COMPUTING THE S MATRIX WITHOUT FEYNMAN GRAPHS*

F. Caravaglios

Theory Division, CERN
Geneva, Switzerland

(Received January 14, 1997)

We discuss some properties of the algorithm ALPHA developed in collaboration with M. Moretti (F. Caravaglios, M. Moretti, *Phys. Lett.* **B358**, 332 (1995); F. Caravaglios, M. Moretti, Oxford University preprint OUTP-9613P, e-Print: hep-ph/9604316).

PACS numbers: 11.80. Fv

1. Introduction

In this section we summarize (in a more heuristic fashion) the basic ideas of the algorithm ALPHA for the computation of the scattering amplitudes, for a more detailed and formal discussion we refer to the paper [1]. Suppose that we want to compute the amplitude of two scalars with momenta p_1 and p_2 going into scalars with momenta p_3 and p_4 within a $\lambda/6\phi^3$ model. The free wave functions for these external states are given by the exponentials $a_i \exp(ip_i x)$, where the a_i are some normalization constants. At the zero perturbative order (*i.e.* at the classical level) these four interacting plane waves will excite additional states, whose wave functions are the plane waves with off-shell momenta given by the sum of the external momenta $p_5 = p_1 + p_2, p_6 = p_1 + p_4, p_7 = p_1 + p_4$. In other words our physical system is described by the superimposition of 7 plane waves

$$\phi(x) = b_j e^{ip_j x} \qquad j = 1, 7. \tag{1}$$

Neglecting all the other states (that are not excited at this perturbative order), the action for this simplified system with 7 states can be written

$$S = \frac{1}{2} p_j^2 b_j^2 - \frac{\lambda}{6} \delta_{ijk} b_i b_j b_k. \tag{2}$$

* Presented at the Cracow International Symposium on Radiative Corrections to the Standard Model, Cracow, Poland, August 1–5, 1996.

The δ_{ijk} is similar to a Kronecker delta: since in our system only those combinations of states which conserve the total momentum interact with coupling λ; thus $\delta_{ijk} = 1$ if $p_i + p_j + p_k = 0$ otherwise the states will not interact and $\delta_{ijk} = 0$. One can verify that this action is well defined and that the scattering amplitude is correctly given by the legendre transform of this function (2) with respect the sources a_i. In particular the equation of motion for the above action is

$$p_i^2 a_i = b_i - \frac{\lambda}{2}\delta_{ijk} b_j b_k. \tag{3}$$

Solving the b_j perturbatively with respect to λ, one can find the value $b_{j,m}$ corresponding to the order λ^m in the coupling constant:

$$b_{j,m} = \frac{1}{2}\frac{\lambda}{p_j^2}\sum_{k,l=1}^{7}\sum_{r,s}^{r+s=m-1} \delta_{j,k,l} b_{k,r} b_{l,s}. \tag{4}$$

We have fixed $b_{j,0} = 1$ when the index j corresponds to an external momentum, otherwise $b_{j,0} = 0$. Higher orders in the coupling constant can be obtained from the iterative formula (4). For instance $b_{5,1}$ is given by (reminding that $p_5 = p_1 + p_2$)

$$b_{5,1} = \frac{\lambda}{p_5^2} b_{1,0} b_{2,0} = \frac{\lambda}{(p_1 + p_2)^2}. \tag{5}$$

Following a derivation [1] very similar to the (2), we can obtain a formula for the scattering amplitude (the truncated connected Green functions) in terms of the $b_{j,m}$. For our specific example

$$A = \sum_{m,r,s}^{m+r+s=1} \frac{\lambda}{6}\delta_{ijk} b_{i,m} b_{j,r} b_{k,s}. \tag{6}$$

For scattering amplitudes with several external legs, we have to compute this sum for higher values of $m + r + s$, according to the power λ^n expected in the final result.

This simple algorithm can be generalized for any lagrangian (also including fermionic particles), and we have used it to compute several physical realistic processes. In the next section we discuss some interesting properties of the algorithm.

2. Some properties of the algorithm ALPHA

The time required for the evaluation of a single matrix element is an important parameter: for Monte Carlo integrations, especially when the phase space contains a lot of variables, the matrix element has to be evaluated in a very large number of points. As a result, one could be even unable to achieve a satisfactory accuracy in the integration within a reasonable time. It is clear that this time significantly depends on the way we write a formula for a computer evaluation: two equivalent mathematical formulas as

$$(a_1 + a_2 + a_3 + \cdots + a_n)^n \tag{7}$$

and

$$\sum_{k_1,k_2,\ldots,k_n}^{k_1+k_2+\cdots+k_n=n} \frac{n!}{k_1!\cdots k_n!} a_1^{k_1}\cdots a_n^{k_n} \tag{8}$$

require very different number of operations; the computer which performs the sums inside the parenthesis before the products is very fast.

Here we estimate the number of operations performed by the algorithm for a process involving N scalar particles and we compare it with the number of Feynman diagrams. For the simplest lagrangian

$$L = \phi\partial^2\phi - \lambda\phi^3 \tag{9}$$

a process of the type

$$\phi(p_1) + \phi(p_2) \to \phi(p_3) + \phi(p_4) + \cdots + \phi(p_N) \tag{10}$$

has N external momenta, and any internal momentum q_i can be written as a linear combination of the external momenta p_j

$$q_i = \sum_{j=1}^{N} a_j p_j . \tag{11}$$

The coefficient a_j can only take two values: 0 or +1 (or −1, depending if the corresponding momentum is ingoing or outgoing). Since each a_j can only have two values, we have 2^N possible choices for the set of a_j in the sum (11). From this set we have to subtract two forbidden choices: one with all the $a_j = 0$ and the other one with all $a_j = +1$ since both would give a null and unacceptable momentum (remembering that $p_1 + p_2 + \cdots + p_N = 0$ for the momentum conservation). Again from the momentum conservation one can easely observe that for each choice, the complementary one obtained changing each a_j from 0 to +1 and *vice versa* from +1 to 0 gives the same internal momentum. Therefore dividing by two the previous set we have that the number of b_j required is $2^{N-1} - 1$. For each iteration in (4) we have to perform $(2^{N-1} - 1)(2^{N-1} - 1)$ operations. Neglecting non-leading terms the number of operations needed to evaluate a matrix element with N external particles increases as $\simeq 4^{N-1}(N)$. Even if an exact calculation of the computation time must take into account several details of the fortran code, one can appreciate how this behavior is in good agreement with Table I, both for the case $\lambda\phi^3$ and $\lambda\phi^3 + \lambda'\phi^4$.

TABLE I

Required CPU time for the computation (single precision) of the matrix elements of a scalar model.

n^o external legs	$\lambda\phi^3$	$\lambda\phi^3 + \lambda'\phi^4$
5	610	2300
6	2840	11250
7	11280	50600
8	48200	230000

We believe that this behavior is remarkable, especially if compared with the number of Feynman diagrams involved in the processes. It is easy to see that such number is given by $(2N - 5)!!$, where N is the number of external legs: in fact if we add a new external leg to a scattering amplitude , we have to multiply the number of diagrams by $(2N - 3)$, since the new leg can be attached to any of the $2N - 3$ internal or external legs of the previous smaller process. Thus the above factorial growth is easy to understand. For a realistic lagrangian the time growth is in general better, since internal symmetries will forbid some interactions and the number of operation for each iteration (4) can be reduced. The standard model processes in Table II show a growth close to 3^N.

TABLE II

Required CPU time for the computation (single precision) of some matrix elements in the Standard Model in electron positron annihilation.

final state	cpu (sec/100000 events)
$e^+e^-e^+e^-$	1120
$e^+e^-e^+e^-\gamma$	3399
$e^+e^-e^+e^-\gamma\gamma$	10729

Therefore the algorithm ALPHA gains a factor with respect the Feynman diagrams which increases as a factorial (the ratio of a factorial and an exponential, still increases as a factorial).

Another nice property of the algorithm which is worthwhile to mention is the good accuracy of the result after all the iterations have been performed. Physical amplitudes of gauge theories often involve significant cancellations among different graphs: each Feynman diagram increases as the initial energy increases. The more particles are in the final state the greater is the power of this growth. But taking the sum of all the contributing diagrams the result do not grow with the energy, since all the powers of the energy cancel each other out. This cancellation among these diagrams needs several digits on a computer machine, and one is forced to use double precision numbers in the fortran code. On the contrary all the $b_{j,m}$ involved in the algorithm are complete physical amplitudes with just one off-shell external leg and various on-shell external legs; thus they do not increase as the energy increases, and the computer machine does not have to deal with huge numbers.

In other words at each iteration there are some *soft* cancellations between quantities which do not have huge powers of the energy.

As a clarifying exercise one could compare the accuracy achieved by a computer in the evaluation of two equivalent formulas

$$(E-(E-1))^8 \tag{12}$$

and

$$\sum_{k=0}^{8}(-1)^k E^{8-k}(E-1)^k \frac{8!}{k!(8-k)!} \tag{13}$$

for $E = 100$.

Only in the former case where the sum inside the parenthesis is performed before the products and only a soft cancellation between E and $E-1$ occurs, one gets an accurate result.

The code ALPHA can compute the connected Green functions for any effective lagrangian by explicitly performing the legendre transform: any kind of interaction, including form factors which have a complicated momentum dependence can be easily implemented.

In addition it is worthwhile to note that even if the algorithm is originally based on the legendre transform formula, its realization into a fortran code can be adapted to more complicated rules which do not come directly from the legendre transform of an effective lagrangian; this will enable the user to modify the code according to his requirements and to apply it for different physical problems.

REFERENCES

[1] F. Caravaglios, M. Moretti, *Phys. Lett.* **B358**, 332 (1995); F. Caravaglios, M. Moretti, Oxford University preprint OUTP-9613P, e-Print: hep-ph/9604316.