COMPUTER ALGORITHMS FOR EUCLIDEAN LATTICE GAUGE THEORY CALCULATIONS

A thesis

submitted in partial fulfilment

of the requirements for the degree

of

Doctor of Philosophy in Physics

in the

University of Canterbury

by

A. G. Riddell

University of Canterbury

February 1988

ABSTRACT

The computer algorithm devised by K. Decker [25] for the calculation of strong coupling expansions in Euclidean lattice gauge theory is reviewed. Various shortcomings of this algorithm are pointed out and an improved algorithm is developed. The new algorithm does away entirely with the need to store large amounts of information, and is designed in such a way that memory useage is essentially independant of the order to which the expansion is being calculated. A good deal of the redundancy and double handling present in the algorithm of ref. [25] is also eliminated.

The algorithm has been used to generate a 14^{th} order expansion for the energy of a glueball with non–zero momentum in Z_2 lattice gauge theory in 2+1 dimensions. The resulting expression is analysed in order to study the restoration of Lorentz invariance as the theory approaches the continuum.

A description is presented of the alterations required to extend the algorithm to calculations in 3+1 dimensions. An eighth order expansion of the Z_2 mass gap in 3+1 dimensions has been calculated. The eighth order term differs from a previously published result.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr. W.R. Moreau for his continuing support and for providing me with the necessary impetus to get this project completed

Correspondence from both Dr. K. Decker and Dr. G. Munster proved extremely helpful in clearing up some of my misconceptions regarding the technical details of mass gap calculations.

Thanks also to Hughan Ross for helping with the preparation of my diagrams.

CONTENTS

CHAPTER 1

INTRODUCTION

One of the most difficult tasks still facing theoretical physicists is the development of a coherent theory of the strong interaction. The major obstacle is the fact that the force is so strong; perturbative calculations are useless for anything but the most small–scale interactions. Hence any theory, once formed, is extremely difficult to test.

The prime candidate for the status of "theory of the strong interaction" is Quantum Chromodynamics, a gauge theory with gauge group SU_3. It has long since been proven [1,2] that non–abelian gauge theories will exhibit asymptotic freedom, the vanishing of coupling strength which occurs in strong interactions at small length scales. The task remains, however, to prove that Q.C.D. also predicts quaark confinement, i.e. infinite interaction strength at infinite separation.

Obviously, non–perturbative techniques are required to investigate the properties of Q.C.D. as the length scale, and so the coupling strength, increases. One such non–perturbative technique is the lattice gauge theory devised independently by Wilson [3] and Polyakov [4]. With the gauge theory set up on the discrete space–time lattice, the strong coupling region becomes very amenable to investigation by means of series expansions in the inverse coupling constant. As hoped, the lattice version of Q.C.D. does display quark confinement. The interaction between a pair of quarks is prorotional to the distance between them and so increases without limit as the quarks are separated [3]. Unfortunately, the lattice version of Q.E.D. also displays the same property.

The hope, then, is that as the lattice spacing is reduced the abelian gauge theory undergoes a phase transition back to a deconfining phase, but the non–abelian theory stays confining right to the continuum. A great deal of effort, therefore, is being

put into studying the phase structure of lattice gauge theories. This work usually takes the form of examining the variation of some property of the theory as the lattice spacing is reduced.

The two quantities most often considered are the string tension and the mass gap. The string tension is a measure of the constant force that exists between two quarks at infinite separation. The mass gap is difference in energy between the vacuum state of the theory and its first excited state. Both quantities should vanish at a deconfining phase transition and remain zero throughout the unconfined phase. There are three main approaches taken to the calculation of these quantities:

(i) Series expansions in the Hamiltonian formulation on a continuous–time lattice.[5]

(ii) Series expansions in the path integral formulation on an infinite Euclidean lattice.[6]

(iii) Monte Carlo simulations on a finite Euclidean lattice.[7]

All the early work in lattice gauge theory was done using strong coupling series expansions [8,9]. Guth, using analytic methods on a Euclidean lattice, proved the existence of a deconfining phase transition for pure lattice Q.E.D. [10]. Then it was recognised that the Monte Carlo techniques used in thermodynamics[11] could be applied to L.G.T..

Early Monte Carlo work by Creutz [12] produced very encouraging results. His numerical calculation of the pure SU_2 string tension showed a rapid but smooth cross–over from agreement with strong coupling results to agreement with the weak coupling behaviour expected from renormalization group calculations. However, subsequent Monte Carlo calculations of the SU_2 string tension [13,14,15], while producing curves of similar shape, produced different numerical values for the string tension. Recently, in fact, more detailed studies [16] have turned evidence that perhaps a deconfining phase transition does occur during the rapid cross–over from strong

coupling to weak coupling regions.

Whatever the case, the results of Monte Carlo simulations are still by no means conclusive. There do exist obstacles to the improvement of Monte Carlo data. In the cross–over region, where interest has centred, the correlation length grows rapidly as the coupling strength is decreased. It is imperitive that the simulation be carried out on a lattice with linear dimensions at least as large as the correlation length or finite size effects will render the results meaningless. Hence, if one wishes to investigate further into the weak coupling region (i.e. enlarge the 'window' between the onset of scaling and dominance of finite size effects) it is necessary to increase the size of the lattice under study. However, present work is already stretching the resources of currently available computers. A major improvement in M.C. results will require either faster computers or much more efficient algorithms. A good deal of effort is being put into both these areas at present with the development of special purpose computers [17] and the experimenting with a wide variety of new methods on conventional computers[18,19,20].

So, despite the initial sucess of M.C. simulations and the incredible array of calculations that are possible with Monte Carlo techniques [21,22], there are still reasons to persevere with analytical calculations on the lattice. The main problem with strong coupling expansions is the fact that their domain of validity fails to extend into the cross–over region. Hence it is necessary to use extrapolation techniques like Pade approximants [23] in order to study the transition to weak coulping. Unfortunately, the extrapolations are not reliable when applied to series for which only a few terms exist. Up to a couple of years ago, the existing strong coupling series for the mass gap had all been calculated by hand and contained only about 4 or 5 terms [8,24].

With the development of computer algorithms for S.C. mass gap calculations [25,26] the situation has changed markedly. There now exists an expansion for the Z_2 mass gap in 2+1 dimensions containing 11 terms [27]. Analytic techniques are now being used to good effect in the study the phase structure of full lattice Q.E.D.

(electrons and all) [28,29]. In short, efficient computer algorithms are making it possible to calculate strong coupling expansions for quantities which have previously only been accessible to M.C. simulations.

The present thesis covers work done in extending the algorithm for mass gap calculations in Euclidean L.G.T. first developed by K.Decker [25]. In chapter 2 the elements of Euclidean strong coupling expansions are presented. Decker's algorithm is reviewed in some detail in chapter 3. Various shortcomings of Decker's algorithm are pointed out in chapter 4, and an improved implementation of his basic ideas is described. The extension of this algorithm to calculations in 3+1 dimensions has been undertaken. Chapter 5 contains a description of the changes required to carry out this extension. It also contains a discussion of some of the as yet unsurmounted obstacles preventing a truely efficient algorithm for 3+1 dimensional calculations. A few new results that have been obtained from my programs are presented in chapter 6. Finally, in chapter 7, we look at the quite different algorithms, based on extracting the S.C. series from resummed series, which might well supercede the present work.

CHAPTER 2

STRONG COUPLING EXPANSIONS IN EUCLIDEAN L.G.T.

The starting point for calculations in any euclidean quantum field theory is the path integral. The path integral for a lattice gauge theory with the Wilson action may be written [6]:

$$Z = \int \prod_{\{l\}} DU_l \prod_{\{P\}} e^{\beta \chi_f(U_p)} \tag{2.1}$$

Where: $\{l\}$ is the set of all links on the lattice

$\{P\}$ is the set of all plaquettes on the lattice

U_l is the group element on the link l

U_P is the product of group elements U_l around the links of the plaquette P

$\chi_f(U)$ is the character of U in the fundamental irrep of gauge group, G.

β is the euclidean coupling constant

The path integral for an infinite lattice obviously shares with that of a continuum quantum field theory the problem that it is an infinite dimensional integral and cannot be evaluated exactly. So, just as in the continuum case, we must approximate it with a power series expansion. In this chapter we shall develop the so–called strong coupling expansion which is commonly employed in euclidean lattice calculations.

2.1 The Character Expansion

We shall begin by employing a little group theory in order to reexpress the

path integral in a form that is more ammenable to the development of a diagram expansion.

The irreducible characters of a group form an orthonormal basis for the expansion of any class function of the elements of the group, a class function being one which satisifies

$$F(U) = F(VUV^{-1}) \quad ; \quad U,V \in G. \tag{2.2}$$

So we may carry out a character expansion of the plaquette action

$$e^{\beta\chi_f(U_p)} = \sum_r c_r(\beta)\, \chi_r(U_P) \tag{2.3}$$

with

$$c_r = \int DU\, \chi_r(U)\, e^{\beta\chi_f(U)} \tag{2.4}$$

Every group has a trivial irrep, $r = 0$, for which $\chi_0(U) = 1$, so we may write

$$\begin{aligned} e^{\beta\chi_f(U_p)} &= c_0(\beta) + \sum_{r \neq 0} c_r(\beta)\, \chi_r(U_P) \\ &= c_0(\beta) \left[1 + \sum_{r \neq 0} a_r(\beta)\, d_r\, \chi_r(U_P)\right] \end{aligned} \tag{2.5}$$

where $a_r(\beta) = \frac{1}{d_r}\, c_r(\beta)/c_0(\beta)$, with d_r being the dimension of the r^{th} irrep. Let us now simplify the notation by writing

$$\sum_{r \neq 0} a_r(\beta)\, d_r\, \chi_r(U) \equiv f(U) \tag{2.6}$$

So that upon substituting the character expansion into Z we obtain:

$$Z = (c_0(\beta))^{N_P} \int \prod_{\{l\}} DU_l \prod_{\{P\}} (1+f(U_P)) \tag{2.7}$$

where N_P is the total number of plaquettes in the lattice.

2.2 The Diagram Expansion of Z.

Having obtained the expression (2.7) for Z, we are now in a position to form a diagram expansion. This will take the form of an expansion in terms of integrals over localized sets of plaquettes obtained by expanding the product $\prod_{\{P\}} (1+f(U_P))$.

Obviously the first term in the expansion of $\prod_{\{P\}} (1+f(U_P))$ is just 1, then we have N_P terms each of just $f(U_P)$, then $^1/_2\, N_P(N_P-1)$ terms involving the product of two $f(U_P)$'s etc. So we write

$$Z = [c_0(\beta)]^{N_P} \sum_{\mathscr{P}} \int \prod_{\{l\}} DU_l \prod_{P \in \mathscr{P}} f(U_P) \tag{2.8}$$

where $\mathscr{P}$ runs over all the possible sets $\{P_1,P_2,.....\}$ of plaquettes on the lattice. It is easy to see that

$$\int \prod_{\{l\}} DU_l \prod_{P \in \mathscr{P}} f(U_P) = \prod_{x_i} \int \prod_{b \in x_i} DU_b \prod_{P \in x_i} f(U_P) \tag{2.9}$$

where x_i enumerates all the connected components of $\mathscr{P}$, a pair of plaquettes being defined to be connected if they share a link.

Lets condense our notation by writing

$$\int \prod_{b \in x_i} DU_b \prod_{P \in x_i} f(U_P) = \Phi(x_i) \tag{2.10}$$

We shall refer to $\Phi(x_i)$ as the activity of the set of connected plaquettes x_i. It is now possible for us to write Z in a very condensed form

$$Z = [c_0(\beta)]^{N_P} \sum_{\mathscr{P}} \prod_{x_i \in \mathscr{P}} \Phi(x_i) \tag{2.11}$$

Now we shall prove the very important result that $\Phi(x_i) = 0$ unless x_i is a closed polymer, i.e. unless every link on every plaquette in x_i is shared by at least one other plaquette in x_i. To prove the assertion we simply consider a set of plaquettes x_i which do not form a closed polymer. Consider one of the unshared links, and let the gauge group operator on the link be U_s. We may write

$$\Phi(x_i) = \int \prod_{\substack{b \in x_i \\ b \neq s}} DU_b \prod_{P \in x_i} f(U_P) \int DU_s \, f(U_{P_s}) \tag{2.12}$$

where P_s is the plaquette which contains the unshared link s. The separation of variables is made possible by the fact that U_{P_s} is the only factor that contains U_s.

Now

$$f(U_{P_s}) = \sum_{r \neq 0} a_r(\beta) \, d_r \, \chi_r(U_{P_s})$$

$$= \sum_{r \neq 0} a_r(\beta) \, d_r \, \chi_r(U'_{P_s} U_s) \tag{2.13}$$

Where U'_{P_s} is the product of the gauge group operators on the three links which, along

with s, make up the plaquette P_S. So we may now write

$$\int DU_S\, f(U_{P_S}) = \sum_{r\neq 0} a_r(\beta)\, d_r \int DU_S\, \chi_r(U'_{P_S} U_P)$$

$$= \sum_{r\neq 0} a_r(\beta)\, d_r \int DU_S\, \chi_r(U'_{P_S} U_S)\, \chi_0(U_S^{-1}) \qquad (2.14)$$

Since $\chi_0(U) = 1$ for any group operator U. We now invoke the character orthogonality relation

$$\int DU\, \chi_r(U)\, \chi_s(VU^{-1}) = \delta_{rs}\, d_r^{-1}\, \chi_r(V) \qquad (2.15)$$

and we have

$$\int DU_s\, f(U_{P_S}) = \sum_{r\neq 0} a_r(\beta)\, d_r\, \delta_{r0}\, d_r^{-1}\, \chi_r(U'_{P_S})$$

$$= 0$$

Hence the activity of a set of plaquettes which do not form a closed polymer is zero, only closed polymers contribute to the path integral. Hence

$$Z = [c_0(\beta)]^{N_P} \sum_{\mathscr{D}} \prod_{x_i \in \mathscr{D}} \Phi(x_i) \qquad (2.16)$$

where $\mathscr{D}$ runs over all the possible sets of closed polymers on the lattice. This is a diagram expansion of Z in terms of closed polymers, from it one may produce a power series in the factor $a_f(\beta)$. Every x_i can be expanded as a series in $a_f(\beta)$, starting at

$[a_f(\beta)]^{N_{x_i}}$, N_{x_i} being the number of plaquettes in x_i. Hence, to generate an expansion of Z up to $[a_f(\beta)]^M$, one need only consider the diagrams which contain up to M plaquettes.

However one very rarely wishes to calculate the path integral Z, interest far more often centres on the free energy per site , given by

$$f = \frac{1}{N_S} \text{Ln } Z$$

$$= \frac{d(d-1)}{2} \text{Ln } c_0(\beta) + \frac{1}{N_S} \text{Ln} \left[\sum_{\mathscr{D}} \prod_{x_i \epsilon \mathscr{D}} \Phi(x_i) \right] \qquad (2.17)$$

where d is the dimensionality of the particular theory being considered, and N_S the total number of sites in the lattice. Let us now form a direct diagram expansion of the second logarithm in (2.17) and see just what sort of digrams are involved.

2.3 The Moment–Cumulant Transformation.

The standard method of forming the diagram expansion of the logarithm is by means of the so called moment–cumulant transformation [30], which we shall examine shortly. However there always seems to be a certain element of magic about the moment–cumulant expansion. It seems to make the contributions from the disconnected diagrams vanish miraculously. When one attempts to sift carefully through the process to really perceive exactly why they vanished it is very easy to become lost in a labyrinth of factorials and cumulants etc. and never get to the bottom of the matter. I have decided, therefore, to first present a more direct proof that the contributions from sets of disconnected objects do vanish in the logarithm, i.e. by counting up the contributions and showing them to be zero.

The logarithm that we are looking at is:

$$\mathrm{Ln}\Big[\sum_{\mathscr{D}}\prod_{x_i\in\mathscr{D}}\Phi(x_i)\Big] = \mathrm{Ln}\Big[1+\sum_{\mathscr{D}\neq\phi}\prod_{x_i\in\mathscr{D}}\Phi(x_i)\Big]$$

where we have singled out the zero–order term, which simply has activity 1. Using the standard series expansion for the logarithm this becomes:

$$\mathrm{Ln}\Big[\sum_{\mathscr{D}}\prod_{x_i\in\mathscr{D}}\Phi(x_i)\Big] = \Big[\sum_{\mathscr{D}\neq\phi}\prod_{x_i\in\mathscr{D}}\Phi(x_i)\Big] - {}^1\!/_2\Big[\sum_{\mathscr{D}\neq\phi}\prod_{x_i\in\mathscr{D}}\Phi(x_i)\Big]^2 + {}^1\!/_3\Big[\sum_{\mathscr{D}\neq\phi}\prod_{x_i\in\mathscr{D}}\Phi(x_i)\Big]^3 - {}^1\!/_4\Big[\sum_{\mathscr{D}\neq\phi}\prod_{x_i\in\mathscr{D}}\Phi(x_i)\Big]^4 + \ldots \qquad (2.18)$$

Consider now some set, $D = (X_1,\ldots,X_n)$, of disconnected diagrams. Obviously the contribution that this set makes to Z, and so also to the first term on the r.h.s. of (2.18) is simply

$$\Phi(\mathscr{D}) = \prod_{i=1}^{n}\Phi(x_i) \qquad (2.19)$$

Let us now look at each of the other terms on the r.h.s. of (2.18), and find how many times the exact expression $\Phi(\mathscr{D})$ appears in each of them. When evaluating the second term on the r.h.s of (2.18) we must sum take the product of every possible pairing of diagrams contributing to Z. Not only does $\mathscr{D}$ contribute to Z, but all possible subsets of $\mathscr{D}$ also contribute. Consider the product of $\Phi(\mathscr{E})$ and $\Phi(\mathscr{F})$ where $\mathscr{E}$ and $\mathscr{F}$ are complementary subsets of $\mathscr{D}$, i.e. $\mathscr{E}\cup\mathscr{F}=\mathscr{D}$ and $\mathscr{E}\cap\mathscr{F}=0$. This product will obviously be among the myriad of terms that will contribute to the squared term in the logarithm, and the value of the product will be equal to $\Phi(\mathscr{D})$. There will be several such complementary pairs , let us denote the number of ways of partitioning a set of n objects into r subsets by ${}^nB_r$. The total number of times $\Phi(\mathscr{D})$ appears in the

second term on the r.h.s. of (2.18), then, will be $(1/2)\ 2\ {}^{n}B_{2}$. Similarly , the number of times it appears in the third term will be $(1/3)\ 3!\ {}^{n}B_{3}$, or more generally, it will appear $(1/r)\ r!\ {}^{n}B_{r}$ times in the r^{th} term. Hence the total number of appearances of $\Phi(\mathscr{D})$ in the expansion of Ln Z is given by

$$1 - {}^{n}B_{2} + 2\ {}^{n}B_{3} - 3!\ {}^{n}B_{4} + \ldots + (-1)^{n-1}\ (n-1)!\ {}^{n}B_{n} = \sum_{r=1}^{n} (-1)^{r-1}\ (r-1)!\ {}^{n}B_{r} \qquad (2.20)$$

Let us now prove that this will always sum to zero. First we must establish a recursion formula for the ${}^{n}B_{r}$. Consider a set of n–1 objects and form all the partitions of this set into r subsets and into r–1 subsets. We will have two collections of the form:

	r–1 subsets		r subsets
1	[a..b][c..d].....[l..m]	1	[a..c][f..g].....[j..m]
2	[a..f][g..k].....[j..m]	2	[a..c][g..h].....[k..m]
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
${}^{n-1}B_{r-1}$	[a..d][f..e].....[c..m]	${}^{n-1}B_{r}$	[a..f][g..h].....[l..m]

Now we wish to form all the partitions of a set of n objects into r subsets. This can be done quite simply by building on the collections displayed above. An n^{th} object, O_{n} , is introduced. Each partition in the first column above is built upon by the addition of a further subset $[O_{n}]$ to form ${}^{n-1}B_{r-1}$ of the partitions of the set of n objects. Then the object O_{n} is placed once inside every subset in every partition on the r.h column, thereby producing all the remaining partitions of the set of n objects. Hence we can see that

$${}^{n}B_{r} = {}^{n-1}B_{r-1} + r\ {}^{n-1}B_{r} \qquad (2.21)$$

We are now ready to carry out the proof that (2.20) is zero for all values of n.

$$\sum_{r=1}^{n} (-1)^{r-1}\,(r-1)!\;{}^{n}B_{r} \;=\; 1 + \sum_{r=2}^{n} (-1)^{r-1}\,(r-1)!\;{}^{n}B_{r}$$

$$= 1 + \sum_{r=2}^{n} (-1)^{r-1}\,(r-1)!\;({}^{n-1}B_{r-1} + r\;{}^{n-1}B_{r})$$

$$= 1 + \sum_{r=2}^{n} (-1)^{r-1}\,(r-1)!\;{}^{n-1}B_{r-1} + \sum_{r=2}^{n-1} (-1)^{r-1}\,r!\;{}^{n-1}B_{r}$$

$$= \sum_{r=1}^{n-1} (-1)^{r}\,r!\;{}^{n-1}B_{r} \;+\; (-1)\sum_{r=1}^{n-1} (-1)^{r}\,r!\;{}^{n-1}B_{r}$$

$$= 0$$

Hence the contribution to Ln Z from any set of disconnected objects contributing to Z is zero.

Having now satisfied ourselves that contributions from disconnected sets of polymers do vanish in the logarithm we may now examine the moment–cumulant transformation.

Simply, the nub of the moment–cumulant transformation is as follows. Form two sets of functions, call one set the moments $\langle \alpha,\beta,\gamma,.....\pi \rangle$, and the other set is dubbed the cumulants $[\alpha,\beta,\gamma,.....\pi]$. The sets must be chosen in such a way that

$$\langle\alpha,\beta,....\psi\rangle = \sum_{P}[\alpha...\delta]\ [\gamma...\varphi]....[\rho...\omega]$$

$$\text{or}\ \ [\alpha,.....,\varphi] = \sum_{P} (-1)^{k-1}\ (k-1)!\ \underbrace{\langle\alpha,...,\beta\rangle\langle\gamma,...,\psi\rangle...\langle\eta,...,\lambda\rangle}_{k\ \text{factors}} \tag{2.23}$$

where P runs over all possible partitions of the set $\{\alpha,\beta,....,\psi\}$ into complementary subsets.

One then defines the generating functional for moments by

$$F(\{\phi_\alpha\}) = \sum_{(\alpha,...,\beta)} (\Pi_i\, n_i!)^{-1}\ \langle\alpha,....,\beta\rangle\ \phi_\alpha.....\phi_\beta \tag{2.24}$$

The sum is over all possible ordered sets of variables, with duplications allowed , and n_i are the multiplicities of the elements of $(\alpha,.....,\beta)$. $F(\{\phi_\alpha\})$ is a generating functional in the sense that

$$\frac{\partial}{\partial\phi_\alpha}\frac{\partial}{\partial\phi_\beta}....\frac{\partial}{\partial\phi_\psi} F(\{\phi_\alpha\})\Big|_{\{\phi_\alpha\}=0} = \langle\alpha,\beta,....,\psi\rangle \tag{2.25}$$

A similar generating functional is defined for the cumulants

$$f(\{\phi_\alpha\}) = \sum_{(\alpha,...,\beta)} (\Pi_i\, n_i!)^{-1}\ [\alpha,.....,\beta]\ \phi_\alpha.....\phi_\beta \tag{2.26}$$

The central theorem of the moment–cumulant transformation is a relation between these two generating functionals

$$1 + F(\{\phi_\alpha\}) = \exp f(\{\phi_\alpha\}) \tag{2.27}$$

The simplest way to satisfy oneself of the truth of this theorem is to differeniate both sides w.r.t. some set of ϕ_α's, and set all the ϕ_α's to zero. The l.h.s. will then be simply $\langle\alpha,\ldots,\beta\rangle$. Upon careful consideration it is possible to see that the r.h.s. will yield $\sum_P [\alpha....\gamma]\,[\eta...\kappa]...[\sigma...\beta]$. Hence the theorem is nothing more than a restatement of the relationship that exists between the moments and the cumulants.

Let us now examine how this transformation may be applied to obtaining a diagram expansion of Ln Z. The critical consideration is the way we define our moments. We shall take the moments (and so also the cumulants) to be functions of polymers X_i, and define their values as follows:

$$\langle X_1,....,X_n\rangle = \begin{cases} 1, & \text{if every pair } X_i,\ X_j \text{ is not connected} \\ 0, & \text{otherwise} \end{cases}$$

Given this definition, it possible to write the path integral in the form

$$Z[\{\Phi(x)\}] = 1 + \sum_{(x_1,\ldots,x_n)} (n!)^{-1} \langle x_1,...,x_n\rangle\, \Phi(x_1).....\Phi(x_n) \qquad (2.28)$$

Note that the sum runs over all possible ordered sets $(x_1,...,x_n)$, hence the need for the factor of $(n!)^{-1}$. Obviously Z is written in exactly the form of the generating functional for the moments, so we may apply the moment–cumulant transformation to it and obtain an expression for Ln Z:

$$\text{Ln } Z[\{\Phi(x)\}] = \sum_{(x_1,\ldots,x_n)} (\Pi_i\, n_i!)^{-1}\, [x_1,.....,x_n]\, \Phi(x_1).....\Phi(x_n) \qquad (2.29)$$

The cumulants may be evaluated from the moments using (2.23). The interesting fact is that the cumulants turn out to be almost the exact opposite of the moments in that $[x_1,.....,x_n]$ is nonzero only if the full set of polymers $\{x_1,....,x_n\}$ is connected, and

zero if even one of the polymers is disconnected from the rest. Hence the diagram expansion of Ln Z is in terms of connected sets of polymers, termed clusters. If we make the definition:

$$a(c) = (\prod_i n_i!)^{-1} [x_1,.....,x_n] \tag{2.30}$$

then we may express the cluster expansion of Ln Z in the compact form:

$$\text{Ln } Z = \sum_C a(C) \prod_{x_i \epsilon C} \Phi(x_i)^{n_i} \tag{2.31}$$

with C running over all possible clusters on the lattice.

All the clusters required for a 16^{th} order calculation of the free energy are presented in a table contained in ref. [6]. A certain amount of information about the phase structure of a lattice gauge theory can be obtained from studying the variation of the free energy as the coupling constant varies. In particular, a first order phase transition ought to result in a singularity in the free energy. For more definite indications of phase changes, however, one must examine the variations of non–local quantities such as the string tension and the mass gap. In the ensuing chapters we shall be examining methods for evaluating series expansions of the mass gap.

CHAPTER 3

COMPUTER EVALUATION OF THE STRONG COUPLING MASS GAP

In the Hamiltonian approach to lattice gauge theory the mass gap arises naturally as the difference between the first excited zero–momentum state and the lowest–energy eigenvalue of the Hamiltonian, but in the Euclidean approach the mass gap has to be obtained by a somewhat more oblique route.

Osterwalder and Seiler [31] prove that if the strong coupling expansion for an expectation value is convergent , then the connected correlation function of two finite–sized lattice states , separated by a distance along some lattice axis is bounded by an exponetial:

$$\langle \theta_1(0)\ \theta_2(t) \rangle - \langle \theta_1(0) \times \theta_2(t) \rangle < \text{const}\ e^{-mt}$$

The constant m is identified as the mass gap of the theory.

Kogut[32] employs the transfer matrix to provide a bridge between the Hamiltonian and Euclidean approaches which enables him to prove that

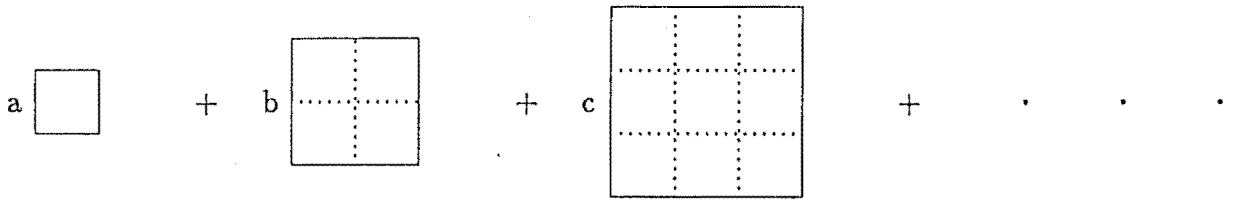
$$\lim_{t \to \infty} \left(\langle \theta_1(0)\ \theta_2(t) \rangle - \langle \theta_1(0) \times \theta_2(t) \rangle \right) = \text{const}\ e^{-mt} \qquad (3.1)$$

The constant m will be the mass gap of the theory provided that θ_1 and θ_2 are zero–momentum states with non–zero projection onto the lowest mass eigenstate of the theory.

3.1 Diagram expansion of the mass gap

Our first task, in order to calculate the mass gap, is to find a zero momentum

lattice state which projects onto the first excited state of the theory. If we choose to take our correlations parallel to the time axis then we must choose lattice states which are local in time. For simplicity we will choose states which are entirely contained within one time–slice of the lattice. In 2+1 dimensions, the spatial latice at each time–slice is obviously a two dimensional plane. Gauge invariant states on this lattice will be, therefore, products of U operators on links around planar loops. One assumes that the lowest mass eigenstate will transform trivially under the symmetry group of the time slice (i.e. the group of the square) and so must be of the form:



The simplest approximation is to take a=1, b,c,.... = 0, i.e. to let the single plaquette be our approximation to the first excited state. A single isolated plaquette, however, is a spatially localized state, and so is of indeterminate momentum. To obtain a state of definite momentum we must perform a fourier transform

$$\theta(p,t) = \sum_{x} e^{i\,p.x}\ \chi_f(U_p(x,t)) \tag{3.2}$$

So, to obtain a state of zero momentum, we simply sum over all the plaquettes in a time–slice:

$$\theta(0,t) = \sum_{x} \chi_f(U_p(x,t)) \tag{3.3}$$

To normalize the state we must divide by $\sqrt{N_t}$ where N_t is the number of plaquettes in a time slice. So our approximation to the zero momentum first excited state is:

$$\theta(0,t) = \frac{1}{\sqrt{N_t}} \sum_{\mathbf{x}} \chi_f(U_P(\mathbf{x},t)) \tag{3.4}$$

Hence the connected correlation function we must evaluate is:

$$\Gamma = \frac{1}{N_t} \sum_{\mathbf{x}_1,\mathbf{x}_2} \left\{ \langle \chi_f(U_P(\mathbf{x}_1,t))\ \chi_f(U_P(\mathbf{x}_2,0)) \rangle - \langle \chi_f(U_P(\mathbf{x}_1,t)) \rangle \langle \chi_f(U_P(\mathbf{x}_2,0)) \rangle \right\}$$

$$= \sum_{\mathbf{x}} \left\{ \langle \chi_f(U_P(\mathbf{x},t))\ \chi_f(U_P(0,0)) \rangle - \langle \chi_f(U_P(\mathbf{x},t)) \rangle \langle \chi_f(U_P(0,0)) \rangle \right\} \tag{3.5}$$

Of course we cannot evaluate this correlation function exactly, we must develop a diagram expansion to approximate the function. The first step in doing so is to note that

$$\frac{\partial}{\partial\beta_1}\frac{\partial}{\partial\beta_2}\cdots\frac{\partial}{\partial\beta_n} \int \prod_l DU_l\ e^{\beta_1\chi_f(U_{P_1})}\ e^{\beta_2\chi_f(U_{P_2})}\cdots e^{\beta_n\chi_f(U_{P_n})} \prod_{P'} e^{\beta\,\chi_f(U_P)} \Big|_{\beta_1,\ldots,\beta_n = \beta}$$

$$= \int \prod_l DU_l\ \chi_f(U_{P_1})\ \chi_f(U_{P_2})\cdots\chi_f(U_{P_n}) \prod_P e^{\beta\,\chi_f(U_P)} \tag{3.6}$$

where P' runs over all plaquettes except $\{p_1,\ldots,p_n\}$, and P runs over the entire set of plaquettes.

The integrand on L.H.S. of eqn. (3.6) may be thought of as the path integral of a theory defined on a lattice in which the coupling constant is uniform except on the set of plaquettes $\{p_1,\ldots,p_n\}$. This path integral will be denoted by $Z[\beta,\beta_1,\ldots,\beta_n]$, and eqn. (3.6) rewritten in the form:

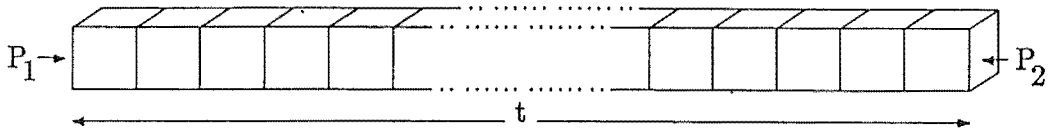
$$\frac{\partial}{\partial\beta_1}\frac{\partial}{\partial\beta_2}\cdots\frac{\partial}{\partial\beta_n} Z[\beta,\beta_1,\ldots,\beta_n] \Big|_{\beta_1,\ldots,\beta_n = \beta} = Z \langle \chi_f(U_{P_1}) \cdots \chi_f(U_{P_n}) \rangle \tag{3.7}$$

From which it follows easily that

$$\frac{\partial}{\partial\beta_1}\frac{\partial}{\partial\beta_2}\mathrm{Ln}(\,Z[\beta,\beta_1,\beta_2])\Big|_{\beta_1,\beta_2=\beta} = \langle\,\chi_f(U_{P_1})\,\chi_f(U_{P_2})\rangle - \langle\,\chi_f(U_{P_1})\rangle\langle\,\chi_f(U_{P_2})\rangle \quad (3.8)$$

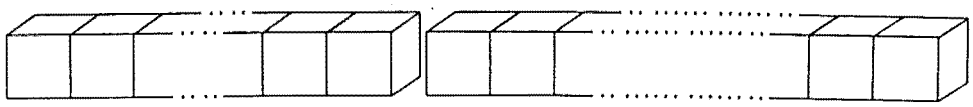
So, in order to obtain a diagram expansion for Γ, we must first produce a diagram expansion for $\mathrm{Ln}(Z[\beta,\beta_1,\beta_2])$ and then differentiate w.r.t. β_1,β_2. Obviously only those diagrams which contain both β_1 and β_2 will contribute to the connected correlation function; the derivatives will eliminate all other diagrams. We also recall that the cluster expansion for Ln Z will contain connected diagrams.

Let us, then, look at the sort of diagrams that will contribute to the correlation function. The lowest–order diagram is a long tube connecting the plaquettes P_1 and P_2



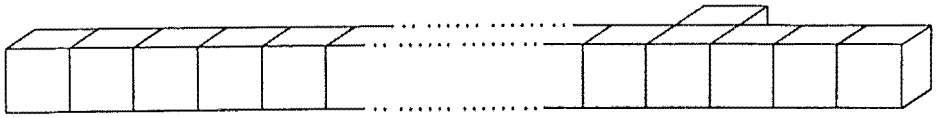
With all plaquettes in the fundamental irrep the contribution made by this diagram will be $\left[\frac{\partial a_1(\beta)}{\partial\beta}\right]^2 (a_0)^{N_t}\,(a_1(\beta))^{4t}$.

The first correction to this leading term is provided by a cluster consisting of two tubes sharing a single plaquette:



There will be t such diagrams, each contributing $-\left[\frac{\partial a_1(\beta)}{\partial\beta}\right]^2 (a_0)^{N_t}\,(a_1(\beta))^{4t+2}$

Then we have diagrams consisting of various 'warts' attached to the tube



And so it goes on.

Obviously, to any finite order, all the diagrams contributing to Γ will consist of long lengths of tube to which a variety of local "decorations" have been attached. It would be convenient to be able to ignore the lengths of tube and concentrate on the decorations alone. This is exactly the approach that is commonly taken when carrying out mass gap calculations, one generates all the possible decorations that will result in corrections to Γ up to some given order. Munster [33] carried out this process by hand to obtain 8^{th} order series for the mass gaps for gauge groups Z_2, Z_3, SU_2, and SU_3 in 2+1 and 3+1 dimensions. More recently Decker[25] has written a computer algorithm to carry out the process automatically for Z_2 in 2+1 dimensions.

It is Decker's algorithm that forms the starting point of the present work. Hence it is essential that we are fully conversant with the Decker's methods before we can start to consider the work I have done.

3.2 Review of Decker's Algorithm

We can divide our examination into three quite clear stages. The first stage involves proving that any cluster contributing to the correlation function Γ may be uniqely characterized by a set of decorations, and that the decorations in turn may be uniquely characterized by a set of cubes on the lattice. We may visualize this process as below :

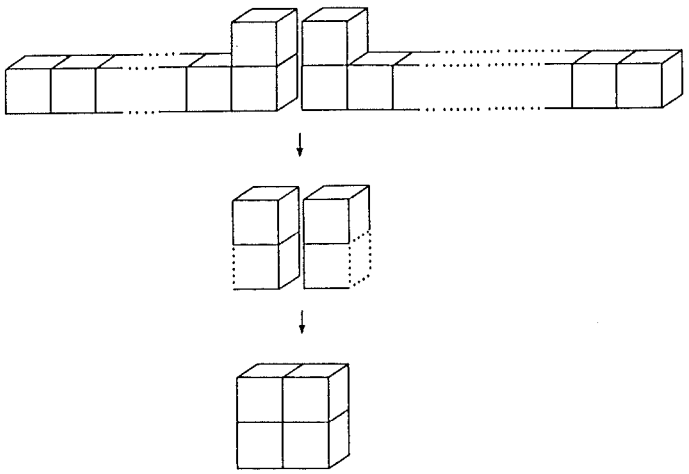


Fig 3.1 Decomposition from cluster to decoration to connected set of cubes

The first stage of our examination, then, involves establishing the uniqueness of this decomposition. It is in the second stage of the examination that we shall begin to look at the processes which actually occur in Decker's program. Knowing that every possible cluster contributing to Γ can be reconstructed from a connected set of cubes, we shall examine the way in which Decker's program generates all the decorations corresponding to a given set of cubes, and then calculates the total contribution made to the mass gap by all the clusters that may be formed from those decorations. We visualize this reconstruction process as:

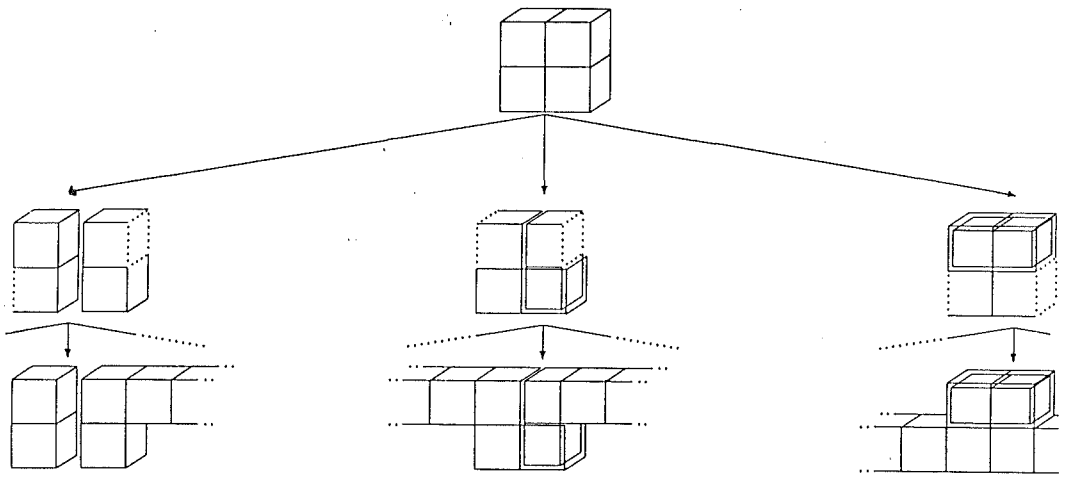


Fig. 3.2 Reconstruction from connected set of cubes to decorations to clusters.

The third and final stage involves examining the method by which one can uniquely generate all the sets of cubes required to produce all contributions to Γ up to a certain order in β.

3.2.1 Decomposition from Cluster to Connected Set of Cubes

Let us begin by defining some notation. Γ, of course, is the connnected plaquette–plaquette correlation function, but the quantity we wish to calculate is not Γ, but the correlation function truncated to some finite order in β. We represent this truncated correlation function by $\hat{\Gamma}$. The full set of all clusters contributing to $\hat{\Gamma}$ will

be designated by $\mathscr{C}$. Hence the first stage of our examination of Decker's algorithm involves taking a typical cluster $c \in \mathscr{C}$ and following through the process whereby it may be decomposed into a collection of one or more local sets of connected cubes.

Decker makes very precise definitions of a variety of diagram parts. Let us begin by informally reviewing some of these definitions.

A Support: It is useful to make a clear distinction between a diagram and it's support. A diagram is a set of plaquettes on the lattice, with each plaquette having been assigned an irrep of the gauge group. The support of this diagram is simply the set of plaquettes. In the case of Z_2 this distinction does not really exist since there is only one non–trivial irrep, however for any other gauge group the distinction is very real.

A Cluster: A cluster C is a collection of polymers :

$$C = (X_1^{n_1}, X_2^{n_2}, ...)$$

Each polymer X_i appears n_i times in the collection, n_i is referred to as the multiplicity of X_i. A cluster is always taken to be connected ,i.e. the support $|C|=|X_1| \cup |X_2| \cup |X_3| \cup .....$ is a connected set of plaquettes.

A Basic tube Part: We shall differ from Decker and define a basic tube part as follows – a basic tube part of length j lattice spacings is a set of 4j+2 plaquettes which entirely enclose a tube of length j lattice spacings and cross–section 1 plaquette, and which runs parallel to the time axis.

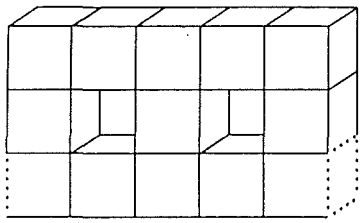
Having got these definitions out of the way, we may now concentrate on the very important task of defining decorations in such a way that there is no ambiguity about the way in which a cluster $c \in \mathscr{C}$ is decomposed into its constituent decorations. Let us start with a bare bones definition on which we may elaborate in order to

remove any ambiguities that may exist in the decomposition process.

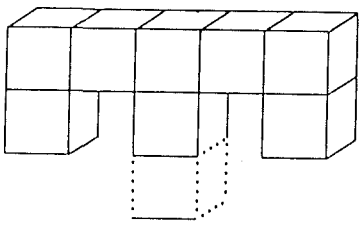
A Decoration: A decoration D is a cluster in which two plaquettes have been singled out and asigned the roles P_{in} and P_{out}. P_{in} and P_{out} are, of course, the two plaquettes at which the decoration would butt onto basic tube parts in the process of reconstructing the full cluster c

There are two essential qualities required of P_{in} and P_{out}. The first is that they must occur only once in the cluster. This means that there must be only 1 polymer X in the cluster whose support $|X|$ contains P_{in}, and similarly for P_{out}. The second requirement is that no basic tube part of any length extending in the positive t direction from P_{out} should share any plaquettes with the decoration except P_{out} itself, similarly for basic tube parts extending in the negative t direction from P_{in}.

That is, we desire a situation like (note that P_{in} and P_{out} are the plaquettes outlined by dotted links)



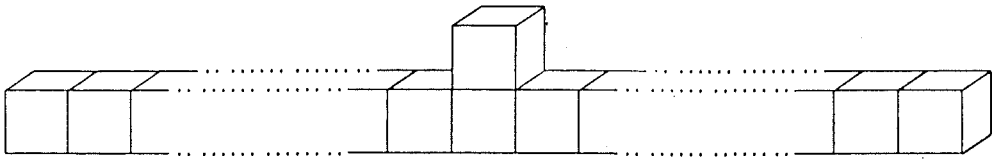
rather than



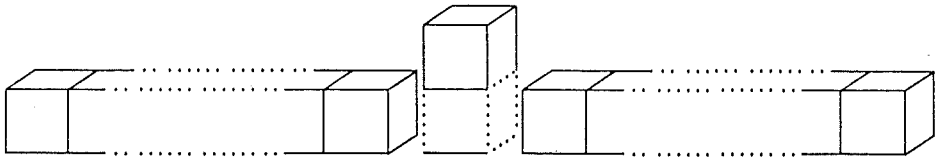
In this latter case P_{in} and P_{out} do not clearly mark the lower the lower–t and upper–t bounds of the decoration whereas they do in the former diagram.

So we now have our bare bones definition of the decoration, but this definition does not as yet preclude all ambiguity in the decomposition of $c \in \mathcal{C}$ into decorations

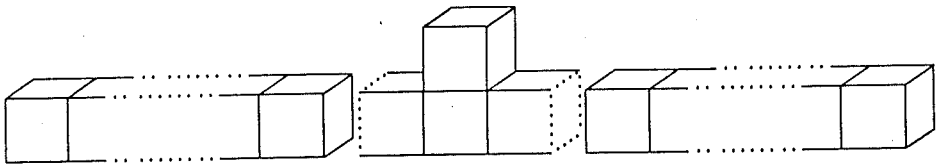
and basic tube parts. Consider the cluster below:



This may be decomposed as

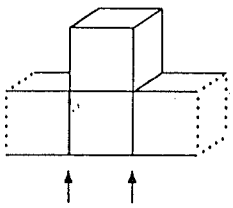


or as



etc.

The essential problem with a decoration like



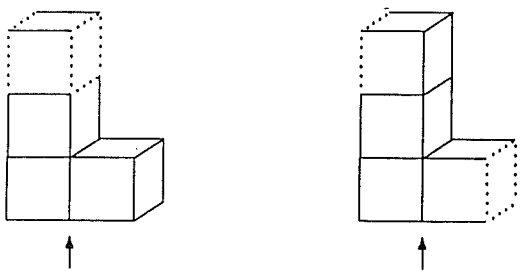
is that we have perfectly valid positions for P_{in} and P_{out} (indicated by the arrows) but have needlessly added lengths of basic tube beyond them. So decorations containing such embedded positions for valid P_{in}'s and P_{out}'s will duplicate the clusters constructed from decorations which have not been needlessly extended with lengths of basic tube. Hence we need to distinguish clearly between those decorations from which lengths of basic tube may be removed to reveal valid P_{in}'s or P_{out}'s , and those decorations for which this cannot be done. The former will be referred to as

reducible and the latter as irreducible. As a first step towards developing a rigourous distinction between these two types of decoration we introduce the concept of a separating bottle–neck.

A Separating Bottle–neck is essentially a space–like loop of four links at which a decoration may be cleanly split into two parts. As with P_{in} and P_{out} , the bottle–neck must be contained in only 1 polymer of the cluster, and that polymer must have multiplicity equal to 1. It is also essential that the plaquette contained within the loop of four links is not contained in the support of the cluster. The loops indicated by the arrows in fig 3.3 are separating bottle–necks.

Fig 3.3 Examples of Separating Bottle–necks.

Essentially a decoration is deemed reducible if it contains a separating bottle–neck that would in fact do service as a valid position for P_{in} or P_{out}.
For example , consider the decorations below



They both contain a separating bottle–neck , but only in the latter case could the existing P_{out} be validly replaced by the bottle–neck. Hence only that decoration is reducible at the bottle–neck.

Let us now formalize the distinction between reducible and irreducible

decorations. We must first define a quantity known as the order of a decoration. Let $t_{P_{in}}$ be the time coordinate of P_{in} and $t_{P_{out}}$ be the time coordinate of P_{out} for some decoration D. Let $||D||$ be the total number of plaquettes contained in the support $|D|$. The order O(D) is defined by the relation:

$$O(D) = ||D|| - 4(t_{P_{out}} - t_{P_{in}}) \tag{3.9}$$

Now we must define the process of expanding a decoration at a bottle–neck. To expand a decoration one splits it in two at the bottle–neck, translates the half above the bottle–neck by one lattice spacing in the positive t direction, and then forms a new diagram by filling the resulting gap with a basic tube section 1 lattice spacing in length.

At last we are in a position to express a rigourous criterion by which to judge whether or not a decoration is reducible. *A decoration is reducible if it contains a separating bottle–neck and the process of expanding the decoration at the bottle–neck does not alter its order.*

Having defined the irreducible decoration we have achieved our goal of being able to uniquely represent any $c \in \mathscr{C}$ by a set of local corrections. Simply, any cluster $c \in \mathscr{C}$ can be uniquely decomposed into a set of one or more irreducible decorations. Obviously there will be many clusters which may be represented by any one set of irreducible decorations $\{D_1, D_2, \ldots\ldots, D_k\}$, but any given cluster will be represented by one and only one set of decorations.

Our next task is to show that any decoration may be uniquely represented by a connected set of cubes. In the interests of clarity we shall differ from Decker's treatment just a little here. Decker considered decorations as being represented by a volume of space on the lattice rather than a set of cubes. He did this in the interests of rigour, so that his treatment would be equally as valid in four dimensions as in

three. However, our task here is to come to an understanding of Decker's algorithm, not to present a fully rigourous treatment.

Let us begin by defining the cube set corresponding to a polymer. Any polymer support can be thought as having been formed by placing a set of cubes onto the lattice and removing various of the plaquettes that are shared by more than one of the cubes. In four dimensions the set of cubes forming a given polymer is not necessarily unique, but in three dimensions it is unique.

In three dimensions, then, we define the cube set $J(|Y|)$ corresponding to the support $|Y|$ of polymer Y as that unique set of cubes which , upon being placed on the lattice with the removal of appropriate shared faces, form $|Y|$.

The cube set corresponding to the support S_c of the cluster $C = (Y_1^{n_1}, Y_2^{n_2}, \ldots, Y^{n_m})$ is simply union of the cube sets of all the constituent polymers:

$$J(S_c) = \bigcup_i J(|Y_i|)$$

In three dimensions, then, any decoration will have one and only one corresponding cube set, namely the cube set of the cluster which constitutes the decoration.

We have now completed the first stage of our understanding of Decker's algorithm, we have shown how any $c \in \mathcal{C}$ may be uniquely represented by a collection of one or more local sets of connected cubes.

Symbolically, the decomposition process is represented by

$$c \rightarrow \{D_1, D_2, \ldots, D_k\} \rightarrow \{J(S_1), J(S_2), \ldots, J(S_k)\}$$

3.2.2 Reconstruction of the Cluster from the Connected Sets of Cubes

We now wish to investigate the process by which all those clusters c may be

reconstructed from a collection of connected sets of cubes.

Let S_J be the set of all cluster supports which share a common cube set J. Obviously all $S \epsilon S_J$ will be of the form

$$S = (|Y_1|^{n_1}, |Y_2|^{n_2}, \ldots\ldots, |Y_m|^{n_m}) \quad ; \quad \bigcup_i J(|Y_i|) = J$$

Hence the process of reconstructing all the $S \epsilon S_J$ from the cube set J is simply a matter of generating all possible connected subsets w_i of J and then systematically producing all possible collections of the form $(w^{n_1}, w^{n_2}, \ldots)$, such that every cube in the set J appears in at least one of the w_i in the collection. There are obviously a number of different algorithms for carrying out this process, some more efficient than others.

Each of the cluster supports generated in this way will result in one or more different clusters upon the assigning of irreps to the constituent plaquettes.

The next step is to chose one of these clusters C_D and generate all the decorations $D = (C_D, P_{in}, P_{out})$. There are two steps to this process. First one must find all the plaquettes in the cluster which satisfy the conditions required of valid P_{in}'s and P_{out}'s. Then each possible P_{in}, P_{out} combination is considered – if the resulting decoration is irreducible then it is stored, if not it is discarded.

In this way one builds up a pool of all the irreducible decorations required to calculate m to some finite order in β. The final step is to systematically choose every possible set of decorations $\{D_1, \ldots\ldots, D_k\}$ from this pool and calculate the contribution made to m by all the clusters represented by $\{D_1, \ldots\ldots, D_k\}$. Let us denote the whole set of such clusters by $\mathscr{C}_k$.

Fortunately all the elements of $\mathscr{C}_k$ make identical contributions to m. Hence we need only calculate the contribution made by one generic element, and then calculate the number of elements of $\mathscr{C}_k$. The product of these two expressions will

then give us the contribution made to m by the set of clusters $\mathscr{C}_k$.

First we shall examine how one determines the number of elements of $\mathscr{C}_k$. This quantity will be denoted N_k and is refered to as the configuration number of $\{D_1,.....,D_k\}$. Basically, N_k is calculated by considering the number of different positions in which the decorations can be placed between two plaquettes separated by temporal distance t. There are some subtleties to this calculation, though. Recall from eqn (3.1) that the mass gap is obtained from the $t \to \infty$ limit of Γ. It is therefore important that the expressions obtained for N_k do not contain end effects which would not be present in the infinite t limit. As always, we can simulate an infinite tube by imposing periodic boundary conditions. So, the correct value for N_k (i.e. the value which results in an expression for $\hat{\Gamma}$ which exponentiates) is obtained by considering the decorations to have been placed on a tube periodic in time with period t lattice spacings. Hence N_k is the number of different positions in which we may put the decorations within 1 period. If k=1, i.e. we have just one decoration, then N_k will always be t; there are t different positions in which we may place the decoration before we start to repeat patterns that have already been seen.

If k>1, however, the value of N_k will depend on both the size and shape of the decorations. Obviously decorations are not allowed to overlap, so the finite lengths of the decorations reduce the number of different ways they may be arranged. The number of arrangements is further reduced by the fact that the *exclusion volume*, or minimum possible seperation, between a pair of decorations is not always zero.

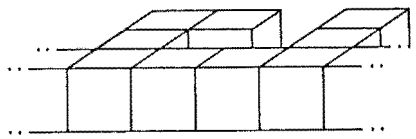


Fig 3.4 A Pair of Decorations With Non–Zero Exclusion Volume.

The number of positions available to the set of decorations, then depends on

the lengths V_i of each decoration D_i and the exclusion volumes V_{ij} between each pair of adjacent decorations D_i,D_j. In general the form of the expressions will be:

$$N_k = \frac{1}{k!} t^k - F(V_1, V_2, \ldots\ldots, V_k, V_{12}, V_{23}, \ldots\ldots, V_{k-1\ k}, V_{k1}) \tag{3.10}$$

Where $F(V_i, V_{ij})$ is a function that takes account of the finite extent of the decorations. The calculation of the functions $F(V_i, V_{ij})$ is a just a matter of simple combinatorics.

Of course, our aim is not to calculate $\hat{\Gamma}$, but to calculate $m = \frac{1}{t} \mathrm{Ln}\ \hat{\Gamma}$. Hence we are only interested in the t–linear term of N_k. This is referred to as the reduced configuration number $N_k^r(V_i, V_{ij})$. A table of the N_k^r values for k = 2,3,4 are given in table 1 of ref. [25].

Now we turn to the other factor, the activity of a generic element of $\mathscr{C}_i$. If we write $\hat{\Gamma}$ in the form $\hat{\Gamma} = a_0 U^{4t} (1+\Delta\hat{\Gamma})$ then m will be of the form:

$$m = \mathrm{Ln}\ a_0 + 4\ \mathrm{Ln}\ U + \Delta m \tag{3.11}$$

where Δm is the t–linear part of $\Delta\hat{\Gamma}$. The contribution made to Δm by any cluster c will be of the form $a(c)\ \hat{\phi}(c)$. The relative activity $\hat{\phi}(c)$ is given by

$$\varphi(c) = \phi(c)/U^{4t} \tag{3.12}$$

where $\phi(c)$ is the activity of the cluster. Fortunately, as shown in ref. [25] , both $a(c)$ and $\hat{\phi}(c)$ factorize into a product over the decorations contained in c:

$$a(c)\ \hat{\phi}(c) = \prod_{j=1}^{k} a(D_j)\ \hat{\phi}(D_j) \tag{3.13}$$

Hence, the contribution made to Δm by the set of clusters $\mathscr{C}_i$ is given by

$$\Delta m(\mathscr{C}_i) = N_k^r(V_1,.....,V_k,V_{12},.....,V_{k1}) \prod_{j=1}^{k} a(D_j)\, \hat{\phi}(D_j) \qquad (3.14)$$

We have now completed the second stage of our examination of Decker's program.

3.3.3 Producing the Cube Sets

The central nub of Decker's program is an algorithm for generating all possible translationally distinct configurations of connected cubes on the lattice with no omissions or duplications. It is this algorithm that we shall now examine.

One could imagine an algorithm which starts by choosing a base cube. The first configuration, then, consists of just the base cube. Pairs of connected cubes are produced by forming sets consisting of the base cube and one of its neighbours. Configurations of three cubes are formed by taking each pair in turn and adding on third neighbouring cubes and so on. Unfortunately, one quickly runs into problems; it is very difficult to avoid duplications. This problem of duplicating configurations has long been a major obstacle to the development of an efficient algorithm for Euclidean lattice calculations. The important point about the algorithm presented in ref. [25] is that it manages to solve the problem.

The solution is contained in a set of rules for determining which cubes may and may not be added to any given configuration. Before presenting the rules, we introduce some new concepts.

When calculating the mass gap to finite order there will be an upper limit to the dimensions of the decorations that need to be considered. Hence we may consider our configurations as being formed on a finite lattice, each of whose unit cubes can be

numbered. The numbering we shall adopt for an LxLxL lattice is:

$$(x,y,z) \rightarrow x + (L+1)y + (L+1)^2 z$$

where (x,y,z) is the bottom, front, left hand corner of the cube. Any configuration can therefore be represented by an ordered set of numbers, $\{a_1,a_2,....,a_k\}$. The ordering of the numbers represents the order in which they have been added to the configuration.

Let us take special note of two particular members of a configuration. The most recently added member of the configuration we shall label r. At least one, and possibly more, members of the configuration will be neighbouring to r on the lattice; that neighbour of r which appears earliest in the configuration we shall label q.

We are now in a position to present the rules for extending a configuration.

A cube s may be added to the configuration $K^{(M)} = (a_1,a_2,....,r)$ if:

(i) $s > a_1$

(ii) $s \notin K^{(M)}$

(iii) s is not neighbouring to any $a_i \in K^{(M)}$ such that a_i appears earlier in the configuration than q.

(iv) s is a neighbour to q or some $a_j \in K^{(M)}$ such that a_j appears later in the configuration than q.

(v) If s is a neighbour of q then $s > r$.

The proof that these rules will garuntee a complete and unique set of connected cube configurations is given in ref. [25].

Those cubes which, according the the rules above, may be added to a given configuration are referred to as being free with respect to the configuration. When implementing his algorithm on a computer, Decker stored the set of points free with respect to each configuration in an associated array he referred to as the Marked Lattice Site List (MLSL). When extending a configuration, the program simply runs

through the associated MLSL, generating one new configuration with each free cube.

It is important to reiterate the fact that I have differed somewhat from Decker in my treatment of these connected configurations. Strictly speaking, a cluster is uniquely characterised by a volume of space on the lattice, not a connected set of elementary cubes. However, a thourough discussion of the ambiguity that arises in 3+1 dimensions is given in chapter 5. In the meantime the problem may be ignored as it only serves to obscure any attempts to come to an understanding of the nature of Decker's algorithm.

I have also chosen not to mention the dual lattice which is introduced in the latter part of ref. [25]. In ref. [25], cubes are considered to be represented by points on the dual lattice. I consider this to be an unnecessary complication, it is quite sufficient to consider cubes as being labelled by the position of their bottom, front, l.h. corners. Hence, I have not introduced the idea of connected point configurations, and have dealt instead with connected configurations of cubes. The cube configurations referred to in this and subsequent chapters can be thought of as entirely equivalent to the point configurations of ref. [25].

The brief description of Decker's algorithm is now complete. In the next chapter we shall investigate the way in which the basic elements of this algorithm may be recombined in a more efficient manner.

CHAPTER 4

THE NEW IMPLEMENTATION

In the previous chapter I have described an algorithm for computer evaluation of the strong coupling expansion of the mass gap. It is now my purpose to examine more closely some of the details of the implementation of the algorithm. Obviously it can be implemented in a number of ways, some more efficient than others. Let us henceforth refer to the implementation described in ref. [25] as $\mathscr{D}$. In section 6 of ref. [25] a flow chart is set out which describes the gross features of $\mathscr{D}$. Throughout this chapter I shall refer repeatedly to that flowchart, which is reproduced at the end of this chapter (appendix 4.1), using it as a convenient way of pinpointing which portion of the program is being discussed at any given time. The algorithm $\mathscr{D}$ will be dissected and the elements reassembled into an improved implementation, culminating in a flow chart significantly different from Decker's. This new implementation will be referred to as $\mathscr{R}$.

4.1 Exploiting the Tree Structure

Let us begin by examining the way in which connected sets of cubes are generated. Note that at step (2) of $\mathscr{D}$ all possible k cube configurations (i.e. configuations containing k elements) are generated and stored. In $\mathscr{D}$ only one size of configuration is treated at any one time. All the decorations derived from k cube configurations are generated and stored before the program moves on to generate any k+1 cube configurations. We can think of this as a 'layered' approach, the whole plane of k cube configurations is filled in before one moves on up to the next layer – the plane of k+1 cube configurations.

It is mentioned in ref. [38] that more than 1.5×10^6 configurations had to be generated in the course of the 16^{th}–order calculation. Obviously not all these

configurations had to be stored at once, but there would have been stages when at least 10^6 configurations would have been in memory. A good deal of computer memory is required to store the three arrays coresponding to a single configuration. Hence, calculations at higher orders will undoubtedly require more memory than is available on current computers. It would be far more desirable to implement the algorithm in such a way that memory requirements were essentially independent of the maximum order to which the mass gap was being calculated. We shall now examine just how we may go about achieving exactly this desired result.

Recall that a k cube configuration is generated by adding a cube onto a k–1 cube configuration, i.e. larger configurations are created by extending smaller ones. We shall refer to the central process of creating a k cube configuation from a k–1 cube configuration as an *extension step.* Recall from chapter 3 that, in order to avoid duplications, only a certain set of cubes may be used to extend any given configuration. A list of these cubes is always stored along with the configuration in an array known as the marked lattice site list (MLSL).

The *extension step*, then, involves four operations

(i)Recalling the k–1 cube configuration and its corresponding MLSL from memory.

(ii)Choosing a free cube from the MLSL and adding it the configuration to form a k cube configuration.

(iii)Generating the MLSL corresponding to this new k cube configuration using the rules set out at the end of chapter 3.

(iv)Storing the k cube configuration and its corrresponding MLSL in memory

As was the case in $\mathscr{D}$, so also in $\mathscr{R}$ is the extension step the fundamental building block in the algorithm for generating all possible connected cube configurations. The essential difference between $\mathscr{D}$ and $\mathscr{R}$ is the order in which the configurations are

generated.

Since larger configurations are generated by extending smaller ones, the configurations may be thought of as residing at the nodes of a tree diagram. Every k cube configuration is derived from one and only one k–1 cube configuration and every k cube configuration in turn produces a set of 1 or more k+1 cube configurations. The tree stems from the single one–cube configuration and radiates out to the ever increasing numbers of larger configurations.

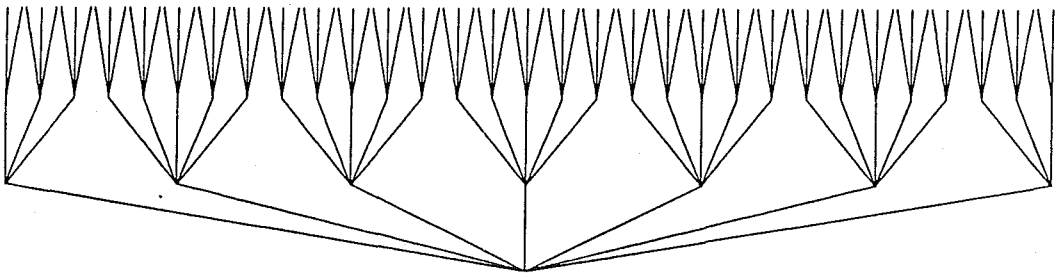


Fig 4.1 Tree Structure Underlying the Generation of Connected Configurations

The branches on the tree diagram above can be considered to represent extension steps. It is immediately obvious from looking at the diagram that only m–1 extension steps are ever required to generate any m cube configuration by starting from the common base point and following the most direct path to the desired configuration. All the memory that would be used in the course of generating the configuration in this manner would be that required to store the information relevant to the m–1 configurations lying in the path between the base point and itself. So, it far more memory efficient to generate configurations by simply following the paths in the tree diagram rather than to generate and store all the configurations in each layer prior to moving on to the next layer.

Having pointed out that this 'tree' approach is a very memory efficient way to generate some randomly chosen configuration, we must determine whether it is possible to systematically generate *all possible* configurations of up to and including p cubes without *ever* having to simultaneously store information on more than p diffrent

configurations.

Obviously one can envisage following a path through the tree which starts by branching out to the extreme left say, and then gradually working back to the right by moving up and down the branches until all the nodes of the tree have been visited. The difficult requirement is the fact that we do not wish to store more than p configurations worth of information. So it is vital to have a method of determining which of the possible extensions of any given configuration have already been generated and which are still to be generated. Fortunately a solution arises quite naturally from the method we have employed for generating configurations. Recall from our definition of the extension step that whenever a new configuration is produced, all those cubes which are free with respect to it (i.e. all its possible extensions) are determined and stored in the corresponding MLSL. Hence, all one need do to be sure of the extensions still to be made from any given configuration is simply to work through the MLSL in one direction and store a pointer indicating which element of the MLSL is to be used at the next extension.

This means that when a configuration is stored, not only do we store the information relating to the identity of the configuration, but *also* the direction in which to branch when that node is encountered again.

Hence, the systematic generation of all the configurations on the tree would proceed as follows:

(i) Starting from the base point, perform p−1 extension steps to obtain a p cube configuration. At each extension step always choose to extend the k cube configuration by adding on the first element of its MLSL, and then move the pointer on to the second element. Store each configuration produced along the way. Also store the MLSL corresponding to each configuration, and the pointer. (see fig 4.2)

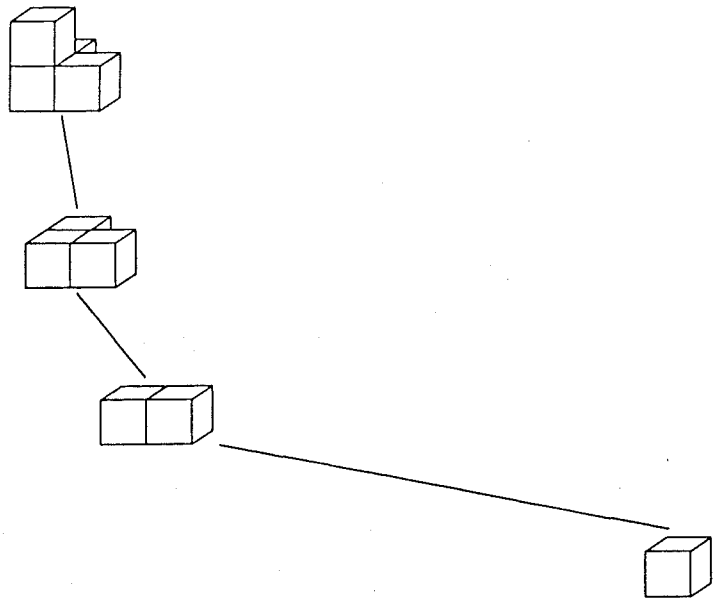


Fig 4.2 The First P Extension Steps in the Systematic Generation of Configurations

(ii) Having formed the first p cube configuration, go back and form the second possible extension of the p–1 cube configuration. Move the MLSL pointer along one place. Then form the third possible extension and so on until the MLSL is completely exhausted. (See Fig 4.3)

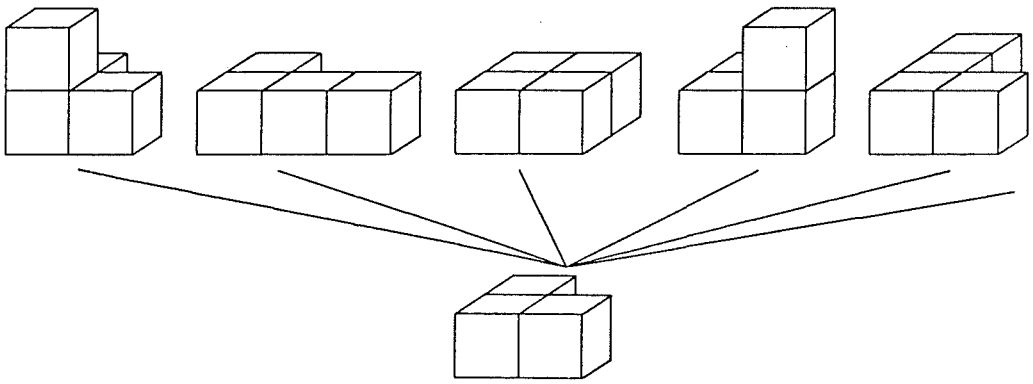


Fig 4.3 Forming All Possible Extensions of the First P–1 Cube Configuration

(iii) Return to the p–2 cube configuration and form a new p–1 cube

configuration – from which another set of p cube configurations can be formed. (See Fig 4.4)

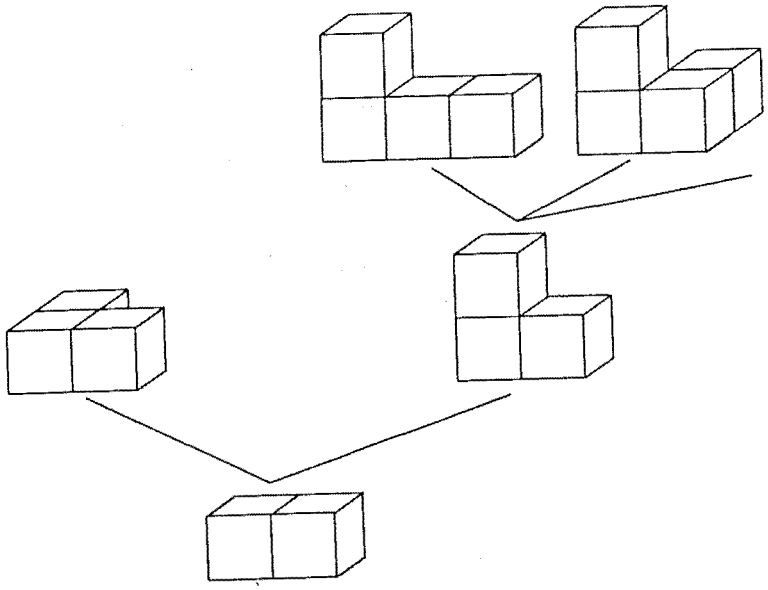


Fig 4.4 Moving on to the Next P–1 Cube Configuration

(iv) Continue this process of making extensions to a k cube configuration until its MLSL is exhausted and then moving back to the k–1 cube configuration etc. until the MLSL corresponding to the base cube is entirely exhausted.

In this way all possible contigurations of up to p connected cubes will be generated.

So we now have a picture of the fundamental alteration I have made in the order in which configurations are generated, and the resultant massive saving in memory usage. At no time is it necessary to store the information relevant to more than p configurations. This contrasts markedly with the massive memory requirements of $\mathscr{D}$. It is important to note that although the memory requirements of the program will increase slighty as the number of terms in the series is increased (the size of the largest decoration will increase) there is absolutely no danger of their ever prohibiting

the calculation of further terms.

Before going on to describe further alterations I have made to the implementation let us introduce some further terms which will enable us to discuss the tree with more precision. We shall make full use of the analogy with a family tree.

The ancestors of a given configuration are all those configurations which lie in the unique path which connects it to the base point.

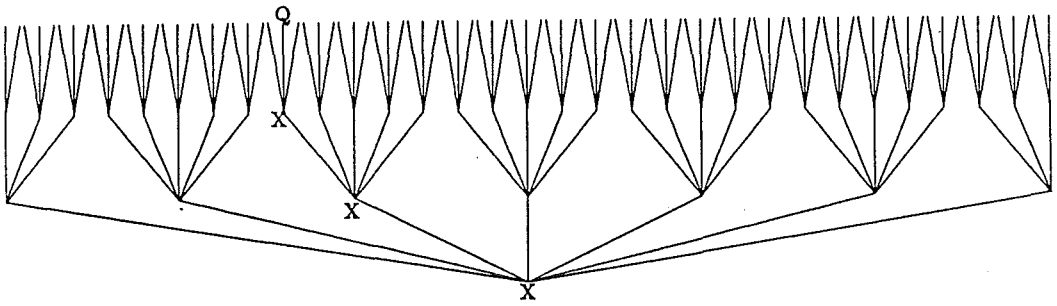


Fig 4.5 Ancestors of Configuration Q Marked by X.

The descendants of a given configuration are all those configurations which count it among their ancestors.

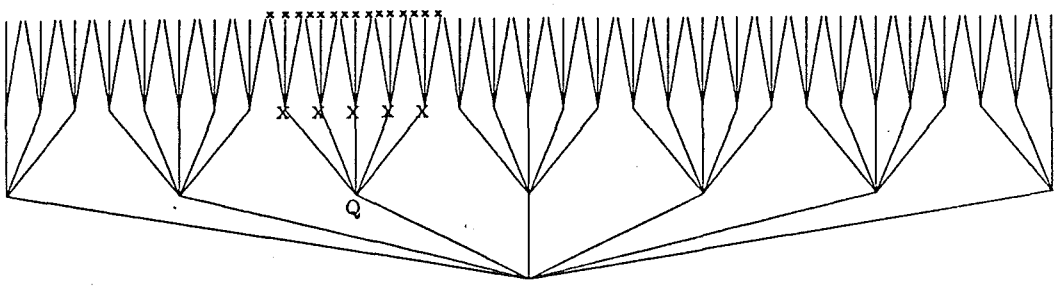


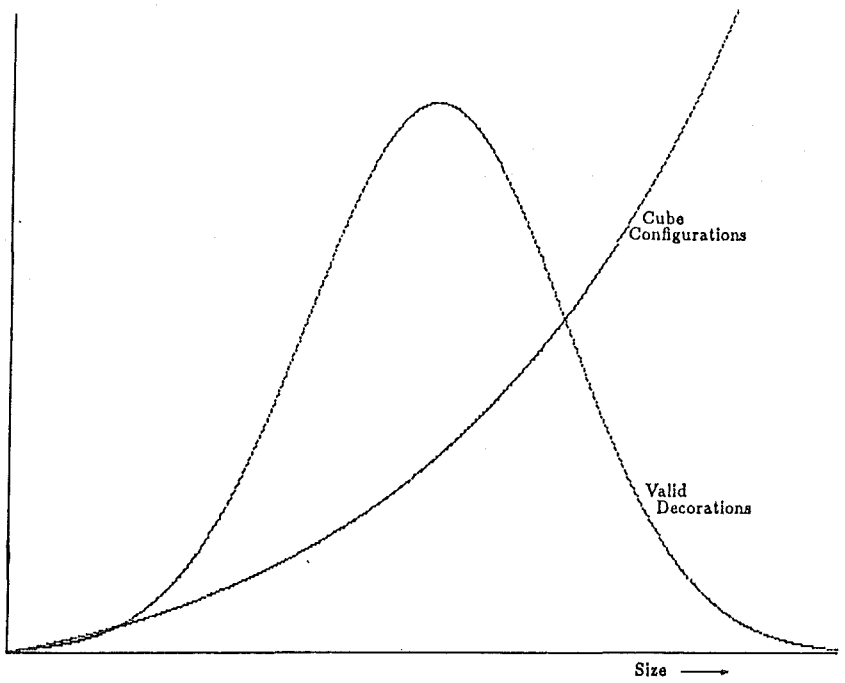
Fig 4.6 Descendants of Configuration Q Marked by X

4.2 Minimizing the Generation of Redundant Decorations

Exploiting the tree structure allowed us to decrease the memory usage of the program, let us now examine the means by which we have increased the speed of the program.

In $\mathscr{D}$, it is necessary to generate all possible k cube configurations for each value of k up to some maximum value p at which no new decorations are found which

contribute at $\mathcal{O}(\beta^m)$ or less. However the number of contributing decorations found at each value of k tends to follow something of a bell–shaped curve as k increases, whilst the number of connected cube configurations increases exponentially with k.



Hence, for large values of k, there are a lot of configurations formed which will not result in any contributions to the $O(\beta^m)$ mass gap. The program would obviously be more efficient if the generating of such fruitless configurations could be avoided. This means identifying those configurations from which it is pointless to make any extension steps, i.e. those whose descendants are all going to be fruitless. We shall dub a configuration with entirely fruitless descendants a *sterile* configuration.

Once a configuration is recognised as being sterile the program could return to its immediate anscestor (its parent configuration) and proceed with the next extension of that configuration. That way, time would not be wasted on the generating and processing of the descendants of the sterile configuration

Obviously the tricky part of the process is designing criteria for determining whether or not a configuration is sterile. We must choose criteria which never result in the discarding of a configuration which does have fruitful descendants, but which do detect as many as possible of the sterile configurations. Unfortunately it is extremely

difficult to design optimal criteria for determining the fruitfullness or otherwise of the descendants of a cube configuration, because such a variety of different decorations may be produced from any one configuration. Hence I altered the program even further so that it *directly generates* decorations.

Let us begin by determining the meaning of the term 'directly generate'. In the implementation that we have developed thus far, the nodes in our tree have been connected cube configurations, and the extension step which has moved us from one node to another has been a process for extending connected cube configurations. Having generated a certain connected cube configuration, the program would then have to branch into those procedures which produce decorations from cube configurations. Directly generating decorations means working with a tree in which the nodes are actually decorations, so upon reaching a node there is no need to branch off to other procedures for producing decorations, the process of moving to the node generated a new decoration.

In $\mathscr{D}$ there are essentially two steps involved in forming the decorations from a connected cube configuration. In step (4), the cube is partitioned to formed clusters, and in step (6) a P_{in}, P_{out} pair is chosen on the cluster to form a decoration. Therefore there are two aspects to the alterations required to produce a program which directly generates decorations.

4.2.1 Direct Generation of Clusters

Let us first consider how we are to produce clusters rather than cube configurations. We must define an extension step which expands a given cluster into one which contains one more cube. There are two essential differences between extending a cube configuration and extending a cluster. The first is that, in general, a cluster will contain more than one polymer, so there is a choice as to which polymer the new cube may be added. An example of this choice is illustrated in Fig 4.7.

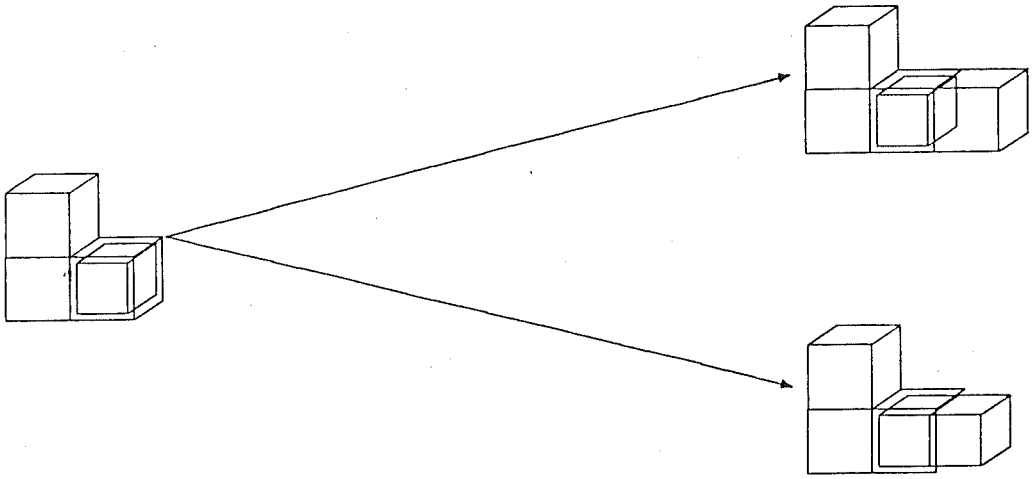


Fig 4.7 Two Different Clusters Resulting From the Addition of the Same Cube

This choice does not exist when extending a connected cube configuration (which is effectively a single–polymer cluster). The second major difference is the fact that a single cube may appear several times in a given cluster, so a cluster may be extended by the repetition of a cube already contained within it.

Fig 4.8 The Extension of a Cluster By the Repeatition of a Cube

First we must determine the set of cubes which may be added to any given cluster, i.e. the contents of the MLSL which will be associated with the cluster. It is vital that we design the contents of the MLSL in such a way as to avoid duplicating clusters. Obviously clusters are not duplicated in the algorithm $\mathscr{D}$. We shall make use of this fact in designing the extension step for clusters. In $\mathscr{D}$ clusters are formed by partitioning connected cube configurations (see fig 4.9).

Recall from chapter 3 that the connected cube configuration from which the cluster C is formed is represented by $J(S_C)$. Recall also that for any given cluster C there is one and only one corresponding cube configuration.

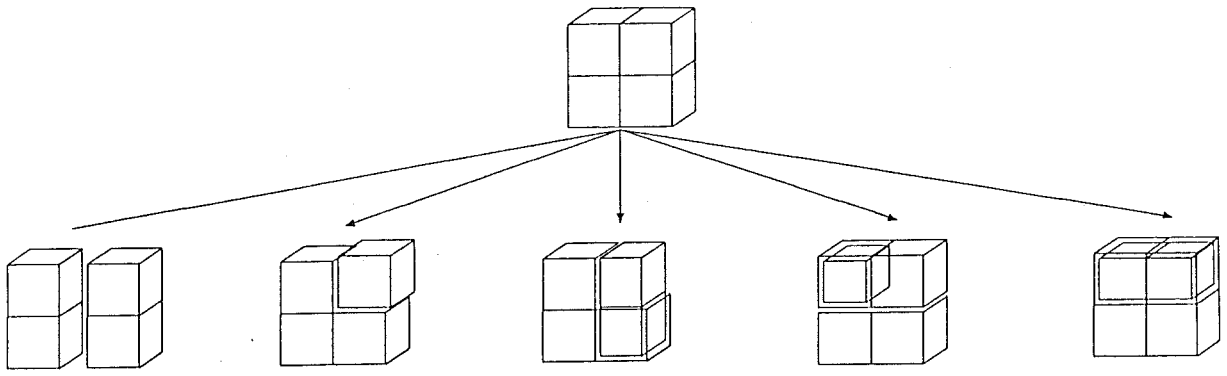


Fig 4.9 Partitioning a Connected Configuration

In the algorithm $\mathscr{D}$ one is assured of not duplicating clusters by the fact that one is certain of not duplicating cube configurations and the fact that clusters formed from two different cube configurations cannot be identical. However, when one is directly extending the clusters themselves, rather than going via the cube configurations, this is no longer necessarily guaranteed. We must define our method of extending clusters in such a way that this division between the clusters corresponding to different cube configurations is maintained. The extension step must be designed such that a cluster C corresponding to some cube configuration $J(S_C)$ may only be extended to clusters C' whose corresponding configurations $J(S_{C'})$ are valid extensions of $J(S_C)$ under the rules set out on page 33.

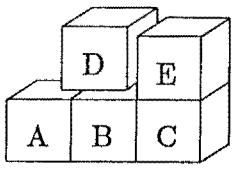
This is obviously achieved by insisting that the cubes which are deemed free with respect to cluster C are just exactly those which are free with respect to $J(S_C)$, i.e. that the MLSL corresponding to C will contain all the cubes which are free with respect to $J(S_C)$. This is the necessary condition to ensure no duplications, the strict order it imposes on the sequence in which the cubes have to be added will ensure that there will be just a single unique path from the base cube to any cluster on the tree.

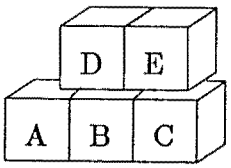
But is it a sufficient condition that ensures we don't miss any clusters? The simple answer is 'not quite'. It would be sufficient if there were no repetitions of cubes

within clusters. To accomodate the sort of extension illustrated in fig 4.8 the MLSL must contain at least one cube which is already a member of the cluster. In fact only the cube which was most recently added to the cluster need be included in the MLSL. This ensures that there is no ambiguity in the order in which cubes are added to a cluster, one can be sure that all the repetitions of any given cube were completed before the next new cube was added to the cluster.

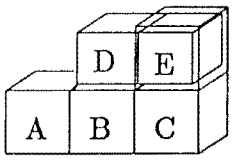
So, the MLSL corresponding to cluster C is the MLSL of $J(S_C)$ with the addition of the cube which constituted the most recent extension of C.

We have now determined the contents of the MLSL, the next task is to consider the way in which these cubes are to be added to the cluster. This means dealing with the first major difference between clusters and cube configurations, namely the fact that clusters generally contain more than one polymer. Basically, given a cluster and a cube, one simply considers each polymer of the cluster in turn, if the cube is connected to the polymer, then form an extended cluster, if not then move on to the next polymer. However, in the case of the final element of the MLSL, the cube which is being repeated, things are not quite so simple. Consider two different clusters:





if we then carry out the repetitions by distributing the cube E over all the polymers of both clusters, then we would form the following cluster twice.



This possibility is averted if we decide to desist from distributing the repeated cube upon finding a polymer already containing the cube.

There is one further aspect to the extending of a cluster which we must consider. Up until now we have only discussed the possibility of extending the cluster by connecting a cube onto an already existing polymer. It is also possible to extend a cluster by forming a whole new polymer.

Fig 4.10 Expanding a Cluster by the Creation of a New Polymer.

The systematic process of finding all the possible expansions of a cluster then proceeds as follows.

(i) F = 1 (F is the pointer to the current member of the MLSL)

(ii) Choose the F^{th} element of the MLSL – the expansion cube

(iii) R = 1 (R points to the polymer row of the cluster to which we are presently making additions)

(iv) If the expansion cube is connected to the R^{th} polymer of the cluster (but is not already contained in the R^{th} polymer) then form the new cluster in which the expansion cube is added to this R^{th} polymer

(v) R → R+1

(vi) Continue at (iv) until reaching the last polymer of the cluster, or (in the

case of the final member of the MLSL) finding the expansion cube to be contained in the R^{th} polymer.

(vii) If the expansion cube has not yet been found to be already a member of one of the polymers then expand the cluster by adding on a new polymer containing just the expansion cube.

(viii) $F \rightarrow F+1$

(ix) Continue at (ii) until the MLSL is exhausted.

The two running variables in this process are F, which indicates which element of the MLSL is presently being being investigated, and R, which designates the last row to which the expansion cube was added. So, provided the MLSL, F and R are stored, the expansion process can be picked up exactly where is was left off. This means that as was the case in section 4.1, when a cluster is saved to memeory, the information which determines the direction in which to branch upon next encountering it is also saved.

So, we now have the necessary elements for the development of a memory efficient algorithm for systemmatically generating clusters. The program systematically moves along the branches of the tree, guided by the clear and simple information stored at each node. However, we wish to create a tree in which all the nodes are decorations, not merely clusters. A few more alterations are still required in order to achieve that goal.

4.2.2 Direct Generation of Decorations

There are basically two ways in which this final step can be undertaken. One could define an ordering on the possible P_{in}, P_{out} pairs which can be chosen from any

cluster and create yet more branching in the tree. On the other hand it is possible to proceed in such a manner that only one decoration is formed from any cluster, and the positions of the P_{in} and P_{out} plaquettes are uniquely defined for any given cluster. I have taken the latter course of action.

In general, several P_{in}, P_{out} pairs can be chosen on any given cluster. The pairs can always be grouped into classes according to the relationship between the spatial (as against temporal) coordinates of P_{in} and the spatial coordinates of P_{out}.

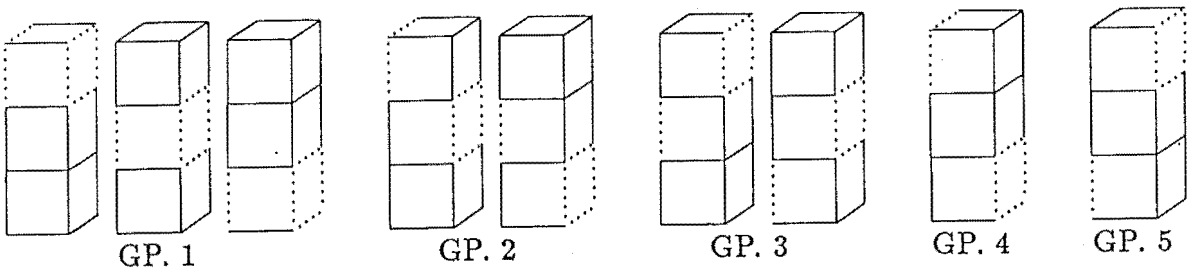


Fig 4.11 Groupings of P_{in} , P_{out} Pairs According to Spatial Relationship

Consider grouping all possible decorations according to this spatial relationship between P_{in} and P_{out}. The number of decorations in any given group which are formed from one given cluster will be less than or equal to the total number of decorations which may be formed from the cluster. However, as can be seen in fig 4.11, the number has not been narrowed down to just one (or no) decoration per cluster per group. Let us subgroup again, according to the actual spatial position of P_{in} and prove that in these new groups there will definitely be no more than one decoration per cluster per group.

For any given set of spatial coordinates there can never be more than one possible P_{in} candidate in a given cluster. Consider attempting to construct a cluster in which two P_{in} candidates sharing the same spatial coordinates. One of the plaquettes must have a higher time coordinate than the other. A length of pure tube extending in the negative–t direction from the plaquette with the higher time coordinate must intersect the other P_{in} candidate. Hence, by the definition of P_{in} given in chapter 3,

the plaquette with the higher time coordinate is not a valid P_{in}. Hence it is impossible to construct a cluster in which two or more potential P_{in} plaquettes share the same spatial coordinates.

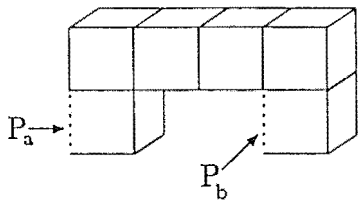


Fig 4.12 P_a and P_b Cannot Both Valid P_{in} Positions

Similarly it is not possible to have more than one valid P_{out} candidate in any given cluster with any given set of spatial coordinates. In our new groupings, the position of P_{in} is specified, the relationship between P_{in} and P_{out} is specified, so the position of P_{out} is effectively specified. Hence, there cannot possibly be more than one decoration per cluster per group.

We can now see how it is possible to implement our program in such a way that as soon as a cluster is formed, there can be no more than one decoration formed from it. Obviously we shall have to to run our program several times, each time specifying a new set of spatial positions for both P_{in} and P_{out}. We thereby treat each of the subgroups described above one at a time. The aim of designing a method of directly generating decorations has now been achieved.

4.3 Tests for Sterility

Recall that the whole purpose of altering the program to directly generate decorations was to enable us to more easily recognise those decorations which will not have any fruitfull descendants. Let us now look at some examples of sterile clusters and thereby gain a feel for the sort of tests required to detect such clusters.

There are two reasons why a decoration will fail to contribute to the $O(\beta^m)$

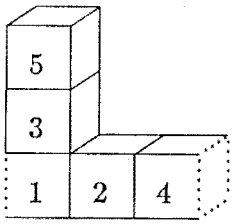
mass gap:

(i) It does not represent a valid irreducible decoration.

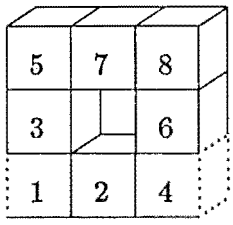
(ii) It contributes at higher order than $O(\beta^m)$.

Therefore sterile clusters will fall into two categories. Let us examine each category in turn.

Consider the decoration below:



The numbers indicate the order in which the cubes would be added to the decoration. This decoration will not contribute to the mass gap, there is a reducing bottle neck between cubes 1 and 2. But, will any of the descendants of this decoration contribute to the mass gap? According to the rules set out in chapter 3 for expanding a cube configuration, new additions to the decoration may not be neighbours of either of cubes 1 and 2. This does not prevent us, however, from building up the decoration until the reducing bottle–neck is no longer reducing. For example, the addition of 3 more cubes would result in the decoration:



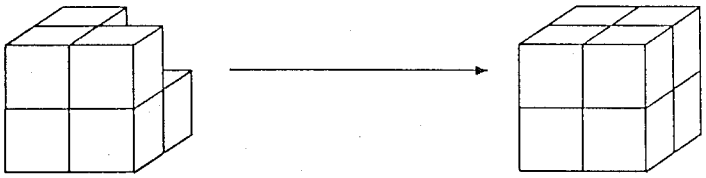
which no longer contains a reducing bottle–neck.

In fact it is not possible to bridge the bottle–neck with the addition of any smaller number of cubes. The latter decoration contributes to the mass gap at $O(\beta^{22})$. Hence, the former decoration will only have fruitfull descendants if we are calculating the mass gap to $O(\beta^{22})$ or higher. In any calculation to less than $O(\beta^{22})$ it would be

sterile.

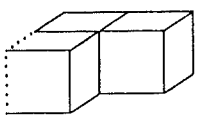
So the tests for sterile clusters belonging to the first category basically concentrate on detecting overly long stretches of pure tube.

Most decorations which already contribute at $O(\beta^m)$ fall into the second category of sterile decorations. However, even in 2+1 dimensions, it is possible to add new cubes to a decoration without increasing its order.

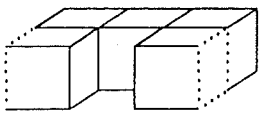


This sort of occurence is even more prevelant in 3+1 dimensions. It is necessary, therefore, that the tests for the second category of sterile clusters take into account the possiblity of there being cubes in the MLSL which have three or more neighbours already present in the cluster. Essentially, the tests for the second category of sterile clusters are actually tests for non–sterility. A decoration contributing at $O(\beta^m)$ or greater is taken to be sterile unless saved by the contents of its MLSL.

There is one specific group among the second category of sterile clusters which ought to be mentioned. There will be clusters from which no decorations can be formed given the restrictions placed on the positions of P_{in} and P_{out}. Consider the cluster illustrated below; if we stipulate that P_{in} and P_{out} must be directly in line then there is no valid position for a P_{out} plaquette .



The addition of two more cubes to the cluster shown above will result in a cluster for which it is possible to define a P_{out}.



If we are calculating the mass gap to $O(\beta^{16})$ this decoration would not contribute, and the cluster would be deemed sterile.

Hence, I have included a check specifically for those clusters which do not have a valid P_{in} or a valid P_{out}. The purpose is to determine how many cubes would have to be added to the cluster before a valid decoration could be formed from it. If the addition of those cubes results in a decoration which contributes beyond $O(\beta^{m})$ then the cluster is sterile.

We now have an idea of the sort of clusters which are likely to prove sterile, and the sort of tests required to detect them. Procedures to detect sterile decorations have been included in the algorithm $\mathcal{R}$.

4.4 Other Advantages of This Approach

It has already been mentioned that the major reasons for rearranging the order of operation in my implementation are to increase the speed and decrease the memory usage of the program. There are other ways in which we benefit from the rearrangement.

There is one major beneficial side–effect of having to split the mass gap calculation into a series of different runs with a definite spatial relationship between P_{in} and P_{out} in each run. When calculating the energy of a particle with momentum p as in ref. [34] or ref. [35], contributions made to the mass gap must be split up according to the spatial relationship between the plaquettes being correlated. The implementation we have developed automatically calculates these different contributions separately, so all the information required for calculating E(p) is provided by the mass gap calculation with no extra effort.

Of course, the calculation of the mass of the 2^{++} (and 1^{+-} for SU(3)) state in four dimensions also requires the separation of these different contributions.

Another fortunate side–effect is that we are able to use symmetry to save a certain amount of calculation time. Consider two groups of decorations contributing

the $O(\beta^k)$ mass gap. All the decorations in both groups have exactly the same position designated for the P_{in} plaquette. In one group the P_{out} plaquette is always translated one lattice spacing in the positive–x direction from P_{in} and in the other group P_{out} is translated one lattice spacing in the positive–y direction from P_{in}. Due to the symmetry of the lattice, the total contribution made to the $O(\beta^m)$ mass gap by each of these two groups of decorations will be identical. The contributions due to the groups in which P_{out} has been translated by $-\hat{Y}$ and by $-\hat{X}$ will also be identical to the other two. Hence, we only have perform to one run of the program in order to evaluate the contribution from four different groups of decorations. Decker's implementation is unable to take advantage of this opportunity to avoid having to generate absolutely all the decorations contributing to the $O(\beta^k)$ mass gap.

This symmetry is also of use in debugging the program, if one tries performing two runs which ought to produce the same result but finds them disagreeing then obviously there is still a bug in the works. Futhermore, one may save to disk the full set of decorations generated in each run. Pinpointing the location of the bug may be faclilitated by knowledge of which decorations are generated in one run, but not in the other.

4.5 Futher Alterations

All the alterations described up to now have been intimately tied up with the tree structure and changing the order in which processes are carried out. I have also made a couple of other quite important changes to the implementation which are more related to the criteria for determining which decorations contribute to the mass gap.

The first alteration we shall examine has had the effect of making the calculation fully automatic. The fact is that a certain amount of the calculation in ref. [25] was carried out by hand. A clear distinction is made in ref. [25] between pure tube contributions and geometrical contributions. Simply , a decoration makes a pure–tube contribution if its corresponding volume is a basic tube part, all other

decorations (i.e. all those whose shape deviates from a straight length of tube) make geometrical contributions.

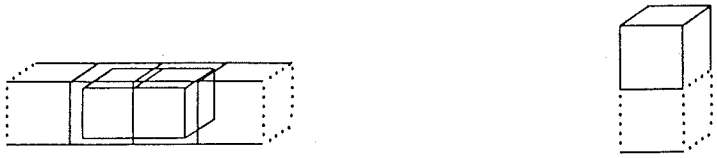


Fig 4.13 A Pure Tube Contribution (left) and a Geometrical Contribution (right)

It is proven in ref. [36] that all clusters involving only pure–tube contributions can be summed exactly in the case of Z_2. Hence Decker's programs were set up to only sum the geometric contributions.

Of course there are also clusters that involve both geometrical and pure–tube contributions; it is these clusters that had to be treated by hand.

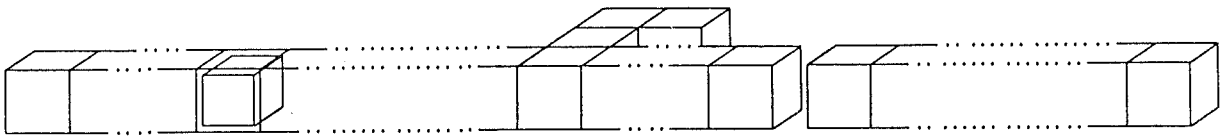


Fig 4.14 A Cluster Containing Both Geometrical and Pure–Tube Contributions.

Table 2 of ref. [25] contains a series of factors $F_k(V_a,V_{ab})$. These factors contain all the terms that arise from addding pure–tube contributions to a cluster containing k geometrical contributions. Obviously, the higher the order one wishes to go to, the more difficult the calculation of the F_k becomes. The evaluation of the $O(\beta^{16})$ factors displayed in ref. [25] involves a good many potential pitfalls, and undoubtedly at higher orders the job could start to become very tricky indeed, and as such would be best avoided if at all possible.

But the avoidance of the potential pitfalls in a hand calculation is not the only reason for including the pure–tube contributions in the automatic calculation.

Note that in step (2) of $\mathscr{D}$ all possible k cube configurations are generated, those resulting only in pure–tube contributions appearing as readily as those producing geometrical contributions. Decker has simply chosen that the program ignore the pure–tube contributions which it generates. Hence, including the pure–tube contributions in the automatic calculation will alter the speed of the program very little.

Therefore, in order to avoid potentially tricky hand calculations and to make fuller use of all the decorations generated, I have chosen to have my program automatically sum all sets of irreducible decorations regardless of whether or not they be pure–tube contributions.

The implementation $\mathscr{D}$ is also severely hamstrung by the fact that sets of more than one decoration are treated quite separately (in step (13)) from single decorations. Let us now examine why this process is both unnecessary and inefficient.

At step (6) of $\mathscr{D}$, reducible decorations will be generated just as readily as reducible decorations. In $\mathscr{D}$, any decoration which is not irreducible is discarded, only completely irreducible decorations being retained. A good many of the reducible decorations discarded would actually consist of two or more reducible decorations connected together. But, it is just exactly sets of two or more decorations which are dealt with at step (13) of $\mathscr{D}$. If it were possible to calculate the contributions of these sets of decorations at their formation, step(13) would be unneccesary.

Therefore, we wish to develop a strategy for treating sets of more than one irreducible decoration on the same footing a single irreducible decorations. Of course, there is a many to one relationship between reducible decorations sets of irreducible decorations, i.e. several different reducible decorations can be decomposed into the same set of irreducible decorations.

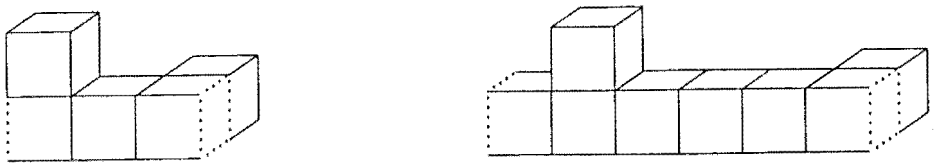


Fig 4.15 Two Decorations Containing Exactly the Same Set of Irreducible Decorations

Therefore, the first requirement of our strategy is a definition of just which reducible decoration is to be taken as the representative of some given set of irreducible decorations. As can be seen in Fig 4.15, the essential difference between the competing reducible decorations is the lengths of pure tube which separate their constituent irreducible decorations. The obvious choice for the representative decoration, therefore, is the one in which the length of pure tube separating any pair of neighbouring irreducible decorations i and j is exactly V_{ij}, their exclusion volume.

The criteria for determining which decorations are to be discarded and which are to be retained must be altered. Obviously Decker's criterion for discarding a decoration is simply that it contain a reducing bottle–neck. We need a somewhat more complex test so that we retain all decorations which consist of either a single irreducible decoration or a set of one or more irreducible decorations separated by lengths of pure tube of length exactly V_{ij}. The steps in our test are as follows:

(i) Identify all the reducing bottle–necks present in the decoration.

(ii) Determine whether there are any pairs of reducing bottle–necks with nothing between them except a length of pure tube. If not then the decoration is automatically retained, but if there any such pairs then continue on to step (iii).

(iii) Choose one pair of reducing bottle–necks which are separated by a length of pure tube, remove one cube from between the pair.

(iv) Close the gap by translating the upper part of the diagram by one lattice spacing in the negative t direction.

(v) Determine whether the lower bottle neck of the pair is still a reducing bottle neck in this new, contracted decoration. If it is then discard the decoration, if not then continue at (iii).

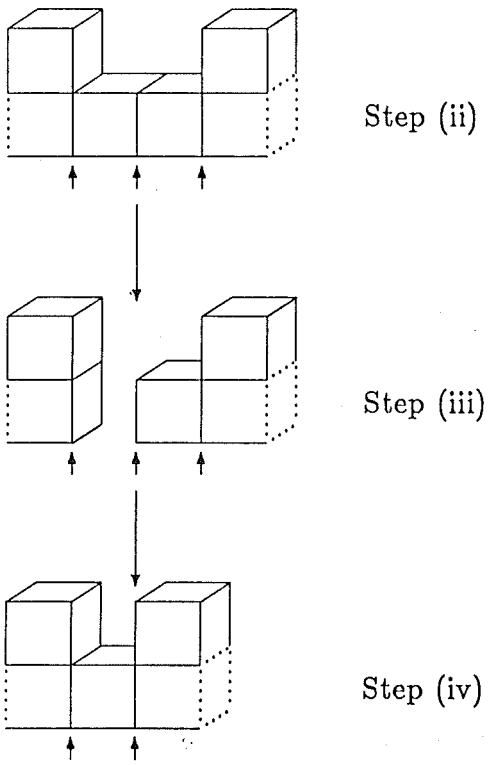


Fig 4.16 Determining That A Decoration Ought To Be Discarded

If all possible such pairs of reducing bottle necks have been exhausted without the decoration being discarded by the test at step (v) then the decoration is equivalent to a set of irreducible decorations separated by stretches of pure tube of lengths exactly V_{ij}.

The program must also be altered in order to actually calculate the contributions of these sets of two or more irreducible decorations. Recall that there are

two factors that must be calculated in order to determine the contribution of some given decoration, namely its activity and its configuration number. Determining the activity of a set of more than one decoration is carried out in exactly the same way as for a single irreducible decoration. One simply calculates the activity of the whole decoration, the fact that the decoration is divided into two or more separate clumps is immaterial.

To calculate the configuration number for a set of more than one irreducible decoration one has to consider the set being 'opened up' at its various reducing bottle necks, and the constituent irreducible decorations being placed independantly along a tube of length t. So the configuration number depends on two factors, the number of irreducible decorations contained in the decoration, and the sum $V+V_E$, where V is the length of the decoration and V_E the exclusion volume between its two ends. A table of values for $N^r(k,V+V_E)$ as functions of $V+V_E$ are given for k = 1,..,7 in table 4.1 .

Hence, step (13) of $\mathscr{D}$ is entirely superfluous, all sets of 2 or more irreducible decorations may be treated on the same footing as single irreducible decorations. But not only is step (13) unnecessary, it represents a serious inefficiency in $\mathscr{D}$. All decorations which I retain as representing ordered sets of irreducible decorations are discarded by Decker's program and then effectively reconstructed at step (13). My implementation avoids this double handling.

4.6 Flow Chart

The implementation $\mathscr{R}$ described in this chapter uses a lot of the same basic procedures as $\mathscr{D}$. It still generates MLSL's, scrutinises decorations for the prescence of reducing bottle necks, and calculates the activities of decorations, etc. However, there are some procedures which are quite central to $\mathscr{D}$ which are not necessary at all in $\mathscr{R}$. The most time–consuming procedure which has been removed in $\mathscr{R}$ is that of forming clusters by partitioning cube configurations. Also avoided is the process of determining

all the valid P_{in} and P_{out} positions on a cluster, as the P_{in} and P_{out} are now predetermined. Of course, some new procedures have been added, such as those for determining the sterility or otherwise of decorations. The process of checking the reducibility of a decoration is also more complex.

More significant, though, than the removing of some procedures and the introduction of others is the severe alteration of the order in which the operation are carried out. This is best illustrated by comparing the flow chart published in ref. [25] with that presented on the following page.

In the following chapter we will consider the extension of this algorithm to calculations in 3+1 dimensions.

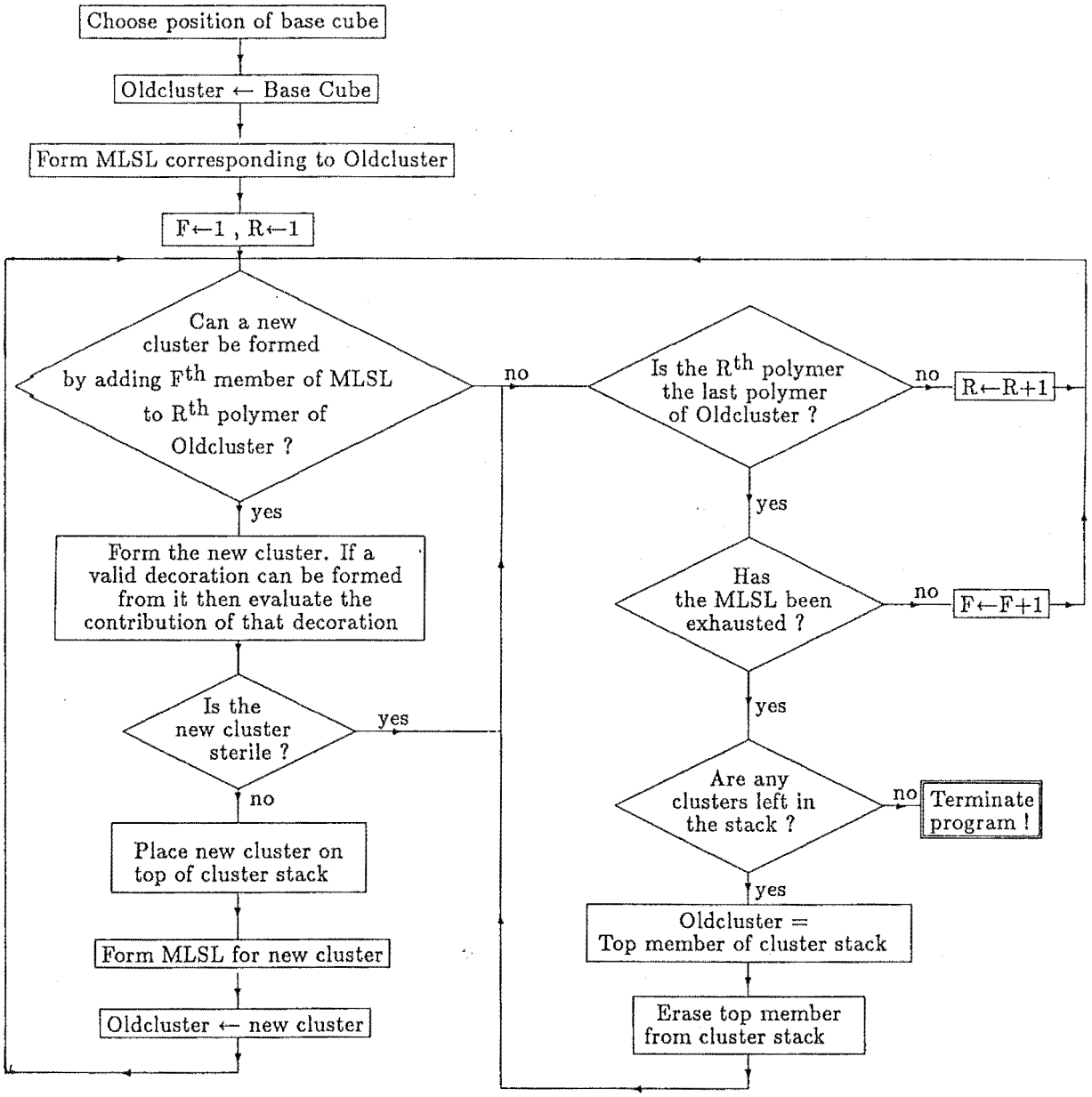


Fig 4.17 Flow Chart for the Algorithm $\mathscr{R}$

TABLE 4.1

REDUCED CONFIGURATION NUMBERS $N^r(K,V)$

$N^r(K,V)$ represents the t–linear term in the expression for the number of ways a set of K irreducible decorations , with total length (including exclusion volumes) V lattice units may be placed on a periodic tube with a period of length t lattice units.

$$N^r(1,V) = 1$$

$$N^r(2,V) = {}^1/_2\,(1 - V)$$

$$N^r(3,V) = {}^1/_6\,(2 - 3\,V + V^2)$$

$$N^r(4,V) = {}^1/_{24}\,(6 - 11\,V + 6\,V^2 - V^3)$$

$$N^r(5,V) = {}^1/_{120}\,(24 - 50\,V + 35\,V^2 - 10\,V^3 + V^4)$$

$$N^r(6,V) = {}^1/_{720}\,(120 - 274\,V + 225\,V^2 - 85\,V^3 + 15\,V^4 - V^5)$$

$$N^r(7,V) = {}^1/_{5040}\,(720 - 1764\,V + 1624\,V^2 - 735\,V^3 + 175\,V^4 - 21\,V^5 + V^6)$$

APPENDIX 4.1

SCHEMATICAL FLOW CHART OF ALGORITHM $\mathscr{D}$

(1) Choose desired maximal order $O(\beta)_{max}$ of computation; $k \leftarrow 0$.

→ (2) $k \leftarrow k+1$; generate the set $\mathscr{K}^{(k)}$ of all link–connected point configurations with k points.

→ (3) Take one $K^{(k)} \in \mathscr{K}^{(k)}$ and map to the dual volume V.

→ (4) Generate one collection $S(\hat{C}) = (|Y_1|^{n_1}, |Y_2|^{n_2},...)$ of polymer supports which has volume V.

→ (5) Determine one cluster $\hat{C}$ which can be constructed from $S(\hat{C})$.

→ (6) Determine one irreducible decoration D which can be constructed from $\hat{C}$.

(7) Compute all contributions to Δm up to $O(\beta)_{max}$ due to the set of clusters represented by (D).

(8) Keep the characteristic data of all those decorations which are needed for the representation of those sets of clusters which are represented by a collection of more than one irreducible decoration and which contribute to Δm up to $O(\beta)_{max}$.

(9) Continue at (6) until no more irreducible decorations D can be obtained from the present cluster $\hat{C}$.

(10) Continue at (5) until all clusters which can be derived from the curent $S(\hat{C})$ have been constructed.

(11) Continue at (4) until all $S(\hat{C})$ which have the current volume V have been generated.

(12) Continue at (3) until $\mathcal{K}^{(k)}$ is exhausted.

(13) Continue at (2) until all corrections to Δm have $O(\beta) > O(\beta)_{max}$.

(14) Compute all contributions to Δm up to $O(\beta)_{max}$ due to those sets of clusters which are represented by (D_1,D_2); (D_1,D_2,D_3);..., according to an analogous scheme.

CHAPTER 5

CALCULATIONS IN 3+1 DIMENSIONS

A computer algorithm for calculating the strong coupling expansion of the free energy of euclidean lattice gauge theories in 3+1 dimensions has been in existence for some time [6]. However, a similar algorithm for mass gap calculations did not exist. In ref. [25], emphasis is placed on the fact that the basic elements of algorithm $\mathscr{D}$ were developed in such a way as to be valid for calculations in 3+1 dimensions as well as 2+1. Despite this, though, the extension of the algorithm to 3+1 dimensional calculations is by no means utterly straightforward. In this chapter I shall discuss the process whereby the algorithm described in chapter 4 has been extended to mass gap calculations in 3+1 dimensions.

5.1 Diagram Expansion for the Mass Gap in 3+1 Dimensions

Let us begin by deriving a strong coupling diagram expansion for the mass gap in 3+1 dimensions. As mentioned in chapter 3, the lowest mass state on the lattice is expected to transform trivially under the the symmetry group of the constant–time sublattice. If the lattice is four dimensional then the lowest mass state has to transform trivially under O, the symmetry group of the three dimensional cube. A single plaquette, therefore, will not suffice since various operators of the cubic group will have the effect of transforming it into a differently oriented plaquette (see fig 5.1)

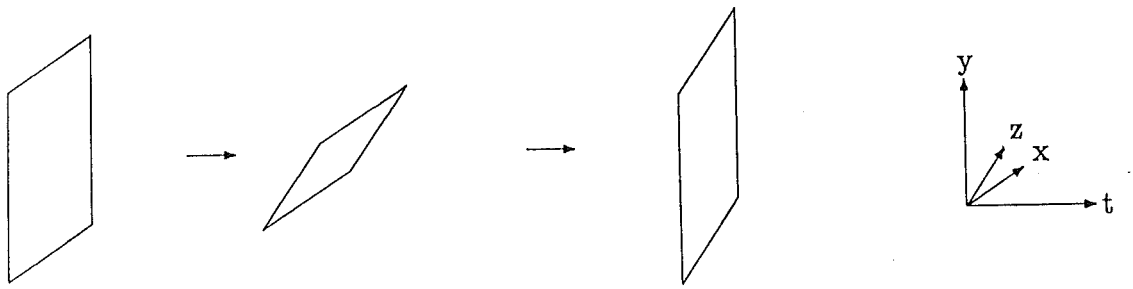


Fig 5.1 Possible Transformations of A Space–Like Plaquette Under O

However, a sum of three differently oriented plaquettes will transform trivially under O.

Hence, our approximation to the zero–nomentum first excited state is:

$$\theta(t) = \frac{1}{\sqrt{3N_t}} \sum_{\mathbf{x}} \left\{ P_{xy}(t,\mathbf{x}) + P_{yz}(t,\mathbf{x}) + P_{zx}(t,\mathbf{x}) \right\} \tag{5.1}$$

where N_t is the total number of sites in one constant–time hyperplane of the lattice, and x runs over all the N sites. Also, we define $P_{ij}(t,\mathbf{x}) = \chi_f(U_{P_{ij}}(\mathbf{x},t))$.

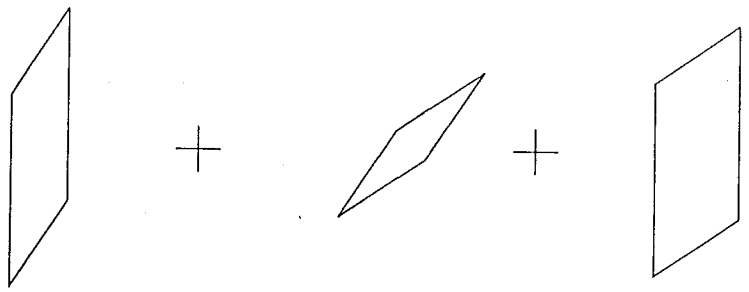


Fig 5.2 Sum of the Three Different Space–Like Plaquettes, An Invariant Under O

The correlation function we consider is

$$\Gamma = \langle \theta(0)\ \theta(t) \rangle - \langle \theta(0) \rangle \langle \theta(t) \rangle$$

$$= \frac{1}{3N_t} \Big[\Big\langle \sum_{\mathbf{x}} \left\{ P_{xy}(0,\mathbf{x}) + P_{yz}(0,\mathbf{x}) + P_{zx}(0,\mathbf{x}) \right\} \sum_{\mathbf{y}} \left\{ P_{xy}(t,\mathbf{y}) + P_{yz}(t,\mathbf{y}) + P_{zx}(t,\mathbf{y}) \right\} \Big\rangle$$

$$- \Big\langle \sum_{\mathbf{x}} \left\{ P_{xy}(0,\mathbf{x}) + P_{yz}(0,\mathbf{x}) + P_{zx}(0,\mathbf{x}) \right\} \Big\rangle \Big\langle \sum_{\mathbf{y}} \left\{ P_{xy}(t,\mathbf{y}) + P_{yz}(t,\mathbf{y}) + P_{zx}(t,\mathbf{y}) \right\} \Big\rangle \Big]$$

$$= \Big\langle P_{xy}(0,0) \sum_{\mathbf{y}} \left\{ P_{xy}(t,\mathbf{y}) + P_{yz}(t,\mathbf{y}) + P_{zx}(t,\mathbf{y}) \right\} \Big\rangle$$

$$-\left\langle P_{xy}(0,0)\right\rangle\left\langle \sum_y \left\{P_{xy}(t,y)+P_{yz}(t,y)+P_{zx}(t,y)\right\}\right\rangle$$

$$= \sum_y \left[\left\langle P_{xy}(0,0)\, P_{xy}(t,y)\right\rangle - \left\langle P_{xy}(0,0)\right\rangle\left\langle P_{xy}(t,y)\right\rangle\right]$$

$$+ 2\sum_y \left[\left\langle P_{xy}(0,0)\, P_{xz}(t,y)\right\rangle - \left\langle P_{xy}(0,0)\right\rangle\left\langle P_{xz}(t,y)\right\rangle\right] \qquad (5.2)$$

So, the various symmetries of the four dimensional lattice enable us to reduce the initial set of nine different correlations down to just two, and to sum over just one of the constant–time hyperplanes, not both. The first correlation on the r.h.s. of eqn. (5.2) is very similar to the correlation involved in the 2+1 dimensional calculation. The contributing clusters will be basically long tubes in xyt–space, with various local decorations appended to them.

The second term on the r.h.s. of eqn. (5.2) is less familiar. It involves correlations between differently oriented plaquettes. The contributing clusters will by necessity contain decorations with an xy plaquette as P_{in} and an xz plaquette as P_{out}, thereby effecting the transformation from tube in xyt–space to tube in xzt–space.

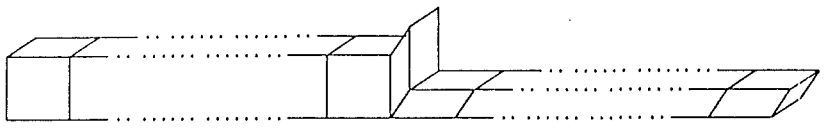


Fig 5.3 A Typical Cluster Contributing to Second Correlation in (5.2)

Let us now examine the major new elements required in a program for automatic calculation of the 3+1 dimensional mass gap which are not present in the 2+1 dimensionsal program.

5.2 The Twisted Boundary Conditions

The first major alteration we shall consider is concerned with calculating the configuration number of a set of two or more decorations. The determination of the exclusion volume between the ends of a decoration is not as straightforward in 3+1 dimensions as it is in 2+1. Recall that the picture we have in the 2+1 dimensional case when determining the configuration number of a set of more than one irreducible decoration is of a long tube periodic in time.

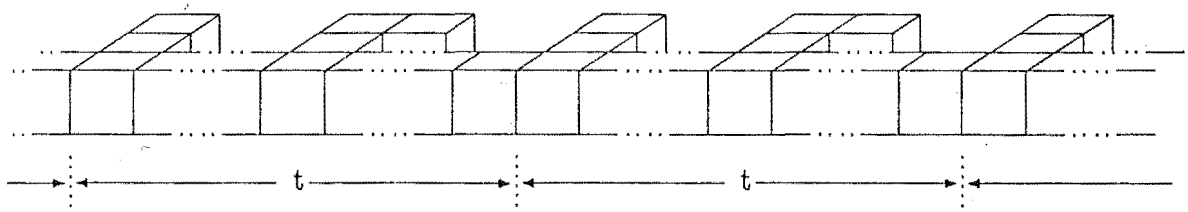


Fig 5.4 The Periodic Tube in 2+1 Dimensions

The upper limit on how far the decorations may be stretched from each other is set by how closely the last decoration in one period may approach the first decoration in the next period.

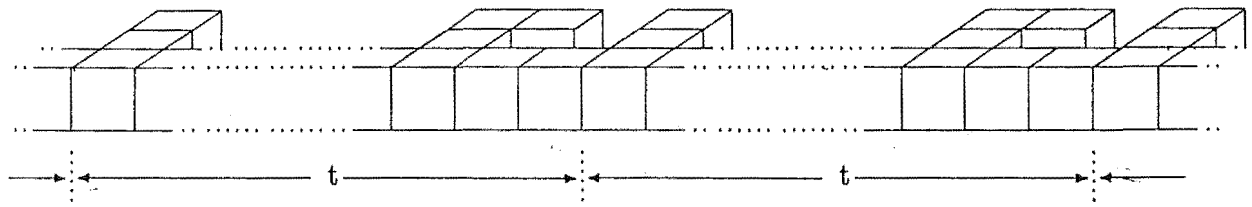
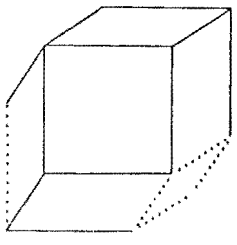


Fig 5.5 Decorations Stretched As Far Apart As Possible

This same picture works perfectly well in 3+1 dimensions for those clusters contributing to the correlation between similarly–oriented plaquettes. However, for the clusters contributing to the second correlation, things are not so simple. Consider a cluster containing the decoration below



Any given period of tube would start with a length of tube in xyt space, but end with a length of tube in xzt space.

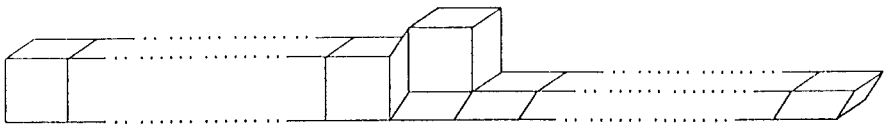


Fig 5.6 A Cluster Contributing To the Correlation Between Different Plaquettes

It is not possible to simply connect another such period of tube onto this first one; their ends are in entirely different three dimensional spaces. Obviously, we need to define our periodic tube more cleverly in such cases. The solution to the problem is to consider the decoration to have 'twisted' the tube, so that beyond P_{out}, the y and z axes have been interchanged. The next period of the tube would therefore be as below:

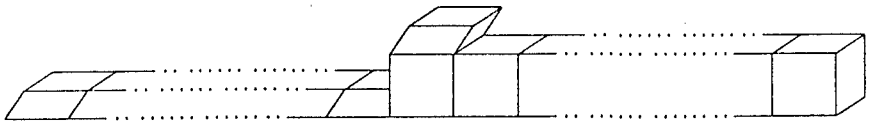


Fig 5.7 Cluster of Fig. 5.6 With the Y and Z Directions Interchanged

The second decoration exchanges the x and y axes back again, so that the third period of the tube will be identical to the first. So the periodic tube would appear as below

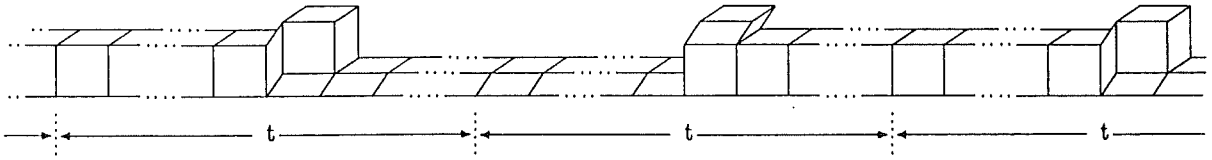


Fig 5.8 A Periodic Tube with Twisted Boundary Conditions

Therefore, this is the picture we must have in mind when determining the exclusion volume between the ends of a decoration in which the P_{in} and P_{out}

plaquettes do not have the same orientation. We must determine how closely the last member of a set of irreducible decorations could approach the first member from below if the first member has been twisted in such a way that its y and z directions have been interchanged.

5.3 The Degeneracy Among Cube Configurations

Because of the form of the Wilson action, the fundamental building block from which we build the diagrams in our expansions is the plaquette, so diagrams may be thought of as closed systems of plaquettes. Fortunately in 2+1 dimensions there is a 1 to 1 correspondence between closed systems of plaquettes and connected sets of cubes. This correspondence enabled us to use the cube as the fundamental building block for closed three dimensional polymers. Unfortunately this 1 to 1 correspondence does not exist in 3+1 dimensions, different configurations of three dimensional cubes may in fact result in the same closed system of plaquettes.

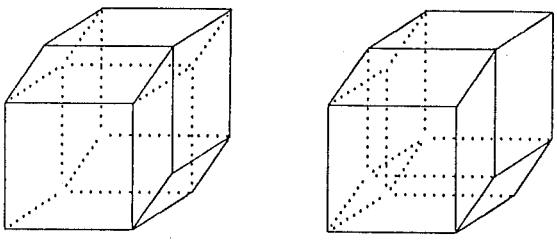


Fig 5.9 One Set of Plaquettes Produced By Two Different Cube Configurations

This means that if we just blithly sum all the contributions from all decorations formed from every possible cube configuration, some decorations are going to be counted several times. The ideal way to solve this problem would be to find a universal rule that chooses one and only one of the several cube configurations which result in a given closed polymer. Such a rule could be applied to every cube configuration generated by our algorithm. If a configuration passed the test it would be kept as the set of cubes corresponding to a certain polymer, otherwise it would be

discarded. However, such a universal criterion has not yet been found, but a certain amount of progress has been made towards it.

Let us refer to cube configurations which produce the same closed system of plaquettes as being degenerate. To understand the progress that has been achieved on the problem of degenerate cube sets, we must examine some of the duality properties of the four dimensional lattice. Every elementary cube on the four dimensional lattice corresponds to a bond on the dual lattice [37]. More specifically, a cube which is at constant X_i is dual to a bond parallel to the X_i axis, and it is always possible to position the dual lattice in such a way that the centre of the cube and the centre point of the dual bond coincide.

This duality enables us to represent any cube configuration by a set of bonds.

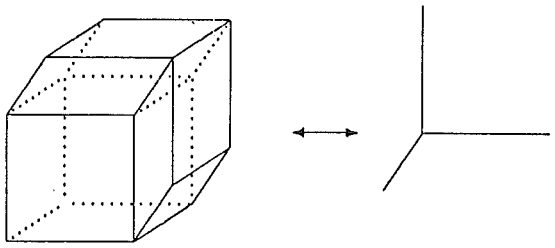


Fig 5.10 Cubes in 4–D May Be Represented by Bonds on the Dual Lattice

Let us divide the possible bond diagrams into three categories:

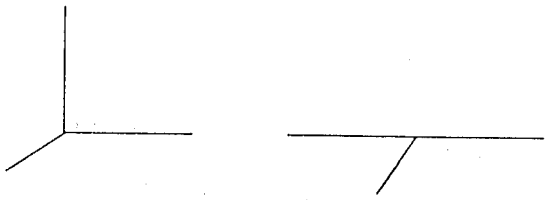
(i) Those in which no site is shared by more than three bonds.

(ii) Those in which at least one site is shared by four bonds, but no site is shared by more than four bonds.

(iii) Those in which at least one site is shared by more than four bonds.

We shall now examine each of these categories in turn.

(i)There are two possible ways to have three bonds sharing a site, either all the bonds are differently oriented, or there are two with the same orientation and one different:



Let us now show that in neither case is it possible to find another set of cubes which will result in the same closed polymer. To do so we must first study the duality transformation more closely. So far we have established that an elementary cube transforms to a bond, but how are the plaquettes which constitute the cube arranged in the dual lattice? The fact is that the six plaquettes forming a cube on the original lattice transform to six plaquettes radiating from its dual bond.

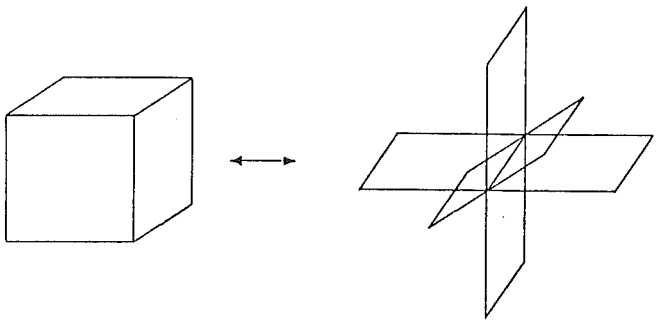


Fig 5.11 An Elementary Cube and Its Analogue on the Dual Lattice

Note that a plaquette in (x_i,x_j) space transforms to a plaquette in (x_k,x_l) space, with no overlap between the sets {i,j} and {k,l}.

So we are now able to fill in the plaquettes on our bond diagrams. First consider the case in which all the bonds are differently oriented. The arrangement of plaquettes on the dual lattice is illustrated in fig. 5.12. The task of finding another set of three cubes that will produce the same closed system of plaquettes is reduced to finding another set of three bonds in this diagram such that every plaquette in the diagram contains one and only one of the bonds. The simple fact is that such a set does not exist.

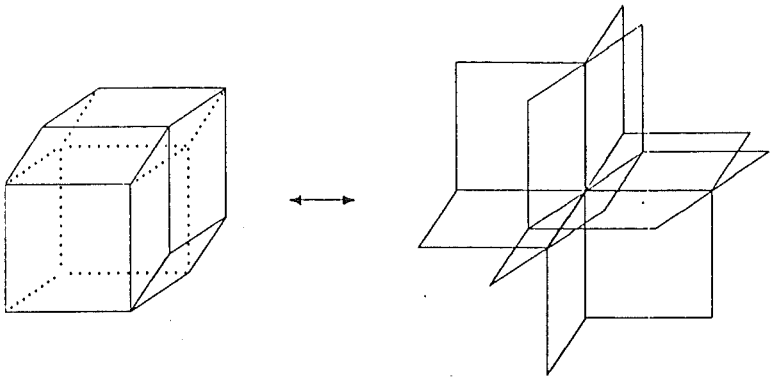
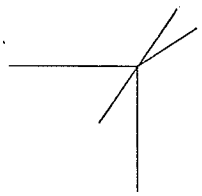


Fig 5.12 A Three–Cube Configuration and its Dual

There is a set of five bonds which fulfills these criteria, namely



but no set of three bonds.

The case is the same for the situation in which two of the bonds have the same orientation. Putting in the plaquettes, we obtain the arrangement illustrated in fig. 5.13.

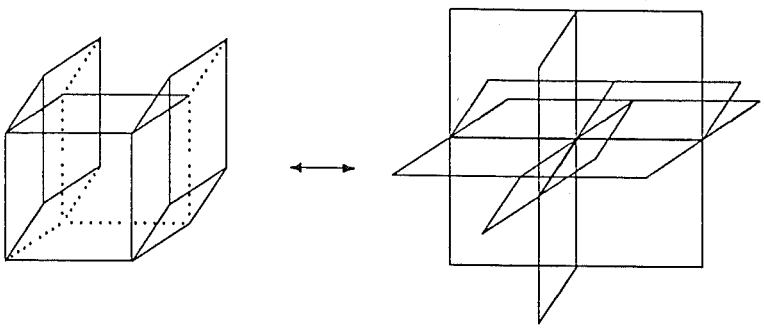


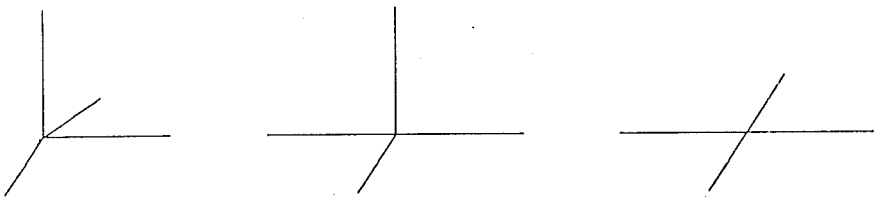
Fig 5.13 Another Configuration of Three Cubes and its Dual

Again it is not possible to find another set of three bonds which fulfill the necessary criteria.

We may conclude, therefore, that if no site in the bond diagram is shared by

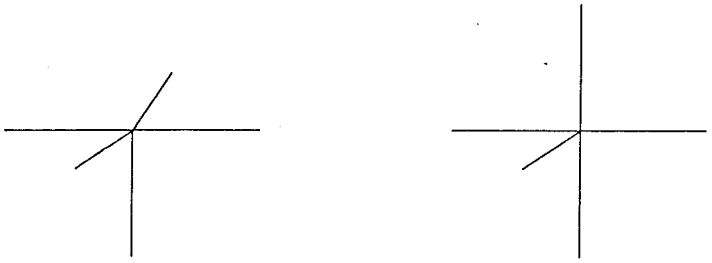
more than three bonds then no other cube set containing the same number of cubes will be degenerate with the set in question.

(ii)It is possible to arrange four bonds sharing a site in three different ways, as shown below:



If the plaquettes are filled in on any of these sets of bonds, it becomes evident in each case that there is a second set of four bonds which have the property that each plaquette in the diagram contains one and only one of the bonds. Hence there is a second set of four cubes which will produce the same closed polymer. Hence we may conclude that if a set of cubes has a corresponding bond diagram in which at least one site is shared by four bonds then there will be at least one other set of cubes that is not only degenerate with it, but also contains exactly the same number of cubes.

(iii)As illustrated below, there are two distinct ways of arranging five bonds to share the same site.

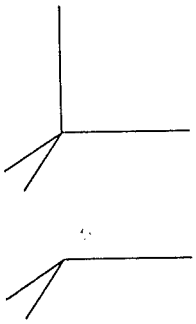


In either case, if the plaquettes are filled in, it is not possible to find another set of five bonds which span the plaquettes, but it always possible to find a set of three bonds which do. Hence if five bonds share a site then the corresponding set of cubes is degenerate with a set of cubes containing less elements. Obviously the same

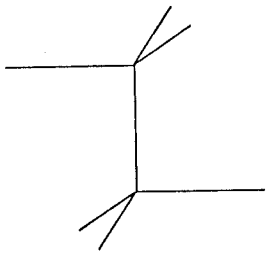
is true if six, seven, or eight bonds share a site as it is always possible to choose any set of five bonds from the set, replace them by their degenerate set of three bonds and thereby reduce the number of cubes in the diagram.

It is now evident just what a useful tool the dual lattice is. If the bond diagram dual to any given set of cubes is in the third category, we may immediately discard it. If it is in the first category we do not need to worry about degeneracy at all.

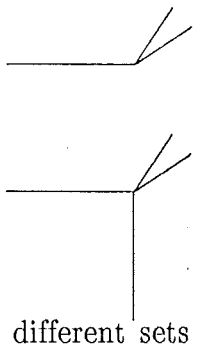
If it is in the second category, however, we need to determine just how many other diagrams are degenerate with it. At first glance it would seem that this is a very simple matter. Every set of four bonds meeting at a site are degenerate with one other set. So, if S is the number of sites shared by four bonds then the degeneracy of the diagram would appear to be 2^S. Unfortunately, the situation is not so simple. Consider the set of bonds below



We may invert the four bonds sharing the one site to produce the following bond configuration:



which still represents the same set of plaquettes. Then we could carry out another inversion to form:



Hence we have a case of three different sets of cubes all producing the same closed polymer, and the corresponding bond diagrams sometimes having one site shared by four bonds and sometimes two sites. Obviously the degeneracy of the sets of cubes is not simply 2^S, with S being the number of four–sites in the coresponding bond diagram. However, it is always possible to determine the degeneracy of a diagram by systematic construction as above.

So, a new addition to the 3+1 dimensional algorithm is a procedure for employing these bond diagrams to check for possible degeneracy of every cube configuration. Upon determining the degeneracy of a given decoration to be D, a factor of $^1/_D$ must be included in the contribution which the decoration makes to the mass gap.

Obviously, as one goes to higher orders, the bond diagrams are going to become increasingly more complex, and evaluating the degeneracy of a given set of cubes will become an increasingly difficult process. So, whilst the degeneracy of cube configurations does not prevent one from being able to produce an algorithm for strong coupling calculations in 3+1 dimensions, it is a major obstacle to improving the efficiency of such a program. This is a stumbling block that has yet to be overcome.

5.4 Other Alterations Required

Apart from the two major changes discussed so far, a great many minor changes had to be made in order to produce an algorithm for calculations in 3+1

dimensions.

For example, in 2+1 dimensions, an elementary cube may be unambiguously represented by the position of its bottom, front, l.h. corner, i.e. the corner nearest the origin. In 3+1 dimensions, any site on the lattice will be the position of the nearest–the–origin corner of four different elementary cubes, one in each of the four 3–D hyperplanes of the 4–D lattice. Hence, to completely specify the identity of a cube, it is necessary to not only specify the position of its nearest–the–origin corner, but also which 3–D space it lies in. Cubes are therefore represented by two numbers, not one.

In 2+1 dimensions, a cube has just 6 neighbours, in 3+1 dimensions, it has 18. Hence the procedure for determining whether or not a given pair of cubes are neighbours had to be modified.

Almost every procedure in the program had to be altered to a greater or lesser extent to allow for the increased complexity of the 3+1 dimensional lattice. However, no fundamental changes were required; the claim in ref. [25] that the algorithm had been formulated in such a way as to be extendable to 3+1 dimensions is correct.

CHAPTER 6

RESULTS

The primary objective of my work has been the development of a fast, memory efficient algorithm for calculating the strong coupling expansions of mass gaps. To quite an extent, this aim has been achieved; the memory usage of the algorithm is utterly minimal. The speed of the program is rather difficult to gauge, all the work having been carried out on an IBM A.T. rather than a main–frame computer as in Decker's case [38].

A very important consideration which we must address is the accuracy of the algorithm. As mentioned in ref. [25], an algorithm for mass gap calculations effectively contains within it an algorithm for the evaluation of the free energy. This portion of the mass gap program may be tested by determining whether it can reproduce previously published expansions of the free energy. I performed this test in both 2+1 and 3+1 dimensions, reproducing the 40^{th} order expression for the Z_2 free energy in 2+1 dimensions [39], and the first 6 non–zero terms in the expression for the 3+1 dimensional Z_2 free energy [6]. With only a few alterations the algorithm for free energy calculations may used for the evaluation of the string tension. Again, my algorithm was able to reproduce published results.

6.1 Restoration of Lorentz Invariance in 2+1 Dimensions

Now let us consider the expressions produced by the full mass gap algorithms. In both 2+1 and 3+1 dimensions, the results of my algorithm are at variance with published results. Let us consider the 2+1 dimensional case first. Recall from chapter 2 that we are producing series expansions in the character expansion co–efficient $a_f(\beta)$. When the gauge group is Z_2 then $a_f(\beta) = \tanh \beta$. Throughout this

chapter we shall follow the notation of ref. [33] and represent tanh β by the single character u. The strong coupling expansion for the Z_2 mass gap in 2+1 dimensions to 14^{th} order in u, produced by the algorithm $\mathcal{R}$, is :

$$M = -4 \text{ Ln } u + u^2 - \frac{15}{2}u^4 + \frac{7}{3}u^6 - \frac{87}{4}u^8 + \frac{251}{5}u^{10} - \frac{331}{2}u^{12} + \frac{2073}{7}u^{14}$$

This does not agree with Decker's expression [38] in the 14^{th} order term. However, the recently published results of Arisue and Fujiwara [27] do aree with the expression above. It seems highly likely, therefore, that it is the expression in ref. [38] which is in error.

The extension of the series for the mass gap enables one to calculate improved estimates of critical exponents. Pade analysis of the strong coupling expansion of the Z_2 mass gap has been carried out in refs. [38] and [27]. The information obtained in calculating the mass gap series also enables us to calculate the energy of a glueball with non–zero momentum and to study the restoration of Lorentz invariance on the lattice [34,35].

The energy of a glueball with momentum **p** is, of course, calculated from the correlation function between latice states with momentum **p**. Recall the expression given in eqn. (3.2) for a state of definite momentum

$$\theta(\mathbf{p},t) = \sum_{\mathbf{x}} e^{i\,\mathbf{p}.\mathbf{x}}\ \chi_f(U_p(\mathbf{x},t))$$

Hence, the correlation function to be calculated is

$$\Gamma_p = \sum_{\mathbf{x}} e^{i\,\mathbf{p}.\mathbf{x}} \left\{ \langle \chi_f(U_p(\mathbf{x},t))\ \chi_f(U_p(0,0)) \rangle - \langle \chi_f(U_p(\mathbf{x},t)) \times \chi_f(U_p(0,0)) \rangle \right\}$$

Let us examine the ways in which the expansion of Γ_p differs from the expression

obtained for Γ. The set of diagrams which contibute to Γ_p will be identical to the set which contribute to Γ, i.e. connected clusters which contain space–like plaquettes at (0,0) and (x,t). We shall refer to these two space–like plaquettes as the *active* plaquettes. Of course, we sum over x, so the relative positions of the active plaquettes will differ from cluster to cluster. The difference between Γ and Γ_p is

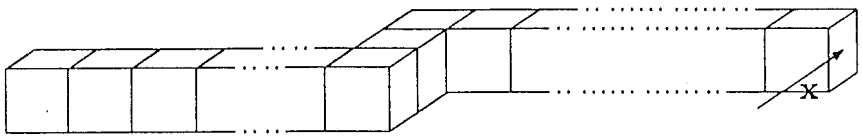


Fig 6.1 A Cluster in Which the Active Plaquettes are not Directly In Line

that in the latter the contributions from clusters in which the two active plaquettes are not directly in line will be multiplied by a factor $e^{i\ p.x}$ where x is the spatial separation of the active plaquettes.

Recall from chapter 4 that the nature of the algorithm $\mathscr{R}$ is such that the contributions to the mass gap are sorted according the spatial relationship between the active plaquettes. Hence, all the information required for the calculation of E(p) is provided automatically by $\mathscr{R}$. The expansion, to 14^{th} order, of E(p) for Z_2 in 2+1 dimensions is:

$$\begin{aligned} E = {} & -4\ \mathrm{Ln}\,u - \mathrm{Ln}(1-u^2) - 8\,u^4 + 2\,u^6 - 22\,u^8 + 50\,u^{10} - 165.67\,u^{12} \\ & + 296\,u^{14} + P^2(\,u^4 + 6\,u^8 - 10\,u^{10} + 56\,u^{12} - 92\,u^{14}\,) \\ & + (P^2)^2\,(-\tfrac{1}{2}u^8 + u^{10} - 8\,u^{12} + 19\,u^{14}\,) - (P^2)^3\,u^{14} \\ & + P_1^2\,P_2^2\,(-u^{10} - 14\,u^{14}\,) + (\,P_1^2\,P_2^4 + P_1^4\,P_2^2\,)\,u^{14} \end{aligned} \qquad (6.1)$$

where $P_i = \sqrt{2\ (1 - \cos\ p_i)}$.

Let us now use this expression for the energy to study the restoration of

Lorentz invariance. Note that eqn (6.1) is of the form

$$E = m + F_1 P^2 + F_2 (P^2)^2 + F_3 (P^2)^3 + F_4 P_1^2 P_2^2 + F_5 (P_1^2 P_2^4 + P_1^4 P_2^2) \qquad (6.2)$$

But we know that the relativistic expression for the energy of a particle is given by

$$E = \sqrt{p^2 + m^2} = m + \frac{p^2}{2m} - \frac{(p^2)^2}{8m^3} + \frac{(p^2)^3}{16m^5} - ..... \qquad (6.3)$$

If Lorentz invariance is to be restored as the lattice approaches the continuum, then the coefficients in eqn (6.2) must behave in such a way that the expression approaches the form of eqn (6.3).

In other words, at the critical point of the theory, where it is believed the continuum is attained, we desire that the following relations hold.

$$\begin{aligned} C_1 &= 2m F_1 = 1 \\ C_2 &= -8 m^3 F_2 = 1 \\ C_3 &= 16 m^5 F_4 = 1 \\ C_4 &= m^3 F_3 = 0 \\ C_5 &= m^5 F_5 = 0 \end{aligned} \qquad (6.4)$$

From eqn (6.1) we obtain the following expressions for the quantities C_1 and C_2:

$$C_1 = -8 u^4 \operatorname{Ln} u (1 + 6 u^4 - 10 u^6 + 56 u^8 - 92 u^{10}) + 2 u^6 (1 - \tfrac{15}{2} u^2 + \tfrac{25}{3} u^4 - \tfrac{307}{4} u^6 + \tfrac{976}{5} u^8) \qquad (6.5)$$

$$C_2 = 256\,(\mathrm{Ln}\,u)^3\,u^8\,(-1 + 2u^2 - 16u^4 + 38u^6)$$
$$- 192\,(\mathrm{Ln}\,u)^2\,u^{10}\,(-1 + \tfrac{19}{2}u^2 - \tfrac{100}{3}u^4)$$
$$+ 48\,\mathrm{Ln}\,u\;u^{12}\,(-1 + 17u^2) + 4u^{14} \qquad (6.6)$$

It is not worth considering the quantities C_3, C_4, C_5, as none of them has an

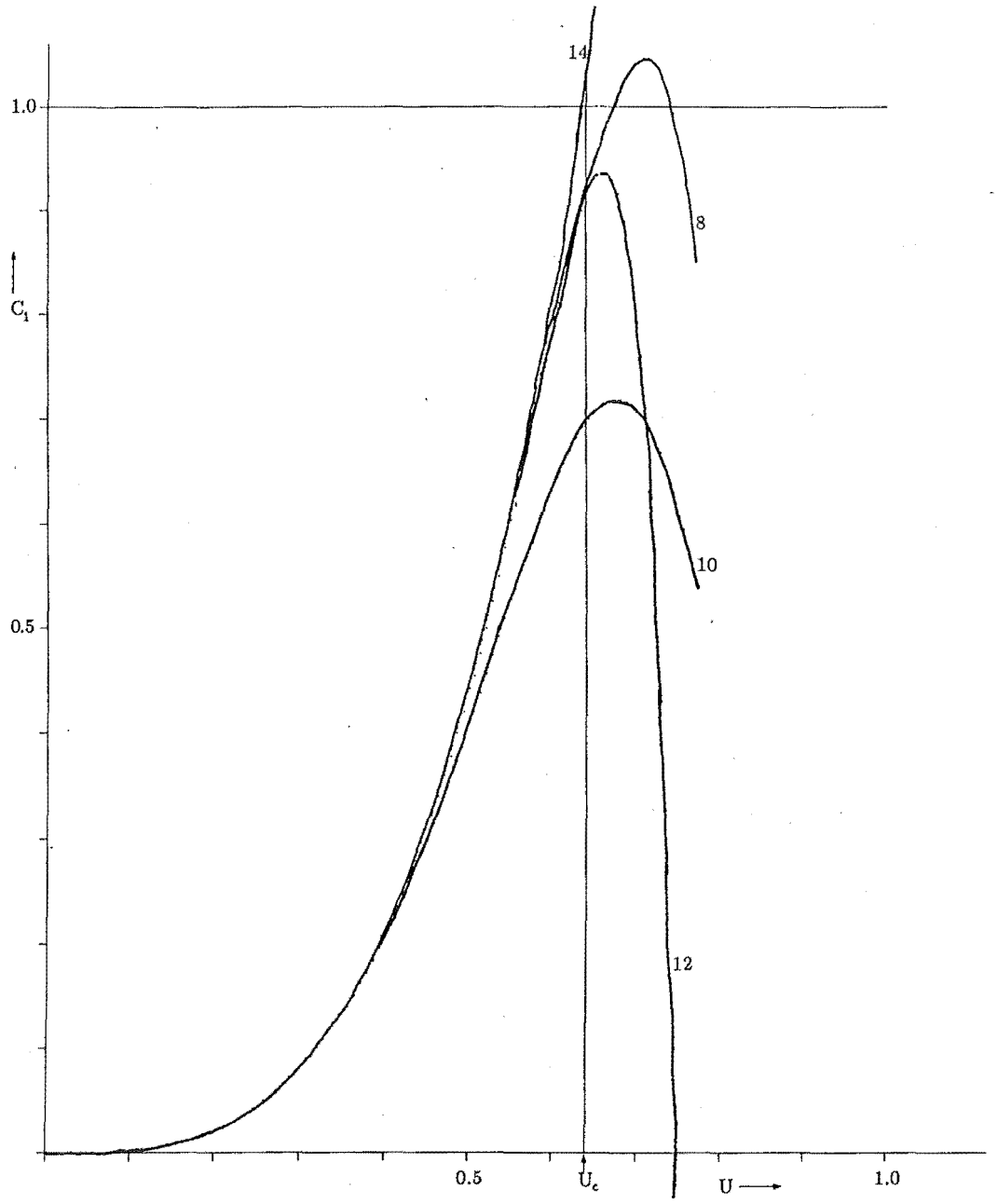


Fig 6.2 8^{th}, 10^{th}, 12^{th} and 14^{th} Order Expansions of C_1

expansion consisting of more than 2 terms to the order we are considering. The

quantities C_1 and C_2 are graphed in Figs 6.1 and 6.2 respectively. In the case of C_1 in particular, the evidence seems to be very much in favour of a full Lorentz symmetry at the continuum. The critical value, $u_c = 0.642$, at which the theory undergoes a second–order phase transition, is marked by a vertical line on the graphs. In Fig 6.1, there is a steady trend from the 10^{th} order series through to the 14^{th} order series

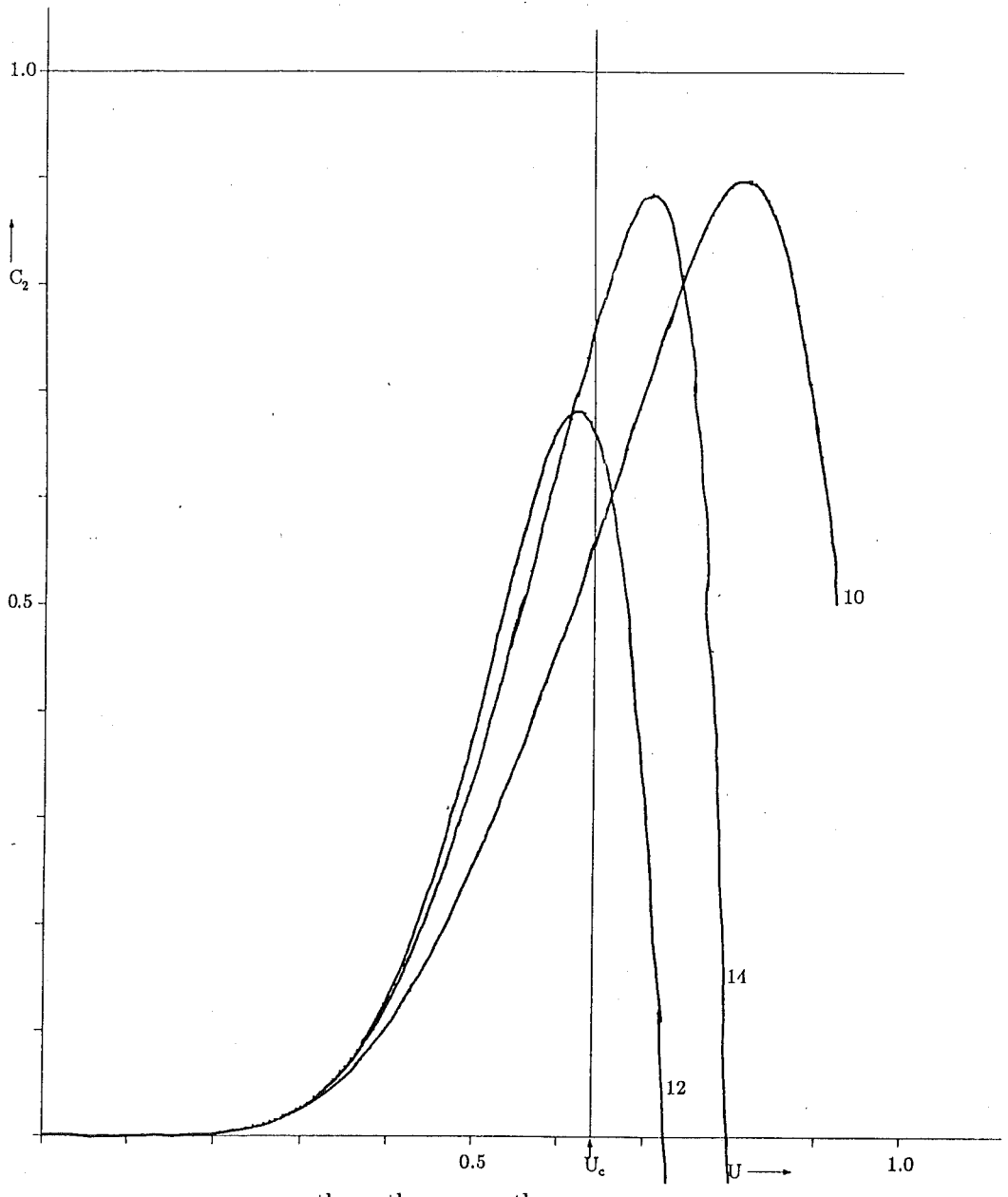


Fig 6.3 10^{th}, 12^{th}, and 14^{th} order expansions of C_2

towards the desired value of 1 at $u = u_c$. In fact, the 14^{th} order series goes very

close to this intercept. In Fig 6.2, there is definitely a steady trend of increasing intercept with increasing order, however, we do not appear as yet to have calculated sufficient terms in the series.

6.2 Discrepancies in the 3+1 Dimensional Mass Gap

Several days of CPU time on the IBM A.T. were required to produce an eighth order expansion of the Z_2 mass gap in 3+1 dimensions. The resulting series is:

$$M = -4 \text{ Ln } u + u^2 - \frac{67}{2}u^4 - \frac{287}{3}u^6 - \frac{2343}{4}u^8$$

This disagrees with Munster's result [24] in the eighth order term. Munster's coefficient of u^8 is $-\frac{2279}{4}$. Obviously the only definitive way to reconcile this difference would be to compare the full set of decorations found by Munster with the set generated by my program and determine which set is deficient. Failing that, though, one can find where the differences lie by decomposing the total coefficient into a sum of terms, each characterised by a different spatial relationship between the active plaquettes at the ends of the contributing clusters. As has already been stated, my program does this automatically. Fortunately, ref [24] contains an effective Hamiltonian, from which this decomposition of the coefficient may be constructed.

The two decompositions are set out in table 6.1. .

Spatial co-ords of P_1	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)
Spatial co-ords of P_2	(0,0,0)	(0,0,1)	(1,0,0)	(0,0,2)	(2,0,0)	(1,0,1)	(1,1,0)	(0,.5,.5)	(1,.5,.5)	(0,1.5,.5)
Orientation of P_1	xy	xy	xy	xy	xy	xy	xy	xy	xy	xy
Orientation of P_2	xy	xy	xy	xy	xy	xy	xy	xz	xz	xz
Result in ref. [24]	$^{507}/_4$	34	10	$^1/_2$	$^1/_2$	1	1	34	2	1
Result from $\mathcal{A}$	$^{539}/_4$	38	8	$^1/_2$	$^1/_2$	1	1	35	2	1

Table 6.1 Decomposing the Conflicting Eighth Order Co–efficients

As one might expect, the majority of the terms are the same in both cases. The facility of this process lies in the fact that task of scouring our set of decorations for possible omissions need now only focus on those sets contributing to the few inconsistent terms in Table 6.1 . There are, according to my program, 993 decorations contributing to the first term in the table, still a very large number among which to search for possible omissions. However, there are only 75 decorations contributing to the third term. After repeated, careful scrutinisation of this set, I am convinced that there are no valid decorations missing from it and can only conclude that it is the result in ref. [24] which is in error. Of course, this is does not conclusively settle the matter of just what exactly is the correct value for the eighth order coefficient, but it does suggest that there are still errors in Munster's calculation, which has already gone through a number of revisions [33].

CHAPTER 7

RESUMMATION SCHEMES AND IMPROVED ALGORITHMS

Thus far we have examined algorithms for directly calculating the first n terms in the strong coupling expansion of the mass gap. This has required the explicit generation and evaluation of every diagram appearing in the strong coupling expansion. In this chapter we shall consider alternatives to this labourious process.

There are methods of reformulating the series expansion in such a way that the contribution from large (even infinite) numbers of strong coupling diagrams may be collected into a single term. These are referred to as resummation schemes – as each contribution to the new series is a sum of terms from the original series.

Let us examine a couple of these schemes. We shall not be considering the resummed series per se, but rather the possibility of finding more efficient algorithms for the computer evaluation of the strong coupling series.

7.1 Inhomogeneous Coupling

The first scheme to consider is an approach we have termed inhomogeneous coupling. The method was first employed in Hamiltonian L.G.T., the work on Z_2 in 2+1 dimensions being published by Moreau et al [40]. The basic idea of inhomogeneous coupling is to divide the plaquettes of the lattice into two distinct groups, known as even and odd plaquettes. The even plaquettes are collected in localised clumps and may be integrated over exactly, the odd plaquettes fill the gaps between the clumps of even plaquettes, and are reintroduced to produce corrections to the integral over the pure even system.

Now let us see how this works in detail. The I.C. analogue to eqn. (2.7) is:

$$Z = [c_0(\beta)]^{N_P} \int \prod_{\{l\}} DU_l \left\{ \prod_{\{P_0\}} (1 + f(U_{P_0})) \prod_{\{P_1\}} (1 + f(U_{P_1})) \right\} \tag{7.1}$$

$\{P_0\}$ being the set of all even plaquettes and $\{P_1\}$ being the set of all odd plaquettes. Instead of expanding both the products within the braces, we now expand only the product $\prod_{\{P_1\}} (1 + f(U_{P_1}))$ to obtain

$$Z = [c_0(\beta)]^{N_P} \sum_{\{\Omega\}} \int \prod_{\{l\}} DU_l \left\{ \prod_{\{P_0\}} (1 + f(U_{P_0})) \prod_{P_1 \epsilon \Omega} f(U_{P_1}) \right\} \tag{7.2}$$

where $\{\Omega\}$ is the set of all possible collections of odd plaquettes on the lattice. Note that in <u>every</u> term in the expansion we integrate over not only a certain local set of odd plaquettes, but also over all the even plaquettes on the lattice. Hence it is imperitive to choose our set of even plaquettes in such a way that the integral

$$\int \prod_{\{l\}} DU_l \prod_{\{P_0\}} (1 + f(U_{P_0}))$$

is not too hard to evaluate. This essentially means choosing our even plaquettes to be in localized clumps, so that the integral above factorizes :

$$\int \prod_{\{l\}} DU_l \prod_{\{P_0\}} (1 + f(U_{P_0})) = \prod_{\{c\}} \int \prod_{l \epsilon c} DU_l \prod_{P_0 \epsilon c} (1 + f(U_{P_0})) \tag{7.3}$$

where $\{c\}$ is the set of all 'clumps' of even plaquettes on the lattice.

The simplest choice is to have the plaquettes collected into corner–connected cubes.

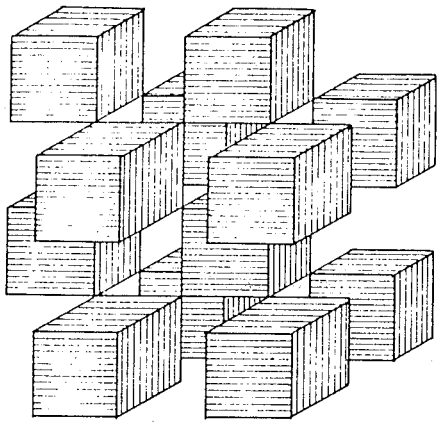


Fig 7.1 Corner Connected Cubes of Even Plaquettes

Let us now examine the diagrams that will will appear in the expansion of Z with this particular choice of the even system. The zeroth order term is

$$Z_0 = \left[C_0(\beta)\right]^{N_P} \left[\int \prod_{l \epsilon C} DU_l \prod_{P \epsilon C} (1 + f(U_{P_0}))\right]^{N_C}$$

$$= \left[C_0(\beta)\right]^{N_P} \left[\text{CUBE}\right]^{N_c} \tag{7.4}$$

with

$$\text{CUBE} \equiv \int \prod_{l \epsilon C} DU_l \prod_{P \epsilon C} (1 + f(U_{P_0})) \tag{7.5}$$

and N_C being the total number of even cubes on the lattice.

Corrections to this zeroth order approximation are obtained by introducing sets of odd plaquettes. Obviously any finite number of odd plaquettes will share links with a finite number of even cubes and the integral

$$\int \prod_{\{l\}} DU_l \prod_{\{P_0\}} (1 + f(U_{P_0})) \prod_{P_1 \epsilon \Omega} f(U_{P_1})$$

will factor into a product over a large number of cubes and some finite number of connected sets of even cubes and odd plaquettes. Hence we must think of the diagrams in our diagram expansion as involving not only the odd plaquettes introduced, but also the even cubes with whom they share links.

Some of the first few non–zero corrections to Z_0 are represented by the diagrams below

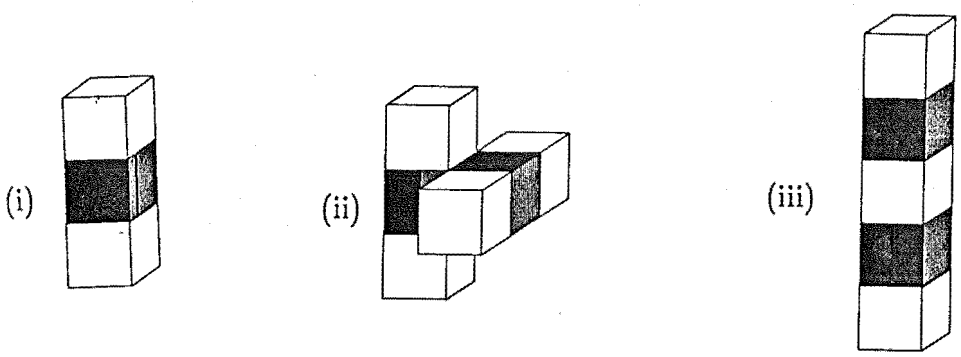


Fig 7.2 Some Low-order Diagrams in the I.C. Expansion
(Odd plaquettes shaded, Even plaquettes white)

The contributions that these diagrams make to the path integral are:

$$\text{(i)}\ \left[C_0(\beta)\right]^{N_p}\left[\text{CUBE}\right]^{N_c-2}\left[F_1(f)\right]^2\left[C_f(\beta)\right]^4$$

$$\text{(ii)}\ \left[C_0(\beta)\right]^{N_p}\left[\text{CUBE}\right]^{N_c-4}\left[F_1(f)\right]^4\left[C_f(\beta)\right]^6$$

$$\text{(iii)}\ \left[C_0(\beta)\right]^{N_p}\left[\text{CUBE}\right]^{N_c-3}\left[F_1(f)\right]^2\left[F_2(f)\right]\left[C_f(\beta)\right]^8$$

where:

$$F_1(f) = \sum_{\mu,\nu \neq 0} C_\mu^5\, C_\nu\, d_\mu\, d_\nu\, N_{\mu\nu f}$$

$$F_2(f) = \sum_{\mu,\nu \neq 0} C_\mu^4 C_\nu^2 d_\mu d_\nu N_{\mu\nu f}$$

$$\text{CUBE} = 1 + \sum_{\mu \neq 0} \left[C_\mu(\beta)\right]^6 d_\mu^2$$

Let us now satisfy ourselves that the resulting series is, in fact, a resummation of the strong coupling series, i.e. that each diagram in the I.C. expansion represents the sum of several S.C. diagrams. First we must note the fact that

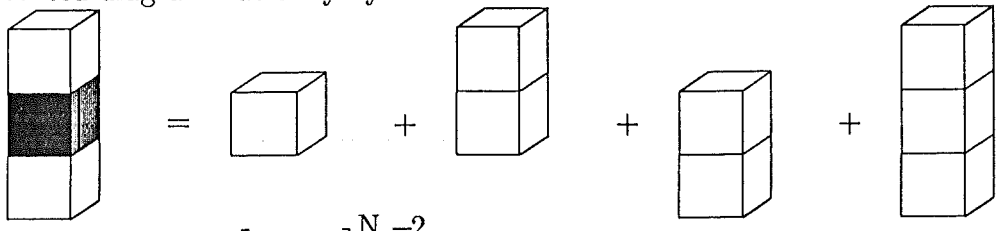
$$\sum_{\mu \neq 0} \left[C_\mu(\beta)\right]^6 d_\mu^2 = S_6 \tag{7.6}$$

where S_6 is defined in ref. [6] to be the contribution made to the S.C. series by the fundamental cube. Consider, now, Z_0

$$Z_0 = \left[1 + S_6\right]^{N_c} \tag{7.7}$$

Upon expanding this product, we would reproduce all the contributions to the S.C. series due to all the possible combinations of these unconnected cubes.

Now consider the first correction : $\left[\text{CUBE}\right]^{N_c-2} \left[F_1(f)\right]^2 \left[C_f(\beta)\right]^4$. The product $\left[F_1(f)\right]^2 \left[C_f(\beta)\right]^4$ may be expanded as a sum of strong coupling terms represented diagrammatically by:



The expansion of the $\left[\text{CUBE}\right]^{N_c-2}$ factor results in adding on all the strong coupling

terms due to all possible combinations of the remaining N_c–2 unconnected cubes.

Any other I.C. diagram could also be expanded in a similar manner, i.e. each diagram in the I.C. expansion does represent a sum of S.C. diagrams.

Of course we have so far been looking at the I.C. expansion of the partition function. It is relatively simple, however, to carry out a moment–cumulant transfomation on the I.C. expansion of Z to obtain the series for Ln Z, and then perform the necessary differentiations to obtain the series for the mass gap, and find that it is also a resummation of the corresponding S.C. series.

On the face of it, then, extracting the S.C. series from the I.C. expansion ought to prove more efficient than calculating it directly. Less diagrams would have to be generated to produce the S.C. series to some given order than would be required in the direct evaluation of the S.C. series.

With this in mind, we proceeded to develop a computer program for evaluating the strong coupling mass gap by this method. Unfortunately, the extra complexity of dealing with I.C. rather than S.C. diagrams resulted in a program which was, if anything, somewhat slower than that described in Chapter 4.

Before leaving the topic of inhomogeneous coupling there is an idea which ought to be discussed. Recall that we may choose to clump the even plaquettes in any manner whatever. Obviously the choice displayed in fig. 7.1 is the easiest choice if one is to obtain an I.C. series containing several terms. On the other hand, one could choose to have a very large clump of even plaquettes, possibly all the plaquettes within a volume measuring 4x4x4 or 5x5x5 lattice units. That is the sort of volume on which finite–lattice calculations are carried out [41]. One could, therefore, obtain an approximation to the integral over the even volume using the methods outlined in ref. [41] and then obtain corrections to the finite lattice result by considering the introdution of the first few sets of odd plaquettes. Hence, the inhomogeneous coupling approach could be applied to enhancing the finite lattice calculation.

7.2 The Mobius Resummation Scheme

The concept of enlarging the volume of the even–plaquette clump leads directly to the next resummation scheme I wish to consider. The scheme in question is the Mobius Resummation scheme presented in ref. [42]. The central result on which this scheme is based is the fact that if one performs the complete integral over some volume of the lattice

$$I(V) \doteq \int \prod_{l \in V} DU_l \prod_{P \in V} (1 + f(U_P)) \tag{7.8}$$

then the result is equal to the sum of the contributions from all possible strong coupling diagrams which fit entirely within V. More importantly, Ln I(V) is equal to the sum over all possible clusters whose supports fit entirely within V

$$H(V) = \text{Ln } I(V) = \sum_{C \in CC(V)} \Psi(C) \tag{7.9}$$

H(V) is referred to as the house of the volume V and CC(V) is the set of all clusters contained entirely within V.

We can see that the method of this resummation will revolve around calculating houses, which are quite explicitly sums of strong coupling diagrams. However, one cannot simply evaluate the house of a large volume of the lattice and extract an infinite–volume S.C. expansion, for the simple reason that the configuration numbers for clusters which do not entirely fill the volume would differ from their values in the infinite volume limit. Hence Arisue and Fujiwara have defined another quantity, referred to as the wall of the volume and defined by

$$W(V) = \sum_{C \in CO(V)} \Psi(C) \tag{7.10}$$

where CO(V) is the set of all clusters whose supports exactly fill the volume V. The relationship between Houses and Walls is given by

$$H(V) = \sum_{V_i \subset V} W(V_i)$$

$$\text{or } W(V) = H(V) - \sum_{V_i \subsetneq V} W(V_i) \tag{7.11}$$

The strong coupling series for the free energy to order n would then be given by

$$f = \sum_{x=1}^{T(\frac{n-2}{4})} \sum_{y=1}^{T(\frac{n-2x}{2x2})} \sum_{z=1}^{T(\frac{n-2xy}{2x2y})} W(x,y,z) \tag{7.12}$$

where T(x) is the integer truncation of the real number x. The walls W(x,y,z) are to be evaluated to only order n.

The walls are evaluated recursively using eqn (7.11). The central task in the process is the calculation of the houses H(x,y,z). Recall H(V) = Ln I(V), and the integral I(V) is just the partition function on the lattice of dimension XxYxZ. Hence, I(V) is given by the sum of the integrals over all closed surfaces contained within that volume. The calculation of the Houses, therefore, involves the same process of finding closed surfaces that is involved in the direct calculation of the S.C. series. The important difference is that in this case the surfaces must be enclosed in a finite volume. This makes the calculation far more ammenable to automation, each of the XYZ unit cubes contained in the volume (X,Y,Z) may be enumerated and all possible different sets of 1,2,3,...,XYZ cubes run through in a systemmatic manner. The algorithm for such a process is very simple, and does not become more complex as we

wish to calculate longer series.

Once I(V) has been calculated, the process of taking the log to form the House and then recursively evaluating Walls can all be carried out with great ease and speed using an algebraic package such as REDUCE.

In short, then, the Mobius resummation provides an excellent means of producing fast, reliable computer algorithms for the calculation of S.C. series. This is borne out by the spectacular achievement published in [27] – a 22^{nd} order calculation of the mass gap and an algorithm at least 1,000 times faster than $\mathscr{D}$.

The algorithm of Arisue and Fujiwara certainly supersedes that of Decker as far as calculations in 2+1 dimensions are concerned. The task remains, however, to design an algorithm based on the Mobius Resummation to carry out calculations in 3+1 dimensions.

Recall that the many–to–one relationship that exists between sets of plaquette connected cubes and closed systems of plaquettes in 3+1 dimensions proved a major stumbling block to the speeding up of my algorithm for 3+1 dimensional calculations. Unfortunately, the problem still occurs if one attempts to extend Arisue and Fujiwara's work to 3+1 dimensions. The Process of calculating the integrals I(V) still requires forming closed surfaces from sets of plaquette connected cubes.

So, despite its other advantages, the Mobius Resummation does not manage to side–step the major obstacle to efficient computer calculations in 3+1 dimensions. There is still every reason to believe that an extension of Arisue and Fujiwara's work will result in a robust and reliable algorithm for 3+1 dimensional calculations.

CHAPTER 8

CONCLUSION

If strong coupling expansions are to remain a viable alternative to Monte Carlo simulations as a tool for studying lattice gauge theories then it is imperitive that one obtain expansions containing sufficient terms as to enable the reliable application of extapolation techniques. Calculating series expansions 'by hand' rapidly becomes impossibly tedious after the first few terms. The obvious alternative is to develop fast and reliable computer algorithms to carry out the calculations. Inevitably the development of these algorithms will advance in a 'leap–frogging' manner, with the first, pioneering programs being improved upon by later workers.

In the context of the present thesis, the pioneering program is that developed by K. Decker [25] for the calculation of the Z_2 mass gap in 2+1 dimensions. To save writing, this algorithm has ben dubbed $\mathscr{D}$. There were a number of ways in which Decker's work could be extended. As is always the case with computer algorithms, there was potential for increasing both the speed and the reliability of the algorithm $\mathscr{D}$. But in this case we are also dealing with a situation in which the pioneering program only goes some of the way to the final goal; although the mass gap for Z_2 in 2+1 dimensions does possess some intrinsic interest, the real motivation of lattice gauge theory is to enable the study of non–abelian gauge theories in 3+1 dimensions. Hence, the tasks of adapting Decker's algorithms to calculations with continuous groups and/or higher dimensions are also available.

In the work presented in this thesis, some progress has been made on most of these possible extensions of $\mathscr{D}$. We shall refer to the new version of the algorithm as $\mathscr{R}$. The most significant achievement has been a radical reduction in memory usage. The algorithm $\mathscr{D}$ requires that all cube configurations containing k cubes be generated

and stored before their contribution to the mass gap is evaluated. Then the program moves on to consider the k+1 cube configurations. If one is calculating the mass gap to high order, this can result in prohibitively large memory requirements. However, Decker's method of generating of all necessary cube configurations without any omissions or duplications is based on strict rules regarding the possible extensions of any given configuration. Hence there is a unique sequence by which any given configuration may be built up cube by cube from a single base cube. It is possible to systematically generate all necessary cube configurations by moving back and forth along the limbs of the tree diagram that may be constructed from these sequences. This latter method of generating configurations requires only a minimal amount of computer memory. At any given point one need only store the information relevant to the few configurations which lie in the unique path from the base cube to the most recently generated configuration.

The exploitation of this tree structure has had the effect of making the memory requirements of the algorithm $\mathscr{R}$ effectively independant of the order to which the mass gap is being evaluated, thus removing a potential barrier to the calculation of higher order expansions.

The speed of the algorithm has been increased in a variety of ways. Most notably, the generation of redundant decorations has been minimized as much as possible. In the algorithm $\mathscr{D}$, a large number of cube configurations are generated which fail to produce any valid contributions to the mass gap. I made a number of alterations to the algorithm designed specifically to avoid wasting time in this way.

The reliability of the algorithm is a matter of some importance. The whole purpose of carrying out the calculations on a computer is to take the strong coupling expansions well beyond what could be achieved 'by hand'. Hence there will not be hand calculations against which to check the algorithm at higher orders. It is desirable, therefore, that the algorithm possess some form of internal check. One of the side–effects of the alterations introduced to speed up the algorithm has been the

emergence of an internal check on the results produced by $\mathscr{R}$. Such a check did not exist for $\mathscr{D}$.

So, the algorithm $\mathscr{R}$ possess greater speed, efficiency and reliability than $\mathscr{D}$. Let us now look at progress that has been made on extending $\mathscr{D}$ to continuous groups and higher dimensions. The fact is that extending $\mathscr{D}$ to continuous groups requires quite fundamental changes to procedure whereby decorations are generated. With Z_2 as the gauge group, there is only one non–trivial irrep and a diagram is uniquely defined simply by listing the plaquettes contained within it. Any group more complex than Z_2, however, contains more than one non–trivial irrep. Therefore, to fully identify a diagram it is necessary not only list its constituent plaquettes, but also the irrep labels attached to the plaquettes. This adds a whole new level of complexity to the task of systematically generating all possible decorations. It also results in a massive increase in the number of decorations contributing to any given order in the expansion. The net effect has been that little progress has been made on the development of an efficient algorithm for calculations with gauge groups other than Z_2.

Extending $\mathscr{D}$ to calculations for Z_2 in 3+1 dimensions is relatively straight forward. A great many minor alterations are required to allow for the greater complexity of the 3+1 dimensional lattice, but no changes are required in the way decorations are defined or generated. However, the 3+1 dimensional lattice does not possess the 1 to 1 correspondence between sets of 3 dimensional cubes and closed systems of plaquettes which exists on the 2+1 dimensional lattice. Hence it is necessary to take care that decorations are not over–counted in the 3+1 dimensional calculation. Although a method does exist for ensuring that over–counting does not occur, it becomes increasingly more time consuming as the decorations become larger. The development of a less unweildy method of avoiding this over–counting problem is really all that is required to make computer generation of strong coupling expansions for Z_2 in 3+1 dimensions as efficient as it presently is in 2+1 dimensions.

with the aid of the new algorithms. The calculation of an 14^{th} order expansion of the energy of a Z_2 'glueball' with non–zero momentum has made it possible to reexamine the restoration of full Lorentz invariance as the 2+1 dimensional Z_2 lattice gauge theory approaches its critical point. The results obtained a few years ago from a tenth order expansion [34,35] were rather difficult to interpret. The 14^{th} order results give a much clearer indication that Lorentz invariance is restored.

The expansion obtained for the Z_2 mass gap in 3+1 dimensions differs from the previously published result. While it has not been possible to resolve the discrepency either way, there are serious reasons to doubt the accuracy of the result published in ref. [24].

As far as the future development of computer algorithms in Euclidean lattice gauge theory is concerned, however, it is very likely that there is little point in continuing with the extension of algorithms based on $\mathscr{D}$. The work of Arisue and Fujiwara [42,27] is already in the process of superceding Decker's algorithm. Not only is the algorithm of ref. [27] much faster than $\mathscr{D}$, it is also a great deal simpler. The fact that the algorithm does not involve such complex processes as searching for bottle–necks and assigning valid positions for P_{in} and P_{out} plaquettes is a great advantage at higher orders when the diagrams tend to become large and convoluted. It is interesting to note that a good deal of progress [26,28,29] has been made in Hamiltonian lattice gauge theory using an algorithm which is effectively a Hamiltonian analogue of that used in ref. [42].

The fact remains, however, that this algorithm has not yet been extended to calculations in 3+1 dimensions, the algorithm presented in this thesis is still the only existing algorithm for computer calculation of strong coupling expansions of the mass gap in 3+1 dimensional Euclidean lattice gauge theory.

REFERENCES

[1] H.D. Politzer Phys. Rev. Letters **26** , 1346 (1973)

[2] D. Gross and F.Wilczek Phys. Rev. Letters **26** , 1343 (1973)

[3] K. Wilson Phys. Rev **D10** , 2445 (1974)

[4] A.M. Polyakov Unpublished

[5] J.M. Drouffe and C. Itzykson Phys Reports **38C** , 133 (1978)

[6] J.M. Drouffe and J–B. Zuber Phys Reports **102** , 1 (1983)

[7] M. Creutz, L. Jacobs and C. Rebbi Phys. Reports **95** , 201 (1983)

[8] J.B. Kogut, D.K. Sinclair and L.Susskind Nuclear Phys. **B114** , 199 (1976)

[9] R. Balian, J.M. Drouffe, and C. Itzykson Phys. Rev. **D11** , 2104 (1975) ; **D19** , 2514 (1979)(E)

[10] A. Guth Phys. Rev. **D21** , 2291 (1980)

[11]K. Binder *in* 'Phase transitions and critical phenomena' ed. C. Domb, M.S. Green Academic Press, London Vol 5B (1976)

[12] M. Creutz Phys. Rev. **D21** , 2308 (1980)

[13] D. Petcher and D.F. Weingarten D22 , 2465 (1980)

[14] F. Gutbrod and I. Montvay Phys. Letters 136B , 411 (1984)

[15] A. Billoire and E Marinari Phys. Ltters 139B , 399 (1984)

[16] M. Grady Alternative Scaling Hypothesis for SU(2) and SU(3) Lattice Gauge Theories , Argonne National Lab Preprint ANL–HEP–PR–87–12 (1987)

[17] E. Marinari The APE Computer and Lattice Gauge Theories , II Universita di Roma Preprint ROM2F/86/005 (1987)

[18] G.Bhanot, K. Bitar, S. Black, P. Carter and R. Salvador Phys. Letters 187B 381 (1987)

[19] S. Duane, A.D. Kennedy, B.J. Pendleton and D. Roweth Phys. Letters 195B 216 (1987)

[20] B. Carpenter, C. Michael and M.J. Teper Phys. Letters 198B , 511 (1987)

[21] I. Montvay Rev. Mod. Phys. 59 , 263 (1987)

[22] J.B. Kogut Five Lite Late–Summer Lectures on Lattices University of Illinois (Urbana–Champaign) Preprint ILL–(TH)–86–#55 (1986)

[23] G.A. Baker Essentials of Pade Approximants Academic Press (1975)

[24] G. Munster Nuclear Phys. B256 , 67 (1985)

[25] K. Decker Nuclear Phys. **B270** , 292 (1986)

[26] A.C. Irving, T.E. Preece and C.J. Hamer Nuclear Phys. **B270** , 536 (1986); Nuclear phys. **B270** , 553 (1986)

[27] H. Arisue and T. Fujiwara Nuclear Phys. **B285** , 253 (1987)

[28] C.J. Burden and C.J. Hamer Hamiltonian Strong Coupling Expansions for (2+1)–Dimensional Quantum Electrodynamics , A.N.U. Preprint (1987)

[29] C.J. Burden Phys. Letters **198B** , 525 (1987)

[30] G. Munster Nuclear Phys. **B180** , 23 (1981)

[31] K. Osterwalder and E. Seiler Annals of Physics **110** , 440 (1978)

[32] J.B. Kogut Rev. Mod. Phys. **51** , 659 (1979)

[33] G. Munster Nuclear Phys. **B190** , 439 (1981); **B200** , 536 (1982) (EA); **B205** , 648 (1982) (E).

[34] N. Kimura and A. Ukawa Nuclear Phys. **B205** , 637 (1982)

[35] K. Seo Nuclear Phys **B209** , 200 (1982)

[36] K. Decker Nuclear Phys. **B240** , 543 (1984)

[37] K. Seo and A. Ukawa Phys. Letters **114B** , 329 (1982)

[38] K. Decker Nuclear Phys. **B257** , 419 (1985)

[39] C. Domb, *in* 'Phase transitions and critical phenomena' ed. C. Domb, M.S. Green Academic Press, New York Vol 3 (1974)

[40] W.R. Moreau, N.I. Churcher and T. Kalotas Nuclear Phys. **B225** , 409 (1983)

[41] P.P. Martin Nuclear Phys. **B220** , 367 (1983)

[42] H. Arisue and T. Fujiwara Progress of Theoretical Phys. **72** , 1176 (1984)