Managing the distribution of recursively calculating matrix elements

L.F. McAven*, M. Schlesinger** and R.D. Kent***

* Department of Physics, University of Windsor, Ontario N9B 3P4, Canada
guest5@server.uwindsor.ca
** Department of Physics, University of Windsor, Ontario N9B 3P4, Canada
msch@server.uwindsor.ca
*** School of Computer Science, University of Windsor, Ontario N9B 3P4, Canada
rkent@server.uwindsor.ca

ABSTRACT

Interaction strengths and material properties are represented mathematically by matrix elements. The calculation of matrix elements is therefore critical in the interpretation and modelling of physical systems. Part of the matrix elements can be factored out as symmetry coefficients of the Racah–Wigner Calculus. One powerful, and widely applicable approach to calculating those coefficients, is a recursive technique which expresses higher power coefficients in terms of lower power ones [1]. A recent recursive algorithm [2] is implemented in RACAH, a package for calculating matrix elements of arbitrary Hamiltonians. We discuss issues involving the use of this algorithm in a distributed environment. A major part of distributed efficiency lies in the management of factorisation, simplification and the recursion, and we concentrate on these aspects. Coefficients frequently reappear in recursion and one needs to balance independence against exact non–reproduction.

Keywords: distributed processing, group theory, matrix elements, recursion, Racah–Wigner calculus

1 Introduction

Symmetries are described mathematically using the theory of groups. The widespread range of symmetries found in diverse systems makes group theory invaluable in understanding characteristics of our universe. Group theory describes aspects of nature from subatomic quarks, through complex molecules to macroscopic materials. It is particularly important for studying spectrosopy, the interaction of materials and electromagnetic radiation which lies at the heart of many modern appliances and technologies.

The new era of nano–technology, in particular integrated nanosystems (ins), makes the influence and understanding of molecular interactions more significant. The recent observation [3] of tin atoms on a copper surface clumping together, and propelling themselves by a chemical reaction between tin and copper, is just one example of a nanosystem ripe for exploitation. We present part of an approach for practically and exactly solving many such systems, using group theory to avoid making approximations.

A useful way to look at the role of group theory is by examining how one exploits symmetries present in a system. The behaviour of the states under operations of symmetry groups allow the states to be labelled by groups. Similarly the symmetries of the interactions mean they can be labelled by groups. As with the states the operators representing the interactions also carry additional *dynamical* information. They are defined with this dynamical information, as well as the group theoretical information, so as to correctly represent the interaction.

The group theoretical matrix elements we refer to in the title are numbers which describe the strength of an interaction between two states. Alternatively one can view matrix elements as providing information about the probability of a transition between one state and another state, due to an interaction. Group theory factors out the geometric or symmetry parts of the matrix elements, those are group theoretical coefficients. The remaining part of the initial matrix element are *dynamical* objects such as radial wavefunctions.

Group theory sets rules about what interactions or transitions are allowed. It tends to be qualitative not quantatative, giving information about when degeneracies are lifted and how, but not what the gap between newly split levels will be. This is because the dynamical part remains independent, and group theory provides no information about it.

In this paper we address the problem of calculating matrix elements, in particular when it is impractical to perform the calculation upon a single modern processor. A software package developed at the University of Canterbury, called RACAH, performs the factorisation process we described above and theoretically can calculate the necessary coefficients. Those are two seperate matters. The process of factorisation or simplifcation, from which the group theoretical coefficients arise, is the simpler. The problem of calculating group theoretical coefficients is more difficult and is computationally impractical for many realistic situations. We discuss how distributed programming may be able to overcome this problem, if not completely then at least to some additional order of magnitude. Distributed programming is also of use for the first problem.

2 Groups

A set of elements and product on that set, together satisfying the requirements of inverse and identity existence and associativity, is a group. The elements might be, for example, operations such as rotations or reflections which leave an object invariant. Associated with any group are representations, which describe different behaviours under the set of operations. In particular irreducible representations, or irreps, form a basis for the possible behaviours. All irreps can be generated by taking the repeated product of a representation called a *primitive* representation ϵ with itself. This both defines the *primitive*, which while not necessarily unique will always exist, as well as allowing one to define the power of an irrep to be the lowest number n for which the irrep appears in the product ϵ^n.

Based on their behaviour under group operations, states and operators may be labelled by irreps. Irreps are multi–dimensional however and can label several states. Such states are distinguished by a *multiplicity* label. This label may correspond to a set of irreps of a set of groups, a basis.

2.1 Group theoretical coefficients

Sometimes it is better to represent the system in one basis than another, for example the JM and the SLJ bases of angular momentum are used for different systems. Generally bases are related by group theoretical transformation coefficients, for the example above by Clebsch–Gordan coefficients ($3jm$ or vector coupling coefficients). If we have a symmetry group G and two possible symmetry chains (bases) H and K then we transform between the bases using transformation coefficients $\langle\lambda\kappa|\lambda\mu\rangle$ as in

$$|\lambda(G)\mu(H)\rangle = \sum_{\kappa} |\lambda(G)\kappa(K)\rangle\langle\lambda\kappa|\lambda\mu\rangle \ . \qquad (1)$$

The general theory of group theoretical transformation coefficients, and their manipulation, is known as the Racah–Wigner calculus or algebra. One associates the different well known types of coefficients ($3jm, 6j, 9j$ etc.) with different relationships between the groups G, H and K. Those groups may be products of groups rather than simple groups. One class of coefficients, the $6j$ or recoupling coefficients (rcc), are particularly important because they can be used to generate other coefficients. Recoupling coefficients relate bases differing in the order in which three irreps are coupled together to give the same overall irrep. For a $6j$ of a group K we consider Eq.(1) with G as a product of three K groups.

2.2 Interactions and matrix elements

In the previous section we mentioned that, based on their behaviour under operations of a group, operators may be labelled by irreps of a group. Let us clarify that statement here.

Interactions play a significant part in physics. For example, they tell us what energy levels there are in systems, what transitions are possible and how a material will react to electromagnetic radiation of some wavelength. Operators are used to mathematically represent the interactions. Hamiltonians are written as a sum of operators each of which represents different interactions. For example

$$\mathcal{H} = \mathcal{H}_1 + \mathcal{H}_2 + \mathcal{H}_3 \qquad (2)$$

where the first part is dependent only on orbital angular momentum, the second only on spin angular momentum, and the third, a spin–orbit term on both spin and orbital angular momentums. Since we want the operators to represent an interaction one defines them with properties reflecting and containing the *dynamical* and group theoretical natures of that interaction.

A matrix element is a function of an interaction, represented by a Hamiltonian, and two states, the initial and final. We write a matrix element as $\langle f|\mathcal{H}|i\rangle$ where i is the initial state, f the final state and $\mathcal{H}$ the Hamiltonian. It is easier to understand the interaction between simpler states so operators are usually defined for simple states. For example consider the interaction between electron spins. If the states are single electrons then the matrix elements of the spin–spin one–body operator are simple. Not all interactions are one–body though, but perhaps more significantly states are usually more complicated.

The Racah–Wigner calculus provides tools for expressing complex states in terms of simple states, for example many–electron, many–shell states in terms of single electron states (for an example of RACAH performing such a reduction see [4]). Furthermore states, operators and thus matrix elements labelled by a group chain ($G \supset K$) can be reduced through the factorising Wigner–Eckart Theorem [1] to *reduced matrix elements*, $\langle f\|\mathcal{H}\|i\rangle$, at a single group level G. It is at this level that one defined the operators.

$$\begin{aligned} & \langle f(G\supset K)|\mathcal{H}(G\supset K)|i(G\supset K)\rangle \qquad (3) \\ = & \sum C_{if}(G\supset K)\langle f(G)\|\mathcal{H}(G)\|i(G)\rangle \end{aligned}$$

The coefficients $C_{if}(G \supset K)$ are coupling coefficients.

3 RACAH and D–RACAH

RACAH is a software package originally developed to calculate coefficients for point groups [1]. Since the version of the early 1980's it has been improved by the inclusion of new algorithms [2] and, more recently, by the addition of components for using the coefficients in calculating matrix elements [4] and fitting energy levels.

At the University of Windsor we are in the process of decomposing RACAH into fundamental components. This is a test case for a software reuse library model [5], [6] which proposes automatic decomposition of C++ code into components which will be stored and reused and adapted as required. Having performed the decomposition we will adapt the components to distributive environments and build a new package which will be called D–RACAH.

Before discussing the improvements we propose let us outline the recursive calculation of group theoretical coefficients.

4 Recursively calculating coefficients

Given an irrep λ of power $p > 1$ we can express λ as the product of the primitive representation and an irrep of power $p-1$. A general transformation coefficient as in Eq.(1) can be reduced to products and sums over coefficients of lower power "at the top". RACAH actually exploits the factorisation of an irrep λ into other than $\epsilon\lambda'$ when possible, i.e. into $\lambda = \lambda_1\lambda_2$. Let us rewrite the form of a general transformation coefficient to $\langle\lambda a\mu|\lambda b\mu\rangle$, where a and b are multiplicities which in general are a chain of irreps of subgroups. At the bottom the chains are both labelled by the irrep μ of some group where all the irreps are 1–dimension so that it is somewhat equivalent to using multiplicity labels associated with no group.

The general recursion relation used by RACAH is a system of equations in the form

$$\sum_a \left(\begin{array}{c|c} \lambda_1\lambda_2 & (\lambda_1\lambda_2) \\ a_1a_2 & r \\ \mu_1\mu_2 & \lambda \\ s & a \\ \mu & \mu \end{array}\right) \left(\begin{array}{c|c} \lambda & \lambda \\ a & b \\ \mu & \mu \end{array}\right) \tag{4}$$

$$= \sum_{b_1b_2} \left(\begin{array}{c|c} \lambda_1 & \lambda_1 \\ a_1 & b_1 \\ \mu_1 & \mu_1 \end{array}\right) \left(\begin{array}{c|c} \lambda_2 & \lambda_2 \\ a_2 & b_2 \\ \mu_2 & \mu_2 \end{array}\right) \left(\begin{array}{c|c} \lambda_1\lambda_2 & (\lambda_1\lambda_2) \\ b_1b_2 & r \\ \mu_1\mu_2 & \lambda \\ s & b \\ \mu & \mu \end{array}\right)$$

where the second term on the left hand side is equal to the coefficient, $\langle\lambda a\mu|\lambda b\mu\rangle$, we want to solve for. This vertical notation makes the functional dependence of the coefficients more apparent. Two coefficients of the same type appear on the right hand side. The two larger coefficients describe the coupling of the λ_1 and λ_2 parts to the original λ.

We have expressed the transformation coefficient containing the irrep λ of power n in terms of brakets containing irreps λ_1 and λ_2. Since λ_1 and λ_2 couple to give λ they cannot both have powers larger than n and provided $n > 1$ at least one can be chosen to be smaller. The significance of being able to choose the factorisation is the speed with which the recursion can reduce the power of brakets being solved for.

The recursion continues until a core set containing primitive irreps is reached. Those are specified by data files or internal formulas or may be calculated by other means [7].

5 Distributing the work around

As we mentioned earlier there are two main parts to the simplification of matrix elements. Firstly one can remove the group theory (geometry) from the system and separate independent interactions. Secondly the group theoretical coefficients that arise in the factorisation can be recursively reduced to a core set of group theoretical coefficients.

In both of those parts there may be circumstances where it is appropriate to make use of distributed programming. One point to note before our discussion is that RACAH stores coefficients in such a way that they can be looked up and not recalculated again and again. If a distributed system is used, as opposed to a parallel system with shared memory, such a lookup becomes more difficult. However with appropriate indexing throughout, processors anywhere can have a small amount of memory allocated to recording calculated coefficients of relevant groups. As processors calculate coefficients the appropriate index could be sent to all relevant places. With transmission times appropriately jiggled (ala travelling salesperson), "neighbouring processors" would be the first to work on the same group.

5.1 Factorisation and simplification

There are really several senses in which a system may be factorised, reduced or simplified. In Eq.(2) we gave a Hamiltonian with three parts. In general it is possible to process different terms in the Hamiltonian separately. They can then be brought back together for calculating something like the energy levels of the system, which require diagonalisation of the matrix (of matrix elements). Another possibility at this stage is the partitioning of the matrix into blocks so that block diagonalisation can be performed. It is well known that for large systems, which result in large matrices, parallel and distributed processing are useful.

The main factorisation we refer to arises through the Wigner–Eckart Theorem (Eq.(3)). RACAH can exploit this theorem to express matrix elements of, for example as performed in [4], a many–electron many–shell system in terms of reduced matrix elements of single–electron orbitals. Also used in such a reduction are factorisations of functions of product groups into products of functions of simple groups. For example, a recoupling coefficient (rcc) of a group $G \times H$ is roughly equal to the product

of an rcc of G and an rcc of H. As an approximate rule, when different groups are involved the calculations can be performed on different processors.

5.2 Recursion for coefficients

The factorisations give rise to numerous group theoretical coefficients, in separating shells for example or in applying the Wigner–Eckart Theorem.

Eq.(4) suggests that the recursive process for calculating general group theoretical transformation coefficients is not simple. Indeed applying this is more complicated that the discussion in section 4 appears. To start with RACAH is programmed with separate forms of Eq.(4) for vector coupling coefficients ($vcc/3jm$) and recoupling coefficients ($rcc/6j$). A sketch of the latter was given in another paper at this conference [7]. Furthermore while this equation is for general transformations, it is easier to use standard relationships to express higher order coefficients (nj, njm types) in terms of the lower order vcc and rcc. For example each of the $9j/4cc$ which appear in the $6j/rcc$ form of the recursion [7] are related to a sum over the product of four $6j/rcc$.

One sees therefore that the number of $6j$ or rcc has potential to grow quickly. However many of the coefficients reappear, and therefore one must be careful to balance independence against exact non–reproduction. Put another way, how independently should processors be allowed to work? Is it possible to tell in advance how many "independent calculations" a processor will perform before it reachs a coefficient being, or already, calculated elsewhere?

The answer to the second question is basically no in general, although some indication of how long it takes to get to coefficients with particular power irreps is possible. The first question is difficult, not only does it depend on the limited knowledge obtained through the second but also on the transmission time, processor types and so on. Each step of the recursion is generally rather simple so it is unlikely one would want to transmit much at each step, because the transmission cost is too relatively.

Returning to the second question, one can be assured that if coefficients are of different powers they are different. This doesn't mean that the coefficients of different powers can be calculated independently, rather that is might be possible to.

Consider for example calculating a coefficient of some "large" power p_l. It is possible to calculate all the coefficients of powers up to a small number p_s on other processors while the recursion is reducing the large coefficient to an expression in terms of p_s power ones which will be known. This is faster than a linear calculation, and is similar to what happens in RACAH when unknown coefficients are calculated for a group where some coefficients have already been calculated. There is potential for significant redundancy in this procedure but with a massively distributed system that doesn't necessarily matter.

6 Summary

We have discussed several places where it is or may be appropriate to make use of distributed programming. It should be pointed out that at the crudest level one can consider parallel programming to be a small scale distributed system without the transmission cost. Thus any improvements in distributed systems would better be applied in a parallel environment if one was large enough. Having stated this, one needs to then realise that it is not at all unusual to want to consider computations easily to large to perform in a practical time on any current parallel machine. At some stage the extreme cost of scaling a parallel machine is outweighed by the availability and relative cheapness of many low end, maybe four processor, machines. Transmission times play a significant role in determining where one draws the line. We aim to provide a tool which can determine available resourses, estimate calculation times taking transmission times into account and perform the calculation in a manner optimal for the environment.

7 Acknowledgements

One of the authors (LFM) appreciates support by The Foundation for Research, Science and Technology (New Zealand) Contract Number: WND801.

REFERENCES

[1] Butler P.H. Point Group Symmetry Applications: Methods and Tables. (Plenum Press, New York, 1981).

[2] Butler P.H., Joyce W.P., McAven L.F. and Searle B.G. "Recursive calculation of non–primitive recoupling and vector–coupling coefficients", to be submitted to J.Phys.A:Math.Gen.

[3] Schmid A.K. *et al* Science Vol. 290, 1561, 2000.

[4] Ross H.J., McAven L.F., Shinagawa K. and Butler P.H. J.Comp.Phys., Vol. 138(2), 332–340, 1996.

[5] Zhang H. "Design and construction of a library–based software reuse model to support grid computing", Masters thesis, School of Computer Science, University of Windsor, Ontario, Canada.

[6] Zhong S. "Software library for reuse-oriented program development", Masters thesis, School of Computer Science, University of Windsor, Ontario, Canada.

[7] McAven L.F., Schlesinger M. and Kent R.D. "The distributive calculation of unitary group recoupling coefficients", International Conference on Computers in Nanoscience (Hilton Head Island, South Carolina, 2001).