E incident on the potential barrier from the left, then, in the vast majority of cases, the particle will be reflected back towards the left from the position q_1. In a small number of cases, however, the particle will tunnel across the barrier from q_1 to q_2 and emerge to the right of the barrier. The probability amplitude for tunnelling is proportional to $e^{-B/2}$, where the exponent B can be estimated using the WKB approximation as

$$B = 2\int_{q_1}^{q_2} dq\sqrt{2m[V(q)-E]}. \tag{2.2}$$

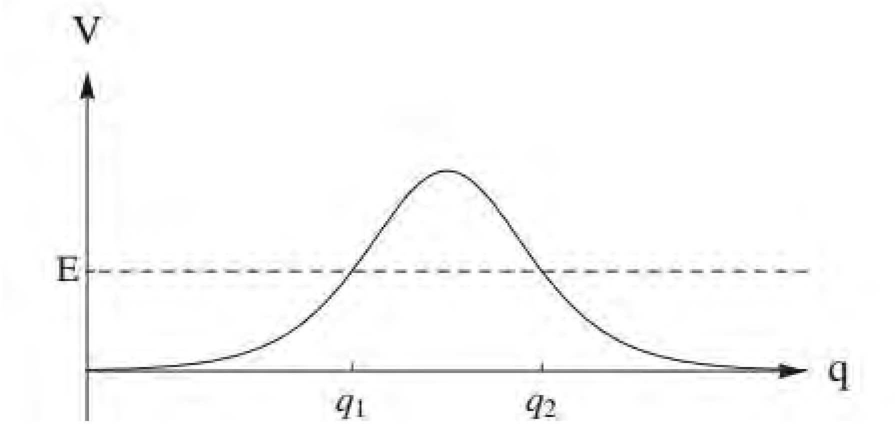

FIGURE 2.1: A potential barrier with classical turning points at $q = q_1$ and $q = q_2$.

2.3 WKB Tunnelling in Multi-Dimensional Quantum Mechanics

A particle within a system of generalised coordinates q^1, q^2, ... , q^N is defined by a Lagrangian given by

$$\begin{aligned} L &= \frac{1}{2}\sum_{j=1}^{N}\left(\frac{dq^j}{dt}\right)^2 - V(q^1, q^2, \ldots, q^N) \\ &= \frac{1}{2}\left(\frac{d\mathbf{q}}{dt}\right)^2 - V(\mathbf{q}), \end{aligned} \tag{2.3}$$

where $\mathbf{q}$ is the generalised vector coordinate.[1]

[1]The Lagrangian is used to define the system since the analysis will be generalised to field-theoretic systems and a lagrangian treatment is more amenable to field-theoretic problems.

To tunnel across a potential barrier from a given point in the coordinate space, a particle can now traverse any one of an infinite number of different paths. The particle carries a different probability amplitude for tunnelling along each of these individual paths. The probability amplitude for tunnelling across the barrier along any given path P is given by $e^{-B[P]/2}$, where the exponent can be estimated using the WKB approximation as

$$B[P] = 2\int_0^{s_f} ds\sqrt{2[V(\mathbf{q}(s)) - E]}, \tag{2.4}$$

the path P being parametrised by a trajectory s such that

$$(ds)^2 = \sum_{j=1}^{N}(dq^j)^2 \equiv (d\mathbf{q})^2 \tag{2.5}$$

and $E = V(\mathbf{q}_0)$, where $\mathbf{q}(0) = \mathbf{q}_0$ defines the initial position of the particle.[2]

It is instructive to limit the analysis only to the most probable escape path as it contributes most significantly to the overall tunnelling probability amplitude [2, 3]. In such a case, $B[P]$ is a global minimum. The goal is therefore to determine the path $\bar{\mathbf{q}}(s)$ which minimises $B[P]$. To that end, Jacobi's principle and Hamilton's principle are useful.

In classical mechanics, Jacobi's principle states that a system governed by the Lagrangian in Eq. 2.3 has a trajectory $\mathbf{q}(s)$ (in the coordinate space) which minimises the functional

$$I = \int_0^{s_f} ds\sqrt{2[E - V(\mathbf{q}(s))]}, \tag{2.6}$$

where $\mathbf{q}_0 = \mathbf{q}(0)$ and $\mathbf{q}_f = \mathbf{q}(s_f)$ are the initial and final positions of the system, respectively. The trajectories $\mathbf{q}(s)$ over which the integration is performed are constrained to satisfy the principle of conservation of energy as

$$E = \frac{1}{2}\left(\frac{d\mathbf{q}}{dt}\right)^2 + V(\mathbf{q}). \tag{2.7}$$

Comparing Eqs. 2.4 and 2.6, it is deduced that $B[P] = 2iI$. Therefore, the trajectory $\bar{\mathbf{q}}(s)$ which minimises the barrier penetration integral $B[P]$ in quantum tunnelling is also

[2]Here, it is assumed that the particle begins to tunnel from the origin.

the trajectory of classical motion (in the coordinate space) governed by the Lagrangian in Eq. 2.3.[3]

In classical mechanics, Hamilton's principle states that a system governed by the Lagrangian in Eq. 2.3 has a trajectory $\mathbf{q}(s)$ (in the coordinate space) which minimises the action

$$S = \int_{t_0}^{t_f} dt \; L(\mathbf{q}, \dot{\mathbf{q}}), \tag{2.8}$$

where $\mathbf{q}(t_0) = \mathbf{q}_0$ and $\mathbf{q}(t_f) = \mathbf{q}_f$ are the initial and final positions of the system, respectively. The trajectories $\mathbf{q}(s)$ over which the integration is performed are, however, not constrained, as in Jacobi's principle.

Therefore, the trajectory $\bar{\mathbf{q}}(s)$ which minimises the barrier penetration integral $B[P]$ in quantum tunnelling also minimises the action

$$S = \int_{t_0}^{t_f} dt \left[\frac{1}{2} \left(\frac{d\mathbf{q}}{dt} \right)^2 - V(\mathbf{q}) \right], \tag{2.9}$$

Performing a formal analytic continuation using $\tau = -it$, the corresponding Euclidean action is obtained as

$$S_E = \int_{\tau_0}^{\tau_f} d\tau \left[\frac{1}{2} \left(\frac{d\mathbf{q}}{d\tau} \right)^2 + V(\mathbf{q}) \right] \tag{2.10}$$

The constraint Eq. 2.7 from Jacobi's principle leads to

$$\frac{1}{2} \left(\frac{d\bar{\mathbf{q}}}{d\tau} \right)^2 = V(\bar{\mathbf{q}}) - E = V(\bar{\mathbf{q}}) - V(\bar{\mathbf{q}}_0). \tag{2.11}$$

so that

[3]At this point, it must be noted that quantum tunnelling is the *actual* physical pheneomenon of interest, and the allusion to Jacobi's principle from classical mechanics is simply a *computational method (carrying no physical significance)* used to determine the trajectory q(s) most likely to be taken during the tunnelling phenomenon.

$$
\begin{aligned}
S_E[\bar{\mathbf{q}}] &= \int_{\tau_0}^{\tau_f} d\tau\ 2[V(\bar{\mathbf{q}}) - V(\mathbf{q}_0)] + \int_{\tau_0}^{\tau_f} d\tau\ V(\mathbf{q}_0) \\
&= \int_{\tau_0}^{\tau_f} d\tau\ \sqrt{\left(\frac{d\bar{\mathbf{q}}}{d\tau}\right)^2} \sqrt{2[V(\bar{\mathbf{q}}) - V(\mathbf{q}_0)]} + \int_{\tau_0}^{\tau_f} d\tau\ V(\mathbf{q}_0) \\
&= \int_{0}^{s_f} ds\ \sqrt{2[V(\bar{\mathbf{q}}) - V(\mathbf{q}_0)]} + \int_{\tau_0}^{\tau_f} d\tau\ V(\mathbf{q}_0).
\end{aligned}
\tag{2.12}
$$

This result gives a relation between the tunnelling exponent B and the Euclidean action.

The particular case of interest is the tunnelling between a local minimum and a turning point $\mathbf{q}_f$ that is not a minimum of V, such as the decay of a bound state in a potential like that in Figure 1.1. In this case, the solution begins at $\tau = -\infty$, but reaches $\mathbf{q}_f$ at a finite value of τ, at which point $\frac{d\bar{\mathbf{q}}}{d\tau} = 0$. Because the Lagrangian is invariant under time reversal, this solution can be continued back to the initial point $\mathbf{q}_i$, which is reached at $\tau = \infty$. This doubles the Euclidean action, so that for the full solution

$$
B = S_E[\bar{\mathbf{q}}] - S_E[\mathbf{q}_0] \qquad \text{(bounce)}, \tag{2.13}
$$

where $S_E[\mathbf{q}_0]$, given by the last integral in Eq. 2.12, is the Euclidean action of the trivial constant solution $\mathbf{q}(\tau) = \mathbf{q}_0$. It should be noted that this relation between B and S_E only holds at their stationary points.

For obvious reasons this solution is called a bounce. The bounce corresponding to a potential like that in Figure 1.1 is shown in Figure 3.1.

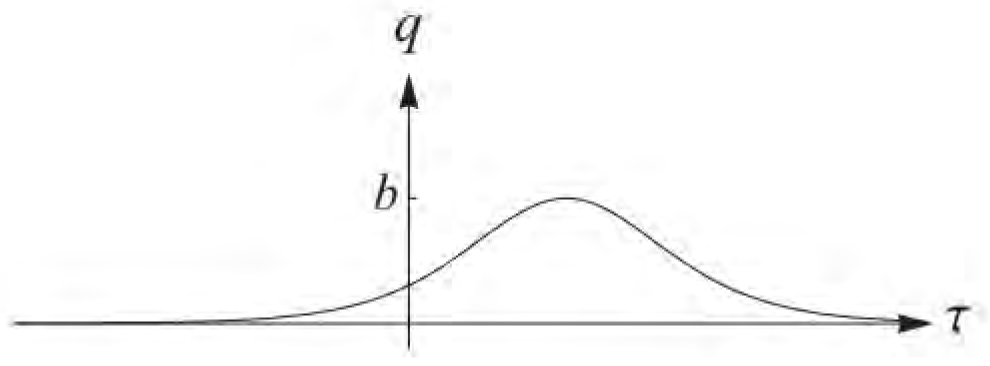

Figure 2.2: The Euclidean bounce solution as a result of metastable decay. The classical turning point is at $q = b$.

2.4 Path Integral Approach to Tunnelling: Instantons

In this section, the pre-exponential factor in the tunnelling amplitude for an instanton with a single degree of freedom is derived using the path integral approach.

As a concrete example, the case of a symmetric double-well potential with potential minima at $x = \pm a$ as shown in Figure 2.3 is considered. For convenience, the potential minima are set to zero so that the term $\int_{\tau_0}^{\tau_f} d\tau\ V(\mathbf{q}_0)$ in Eq. 2.12 is zero and the barrier penetration integral is obtained simply from the Euclidean action.

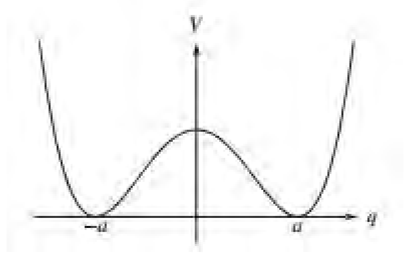

Figure 2.3: A double-well potential.

It is instructive to consider the matrix elements

$$\langle a|e^{-iHt}|a\rangle = \langle -a|e^{-iHt}|-a\rangle \tag{2.14}$$

and

$$\langle a|e^{-iHt}|-a\rangle = \langle -a|e^{-iHt}|a\rangle, \tag{2.15}$$

where $|\pm a\rangle$ are the position eigenstates with eigenvalues $x = \pm a$ respectively.

According to the path integral formalism of quantum mechanics,

$$\langle \pm a|e^{-iHt}|a\rangle = \int [dq(\tau)] e^{iS[q]}, \tag{2.16}$$

Under a Wick rotation, the problem is switched into Euclidean space as follows:

$$\langle \pm a|e^{-HT}|a\rangle = \int [dq(\tau)] e^{-S_E[q]}, \tag{2.17}$$

where T is the imaginary time and the integration is over trajectories such that $q(-T/2) = a$ and $q(T/2) = \pm a$.

Expansion of the matrix element on the left hand side in terms of energy eigenstates produces

$$\langle \pm a|e^{-HT}|a\rangle = \sum_n e^{-E_n T}\langle \pm a|n\rangle\langle n|a\rangle. \tag{2.18}$$

Fr large T, only the smallest energy eigenvalues significantly contribute to the matrix element. Calling the lowest even and odd energy eigenstates by $|+\rangle$ and $|-\rangle$, respectively,

$$\langle a|e^{-HT}|a\rangle = |\langle a|+\rangle|^2 e^{-E_+T} + |\langle a|-\rangle|^2 e^{-E_-T} \tag{2.19}$$

and

$$\langle -a|e^{-HT}|a\rangle = \langle -a|+\rangle\langle +|a\rangle e^{-E_+T} + \langle -a|-\rangle\langle -|a\rangle e^{-E_-T} \tag{2.20}$$

so that

$$\frac{\langle a|e^{-HT}|a\rangle + \langle -a|e^{-HT}|a\rangle}{\langle a|e^{-HT}|a\rangle - \langle -a|e^{-HT}|a\rangle} = e^{(E_- - E_+)T}, \tag{2.21}$$

where $\langle a|\pm\rangle = \pm\langle -a|\pm\rangle$ and $|\langle a|+\rangle| = |\langle a|-\rangle|$ have been used.

The goal is to extract the difference $E_- - E_+$ in the lowest energy eigenvalues. To that end, the matrix elements on the left hand side are evaluated using path integrals. Each of the path integrals are approximated by a sum of Gaussian integrals about their stationary points. Given a Euclidean solution $\bar{q}(\tau)$, it is possible to write

$$q(\tau) = \bar{q}(\tau) + \sum_n c_n \psi_n(\tau), \tag{2.22}$$

where $\psi_n(\tau)$ is an eigenmode with eigenvalue λ_n of

$$\left.\frac{\delta^2 S}{\delta q(\tau)\delta q(\tau')}\right|_{q=\bar{q}(\tau)} = -\frac{d^2}{d\tau^2} + V''(\bar{q}(\tau)) \equiv S''(\bar{q}) \tag{2.23}$$

Shifting from $q(\tau)$ to the the values c_n,

$$[dq] = \prod_n \frac{dc_n}{\sqrt{2\pi}} \tag{2.24}$$

Therefore, the contribution to the path integral from this stationary point is

$$I = \int \prod_n \frac{dc_n}{\sqrt{2\pi}} e^{-[S(\bar{q}) + \frac{1}{2}\Sigma_k \lambda_k c_k^2 + \cdots]}, \tag{2.25}$$

so that

$$\begin{aligned} I &= e^{-S(\bar{q})} \prod \lambda_n^{-1/2} \, [1 + \cdots] \\ &= e^{-S(\bar{q})} \, [\det\, S''(\bar{q})]^{-1/2} \, [1 + \cdots]. \end{aligned} \tag{2.26}$$

For $\langle a|e^{-HT}|a\rangle$, the stationary point is the trivial constant solution $q_0(\tau) = a$. So, the contribution to the path integral is

$$I_0 = [\det\, S''(q_0)]^{-1/2} \tag{2.27}$$

For $\langle -a|e^{-HT}|a\rangle$, the stationary point is the instanton solution such that q_1 extends from $-a$ at $\tau = -T/2$ to a at $\tau = T/2$. So, the contribution to the path integral ought to be

$$e^{-S_1} \, [\det\, S''(q_1)]^{-1/2}, \tag{2.28}$$

where S_1 is the Euclidean action of the instanton. However, there is a zero mode of S'' as follows:

$$\psi_0(\tau) = N^{-1/2} \, \frac{dq_1}{d\tau}, \tag{2.29}$$

reflecting the broken τ-translation symmetry. Given the zero eigenvalue, det S'' vanishes and the pre-exponential factor blows to infinity.

The solution to the problem is given in [14]. The resulting contribution to the path integral is

$$I_1 = e^{-S_1} \, [\det\, S''(q_0)]^{-1/2} KT, \tag{2.30}$$

where

$$K = \left(\frac{N}{2\pi}\right)^{1/2} \left[\frac{\det{}' S''(q_1)}{\det S''(q_0)}\right]^{-1/2} \tag{2.31}$$

where the prime on the determinant indicates that only nonzero modes are to be included.

In addition to these, there are approximate stationary points that must also be considered. It turns out that one needs to focus on a configuration with n instantons and anti-instantons and the contribution from all configurations with n instantons and anti-instantons is

$$I_n = e^{-nS_1}\ [\det\ S''(q_0)]^{-1/2}\ K^n \frac{T^n}{n!}. \tag{2.32}$$

Now, taking all the contributions from the stationary and approximately stationary points, the matrix elements are given by

$$\begin{aligned} \langle a|e^{-HT}|a\rangle &= \sum_{\text{even} n} I_n \\ &= [\det\ S''(q_0)]^{-1/2} \sum_{\text{even} n} \frac{[e^{-S_1}KT]^n}{n!} \\ &= [\det\ S''(q_0)]^{-1/2} \cosh\ [e^{-S_1}KT] \end{aligned} \tag{2.33}$$

and

$$\begin{aligned} \langle -a|e^{-HT}|a\rangle &= \sum_{\text{odd} n} I_n \\ &= [\det\ S''(q_0)]^{-1/2} \sum_{\text{odd} n} \frac{[e^{-S_1}KT]^n}{n!} \\ &= [\det\ S''(q_0)]^{-1/2} \sinh\ [e^{-S_1}KT] \end{aligned} \tag{2.34}$$

Therefore,

$$e^{(E_- - E_+)T} = \exp\ [2KTe^{-S_1}], \tag{2.35}$$

so that

$$\Delta = E_- - E_+ = 2Ke^{-S_1} \tag{2.36}$$

It is noteworthy that the exponent S_1 is given by the WKB approximation. The crucial new result is the pre-exponential factor in Eq. 2.31.

2.5 Path Integral Approach to Tunnelling: Bounces

For the decay of a particle from a metastable minimum $V(q = a) = 0$ of the potential energy, the energy at the metastable minimum is a complex number E such that

$$\text{Im } E = -\frac{\Gamma}{2}. \tag{2.37}$$

Arguments which are similar to those of the previous section give

$$\langle a|e^{-HT}|a\rangle = \int [dq(\tau)]e^{-S[q]} = [\det\, S''(q_0)]^{-1/2} \exp\Big[KTe^{-S(q_b)}\Big], \tag{2.38}$$

where

$$K = \left(\frac{N}{2\pi}\right)^{1/2} \left[\frac{\det'\, S''(q_1)}{\det\, S''(q_0)}\right]^{-1/2}, \quad \textbf{(incorrect)}. \tag{2.39}$$

It is possible to extract E_0 from the coefficient of T in the dominant exponential at large T, obtaining

$$E_0 = -\left[\lim_{T\to\infty} \frac{1}{2T} \ln\, \det\, S''(q_0)\right] - Ke^{-S(q_b)} = \frac{1}{2}\sqrt{V''(a)} - Ke^{-S(q_b)}. \tag{2.40}$$

It turns out that $S''(q_b)$ has a mode with negative eigenvalue. The solution to the problem of negative modes is given in [14]. The corrected expression for K is

$$K = \frac{i}{2}\left(\frac{N}{2\pi}\right)^{1/2} \left[\frac{\det'\, S''(q_1)}{\det\, S''(q_0)}\right]^{-1/2}. \tag{2.41}$$

Therefore, the decay width is

$$\Gamma = -2 \text{ Im } E_0 = \left(\frac{S(q_b)}{2\pi}\right)^{1/2} \left[\frac{\det' S''(q_1)}{\det S''(q_0)}\right]^{-1/2} e^{-S(q_b)} \tag{2.42}$$

under the assumption that $V(q = a) = 0$.

For the general case of $V(q = a) \neq 0$,

$$\Gamma = \left(\frac{B}{2\pi}\right)^{1/2} \left|\frac{\det{}' S''(q_b)}{\det S''(q_0)}\right|^{-1/2} e^{-B} \tag{2.43}$$

where B is the difference between the bounce action and that of the trivial solution.

2.6 Extension to Field Theory

The formalism in the previous sections was developed for a system with finitely many degrees of freedom. However, false vacuum decay is a problem in quantum field theory. Therefore, it is instructive to generalise the developments of the preceding sections to a system with continuously many degrees of freedom. The three crucial parameters that transform are the following (for a theory of a single scalar field in $D + 1$ dimensions):

- Coordinates q^j become field configurations $\phi(\vec{x})$.
- The tunnelling path $\vec{q}(\tau)$ becomes $\phi(\vec{x}, \tau)$.
- The potential energy $V(\vec{q})$ becomes $U[\phi(\mathbf{x})] = \int d^D x \left[\frac{1}{2}(\nabla\phi)^2 + V(\phi)\right]$

Chapter 3

Equation of Motion of the Classical Bounce Solution

3.1 Euclidean Classical Action

The action $S[\phi]$ in Minkowski space for a scalar field ϕ is given by

$$S[\phi] = \int d^4x \left(\frac{1}{2}(\partial_\mu \phi)^2 - U(\phi) \right). \tag{3.1}$$

Under a Wick rotation $\tau = it$, the derivative transforms as $(\partial_\tau, \partial_i) = (-i\partial_t, \partial_i)$ and the Euclidean classical action $S_{\text{cl}}[\phi]$ for the same scalar field ϕ is obtained as

$$S_{\text{cl}}[\phi] = \int d^4x \left(\frac{1}{2}(\partial_\mu \phi)^2 + U(\phi) \right). \tag{3.2}$$

It must be noted that the transformation of the underlying Minkowski space into Euclidean space via analytic continuation into the imaginary temporal axis is simply a computational technique that simplifies the analysis of semiclassical tunnelling problems and carries no physical significance whatsoever.

3.2 Scalar Field Potential for False Vacuum Decay

The simplest model for false vacuum decay [5, 6] considers a form of the field potential $U(\phi)$ as shown qualitatively in Figure 3.1. A standard form of the field potential $U(\phi)$

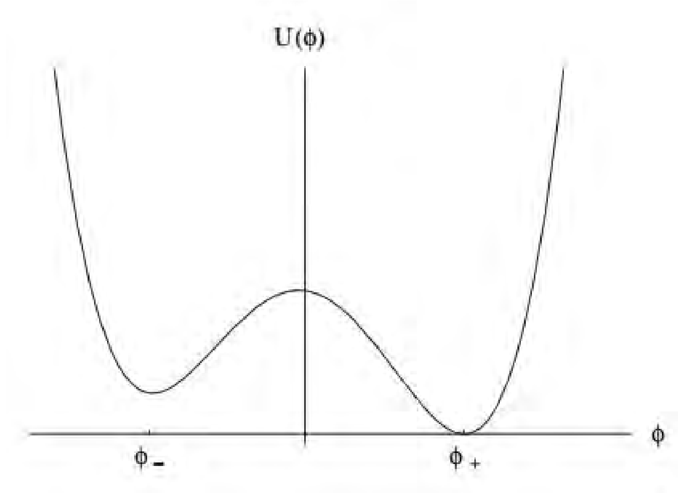

FIGURE 3.1: A quartic field potential $U(\phi)$ with its non-degenerate local minima at $\phi = \phi_-$ and $\phi = \phi_+$.

consistent with Figure 3.1 and widely considered in the existing literature [4–7] is the following:

$$U(\phi) = \frac{\lambda}{8}(\phi^2 - a^2)^2 - \frac{\epsilon}{2a}(\phi - a), \tag{3.3}$$

where λ is the coupling constant and ϵ represents the effect of a constant external source on the field potential $U(\phi)$. The physical interpretation of ϵ as an external source can be appreciated by considering the form of the field potential $U(\phi)$ for $\epsilon = 0$: when $\epsilon = 0$, the field potential $U(\phi)$ is a symmetric double-well potential with local minima at $\phi = -a$ and $\phi = a$, such that $U(\phi_-) = U(\phi_+) = 0$.

3.3 Equation of Motion of the Classical Bounce Solution

The equation of motion of the classical bounce solution can be derived for the scalar field potential in 3.3 by application of the principle of least action to the Euclidean classical action in 3.2. However, the calculations are enormously simplified if the scalar field potential in 3.3 and the Euclidean classical action in 3.2 are each rescaled as in sections 3.3.1 and 3.3.2, respectively. The equation of motion of the classical bounce solution is then calculated in section 3.3.3.

3.3.1 Rescaling of the potential $U(\phi)$

Expansion of the scalar field ϕ about the false vacuum $\phi = \phi_-$ as follows

$$\phi = \phi_- + \varphi \tag{3.4}$$

is useful because the linear term in the Taylor expansion of the scalar field potential $U(\varphi)$ vanishes:

$$\begin{aligned} U(\varphi) = U(\phi_-) + U'(\phi_-)(\varphi - \phi_-) + \frac{U''(\phi_-)}{2}(\varphi - \phi_-)^2 + \\ \frac{U'''(\phi_-)}{6}(\varphi - \phi_-)^3 + \frac{U''''(\phi_-)}{24}(\varphi - \phi_-)^4 + \cdots \\ = U(\varphi) = U(\phi_-) + \frac{U''(\phi_-)}{2}(\varphi - \phi_-)^2 + \\ \frac{U'''(\phi_-)}{6}(\varphi - \phi_-)^3 + \frac{U''''(\phi_-)}{24}(\varphi - \phi_-)^4 + \cdots \end{aligned} \tag{3.5}$$

Now, keeping terms up to dimension four,[1] the potential $U(\varphi)$ is given by

$$U(\varphi) = \frac{m^2}{2}\varphi^2 - \eta\, \varphi^3 + \frac{\lambda}{8}\varphi^4, \tag{3.6}$$

where

$$m^2 = \frac{\lambda}{2}(3\phi_-^2 - a^2) \qquad \eta = \frac{\lambda}{2}|\phi_-|.$$

3.3.2 Rescaling of the action $S_{\mathbf{cl}}(\phi)$

Rescaling the field φ and the space-time coordinates x as follows

$$\bar{x} = mx \qquad \varphi = \frac{m^2}{2\eta}\Phi. \tag{3.7}$$

is also useful because the classical Euclidean action in terms of the dimensionless quantities $\bar{x}$ and Φ simplifies to:

$$S_{\rm cl}[\Phi] = \left(\frac{m^2}{4\eta^2}\right) \int d^4\bar{x} \left[\frac{1}{2}(\bar{\partial}_\mu \Phi)^2 + \frac{1}{2}\Phi^2 - \frac{1}{2}\Phi^3 + \frac{\alpha}{8}\Phi^4\right], \tag{3.8}$$

[1]In four-dimensional space-time, the mass dimensions of the couplings are: $[\lambda] = 0$, $[a] = 1$, and $[\epsilon] = 4$.

where the quartic coupling strength is determined by the dimensionless quantity

$$\alpha = \frac{\lambda m^2}{4\eta^2} = 1 - \frac{\epsilon}{2\lambda a^4} + \cdots , \tag{3.9}$$

and the dimensionless potential is given by

$$U(\Phi) = \frac{1}{2}\Phi^2 - \frac{1}{2}\Phi^3 + \frac{\alpha}{8}\Phi^4. \tag{3.10}$$

Figure 3.2 shows some plots, for various values of α, of $U(\Phi)$.

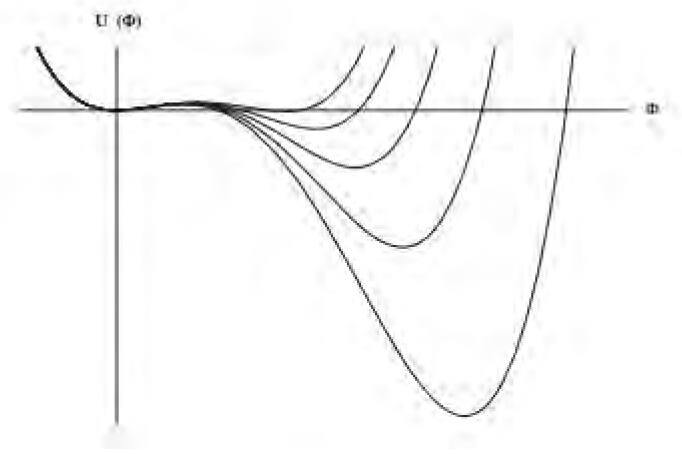

FIGURE 3.2: Plots of $U[\Phi] = \frac{1}{2}\Phi^2 - \frac{1}{2}\Phi^3 + \frac{\alpha}{8}\Phi^4$, for $\alpha = 0.6, 0.7, 0.8, 0.9, 0.99$. As α approaches 1, the rescaled potential tends to the double-well potential.

3.3.3 Equation of motion of the spherically symmetric classical bounce solution

In this section, the equation of motion of the rescaled classical bounce $\Phi_{\text{cl}}(r)$ in 3.7 is obtained from the stationary point of the rescaled classical Euclidean action in 3.8 under the assumption that *the classical bounce is spherically symmetric.* Expressing the angular measures in radians, the transformation from Cartesian coordinates $\{x^1, x^2, x^3, x^4\}$ to spherical polar coordinates $\{r, \phi_1, \phi_2, \phi_3\}$ is given by:[2]

[2]The principle of least action leads to the Euler-Lagrange equations which are conditions to what is integrated over all space. So there is no need to transform the volume element d^4x. However, the classical bounce is radially symmetric, and the equation of motion takes a simpler form in spherical polar coordinates than in Cartesian coordinates, hence the decision to transform coordinates, the volume element, the action and ultimately the Lagrangian.

$$\begin{aligned}
x_1 &= r\ \cos(\phi_1)\\
x_2 &= r\ \sin(\phi_1)\ \cos(\phi_2)\\
x_3 &= r\ \sin(\phi_1)\ \sin(\phi_2)\ \cos(\phi_3)\\
x_4 &= r\ \sin(\phi_1)\ \sin(\phi_2)\ \sin(\phi_3)\ \sin(\phi_4)
\end{aligned} \tag{3.11}$$

Therefore, the spherical volume element in 4-dimensional Euclidean space is found from the Jacobian of the transformation as follows:

$$\begin{aligned}
d^4x &= \left|\det\frac{\partial(x_i)}{\partial(r,\phi_j)}\right| dr\ d\phi_1\ d\phi_2\ d\phi_3\\
d^4x &= r^3\ \sin^2(\phi_1)\ \sin(\phi_2)\ dr\ d\phi_1\ d\phi_2
\end{aligned} \tag{3.12}$$

Therefore,

$$\begin{aligned}
S_{\rm cl}[\Phi] =& \left(\frac{m^2}{4\eta^2}\right)\int d^4\bar{x}\left[\frac{1}{2}(\bar{\partial}_\mu\Phi)^2+\frac{1}{2}\Phi^2-\frac{1}{2}\Phi^3+\frac{\alpha}{8}\Phi^4\right]\\
=& \left(\frac{m^2}{4\eta^2}\right)\int_0^{2\pi} d\phi_3 \int_0^{\pi} d\phi_2\ \sin(\phi_2)\int_0^{\pi} d\phi_1\ \sin^2(\phi_1)\\
& \qquad \int_0^{r_f} r^3\ dr\ \left(\frac{1}{2}(\partial_r\Phi)^2+\frac{1}{2}\Phi^2-\frac{1}{2}\Phi^3+\frac{\alpha}{8}\Phi^4\right)\\
=& (2\pi)(2)\left(\frac{\pi}{2}\right)\left(\frac{m^2}{4\eta^2}\right)\int_0^{r_f} dr\ r^3\ \left(\frac{1}{2}(\partial_r\Phi)^2+\frac{1}{2}\Phi^2-\frac{1}{2}\Phi^3+\frac{\alpha}{8}\Phi^4\right)\\
=& (2\pi^2)\left(\frac{m^2}{4\eta^2}\right)\int_0^{r_f} dr\ r^3\ \left(\frac{1}{2}(\partial_r\Phi)^2+\frac{1}{2}\Phi^2-\frac{1}{2}\Phi^3+\frac{\alpha}{8}\Phi^4\right),
\end{aligned} \tag{3.13}$$

where the radial integral of the classical Euclidean action $S_{\rm cl}[\Phi]$ is integrated from $r = 0$ to $r = r_f$.

Therefore, the Lagrangian $\mathcal{L}[\Phi]$ of the system in spherical polar coordinates is given by

$$\mathcal{L}[\Phi] \propto r^3\ \left(\frac{1}{2}(\partial_r\Phi)^2+\frac{1}{2}\Phi^2-\frac{1}{2}\Phi^3+\frac{\alpha}{8}\Phi^4\right) \tag{3.14}$$

The classical bounce solution $\Phi_{\rm cl}(r)$ corresponding to the extremum of the classical Euclidean action $S_{\rm cl}[\Phi]$ can therefore be found using the Euler-Lagrange equation for the scalar field Φ as follows:

$$\frac{\partial \mathcal{L}}{\partial \Phi_{\text{cl}}} - \partial_r\left(\frac{\partial \mathcal{L}}{\partial(\partial_r \Phi_{\text{cl}})}\right) = 0 \tag{3.15}$$

$$r^3\left(\Phi_{\text{cl}} - \frac{3}{2}\Phi_{\text{cl}} + \frac{\alpha}{2}\Phi_{\text{cl}}^3\right) - \partial_r\left(r^3\partial^r\Phi_{\text{cl}}\right) = 0$$

$$-\Phi_{\text{cl}}'' - \frac{3}{r}\Phi_{\text{cl}}' + \Phi_{\text{cl}} - \frac{3}{2}\Phi_{\text{cl}}^2 + \frac{\alpha}{2}\Phi_{\text{cl}}^3 = 0. \tag{3.16}$$

The boundary conditions which the classical bounce solution $\Phi_{\text{cl}}(r)$ must satisfy are as follows:

$$\Phi_{\text{cl}}'(0) = 0, \tag{3.17}$$

$$\Phi_{\text{cl}}(r) \to \Phi_- \equiv 0, \text{ as } r \to \infty. \tag{3.18}$$

The nature of the boundary conditions can be appreciated from the following facts:

1. $\Phi_{\text{cl}}(r)$ interpolates between the false and true vacuum as r goes from 0 to ∞. Therefore, the boundaries of the system are at $r = 0$ and $r = \infty$.
2. The system is assumed to occupy the false vacuum for an indefinite period of time before the decay spontaneously starts towards the true vacuum. Therefore, the field starts towards the true vacuum from rest, hence the boundary condition in 3.17.
3. The Euler-Lagrange equation in 3.15 follows from the principle of least action only if the scalar field $\Phi_{\text{cl}}(r)$ vanishes at infinity, hence the boundary condition in 3.18.

In the following chapter, the equation of motion 3.16 is solved numerically to obtain the classical bounce solution $\Phi_{\text{cl}}(r)$.

3.4 The Thin-Wall Approximation

3.4.1 Motivation

The exact analytical computation of the classical bounce solution $\Phi_{\text{cl}}(r)$ for any non-trivial field theory is a technically difficult problem - no such computations have yet not been performed. However, various approximation schemes, chief among them the thin-wall approximation (expanding Φ_{cl} about the point $\alpha = 1$, where the two vacua are degenerate), have been used to compute the classical bounce solution analytically. Although the goal of the thesis is to review an exact computational (numerical) technique for the classical bounce solution[9], the thin-wall approximation is still reviewed for completeness and as a conduit into the existing literature on the topic.

3.4.2 Physical interpretation

In the limit that $\epsilon \ll \lambda a^4$, the potential energy difference $U(\phi_-) - U(\phi_+)$ between the false and true vacua is given by

$$U(\phi_-) - U(\phi_+) = \epsilon\left[1 + \mathcal{O}\left(\frac{\epsilon}{\lambda a^4}\right)\right]. \tag{3.19}$$

Equation 3.19 can be physically interpreted to mean that the bubbles of true vacuum within the false vacuum have thin walls compared to their radius. Quite appropriately, this small ϵ limit is known as the "thin-wall" approximation [5, 6].

3.4.3 Derivation

In the following, the potential minima $\phi = \phi_-$ and $\phi = \phi_+$ are first calculated and then used to determine the potential energy difference $U(\phi_-) - U(\phi_+)$.

The minima of the field potential $U(\phi)$ can be expressed in terms of the parameters λ, a and ϵ by finding the stationary points of the potential field $U(\phi)$:

$$\begin{aligned} \frac{dU}{d\phi} &= 0 \\ \frac{\lambda}{4}(\phi^2 - a^2)(2\phi) - \frac{\epsilon}{2a} &= 0 \\ \phi(\phi^2 - a^2) &= \frac{\epsilon}{\lambda a} \\ \phi^3 - \phi a^2 - \frac{\epsilon}{\lambda a} &= 0. \end{aligned} \tag{3.20}$$

Equation 3.20 can be solved by first finding the minima for when $\epsilon = 0$ and then finding the perturbation of each of these minima when ϵ is small. Therefore, under the assumption that $\phi_\pm = \phi^0_\pm + \epsilon\phi^1_\pm + O(\epsilon^2)$ and that the derivative is zero to linear order in ϵ,

- to 0th order in ϵ,

$$\begin{aligned}\phi^3 - \phi a^2 - \frac{\epsilon}{\lambda a} &= 0\\ (\phi^0)^3 - (\phi^0)a^2 &= 0\\ \phi^0 &= 0, \pm a. \end{aligned} \tag{3.21}$$

- to 1st order in ϵ,

$$\begin{aligned}\phi^3 - \phi a^2 - \frac{\epsilon}{\lambda a} &= 0\\ (\phi^0 + \epsilon\phi^1)^3 - (\phi^0 + \epsilon\phi^1)a^2 - \frac{\epsilon}{\lambda a} &= 0\\ (\phi^0)^3 + 3(\phi^0)^2(\epsilon\phi^1) - a^2\phi^0 - a^2\epsilon\phi^1 - \frac{\epsilon}{\lambda a} &= 0.\end{aligned}$$

Therefore, for $\phi^0 = \pm a$,[3]

$$3a^2\phi^1 - a^2\phi^1 = \frac{1}{\lambda a} \quad \Longrightarrow \quad \phi^1 = \frac{1}{2\lambda a^3}.$$

Therefore, the two minima are $\phi_\pm = \pm a(1 \pm \frac{\epsilon}{2\lambda a^4} + ...)$. So, the difference in potential energy $U(\phi_-) - U(\phi_+)$ between the true and false vacua is given by:

$$U(\phi_-) - U(\phi_+) = \frac{\lambda}{8}(\phi_-^4 - \phi_+^4) - \frac{\lambda a^2}{4}(\phi_-^2 - \phi_+^2) - \frac{\epsilon}{2a}(\phi_- - \phi_+)$$

Now,

$$\begin{aligned}\phi_- - \phi_+ &= -a(1 - \frac{\epsilon}{2\lambda a^4} + \dots) - a(1 + \frac{\epsilon}{2\lambda a^4} + \dots) = -2a + \dots, \text{ and}\\ \phi_- + \phi_+ &= -a(1 - \frac{\epsilon}{2\lambda a^4} + \dots) + a(1 + \frac{\epsilon}{2\lambda a^4} + \dots) = \frac{\epsilon}{\lambda a^3} + \dots \ ,\end{aligned}$$

so that

$$\phi_-^2 - \phi_+^2 = -\frac{2\epsilon}{\lambda a^2} + \dots \ .$$

Furthermore,

$$\phi_-^2 + \phi_+^2 = [-a(1 - \frac{\epsilon}{2\lambda a^4} + \dots)]^2 + [a(1 + \frac{\epsilon}{2\lambda a^4} + \dots)]^2 = 2a^2 + \dots \ ,$$

[3] $\phi^0 = 0$ is a local maximum, so it is not needed to consider how it shifts.

so that

$$\phi_-^2 - \phi_+^2 = -\frac{4\epsilon}{\lambda}.$$

Therefore,

$$U(\phi_-) - U(\phi_+) = \epsilon + \dots \ . \tag{3.22}$$

Therefore, ϵ can be interpreted as the potential energy difference or barrier between the two classical vacua in the limit that $\epsilon \ll \lambda a^4$.

Chapter 4

Numerical Computation of the Classical Bounce Solution

The goal of this chapter is to present two alternative numerical techniques - the shooting method and the finite-difference method - to solve the equation of motion 3.16 reproduced below:

$$-\Phi''_{\text{cl}} - \frac{3}{r}\Phi'_{\text{cl}} + \Phi_{\text{cl}} - \frac{3}{2}\Phi^2_{\text{cl}} + \frac{\alpha}{2}\Phi^3_{\text{cl}} = 0,$$

with the boundary conditions 3.17 and 3.18, also reproduced below:

$$\begin{aligned} &\Phi'_{\text{cl}}(0) = 0, \\ &\Phi_{\text{cl}}(r) \to \Phi_- \equiv 0, \ \text{ as } r \to \infty. \end{aligned}$$

4.1 The Shooting Method

4.1.1 Algorithm

The solution to the differential equation 3.16 via the shooting method has been implemented in C/C++ in section 4.1.2. C/C++ does not built-in functions to directly implement the shooting method. Therefore, the various aspects of the shooting method that have found use in the C/C++ implementation in section 4.1.2 are discussed below, one by one.

In the shooting method, a number of iterations of *some specific computation* must be performed to obtain the solution of the equation of motion 3.16 to a reasonable degree of precision. *That specific computation* is the numerical integration of 3.16 using 4th order Runge-Kutta, *starting at some very small* $r = r_0$, and *ending at some very large* $r = r_f$.The choice of a *non-zero* initial value of r and the choice of a *finite* final value of r is explained later in the section.

The first iteration of the numerical integration uses the initial condition 3.17 given by $\Phi'_{\text{cl}}(0) = 0$ and a guess of the value of $\Phi_{\text{cl}}(0) \equiv \Phi_0$. After the first iteration of the numerical integration is performed, the offset in the value of $\Phi(r_f)$ from the value in the boundary condition 3.18 is used to inform the choice of the new value of Φ_0 for the next iteration of the algorithm. This is the essential link that ties the various iterations of the numerical integration. In this way, the iterations continue to adjust the value of Φ_0 until the boundary condition 3.18 is satisfied.

The preceding exposition of the nature of the iterations implies that the shooting method converts a boundary value problem into an initial-value problem, such that each of the iterations of the numerical integration solves an initial value problem for some value of Φ_0. Therefore, the goal of the shooting method is to determine the true value of Φ_0 (to a reasonable degree of precision) that, when along with 3.17 form the initial conditions of the equation of motion 3.16, satisfy the boundary condition 3.18.

The refinement of the value of Φ_0 is mediated by a root finding algorithm. The bisection method is particularly well suited for this particular differential equation. A simple plotting of the solutions of the equation for different values of Φ_0 reveals the interval (bounded both from below and from above) of Φ within which the true value of Φ_0 is located. Then, the bisection method narrows down this interval in half after each iteration. In spite of the slow convergence of this method when compared to the Newton-Raphson method or the secant method, this method is guaranteed to converge to the true value of Φ_0.

The 4th-order Runge-Kutta method is only applicable for 1st-order ordinary differential equations. Therefore, the second order ordinary differential equation 3.16 must be converted into a system of coupled first-order differential equations in order to use the 4th-order Runge-Kutta method:

$$-\Phi''_{\text{cl}} - \frac{3}{r}\Phi'_{\text{cl}} + \Phi_{\text{cl}} - \frac{3}{2}\Phi^2_{\text{cl}} + \frac{\alpha}{2}\Phi^3_{\text{cl}} = 0$$

$$\implies \begin{cases} \Theta'_{\mathrm{cl}} = -\dfrac{3}{r}\Theta_{\mathrm{cl}} + \Phi_{\mathrm{cl}} - \dfrac{3}{2}\Phi^2_{\mathrm{cl}} + \dfrac{\alpha}{2}\Phi^3_{\mathrm{cl}} & (4.1\mathrm{a}) \\ \Phi'_{\mathrm{cl}} = \Theta_{\mathrm{cl}} & (4.1\mathrm{b}) \end{cases}$$

for the boundary conditions

$$\begin{cases} \Theta_{\mathrm{cl}}(0) = 0, & (4.2\mathrm{a}) \\ \Phi_{\mathrm{cl}}(r) \to \Phi_- \equiv 0, \ \text{ as } r \to \infty. & (4.2\mathrm{b}) \end{cases}$$

Therefore, the system 4.1a, 4.1b of coupled first-order differential equations must be numerically integrated *concurrently* using 4th-order Runge-Kutta method in the application of the shooting method.

The numerical integration is started at some very small $r = r_0$, and not at $r = 0$, because the term $-\frac{3}{r}\Phi'_{\mathrm{cl}}$ in 3.16 blows up at $r = 0$. Therefore, 3.16 cannot be integrated from $r = 0$ to $r = r_0$. An alternative computational technique must be used to calculate $\Phi_{\mathrm{cl}}(r_0)$ from the estimated value of $\Phi_{\mathrm{cl}}(0) \equiv \Phi_0$. One such technique employs the Taylor expansion of $\Phi_{\mathrm{cl}}(0)$.

The Taylor expansion of $\Phi_{\mathrm{cl}}(r_0)$ is given by

$$\Phi_{\mathrm{cl}}(r_0) = \Phi_{\mathrm{cl}}(0) + \Phi'_{\mathrm{cl}}(0)(r_0) + \frac{\Phi''_{\mathrm{cl}}(0)}{2}(r_0)^2 + \mathcal{O}(r_0^3) \tag{4.3}$$

Now, the boundary condition 3.17 states that $\Phi'_{\mathrm{cl}}(r_0) = 0$. Also, the equation of motion 3.16 implies that

$$\begin{aligned} \Phi''_{\mathrm{cl}}(0) &= -\lim_{r\to 0}\frac{3}{r}\Phi'_{\mathrm{cl}}(r) + \Phi_{\mathrm{cl}}(0) - \frac{3}{2}\Phi^2_{\mathrm{cl}}(0) + \frac{\alpha}{2}\Phi^3_{\mathrm{cl}}(0) \\ &= -3\lim_{r\to 0}\frac{\Phi'_{\mathrm{cl}}(r) - \Phi'_{\mathrm{cl}}(0)}{r} + \Phi_{\mathrm{cl}}(0) - \frac{3}{2}\Phi^2_{\mathrm{cl}}(0) + \frac{\alpha}{2}\Phi^3_{\mathrm{cl}}(0) \\ &= -3\Phi''_{\mathrm{cl}}(0) + \Phi_{\mathrm{cl}}(0) - \frac{3}{2}\Phi^2_{\mathrm{cl}}(0) + \frac{\alpha}{2}\Phi^3_{\mathrm{cl}}(0) \\ \implies \Phi''_0(0) &= -3\Phi''_0 + \Phi_0 - \frac{3}{2}\Phi_0^2 + \frac{\alpha}{2}\Phi_0^3 \\ \implies \Phi''_0(0) &= \frac{1}{4}\left(\Phi_0 - \frac{3}{2}\Phi_0^2 + \frac{\alpha}{2}\Phi_0^3\right) \end{aligned}$$

Therefore,

$$\begin{aligned}\Phi_{\text{cl}}(r_0) &= \Phi_0 + \frac{r_0^2}{8}\left(\Phi_0 - \frac{3}{2}\Phi_0^2 + \frac{\alpha}{2}\Phi_0^3\right) + \mathcal{O}(r_0^3) \\ &= \Phi_0 + \frac{r_0^2}{16}\left(\alpha\Phi_0^3 - 3\Phi_0^2 + 2\Phi_0\right) + \mathcal{O}(r_0^3) \qquad (4.4)\end{aligned}$$

Therefore, 4.4 is used to obtain $\Phi_{\text{cl}}(r_0)$ from the estimated value of Φ_0 before the numerical integration of 3.16 is started at $r = r_0$. Furthermore, the system 4.1a, 4.1b of coupled differential equations must be numerically integrated *concurrently*. Therefore, the expression for $\Phi'_{\text{cl}}(r_0) = \Theta_{\text{cl}}(r_0)$ is also required:

$$\Phi'_{\text{cl}}(r_0) = \frac{r_0^2}{8}\left(\alpha\Phi_0^3 - 3\Phi_0^2 + 2\Phi_0\right) + \mathcal{O}(r_0^3) \qquad (4.5)$$

The numerical integration is ended at some very large $r = r_f$, and not at $r = \infty$, because the integration will never be completed otherwise by the computing machine. The value of r_f is chosen such that the difference $|\Phi(2r_f) - \Phi(r_f)|$ is negligible.

4.1.2 C/C++ implementation of the algorithm

The source code in Appendix A.1 is explained line by line in the following.

Lines 1 to 10:	All the required libraries are declared. The GNU Multiple Precision Arithmetic Library (GMP) library is declared in lines 7 and 8. GMP is a free library for arbitrary-precision arithmetic, operating on signed integers, rational numbers, and floating point numbers. The basic interface is for C but wrappers also exist for C++. The object-oriented nature of C++ allows for compact code as compared to the C code, hence the use of the wrapper.

Lines 12 to 13:	The two functions of the program are defined.
Line 12:	This function uses the Runge-Kutta method to calculate *phi* at discrete values of *r*. The variable *count* of type *string* counts the current number of times the function *runge_kutta* is called.
Line 13:	This function is called from within the function *runge_kutta* in order to calculate the local variables *k1*, *k2*, *k3*, *k4* within the function *runge_kutta* method. The function computes and returns the right-hand side of equation 4.1a. The right-hand-side of equation 4.1a is written in factorised form to minimise the number of computations and speed up the running of the program.
Line 15:	The main function starts.
Line 17:	*alpha* is a user-defined constant. *alpha* is declared as a variable of class *mpf_class* which is included in the C++ interface to the GMP library. Variables of the class *mpf_class* are of arbitrary precision, so *alpha* can subsequently be manipulated with the other high-precision real numbers.
Lines 18 to 19:	These lines initialise the lower bound *phi0_lower* and the upper bound *phi0_upper* of the true, but yet unknown, value of *phi0*. *phi0_lower* and *phi0_upper* are chosen such that *phi* does not diverge to infinity as *r* tends to infinity. The number 10 indicates that *phi0_lower* and *phi_upper* are in base 10. The number 500 indicates that 500 bits are reserved for the variables *phi_lower* and *phi_upper*. 500 binary bits corresponds to 150 decimal places so the subsequent computations are guaranteed be very precise.
Line 21:	This line calls the function *runge_kutta* for the lower bound *phi0_lower* of *phi0*. The argument 1 indicates that this is the first time the function *runge_kutta* is called.
Line 22:	This line calls the function *runge_kutta* for the upper bound *phi0_upper* of *phi0*. The argument 2 indicates that this is the second time the function Lines *runge_kutta* is called.

Lines 24 to 31:	These lines implement the root-finding algorithm - the bisection method. The bisection method is used to generate finer and finer approximations to the true value of *phi0* down to 100 decimal places. The bisection method within the *for* loop calls the function *runge_kutta* 100 times for the 100 iterations of the *for* loop.
Line 26:	This line bisects *phi0_lower* and *phi0_upper* to generate a finer approximation to the true value of *phi0*. Each time the interval is bisected, the precision of the value of *phi0* increases by 1 decimal place.
Line 27:	This line is miscellaneous code used to convert an *int* type to a *string* type. This is necessary because the function *runge_kutta* requires a *string* type rather than an *int* type as its third argument.
Line 28:	This line call the function *runge_kutta* for the current value of *phi0*.
Lines 29:	The *if* condition checks if the current value of *phi0* bounds the true value of *phi0* from below.
Lines 29:	The *else* condition checks if the current value of *phi0* bounds the true value of *phi0* from above.
Line 33:	The main function ends.

Lines 35 to 121:	This block of code defines the function *runge_kutta*.
Line 37:	This line initialises r to 0.001 for the Taylor expansion at small r.
Line 38:	This line uses the Taylor expansion in equation 4.4 to calculate *phi* at r = 0.001.
Line 39:	This line uses the Taylor expansion in equation 4.5 to calculate *theta* at $r = 0.001$.
Lines 42 to 73:	This block of code is miscellaneous. It sets the file input and output to write the data to a text file. It must be noted that this block of code is specific to the Ubuntu environment.
Lines 75 to 79:	This block of code is miscellaneous. It opens an interface that one can use to send commands as if they were typing into the gnuplot command line. "The -persistent" keeps the plot open even after this C program terminates. Line 78 sends commands to gnuplot one by one.

Lines 81 to 83:	These lines initialise the variables which determine if the current approximation to *phi0* bounds the true value of *phi0* from above or from below. *boundType*= 1 indicates that *phi* has diverged.
Lines 85 to 113:	In the *while* loop, the Runge-Kutta method is run from r= 0.001 to 150.0 in steps of 0.001.
Lines 87 to 94:	This block of code calculates the coefficients k_1 through k_4 of the Runge-Kutta method for each of the differential equations 4.1a, 4.1b at each value of r.
Lines 96 to 98:	This block of code uses the coefficients from the previous block to calculate *phi* and *theta* at the value of r which is one-step size from the current value of r.
Lines 100 to 103:	Having played around with the program and generating numerous solutions of the second-order differential equation, the following has been learned: If *phi0* is less than -10.0, then *phi* shoots off to negative infinity If *phi0* is greater than 10.0, then *phi* shoots off to positive infinity. Therefore, the *while* loop is *broken* if *phi*, during any iteration of the *while* loop, returns an NaN value
Line 105:	The *if* condition implies that *phi* either shoots towards positive infinity or stays level.
Line 106:	The *if* condition implies that *phi* shoots towards negative infinity.
Line 107:	The *if* condition implies that *phi* gradually rises up towards positive infinity
Lines 111 to 112:	This block of code is miscellaneous. The precision can be modified to other values - I have used 3 decimal places and 8 decimal places to generate precise and representative plots while simultaneously avoiding space wastage
Lines 117 to 118:	This block of code is miscellaneous. It is used to check that the computation has indeed been completed for each value of *phi0*. It outputs the value of *phi0* at the console/terminal. The precision has been set to 100 decimal places. This is because 100 iterations will produce a value of *phi0* that is precise by 100 decimal places.

4.2 The Finite-Difference Method

4.2.1 Algorithm

The solution to the differential equation 3.16 via the finite-difference method has been implemented in Mathematica in section 4.2.2. Mathematica has the built-in function *NDSolve‘FiniteDifferenceDerivative[Derivative[n], rGrid, pGrid]*, which computes symbolic equations for the n-th derivative of $\Phi_{\mathrm{cl}}(r)$ as a set of finite-difference equations on a one-dimensional grid using the values on the grid of the array *rGrid* of *numerical* values of r and of the array *pGrid* of *symbolic* values of $\Phi_{\mathrm{cl}}(r)$. The finite-difference equations can then be simultaneously solved to obtain the *numerical* values of the array *pGrid* which can be plotted against the corresponding values of *rGrid* to obtain the solution $\Phi_{\mathrm{cl}}(r)$ of the differential equation 3.16.

4.2.2 Mathematica implementation of the algorithm

The source code in Appendix A.2 is explained line by line in the following.

Line 1:	*alpha* is a user-defined constant.
Line 3:	*rmin* is the value of r from which the finite-difference equations are evaluated after the Taylor expansion is performed.
Line 4:	*rmax* is the final value of r.
Line 5:	*rdivisions* is the the number of intervals over the range of r.
Line 6:	*dr* is the stepsize.
Line 8:	This line creates a one-dimensional grid of values of r and stores as an array *rGrid*.
Line 9:	This line creates a one-dimensional grid of values of *phi* and stores as an array *pGrid*.
Line 11:	This line assigns the coefficient of $\Phi''_{\mathrm{cl}}(r)$ of the ordinary differential equation 3.16 to the function a.
Line 12:	This line assigns the coefficient of $\Phi'_{\mathrm{cl}}(r)$ of the ordinary differential equation 3.16 to the function b.

Line 13:	This line assigns the coefficient of $\Phi_{\rm cl}(r)$ of the ordinary differential equation 3.16 to the function c.
Line 14:	This line assigns the coefficient of $\Phi^2_{\rm cl}(r)$ of the ordinary differential equation 3.16 to the function d.
Line 15:	This line assigns the coefficient of $\Phi^3_{\rm cl}(r)$ of the ordinary differential equation 3.16 to the function e.

Line 17:	This line stores the value of the function a in a one-dimensional grid of an array *aGrid*.
Line 18:	This line stores the value of the function b in a one-dimensional grid of an array *bGrid*.
Line 19:	This line stores the value of the function c in a one-dimensional grid of an array *cGrid*.
Line 20:	This line stores the value of the function d in a one-dimensional grid of an array *dGrid*.
Line 21:	This line stores the value of the function e in a one-dimensional grid of an array *eGrid*.

Line 23:	This line computes symbolic equations for $\Phi'_{\rm cl}(r)$ at the grid points of r (by means of a built-in function that uses the finite-difference method) and stores as an array *dpdr*.
Line 24:	This line computes symbolic equations for $\Phi''_{\rm cl}(r)$ at the grid points of r (by means of a built-in function that uses the finite-difference method) and stores as an array *dpdr2*.

Line 26 :	This line uses the boundary condition on $\Phi'_{\rm cl}(rmin)$ to reformulate the finite difference equation at $r = rmin$.
Line 27:	This line uses the boundary condition on $\Phi_{\rm cl}(rmax)$ to reformulate the finite difference equation at $r = rmax$.
Line 28:	This line combines the boundary conditions in the previous two lines.

Line 30:	A set of equations is created using the ordinary differential equation. A table is created, with each entry corresponding to an interior grid point. Each entry in the table becomes an equation, from the discretized ODE. The knowledge of the boundary conditions is then used to eliminate the values of phi on the boundary.
Line 33:	The interior values of *phi* are substituted into the equations that determine the boundary values of *phi*.
Line 34:	A table of coordinates is created.
Line 35:	The solution is plotted.

4.3 Results and Discussion

4.3.1 Results

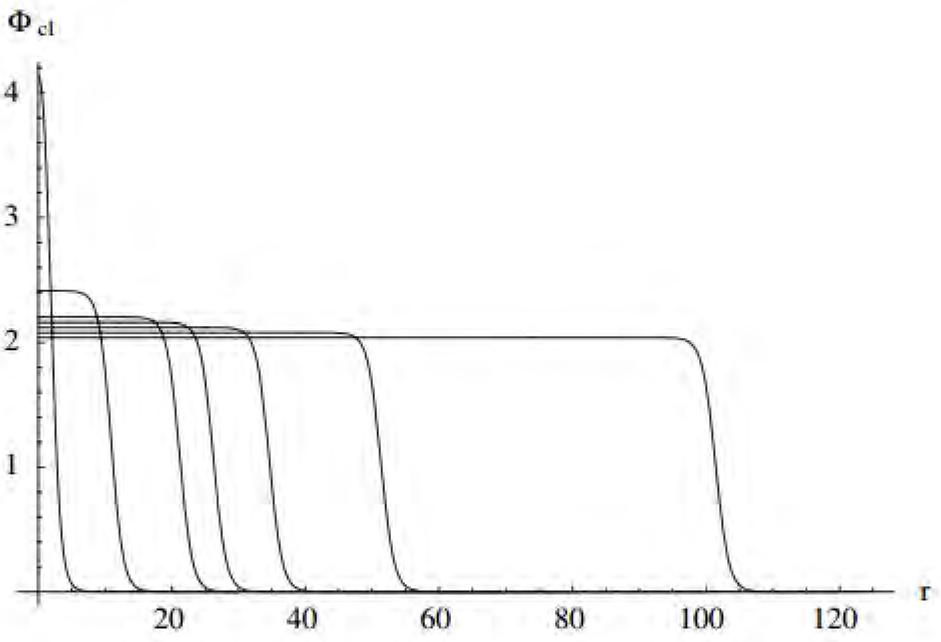

FIGURE 4.1: Plots of the bounce solution $\Phi_{\rm cl}(r)$ for $\alpha = 0.5, 0.9, 0.95, 0.96, 0.97, 0.98, 0.99$, with the plateau ending farther to the right for increasing α.

Figure 4.1 shows the plots which are expected for different values of α [9]. The source code for the shooting method generates the expected plots for smaller values of α (in a limited region of the r-axis, as discussed below). For larger values of α, the plots do not even touch the r-axis, but show rapidly oscillating behaviour for large values of r - as r increases, the plots continue to oscillate with decreasing amplitude around any one of two horizontal lines given by $\Phi_{\rm cl}(r) = k$, where the *non-zero* constant k is the solution to the cubic equation

$$k - \frac{3}{2}k^2 + \frac{\alpha}{2}k^3 = 0. \tag{4.6}$$

For smaller values of α, the plots do not simply remain stable at $\Phi_{\rm cl} = 0$, but suddenly shoot upwards and show similar oscillating behaviour as for the larger values of α.

The cubic equation 4.6 is obtained from the equation of motion 3.16 by setting $\Phi'_{\rm cl}$ and $\Phi''_{\rm cl}$ equal to 0. The oscillations tending towards $\Phi_{\rm cl}(r) = k$ is an expected behaviour (if the oscillations can be justified in the first place, but they cannot be justified), because these lines define the equilibrium solution of the equation of motion 3.16 when the boundary condition 3.17 is satisfied. However, the sudden oscillating behaviour itself is unexpected and is worthy of further investigation.

The possible cause of the unexpected behaviour of the plots lies in the *stiffness* of the ordinary differential equation 3.16. Shooting methods are notorious for causing stiffness issues in the solutions of many a differential equation. The essence of stiffness is the rapid propagation of error in the value of $\Phi_{\rm cl}(r)$ starting from the (negligibly small) error in the value of $\Phi_{\rm cl}(0)$. This behaviour can be remedied with the help of the so-called multiple shooting method, which is a variant of the standard shooting method [1]. In the multiple shooting method, the $r-$axis is divided into a number of equal intervals and the shooting method is applied to each of these intervals independently of the others. In general, there is a difference in the values of $\Phi_{\rm cl}(r)$ at the junction of any two intervals. These differences are used to compute finer and finer approximations to the required solution of the differential equation. The division of the $r-$axis into many small intervals limits the propagation of errors and thus a reliable and accurate solution to the differential equation is obtained. Indeed, the multiple shooting method is known to solve the worst problems of the standard shooting method.

The source code for the finite-difference method generates the expected plots only if the stepsize is chosen wisely. Too large a stepsize leads to an inaccurate plot because the error in the values of $\Phi_{\rm cl}(r)$ at all of the grid points is greater for a larger stepsize. On the other hand, too small a stepsize also leads to an incorrect plot.

4.3.2 Comparison of the two methods

Both the shooting method and the finite difference method are excellent techniques for solving differential equations. The shooting method uses a standard root-finding algorithm (such as the bisection method) to find the missing initial condition which satisfies the differential equation and is consistent with the boundary conditions. Also, the shooting method uses a numerical integration algorithm such as Euler's method or the 4th-order Runge Kutta method. On the other hand, the finite difference method creates a grid of points on the axis of the independent variable and then approximates the differential equation of interest using finite-difference derivatives (such as the forward difference, backward difference, central difference, etc.) at each of these grid points. The resulting set of finite difference equations, one equation at each point of the grid, are simultaneously solved to produce the solution of the differential equation.

The shooting method is a very popular technique for solving boundary-value problems. However, a standard numerical package for the shooting method does not exist. This is because it can quite often cause difficulties such as stiffness problems, in which cases only a *customised* solution using the shooting method can solve the differential equation of interest. On the other hand, the finite-difference method is a very popular technique for solving differential equations *in general*, and standard programming software such as Mathematica, Matlab, etc. are replete with numerical packages for the finite-differente method. Therefore, the finite difference method is easier to implement in a scientific programming software such as Mathematica.

Furthermore, numerical solution of a non-linear differential equation with boundary conditions using programming languages such as C/C++ and Java is, in itself, a computational challenge. Not only is it necessary to use specific programming environments (Linux, MINGW, etc. for C/C++, Eclipse, etc. for Java), but also appropriate third-party numerical libraries quite often need to be found and its functionality learnt from the user documentation and then used in the program. On the other hand, Mathemtica already has built-in numerical packages for all kinds of sophisticated mathematical computations and it can also be run on all environments. Therefore, Mathematica is particularly well suited to solve the differential equation, even for the shooting method.

Chapter 5

Conclusion

This thesis reviews the computation of the classical bounce solution for a quartic scalar field potential which is offset by an external source. In chapter 1, false vacuum decay is discussed and cosmic inflation - an example consisting of the application of false vacuum decay - is mentioned. In chapter 2, the decay rate per unit volume per unit time is shown to consist of a dominant exponential contribution due to the action evaluated on the classical bounce solution and a prefactor of functional determinants due to quantum fluctuations about the classical bounce solution. Chapters 3 and 4 and Appendix A focus exclusively on the computation of the classical bounce solution. Appendix B focuses exclusively on the calculation of the pre-factor of functional determinants.

The thesis can be improved in several possible ways, Firstly, the classical bounce solution for the quartic-offset potential model should be calculated reliably and to a high degree of precision. The multiple shooting method must be implemented to solve the stiffnes problem of the equation of motion of the classical bounce solution. The source code for the finite difference method must also be reviewed to identify the reason for the ill-behaved nature of the solutions for very small stepsizes. Furthermore, this thesis only presents, in appendix B, the analysis of the functional determinants for one-dimensional operators. The prefactor of functional determinants for the quartic-offset potential model should therefore be calculated. Lastly, the scalar field potential should be modified to consider the decay of a system with three or more local minima.

Appendix A

Source Code for the Computation of the Classical Bounce Solution

A.1 C/C++ Program for the Shooting Method

```
#include <stdlib.h>
#include <iostream>
#include <fstream>
#include <sstream>
#include <string.h>
#include <string>
#include <gmp.h>
#include <gmpxx.h>

using namespace std;

int runge_kutta(mpf_class phi0, mpf_class alpha, string count);
mpf_class get_f_theta(mpf_class r, mpf_class phi, mpf_class theta, mpf_class
    alpha);

int main() {

    mpf_class alpha = 0.50;
    mpf_class phi0_lower("0", 500, 10);
    mpf_class phi0_upper("5", 500, 10);

   int a = runge_kutta(phi0_lower, alpha, "1");
   int b = runge_kutta(phi0_upper, alpha, "2");

   int i;
   for(i=3; i<100; i++){
       mpf_class phi = (phi0_lower + phi0_upper)/2;
       stringstream ss; ss << i; string s = ss.str();
       int c = runge_kutta(phi, alpha, s);
       if(c==0)    phi0_lower = phi;
      else        phi0_upper = phi;
   }
   return 0;
}

int runge_kutta(mpf_class phi0, mpf_class alpha, string count){

   mpf_class r = 0.001;
   mpf_class phi = phi0 + ((r*r)/16.0)*(phi0)*(2.0-(phi0)*(3.0-(alpha*phi0)));
   mpf_class theta = (r/8.0)*(phi0)*(2.0-(phi0)*(3.0-(alpha*phi0)));
   mpf_class stepSize = 0.001;

   char fileName[1000];    strcpy(fileName, "Iteration ");
    strcat(fileName, count.c_str());
   strcat(fileName, ".dat");
   ofstream textfile;
   textfile.open(fileName);
   textfile << fixed;
   textfile.precision(100);   textfile << "# phi(0) = " << phi0 << "\n";
   textfile << "# r\t\tphi\n";
   textfile << fixed;
   textfile.precision(3);     textfile << r   << "\t\t";
   textfile.precision(8);     textfile << phi << "\n"   ;
    char exportCommand[1000]; strcpy(exportCommand, "set output '");
    strcat(exportCommand, "Iteration ");
   strcat(exportCommand, count.c_str());
   strcat(exportCommand, ".png");
   char plotCommand[1000];
   strcpy(plotCommand, "plot '");
   strcat(plotCommand, fileName);
   strcat(plotCommand, "' using 1:2 with line");
   mp_exp_t exponent;
   char *pointerToFileName = mpf_get_str(NULL, &exponent, 10, 0,
    phi0.get_mpf_t());
   char titleCommand[1000];
   strcpy(titleCommand, "set title \"");
   strcat(titleCommand, "phi(0) = ");
   strcat(titleCommand, pointerToFileName);
   strcat(titleCommand, "\"");
   char legendCommand[1000];
   strcpy(legendCommand, "set key off");
   char gridCommand[1000];    strcpy(gridCommand, "set grid");
   char xlabelCommand[1000];  strcpy(xlabelCommand, "set xlabel \"r\"");
   char ylabelCommand[1000];  strcpy(ylabelCommand, "set ylabel \"phi\"");
   char pngCommand[1000];     strcpy(pngCommand, "set term png");
   char x11Command[1000];     strcpy(x11Command, "set term x11");
   char * commandsForGnuplot[] = {titleCommand, legendCommand, gridCommand,
    xlabelCommand, ylabelCommand, pngCommand,            exportCommand,
    plotCommand, x11Command};

   FILE * gnuplotPipe = popen("gnuplot -persistent", "w");
    int i;
   for (i=0; i < 9; i++){
      fprintf(gnuplotPipe, "%s \n", commandsForGnuplot[i]);
   }

   mpf_class phiBefore = phi;
   int  boundType  = 0;
   bool boundCheck = false;

   while(r<=150.0){

        mpf_class k_1_phi   = theta;
        mpf_class k_1_theta = get_f_theta(r , phi, theta, alpha);
        mpf_class k_2_phi   = theta+(0.5*k_1_theta*stepSize);
        mpf_class k_2_theta = get_f_theta(r+(0.5*stepSize),
    phi+(0.5*k_1_phi*stepSize), theta+(0.5*k_1_theta*stepSize), alpha);
        mpf_class k_3_phi   = theta+(0.5*k_2_theta*stepSize);
        mpf_class k_3_theta = get_f_theta(r+(0.5*stepSize),
    phi+(0.5*k_2_phi*stepSize), theta+(0.5*k_2_theta*stepSize), alpha);
        mpf_class k_4_phi   = theta+(k_3_theta*stepSize);
        mpf_class k_4_theta = get_f_theta(r+stepSize, phi+(k_3_phi*stepSize),
    theta+(k_3_theta*stepSize), alpha);

        r = r + stepSize;
        phi = phi + (k_1_phi + (2.0*(k_2_phi + k_3_phi)) +
    k_4_phi)*(stepSize/6.0);
        theta = theta + (k_1_theta + (2.0*(k_2_theta + k_3_theta)) +
    k_4_theta)*(stepSize/6.0);

        if(phi<-10.0 || phi>10.0){
         boundType = 1;
```

```
            break;
        }

        if((r>=149.0) && (phi>=phi0))   boundType = 1;
        if((r>=149.0) && (phi<0.0))     boundType = 1;
        if(phiBefore<phi)               boundCheck = true;
        if((r>=149.0) && (boundCheck==false)) boundType = 1;
        phiBefore = phi;

        textfile.precision(3);     textfile << r   << "\t\t";
        textfile.precision(8);     textfile << phi << "\n"   ;
    }

    textfile.close();

     cout.precision(100);      cout << fixed;
     cout << "Pass: " << count << ": " << phi0 << "\r\n";

    return boundType;
}

mpf_class get_f_theta(mpf_class r, mpf_class phi, mpf_class theta, mpf_class
    alpha){
     return ((phi*(((phi/2.0)*((alpha*phi)-3.0))+1.0)) - ((3.0/r)*theta));
}
```

A.2 Mathematica Program for the Finite-Difference Method

```
alpha = 0.50;

rmin = 0.000001;
rmax = 60.000001;
rdivisions = 60;
dr = (rmax - rmin)/rdivisions;

rGrid = Range[rmin, rmax, dr];
pGrid = Array[p, rdivisions + 1, {rmin, rmax}];

a[r_] := -1;
b[r_] := -(3/r);
c[r_] := 1;
d[r_] := -1.5;
e[r_] := alpha/2;

aGrid = Map[a, rGrid];
bGrid = Map[b, rGrid];
cGrid = Map[c, rGrid];
dGrid = Map[d, rGrid];
eGrid = Map[e, rGrid];

dpdr = NDSolve`FiniteDifferenceDerivative[Derivative[1], rGrid, pGrid];
dpdr2 = NDSolve`FiniteDifferenceDerivative[Derivative[2], rGrid, pGrid];

pleft = NSolve[dpdr[[1]] == 0, pGrid[[1]]];
pright = NSolve[pGrid[[-1]] == 0, pGrid[[-1]]];
boundary = Join[pleft[[1]], pright[[1]]];

equations = Map[(0 == #) &, Flatten[Table[aGrid[[i]]*dpdr2[[i]] +
bGrid[[i]]*dpdr[[i]] + cGrid[[i]]*pGrid[[i]] + dGrid[[i]]*(pGrid[[i]]^2) +
eGrid[[i]]*(pGrid[[i]]^3), {i, 2, Length[rGrid] - 2}]] /. boundary];
intSol = FindRoot[equations[[;; Length[rGrid] - 3]], Map[{#, 1} &, Apply[
Union, Map[Variables, equations [[All, 2]] ] ] ]];
boundarySol = (boundary /. intSol);
solutionArray = (U /. Join[intSol, boundarySol]);
dataPoints = Table[{rmin + i*dr, solutionArray[[i + 1]]}, {i, 0,
Length[rGrid] - 1}];
ListPlot[dataPoints, AxesLabel -> {Style[x, Medium, Blue], Style[u,
Medium, Blue]}, PlotRange -> All];
```

Appendix B

Gel'fand-Yaglom Theorem

In this appendix, a computational method - the so-called Gel'fand Yaglom theorem - for the functional determinants of one dimensional operators is presented. In section B.1, the theorem is stated. In section B.2, the theorem is proved [8, 11]. In section B.3, an example is used to illustrate the theorem.

B.1 Statement of the Theorem

Given operators $\mathcal{O}_1$ and $\mathcal{O}_2$ on the intervals $x \in [0, L_1]$ and $x \in [0, L_2]$ respectively, each with Dirichlet boundary conditions as follows:

$$\mathcal{O}_i\ \psi_n^{(i)}(x) = \lambda_n^{(i)}\ \psi_n^{(i)}(x), \qquad \psi_n^{(i)}(0) = \psi_n^{(i)}(L_i) = 0, \qquad i \in \{1, 2\}, \tag{B.1}$$

the ratio of the functional determinants of the operators $\mathcal{O}_1$ and $\mathcal{O}_2$ is

$$\frac{\det\ (\mathcal{O}_1)}{\det\ (\mathcal{O}_2)} = \frac{\phi_1(L_1)}{\phi_2(L_2)}, \tag{B.2}$$

where $\phi_1(x)$ and $\phi_2(x)$ are given by the related initial value problems:

$$\mathcal{O}_i\ \phi_i(x) = 0; \qquad \phi_i(0) = 0,\ \phi_i^{'}(0) = 1; \qquad i \in \{1, 2\}. \tag{B.3}$$

It is instructive to pause and appreciate the significance of the above theorem. Basically, the Gel'fand-Yaglom allows one to bypass the explicit calculation of the infinite discrete sets of eigenvalues $\{\lambda_n^{(i)}\}$ - which may, in many instances, turn out to be a formidable (and nigh impossible) task - and still obtain the ratio of the functional determinants by solving related initial value problems B.3 - which are simple to evaluate analytically or implement numerically. Thus, the appeal of the Gel'fand-Yaglom theorem lies in its remarkable simplicity and practical utility - it is straightforward to implement numerically.

B.2 Proof of the Theorem

B.2.1 ζ-function regularisation

In Chapter 2, functional determinants are defined in terms of the functional integration of trajectories in the context of the path integral formalism of quantum field theory. In this section, an alternative definition of functional determinants in terms of the zeta function regularisation of operators in the context of the spectral theory of functional analysis is used. Firstly, the necessity of regularisation for the purposes of calculating the determinant is motivated, and then a rigorous definition of functional determinants in terms of a generalised Riemann zeta function is presented, finally ending with a discussion of the regularisation procedure.

As mentioned in Chapter 2, the functional determinant det $\mathcal{O}$ of an operator $\mathcal{O}$ is given simply by the product of its spectrum of eigenvalues $\{\lambda_n\}$ as follows:

$$\det \mathcal{O} = \prod_{n=1}^{\infty} \lambda_n \tag{B.4}$$

In general, the product in equation B.4 is divergent. Therefore, the technique of regularisation must be implemented on the problem to extract a finite result from the infinite product of the eigenvalues. The procedure of regularisation calls for a function that depends on the spectra of the eigenvalues - a so-called spectral function. To that end, the generalised Riemann zeta function is chosen for reasons which will become apparent after the following formal manipulations on the zeta function are observed.

The zeta function $\zeta_{\mathcal{O}}(s)$ of an operator $\mathcal{O}$, one prominent example of spectral functions, is defined as:

$$\zeta_{\mathcal{O}}(s) \equiv \text{trace}\left\{\frac{1}{\mathcal{O}^s}\right\} = \sum_{n=1}^{\infty} \frac{1}{\lambda_n^s} \tag{B.5}$$

such that s is a complex parameter and the zeta function at points where the function does not exist is defined via analytic continuation.

Definition B.5 can be used to calculate the derivative of the zeta function with respect to s as follows:

$$\zeta'_{\mathcal{O}}(s) = -\sum_{n=1}^{\infty} \frac{\ln(\lambda_n)}{\lambda_n^s} \tag{B.6}$$

so that

$$\zeta'_{\mathcal{O}}(0) = -\ln\Bigg(\prod_{n=1}^{\infty} \lambda_n\Bigg) \tag{B.7}$$

Therefore, the functional determinant det $\mathcal{O}$ is given by:

$$\det\, \mathcal{O} = \exp\,(-\zeta'_{\mathcal{O}}(0)) \tag{B.8}$$

Equation B.8 states that the determinant det $\mathcal{O}$ can be found by calculating the derivative of the associated ζ-function at its origin. This is the reason for choosing the generalised Riemann zeta function for the purposes of regularisation. The crux of the zeta function regularisation involves the introduction of the regulator s and taking the limit in which s goes away after the formal manipulations B.6 and B.7 have been performed. In this way, the required value has been decoded from the divergent product of the infinite set of eigenvalues in B.4.

Although the operator $\mathcal{O}$ has been assumed to live in an infinite-dimensional vector space, the formalism of the zeta function and definition B.8 of the functional determinant is also valid for the case of an operator from a finite-dimensional vector space. However, in that case, the product of the eigenvalues is already convergent, so the zeta function regularisation is apparently not of much use.

B.2.2 Contour integration

Equation B.5 defines the zeta function $\zeta_{\mathcal{O}}(s)$ in terms of an infinite sum of terms of the form $\frac{1}{\lambda_n^s}$. This equation bears a striking resemblance to Cauchy's residue theorem in the following form:

$$\frac{1}{2\pi i}\int_{\gamma} f(\lambda)\ \mathrm{d}\lambda\ = \sum_{n=1}^{\infty}\ \mathrm{Res}\ (f, \lambda_n), \tag{B.9}$$

where Res (f, λ_n), in correspondence with equation B.5, is the n-th residue $\frac{1}{\lambda_n^s}$ of a function $f(\lambda)$ at $\lambda = \lambda_n$ in the complex λ-plane. The function $f(\lambda)$ is constrained to be analytic, except for poles or isolated singularities located at $\lambda = \lambda_n$, within a region enclosed by the contour γ, as shown in Figure B.2. γ must be a simple, closed, and positive (circling counterclockwise) contour in order for Cauchy's residue theorem to hold. Furthermore, the eigenvalues range over the set of all real positive numbers. Therefore, the contour γ encloses the positive segment of the real axis.

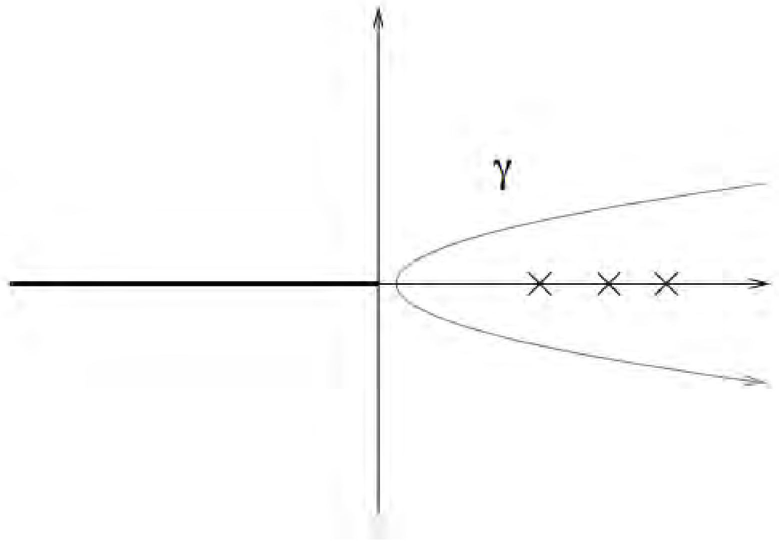

FIGURE B.1: Contour γ defined around crossed positive eigenvalues in the complex λ-plane.

Matching the left-hand sides of equation B.5 and equation B.9, the following expression for $\zeta_{\mathcal{O}}(s)$ is obtained:

$$\zeta_{\mathcal{O}}(s) = \frac{1}{2\pi i}\int_{\gamma} f(\lambda)\ \mathrm{d}\lambda \tag{B.10}$$

To calculate $\zeta_{\mathcal{O}}(s)$, the functional form of $f(\lambda)$ must be chosen. For simplicity, the poles at $\lambda = \lambda_n$ are assumed to be of order 1. Therefore, the denominator of $f(\lambda)$ is some function $\mathcal{F}(\lambda)$ such that the zeroes, each of order 1, of $\mathcal{F}(\lambda)$ are the poles $\lambda = \lambda_n$. Furthermore, $\mathcal{F}'(\lambda_n) \neq 0$ as the poles are simple. Therefore, the functional form

$$f(\lambda) = C(\lambda)\ \frac{\mathcal{F}'(\lambda)}{\mathcal{F}(\lambda)} = C(\lambda)\ \frac{\mathrm{d}}{\mathrm{d}\lambda}\ln\mathcal{F}(\lambda) \tag{B.11}$$

is a suitable ansatz for $f(\lambda)$. The residues of $f(\lambda)$ at the poles $\lambda = \lambda_n$ can be found by Taylor expanding $\mathcal{F}(\lambda)$ and $\mathcal{F}'(\lambda)$ about $\lambda = \lambda_n$ as follows:

$$\begin{aligned}\frac{\mathcal{F}'(\lambda)}{\mathcal{F}(\lambda)} &= \frac{\mathcal{F}'(\lambda_n) + (\lambda-\lambda_n)\mathcal{F}''(\lambda_n) + \frac{(\lambda-\lambda_n)^2}{2}\mathcal{F}'''(\lambda-\lambda_n) + \dots}{\mathcal{F}(\lambda_n) + (\lambda-\lambda_n)\mathcal{F}'(\lambda_n) + \frac{(\lambda-\lambda_n)^2}{2}\mathcal{F}''(\lambda-\lambda_n) + \dots} \\ &= \left(\frac{1}{\lambda-\lambda_n}\right)\left(\frac{\mathcal{F}'(\lambda_n) + (\lambda-\lambda_n)\mathcal{F}''(\lambda_n) + \frac{(\lambda-\lambda_n)^2}{2}\mathcal{F}'''(\lambda-\lambda_n) + \dots}{\mathcal{F}'(\lambda_n) + \frac{(\lambda-\lambda_n)}{2}\mathcal{F}''(\lambda-\lambda_n) + \dots}\right), \qquad \text{(B.12)}\end{aligned}$$

where the fact that $\mathcal{F}(\lambda_n) = 0$ is used, so that

$$\begin{aligned}\operatorname{Res}(f, \lambda_n) &= \lim_{\lambda\to\lambda_n}\Big((\lambda-\lambda_n)\; f(\lambda)\Big) \\ &= \lim_{\lambda\to\lambda_n}\left((\lambda-\lambda_n)\; C(\lambda)\;\frac{\mathcal{F}'(\lambda)}{\mathcal{F}(\lambda)}\right) \\ &= \lim_{\lambda\to\lambda_n}\left(C(\lambda)\;\frac{\mathcal{F}'(\lambda_n) + (\lambda-\lambda_n)\mathcal{F}''(\lambda_n) + \frac{(\lambda-\lambda_n)^2}{2}\mathcal{F}'''(\lambda-\lambda_n) + \dots}{\mathcal{F}'(\lambda_n) + \frac{(\lambda-\lambda_n)}{2}\mathcal{F}''(\lambda-\lambda_n) + \dots}\right) \\ &= C(\lambda).\end{aligned} \qquad \text{(B.13)}$$

Therefore, one of the simplest possible functional forms of $f(\lambda)$ is given by

$$f(\lambda) = \lambda^{-s}\,\frac{\mathcal{F}'(\lambda)}{\mathcal{F}(\lambda)} \qquad \text{(B.14)}$$

The complex function $f(\lambda)$ is multivalued due to the pre-factor λ^{-s}, which takes different values at angles θ and $\theta + 2\pi$. Therefore, the domain of $f(\lambda)$ is chosen to be $(-\pi, \pi)$ and place a branch cut on the negative real axis, marked as a solid black line, in Figure B.2. Therefore, the contour γ cannot cut across the negative real axis.

Furthermore, the pre-factor λ^{-s} in $f(\lambda)$ introduces an additional pole at $\lambda = 0$ of order $s + 1$. Therefore, the contour γ cannot pass through the origin in Figure B.2.

The contour is now deformed $\gamma \to \gamma_-$ such that it encloses the negative real λ-axis, rather than the positive real axis, see Fig. 4.1. When shifting the upper and lower half of the γ_--contour towards the branch cut at the negative real λ-axis, the integrands pick up a phase of $e^{i\pi s}$ and $e^{i\pi s}$, respectively. Thus Eq. B.10 becomes

$$\begin{aligned}\zeta_{\mathcal{O}}(s) =& \frac{1}{2\pi i}\left[e^{-i\pi s}\int_{-\infty}^{0} \mathrm{d}\lambda\ \lambda^{-s}\frac{\mathrm{d}\ \ln\mathcal{F}(\lambda)}{\mathrm{d}\lambda} + e^{i\pi s}\int_{0}^{-\infty} \mathrm{d}\lambda\ \lambda^{-s}\frac{\mathrm{d}\ \ln\mathcal{F}(\lambda)}{\mathrm{d}\lambda}\right] \\ =& \frac{sin(\pi s)}{\pi}\int_{0}^{-\infty} \mathrm{d}\lambda\ \lambda^{-s}\frac{\mathrm{d}\ \ln\mathcal{F}(\lambda)}{\mathrm{d}\lambda}. \end{aligned} \tag{B.15}$$

There is a pole at $\lambda = 0$ of order s, which is why the deformed contour maintains its orientation with respect to the origin. Also, the contour is deformed in such a way that it still cuts through a point on the real axis in between 0 and λ_1.

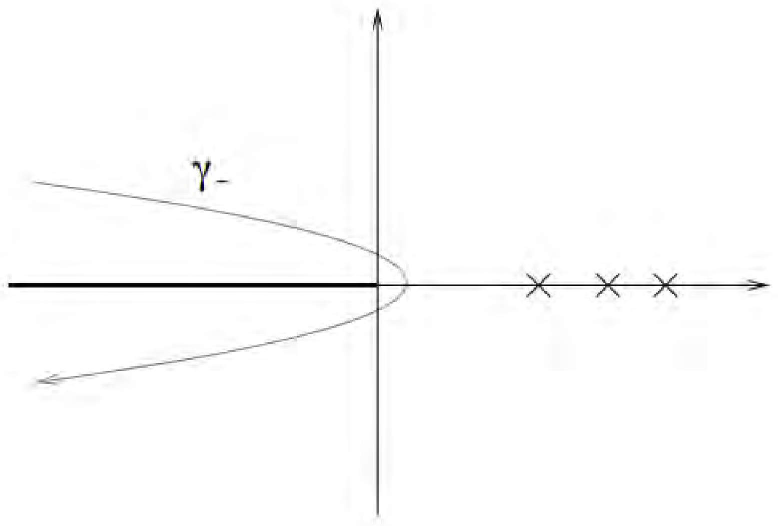

FIGURE B.2: Contour γ_- defined by deforming contour γ away from the positive eigenvalues.

Using definition B.8, the functional determinant det $\mathcal{O}$ of the operator $\mathcal{O}$ can now be calculated as

$$\begin{aligned}\det\ \mathcal{O} &= \exp\ (-\zeta'_{\mathcal{O}}(0)) \\ &= \exp\ \Big(-\frac{\mathrm{d}}{\mathrm{ds}}\zeta_{\mathcal{O}}\Big|_{s=0}\Big) \\ &= \exp\ \Big(\ln\mathcal{F}(0) - \ln\mathcal{F}(-\infty)\Big) \\ &= \frac{\mathcal{F}(0)}{\mathcal{F}(-\infty)}. \end{aligned} \tag{B.16}$$

The calculation of $\mathcal{F}(-\infty)$ can be circumvented if the ratio of the functional determinants of operators $\mathcal{O}_1$ and $\mathcal{O}_2$ is evaluated instead and the term $\mathcal{F}(-\infty)$ is assumed to be the same for both operators so that

$$\frac{\det\, \mathcal{O}_1}{\det\, \mathcal{O}_2} = \frac{\mathcal{F}_1(0)}{\mathcal{F}_2(0)} \tag{B.17}$$

In typical physical problems, the operator usually takes the form of a Hamiltonian and the corresponding functional determinant is normalised with respect to the functional determinant for the free Hamiltonian operator $\mathcal{O}_{free}$. Also, the term $\mathcal{F}(-\infty)$ is typically independent of the potential. Therefore, the assumptions used to derive equation B.17 are valid in typical physical problems.

An explicit functional form for $\{\mathcal{F}_i(\lambda)\}$ is now required to complete the derivation of the Gel'fand-Yaglom theorem. To this end, operators $\mathcal{O}_1$ and $\mathcal{O}_2$ are considered on the intervals $x \in [0, L_1]$ and $x \in [0, L_2]$ respectively, each with Dirichlet boundary conditions as follows:

$$\mathcal{O}_i\ \psi_n^{(i)}(x) = \lambda_n^{(i)}\ \psi_n^{(i)}(x), \qquad \psi_n^{(i)}(0) = \psi_n^{(i)}(L_i) = 0, \qquad i \in \{1, 2\}, \tag{B.1}$$

The required functions $\{\mathcal{F}_i(\lambda)\}$ must equal 0 at $\lambda = \lambda_n^{(i)}$ and have a finite value at $\lambda = 0$. All the functions in the sets $\{\psi_n^{(i)}(0)\}$ and $\{\psi_n^{(i)}(L_i)\}$ satisfy the former criteria, but none of these are defined for $\lambda = 0$. This is because the proof, from the start, has been presumed to exclude non-positive eigenvalues.

Therefore, in search for an alternative functional form for $\mathcal{F}(\lambda)$, initial value problems are defined with the same operators $\mathcal{O}_1$ and $\mathcal{O}_2$, but with Cauchy boundary conditions:

$$\mathcal{O}_i\ \phi_n^{(i)}(x) = \kappa_n^{(i)}\ \phi_n^{(i)}(x), \qquad \phi_n^{(i)}(0) = 0,\ \phi_n'^{(i)}(0) = 1, \qquad i \in \{1, 2\}. \tag{B.18}$$

Once more, the required functions $\{\mathcal{F}_i(\lambda)\}$ must equal 0 at $\lambda = \lambda_n^{(i)}$ and have a finite value at $\lambda = 0$. The functions in the set $\{\phi_n^{(i)}(0)\}$, however, equal 0 at $\kappa = \kappa_n$. As such, none of the functions from $\{\phi_n^{(i)}(0)\}$ are valid choices for the functions $\{\mathcal{F}_i(\lambda)\}$.

In light of the above discussion, the corresponding sets of eigenvalues in problems B.1 and B.18 must be made equal to each other by setting $\phi_n^{(i)}(L_i) = 0$. It may appear that the functions $\{\phi_n^{(i)}(x)\}$ are uniquely determined by the initial conditions in B.18 and cannot necessarily satisfy $\phi_n^{(i)}(L_i) = 0$. However, $\phi_n'^{(i)}(0)$ can be rescaled to allow for $\phi_n^{(i)}(L_i) = 0$ and the normalisation constants are dropped as a ratio of functional determinants is evaluated.

Therefore, the functions $\mathcal{F}_i(\lambda)$ take the form

$$\mathcal{F}_i(\lambda) \equiv \phi^{(i)}_{\lambda=0}(L_i) \equiv \phi_i(L_i) \tag{B.19}$$

such that

$$\mathcal{O}_i \; \phi_i(x) = 0; \qquad \phi_i(0) = 0, \; \phi_i^{'}(0) = 1; \qquad\qquad i \in \{1, 2\}. \tag{B.3}$$

Therefore,

$$\frac{\det\ (\mathcal{O}_1)}{\det\ (\mathcal{O}_2)} = \frac{\phi_1(L_1)}{\phi_2(L_2)}, \tag{B.2}$$

B.3 Example - The Infinite Square Well

It is instructive compute the determinant of the massive Helmholtz operator $[-\frac{d^2}{dx^2}+m^2]$ relative to the determinant of the massless operator $[-\frac{d^2}{dx^2}]$, both of which lie on the interval $x \in [0, L]$. Using Dirichlet boundary conditions as in B.1, the corresponding eigenvalues $\lambda_n^{(i)}$ are given by $[m^2 + (\frac{n\pi}{L})^2]$ and $[(\frac{n\pi}{L})^2]$ respectively. Therefore, the ratio of the functional determinants evaluates as follows:

$$\frac{\det[-\frac{d^2}{dx^2}+m^2]}{\det[-\frac{d^2}{dx^2}]} = \prod_{n=1}^{\infty} \frac{[m^2 + (\frac{n\pi}{L})^2]}{[(\frac{n\pi}{L})^2]} = \prod_{n=1}^{\infty} \left[1 + \left(\frac{mL}{n\pi}\right)^2\right] = \frac{\sinh(mL)}{mL} \tag{B.20}$$

The result given above has been obtained using an explicit form for the eigenvalues. However, the ratio of the functional determinants can also be calculated (without resorting to the eigenvalues) using the Gel'fand-Yaglom theorem. To do so, the related initial value problems as in B.3 are solved:

$$\left[-\frac{d^2}{dx^2} + m^2\right]\phi_1(x) = 0 \qquad \Longrightarrow \qquad \phi_1(x) = \frac{\sinh(mx)}{m} \tag{B.21}$$

$$\left[-\frac{d^2}{dx^2}\right]\phi_2(x) = 0 \qquad \Longrightarrow \qquad \phi_2(x) = x \tag{B.22}$$

Therefore, the ratio of the functional determinants using B.2 evaluates to become:

$$\frac{\det[-\frac{d^2}{dx^2}+m^2]}{\det[-\frac{d^2}{dx^2}]} = \frac{\phi_1(L)}{\phi_2(L)} = \frac{\sinh(mL)}{mL}. \tag{B.23}$$

The results in equations B.20 and B.23 clearly agree, but the point is that if m^2 were replaced by a nontrivial potential $V(x)$, the first approach, from the eigenvalues, would be extremely difficult, while the Gelfand-Yaglom approach is still easy.

Bibliography

[1] Ascher, U., Mattheij, R., and Russell, R. (1987). *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations.* Society for Industrial and Applied Mathematics.

[2] Banks, T. and Bender, C. M. (1973). Coupled anharmonic oscillators. ii. unequal-mass case. *Phys. Rev. D*, 8:3366–3378.

[3] Banks, T., Bender, C. M., and Wu, T. T. (1973). Coupled anharmonic oscillators. i. equal-mass case. *Phys. Rev. D*, 8:3346–3366.

[4] Callan, C. G. and Coleman, S. (1977). Fate of the false vacuum. ii. first quantum corrections. *Phys. Rev. D*, 16:1762–1768.

[5] Coleman, S. (1977). Fate of the false vacuum: Semiclassical theory. *Phys. Rev. D*, 15:2929–2936.

[6] Coleman, S. (1988). *Aspects of Symmetry.* Cambridge University Press.

[7] Coleman, S. and De Luccia, F. (1980). Gravitational effects on and of vacuum decay. *Phys. Rev. D*, 21:3305–3315.

[8] Dunne, G. V. (2008). Functional determinants in quantum field theory. *J. Phys.*, A41:304006.

[9] Dunne, G. V. and Min, H. (2005). Beyond the thin-wall approximation: Precise numerical computation of prefactors in false vacuum decay. *Phys. Rev. D*, 72:125004.

[10] Guth, A. H. (1981). Inflationary universe: A possible solution to the horizon and flatness problems. *Phys. Rev. D*, 23:347–356.

[11] Kirsten, K. and Loya, P. (2008). Computation of determinants using contour integrals. *Am. J. Phys.*, 76:60–64.

[12] Langer, J. (1969). Statistical theory of the decay of metastable states. *Annals of Physics*, 54(2):258 – 275.

[13] Langer, J. (2000). Theory of the condensation point. *Annals of Physics*, 281(12):941 – 990.

[14] Weinberg, E. (2015). *Classical Solutions in Quantum Field Theory.* Cambridge University Press.