Prepared for submission to JHEP

Yonsei HEP-COSMO Coding Study Final Project - Running Gauge Coupling via the Numerical Integration of the Renormalization Group Equation

Dhong Yeon Cheong[a]

[a] *Department of Physics, Yonsei University, Seoul, South Korea*

E-mail: dhongyeoncheong@gmail.com

Abstract: In this final project I conducted a numerical integration in Python 3.6 to evaluate the Running Gauge Coupling. This document contains an introduction to Python 3.6, focusing on its pros and cons. Then, I briefly introduce the physics in calculating the Running Gauge Coupling, followed with an explanation with the algorithm of my code. The results show that by increasing the energy scale, there is no unification point in the range $M_t = 170.85GeV, \xi = 0$ to $M_t = 170.85GeV, \xi = 10^4$.

Contents

1 Summary of Python 3.6

Python is a programming language developed in 1991. It is an interpreted language that feature a dynamic system. Python's language is simple and compact compared to other programming languages, such as C, C++, etc. This lets beginners to easily write short codes efficiently. This dynamic typing automatically confirms the data type while running, which eliminates the need to define a specific one for each variable.

Python also supports Object oriented programming(OOP), which is a programming paradigm that interprets a code as a set of objects, and these objects interact with each other. This allows the programmer to use classes, objects, and methods to create a more efficient code.

In contrast, Python requires a fairly long computing time. This is due to the fact that the program uses an "interpreter" instead of a compiler. This feature is critical when computing large data structures. To compensate with this weakness, programmers tend to interchange between Python and other languages(ex. C, C++) when needed. Cython uses this method, where the basic functional structure is written in Python, and the computing code is compiled in C. This gives the advantage of both the simple grammar of Python, and the fast computing of C.

2 Running Gauge Coupling

2.1 Renormalization Group Equation

The purpose of this project is to perform a running for the fundamental gauge coupling constants in the Standard Model, based on the Higgs Inflation model, which takes the action form as (2.1)

$$S = \int d^4x\sqrt{-g}\Big[\frac{M_P^2}{2}R + \frac{1}{2}\xi\phi^2 R - \frac{1}{2}g^{\mu\nu}\partial_\mu\phi\partial_\nu\phi - \frac{1}{4}\lambda\phi^4\Big] \tag{2.1}$$

where ξ is the nonminimal coupling, λ is the quadratic coupling, and ϕ is the Higgs field in unitary gauge. [1] The fundamental objective of this is to obtain a numerical solution of the Renormalization Group Equation. The Renormalization Group Equation for our Higgs Inflation model takes the form as the following (2.2),

$$\frac{dg}{d(ln(\mu/M_t))} = \frac{\beta_g}{1+\gamma} \tag{2.2}$$

where g is the running coupling constant. and β is the rate of change of the renormalized coupling at the scale M_t corresponding to a fixed bare coupling.[2] The γ is the Higgs field anomalous dimension. This equation describes the flow of a modified coupling constant as a function of μ.

2.2 Running Gauge Couplings

Now, as I determined the equation needed to be solved, we need to specify each β function and γ functions. These functions are obtained by conducting a loop correction[2]. I only considered the one-loop corrections, which leads to the following forms, [1]

$$\begin{aligned}
\beta_\lambda &= \frac{1}{16\pi^2}\Big[6(1+3s_\phi^2)\lambda^2 + 12\lambda y_t^2 - 6y_t^4 - 3\lambda(3g_2^2+g_1^2) + \frac{3}{8}(2g_2^4 + (g_2^2+g_1^2)2)\Big] \\
\beta_{g1} &= \frac{1}{16\pi^2}\Big[\frac{81+s_\phi}{12}g_1^3\Big] \\
\beta_{g2} &= \frac{1}{16\pi^2}\Big[\frac{s_\phi - 39}{12}g_2^3\Big] \\
\beta_{g3} &= -\frac{7}{16\pi^2}g_3^3 \\
\beta_{y_t} &= \frac{y_t}{16\pi^2}\Big[\Big(\frac{23}{6}+\frac{2}{3}s_\phi\Big)y_t^2 - \Big(8g_3^2 + \frac{9}{4}g_2^2 + \frac{17}{12}g_1^2\Big)\Big] \\
\beta_\xi &= \frac{1}{16\pi^2}\Big[6(1+s_\phi^2)\lambda + 6y_t^2 - \frac{3}{2}(3g_2^2+g_1^2)\Big]\Big(\xi+\frac{1}{6}\Big) \\
\gamma &= -\frac{1}{16\pi^2}\Big(\frac{9}{4}g_2^2 + \frac{3}{4}g_1^2 - 3y_t^2\Big) \\
s_\phi &= \frac{1+\xi\phi^2/M_P^2}{1+(1+6\xi)\xi\phi^2/M_P^2}
\end{aligned} \tag{2.3}$$

Here, y_t is the top Yukawa coupling constant, $g_{1,2,3}$ are the SM gauge couplings(strong, weak, electromagnetic in order), and s_ϕ is the suppression factor. For this project, I set the renormalization scale $\mu = \frac{y_t}{\sqrt{2}}\phi$, which is the Higgs vev term, to minimize the radiative correction.

The initial conditions for the values are

$$\begin{aligned}
\lambda(\mu = M_t) &= 0.12604 + 0.00206\Big(\frac{M_H}{GeV} - 125.15\Big) - 0.00004\Big(\frac{M_t}{GeV} - 173.34\Big) \\
y_t(\mu = M_t) &= 0.93690 + 0.00556\Big(\frac{M_t}{GeV} - 173.34\Big) - 0.00003\Big(\frac{M_H}{GeV} - 125.15\Big) - 0.00042\Big(\frac{\alpha_s(M_Z) - 0.1184}{0.0007}\Big) \\
g_1(\mu = M_t) &= 0.35830 + 0.00011\Big(\frac{M_t}{GeV} - 173.34\Big) - 0.0002\Big(\frac{M_W/GeV - 80.384}{0.014}\Big) \\
g_2(\mu = M_t) &= 0.64779 + 0.00004\Big(\frac{M_t}{GeV} - 173.34\Big) + 0.00011\Big(\frac{M_W/GeV - 80.384}{0.014}\Big) \\
g_3(\mu = M_t) &= 1.1666 + 0.00314\Big(\frac{\alpha_s(M_Z) - 0.1184}{0.0007}\Big) - 0.00046\Big(\frac{M_t}{GeV} - 173.34\Big)
\end{aligned} \tag{2.4}$$

with the PDG values

$$\begin{aligned}
M_W = 80.385GeV, \qquad M_Z = 91.1876GeV, \\
M_H = 125.09GeV, \qquad \alpha_s(M_Z) = 0.1182
\end{aligned} \tag{2.5}$$

As this simulation is based on the Higgs inflation model, the top quark mass M_t and the nonminimal coupling constant xi are free parameters. The expected range for these parameters are $M_t = 170 \sim 172GeV$ and $\xi = 0 \sim 44000$. In summary, the running process is evaluating the values of the running gauge coupling constants while varying $\mu = \frac{y_t}{\sqrt{2}}\phi$ for fixed free parameters M_t and ξ.

3 Algorithm - Code

3.1 Numerical Integration - Euler's Method

As we need to evaluate the value g, I performed the integration as to be

$$g(t+h) - g(t) = \int_t^{t+h} \frac{\beta}{1+\gamma} dt \approx h\frac{\beta(t)}{1+\gamma(t)} \tag{3.1}$$

This method is derived via the Taylor expansion and the left Riemann sum rule. Although there are more precise methods for numerical integration(ex. Gauss Quadrature Rule), I chose the simplest numerical integration method as the computing speed of Python 3.6 couldn't bear with other integration methods. For example, for the Gauss quadrature rule, I needed to call(or define) the Legendre polynomials, which would increase the computing time to an unbearable region.

For this project, I set the region $t = 0 \sim 44$, and the step size to be $h = 0.00001$. To deal with the t dependence of the integrand, by using the scaling I defined earlier, ϕ takes the form

$$\phi = \frac{\sqrt{2}}{y_t} M_t \exp(t) \tag{3.2}$$

and this value gives the t dependence of the integrand.

3.2 Code

In this study, I learned that to deal with the slow speed, it is possible to use Numpy arrays and calculations. Thus, I defined Numpy arrays that hold data for the functions needed in the calculation. I defined a class that will compute all needed initial values, functions, and integration methods. Among this class, I evaluated the Euler's numerical integration method as

```
def Running(self, h):
        self.IH += h *self.Beta_gamma[0]
        self.g1 += h * self.Beta_gamma[1]
        self.g2 += h * self.Beta_gamma[2]
        self.g3 += h * self.Beta_gamma[3]
        self.yt += h * self.Beta_gamma[4]
        self.LG += h * self.Beta_gamma[5]
        self.t += h
```

Then, I defined a function that evaluates the prior Running process for the given range.

```
def Data_writing(Mt, xi, Nend, h):
        RGE = Runnin_Gauge(Mt, xi, 0)
        k = int(1/h * Nend)
        IH = np.empty(k)
        yt = np.empty(k)
        t = np.empty(k)
        g1 = np.empty(k)
        g2 = np.empty(k)
        g3 = np.empty(k)
        phi = np.empty(k)
        G = np.empty(k)

        for i in range(k):
                RGE.Beta_funct()
                IH[i] = RGE.IH
                g1[i] = RGE.g1
                g2[i] = RGE.g2
                g3[i] = RGE.g3
                yt[i] = RGE.yt
                t[i] = RGE.t
                phi[i] = RGE.phi
                G[i] = np.exp(-RGE.LG)
                RGE.Running(h)
        return IH, g1, g2, g3, yt, t, phi, G
```

I conducted this for various ξ values, and created csv files with the data. Then, I plotted the data using `matplotlib.pyplot`.

4 Running Results

Figure 1. and Figure 2. show the results of the running gauge coupling. Figure 1. implies that in the range of $\xi = 0 \sim 10000$, the graph of each coupling constant is almost identical with each other. This implies that in the range $\xi = 0 \sim 10000$ for $M_t = 170.85GeV$, there is no unification where $g_{1,2,3}$ merge at the same point.

Figure 2. shows the running results of the quadratic coupling constant λ. There are some anomalies in which I could not account. The coupling constant value becomes negative in the $15 \sim 44$ region. By analyzing the $33 \sim 39$ region where the minimum exists, there is no tendency for the λ graph when increasing t.

References

[1] Jinsu Kim, Tae Geun Kim, Seong Chan Park, *Top quark mass from Primordial Black Holes in Critical Higgs Inflation.*

[2] Peskin, Schroeder, *An Introduction to Quantum Field Theory*, Westview Press (2016).

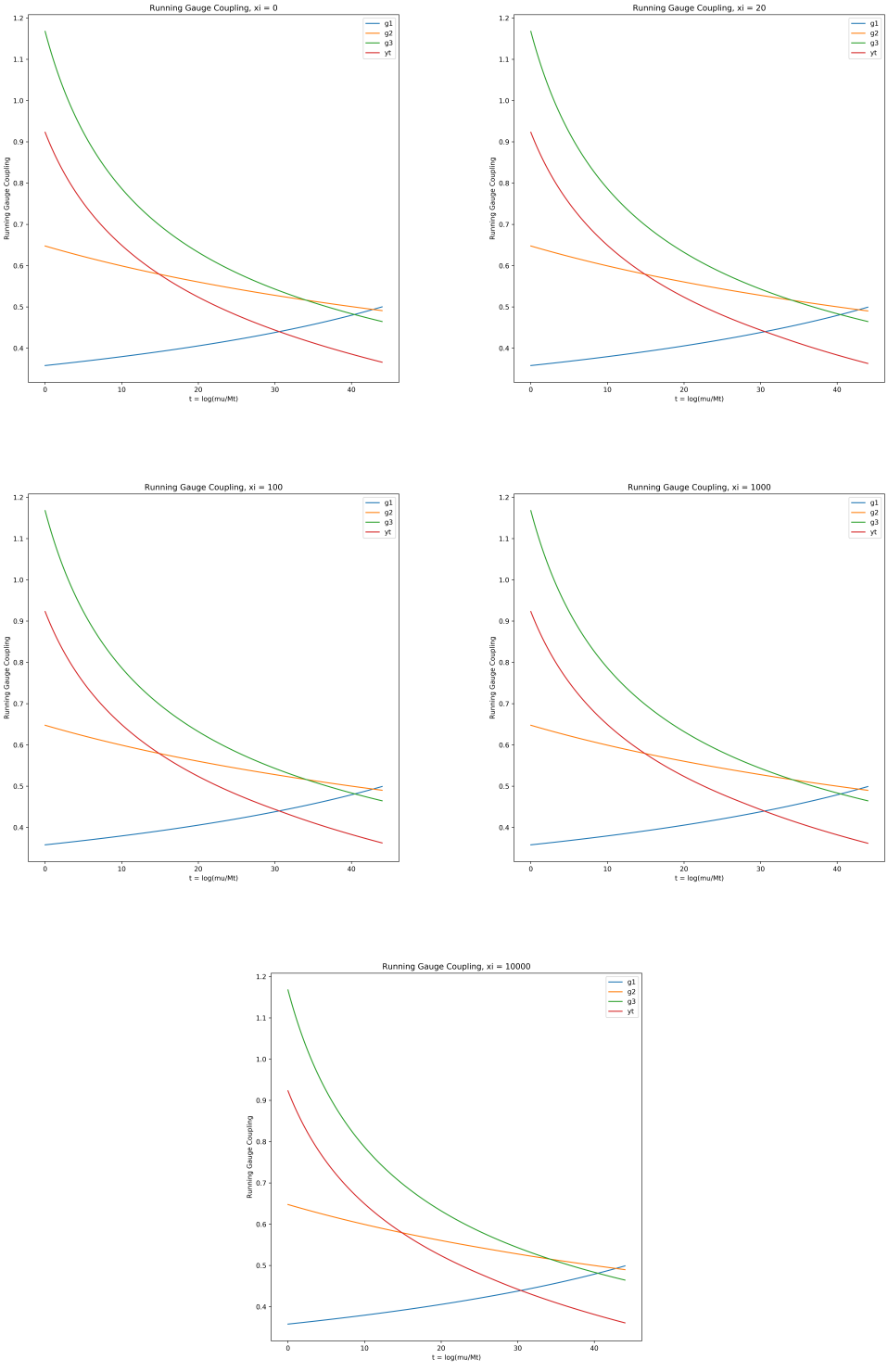

Figure 1. Running Gauge Coupling for $\xi = 0, \xi = 10$(top), $\xi = 100, \xi = 1000$(middle), $\xi = 10000$(bottom)

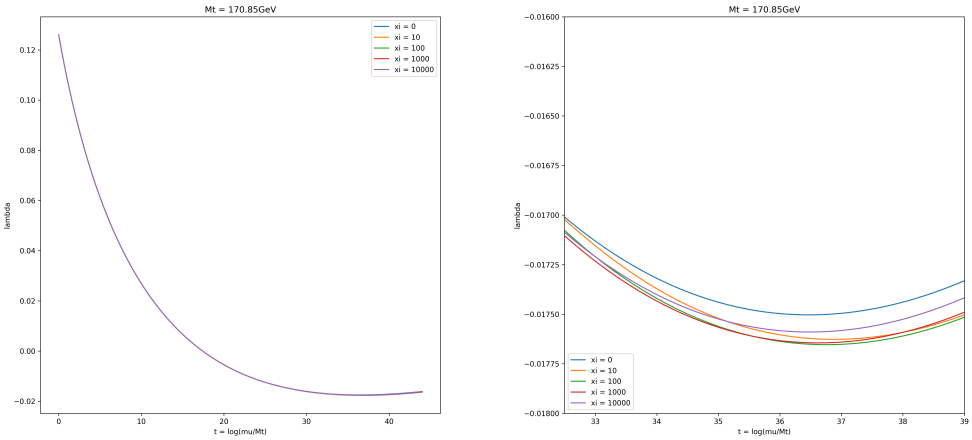

Figure 2. Running λ for $\xi = 0$, $\xi = 10$, $\xi = 100$, $\xi = 1000$, $\xi = 10000$