Modelling Mass Generation

Anuradha Ratnaweera and L.C.R. Wijewardhana
Institute of Fundamental Studies, Kandy 20000
Email: anuradha@ifs.ac.lk

1. Introduction

Since the advent of computers, it has been possible to model natural phenomena mathematically. As most natural systems are under the effect of many factors, such models are extremely complicated and in many cases, a complete analysis is impossible. On the other hand, numerical techniques can be used to solve the equations describing these models. In the present paper, we will demonstrate the use of computational tools, through an example in quantum mechanics. The physical and mathematical background of the problem will be outlined in Sections 2 and 3. The implementation of the calculations is overviewed in Section 4.

2. The Theoretical Background

According to the so-called "standard theory" of particle physics, there are two types of elementary particles, quarks and leptons. They come in three families and feel the four types of forces listed below.

- Strong interaction
- Electro-magnetic interaction
- Weak interaction
- Gravitational interaction

These forces are carried by gluons, photons, intermediate vector bosons W and Z and gravitons respectively.

The mathematical model which describes electromagnetic and weak interactions has a local SU(2) × U(1) symmetry and it is called the Electro-weak model. The strong interaction is described Quantum Chromodynamics (QCD), and this model has a local SU(3) symmetry. These two quantum field theories, taken together, are called the standard model. When the symmetries are exact, the gauge particles should remain massless, whereas experimental evidence shows that the W and Z particles have masses of 80 GeV and 90 GeV, although gluons and photons remains massless. It was Abdul Salam and Steven Weinberg who figured out a way to keep the gauge symmetries exact and at the same time make W and Z massive, by a process known as spontaneous symmetry breaking. The idea of this mechanism is to keep the equations of motion symmetric although the solutions may not. Salam and Weinberg introduced a particle called Higgs that distorts the vacuum and makes the minimum energy configuration violate the symmetry.

Theories with the elementary Higgs particle are not aesthetically appealing as they suffer from naturalness problems. One option is to assume that there is another hidden symmetry called the supersymmetry. Although supersymmetry does not give rise to naturalness problems, there is no experimental evidence for its existence. Another possible assumption is that the Higgs particles are bound states of fermion pairs and therefore not elementary. The force that binds these fermions is called Technicolor and these fermions are called Techni fermions. Techni force is the binding force of Techni quarks. These Techni bound states deform the vacuum and make W and Z particles massive by spontaneously breaking the electro-weak symmetry.

3. The Gap Equation

The running coupling strength α measures the force acting between the Techni quarks. At high momenta (small distances) it goes to zero and at low momenta (long distances) it goes to an fixed point α^*.

The renormalization group (RG) equation for the running coupling is

$$\mu\frac{\partial\alpha(\mu)}{\partial\mu}=-b\alpha^2-c\alpha^3-d\alpha^4... \tag{1}$$

The first two coefficients are given by

$$b=\frac{1}{6\pi}(11N-2N_f)$$

$$c=\frac{1}{24\pi^2}\left(34N^2-10NN_f-3\frac{N^2-1}{N}N_f\right)$$

where N_f flavours of fermions in the fundamental representation. Also fixed point α^* is given by

$$\alpha^*=-\frac{b}{c} \tag{2}$$

The explicit solution of the above equation with the first two terms of the RG equation can be written as

$$\alpha=\alpha^*\left[W\left(\mu^{\alpha^*b}/e\Lambda^{\alpha^*b}\right)+1\right]^{-1} \tag{3}$$

where $W(x)=F^{-1}(x)$ with $F(x)=xe^x$ and Λ is an introduced scale.

The Techni quark mass Σ is given by the gap equation

$$\Sigma(p)=\frac{1}{2\alpha_c}\left[\int_0^p\frac{k^3\alpha(p)\Sigma(k)}{p^2[k^2+\Sigma^2(k)]}dk+\int_p^M\frac{k\alpha(k)\Sigma(k)}{k^2+\Sigma^2(k)}dk\right] \tag{4}$$

This is a non-linear integral equation and $\Sigma=0$ is trivially a solution. It is well known that a linear integral equation usually has a unique solution and the problem can be reduced to a set of linear equations. However, in this particular case only stable solutions, for which the iterative process

$$\Sigma_{n+1}(p)=\frac{1}{2\alpha_c}\left[\int_0^p\frac{k^3\alpha(p)\Sigma_n(k)}{p^2[k^2+\Sigma_n^2(k)]}dk+\int_p^M\frac{k\alpha(k)\Sigma_n(k)}{k^2+\Sigma_n^2(k)}dk\right] \tag{5}$$

converges, are of interest. Here α_c is a critical value given by

$$\alpha_c=\frac{2\pi N}{3(N^2-1)} \tag{6}$$

It was verified that when $\alpha^*>\alpha_c$, the mass Σ is nonzero and therefore the chiral symmetry is spontaneously broken.

The higher order terms of the RG equation were also analysed by finding the minimum number of flavours N_f for a given N.

4. Implementation

Several types of software were used to do the above calculations, and the contribution from each is shortly described below.

MATLAB: The fourth generation language of MATLAB was used to handle symbolic manipulations. Its symbolic toolbox internally calls the MAPLE library. The explicit solution for the RG equation was obtained by this method. First a linear approximation was used to solve the gap equation and later it was replaced by a fifth order approximation. The calculation of the coefficients for this process and the algebraic complexity of the higher order calculations were again handled by this toolbox. Some of the graphics were also obtained with MATLAB.

C Programming Language: The gap equation was solved with a C program. Newton's method was used to evaluate the function W that was necessary to calculate the coupling α. The MATLAB program used to find the coefficients for the fifth order approximation was generated by another C program. Finally, the higher order calculations of the RG equations were treated with the Newton's method.

Origin: This spreadsheet like package integrates several mathematical routines, plotting routines and an object oriented language, Labtalk. The generated masses S had to be checked for several scaling behaviours and this was done using the non-linear curve fitting facilities available in origin. Its graphical facilities were very useful to visualise the results.

System: All the experiments were carried out on an IBM compatible PC (Pentium) running under Windows 95. Parallel experiments were also performed on an IBM workstation in the University of Cincinnati.

Other: Programming languages such as Basic and Pascal were sometimes used for minor calculations. In addition, packages such as MS Excel, Mathcad were also proved useful. Auxiliary packages supplied with the operating system were also sometimes used.

5. Discussion

The example described in this article is only a part of a larger experiment. The use of appropriate software, ranging from third generation programming languages to application packages, was demonstrated.

These experiments proved that it is possible to use personal computers (PCs) in advanced scientific work. Although the Windows 95 operating system was used, most of the first stages were done using the DOS based software such as Borland Pascal. Only the standard functions were used to preserve compatibility as the programmes were implemented on different computers.

Appendix

An Overview of the Available Computational tools

The early computers were controlled by the assembly language and machine code. However, at present, scientists hardly use them, with the exception of rare cases where high optimisations are necessary. FORTRAN, the first programming language, became widely popular among the mathematicians, and even today, many of them think that it is *the* language for numerical processing. The development of the "numerical recipes" library, which was a collection of routines necessary for numerical processing is another reason for its popularity.

While FORTRAN was progressing with mathematicians, computer scientists were interested in the C language, which is undoubtedly the best language for system programming. It was developed along with the UNIX system, which was written entirely in C. Because of the highly flexible nature of the language, it is easy to make mistakes with C, if used carelessly. On the other hand, some mathematicians had enough patience to learn C and use its strength in numerical processing. The conversion of the "numerical recipes" routines to C was a major step in popularising its capabilities for scientific work. Almost all of the programs described in this article were written in C.

Comprehensive computer programming is necessary in advanced mathematical modelling. Therefore, the necessity of some tools for the non-specialists was required, and it gave birth to the so-called fourth generation languages[1]. These languages provide many library routines, so that a great deal can be accomplished with only a few lines of code. At the same time, the control over the computer is lost

[1] The machine language and the assembler languages are called first and second generation languages respectively. Programming languages such as FORTRAN, C and PASCAL are categorised under third generation languages.

as these routines have their own ways of doing things. Two popular fourth generation languages are MATLAB and MATHEMATICA. Similar to the UNIX operating system, the programming language in MATLAB is written with the open system approach so that all the routines are open and can be modified if necessary. This also improves the reliability of the system, for anyone can check the routines to ensure that they are correct.

Apart from the fourth generation languages, many application packages are currently used for numerical analysis. One of the drawbacks of such packages is that one cannot see what is going on inside them, and for this reason, they are sometimes called "black box software". On the other hand, this hides unnecessary computational details from the non-specialist. Packages such as ORIGIN and STATISTICA are well accepted as scientific software and even Microsoft Excel is now widely used for statistical analysis.

Along with the rapid development of computing software, numerical techniques were also highly improved. Proper methods of error analysis were introduced to obtain results that are more rigorous. The celebrated "shadowing lemma" is a good example, which shows the consistency of using numerical methods to analyse dynamical systems.

The power of the computers is also rapidly increasing. Even a personal computer is powerful enough to perform do a fair amount of advanced work.