Optimizations for the Computation of Radiative Corrections

Thomas Hahn
Max-Planck-Institut für Physik

When computing radiative corrections, the Running Time of a program is basically $T = N\tau + \varepsilon$, where

$N =$ Number of points in phase and parameter space,
$\tau =$ Time for one point in phase and parameter space.

Consider typical numbers: $N \sim 10^5 \ldots 10^7$, $\tau \sim 100$ ms.

This Talk presents two techniques:

1. Parallelization: $N \to \dfrac{N}{N_{\text{processors}}}$

2. Fermionic Objects: $\tau \sim \mathcal{O}(n_F^2) \to \tau \sim \mathcal{O}(n_F)$

Thomas Hahn: Optimizations for the Computation of Radiative Corrections. Talk given at RADCOR/Loops & Legs, September 2002.

Parallelization

Programs which compute radiative corrections often contain "naturally" parallel structures.

Example: the same calculation is performed for each point in parameter space:

```
do tanBeta = 2, 20
  xsec = CrossSection(tanBeta)
  print *, tanBeta, xsec
enddo
```

Note: The cross section can be computed completely independently for different values of `tanBeta`.

Such programs are known as "essentially parallel."

Parallelization of such programs is very simple on an SMP architecture: One machine with many processors.

Due to high demand in areas such as commercial Web hosting, such machines are increasingly popular and cheap. In fact, there may already be one at your institute!

Programming Technique

- The parent process executes the main loop.
- In each cycle, the parent forks off a child process to do the actual calculation.
- The operating system automatically sends the concurrent processes on the available processors.

The implementation in Fortran is so simple because each child process runs completely independently:

- No need to worry about shared data (mutexing etc.), or reentrant code. This is particularly important in Fortran 77, where it is in general not possible to control either.
- But: there is no way for the child process to communicate its results back to the parent process.
- To avoid mixing up the output, each child process must write its results to a different file.

  Easy solution: parent process opens files explicitly, redirects standard output if necessary.

Example Program

```
seqnum = 0
processes = 0
processors = 4

do tanBeta = 2, 20

  if( processes .lt. processors ) then
    processes = processes + 1
  else
    pid = wait(0)
  endif

  seqnum = seqnum + 1

  if( fork() .eq. 0 ) then


    write(log, '("log.",I5.5)') seqnum
    open(6, file=log)
    xsec = CrossSection(tanBeta)
    print *, tanBeta, xsec
    return
  endif

enddo

do i = 1, processes
  pid = wait(0)
enddo
```

If a free processor is available, fork next job immediately,

otherwise wait for one of the running child processes to finish.

Child process executes the if-statement, parent skips to next cycle of main loop.

Redirect standard output so results won't get mixed up.

Child process terminates here.

Before the parent terminates, it waits for all child processes to finish.

Performance

- A fork is negligible in terms of CPU time unless very short blocks are parallelized.
- The number of child processes running concurrently should not exceed the number of processors in the system, otherwise CPU time will unnecessarily be diverted into switching between processes.

Conclusions

- Programs for computing radiative corrections are often natural candidates for parallelization.
- On SMP machines, it is very easy to parallelize Fortran programs with only few changes in the code.
- It is not difficult to attain near-optimal speed-up, i.e. $T \to \frac{T}{N}$ on N processors.

Fermionic Objects

An amplitude containing external fermions has the form

$$\mathcal{M} = \sum_{i=1}^{n_F} c_i F_i \quad \text{where} \quad F_i = (\text{Product of}) \langle u | \Gamma_i | v \rangle .$$

The usual way to compute $|\mathcal{M}|^2$ is to use the trace technique:

$$|\mathcal{M}|^2 = \sum_{i,j=1}^{n_F} c_i^* c_j F_i^* F_j$$

where

$$F_i^* F_j = \langle v | \bar{\Gamma}_i | u \rangle \langle u | \Gamma_j | v \rangle = \mathrm{Tr}\left(\bar{\Gamma}_i \, |u\rangle\langle u| \, \Gamma_j \, |v\rangle\langle v|\right) .$$

Advantage: method is independent of any representation.

Problem: one has to compute n_F^2 of the $F_i^* F_j$ and not just n_F F_i's as for $\mathcal{M}$ alone.

Take $n_F = 20$, and we have to compute 400 traces.

Things get worse the more vectors are in the game, e.g. in multi-leg final states, or with polarization effects:
Basically all combinations of vectors can appear in the Γ_i, thus $n_F \sim$ (# of vectors)! .

Alternatives to Trace Technique

Obvious way: plug in 4-dim. representation for $|u\rangle$, $|v\rangle$, Γ_i and do the matrix algebra.

However:

- Tricky to implement because in general Lorentz indices connect different fermion chains.
- Calculational efficiency not perfect because 4-dim. representation contains redundancy (γ-matrices with whole blocks of zeros).

Fermions need to be represented by 2-dim. objects only, and they come in two kinds: left- and right-handed.

Therefore it is natural to use the Weyl representation.

- Why not use the full Weyl–van-der-Waerden formalism, where all fermionic objects are 2-dim. already in the Feynman rules?

  Answer: No control over D-dimensional objects, can only do tree level in WvdW.

Here we stay in D (or 4) dimensions during the algebraical simplification. We go to the Weyl representation only just before the numerical evaluation, and only for the purpose of computing the F_i's efficiently.

Sigma Chains

Define Sigma Chains as generalized 4-dim. sigma matrices:

$$\sigma_\mu = (\mathbb{1}, -\vec{\sigma})\,, \qquad \sigma_{\mu\nu\rho\ldots} = \sigma_\mu \overline{\sigma}_\nu \sigma_\rho \cdots\,,$$
$$\overline{\sigma}_\mu = (\mathbb{1}, +\vec{\sigma})\,, \qquad \overline{\sigma}_{\mu\nu\rho\ldots} = \overline{\sigma}_\mu \sigma_\nu \overline{\sigma}_\rho \cdots\,.$$

Define 2-component Spinors as

$$\langle u| \equiv \left(\langle u_+|\,, \langle u_-|\right)\,, \qquad |v\rangle \equiv \begin{pmatrix} |v_-\rangle \\ |v_+\rangle \end{pmatrix}.$$

Using the chiral representation

$$\gamma_\mu = \begin{pmatrix} 0 & \sigma_\mu \\ \overline{\sigma}_\mu & 0 \end{pmatrix}, \quad \omega_+ = \begin{pmatrix} \mathbb{1} & 0 \\ 0 & 0 \end{pmatrix}, \quad \omega_- = \begin{pmatrix} 0 & 0 \\ 0 & \mathbb{1} \end{pmatrix}$$

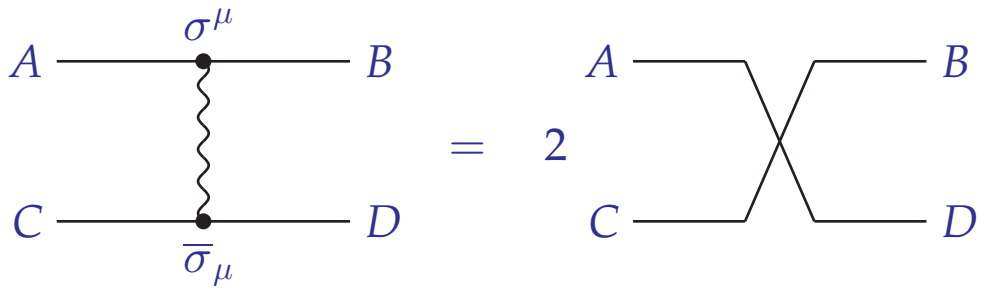
it is easy to show that every chiral 4-dim. Dirac chain can be converted to a single 2-dim. sigma chain:

$$\langle u|\,\omega_-\gamma_\mu\gamma_\nu\cdots|v\rangle = \langle u_-|\,\overline{\sigma}_{\mu\nu\ldots}\,|v_\pm\rangle\,,$$
$$\langle u|\,\omega_+\gamma_\mu\gamma_\nu\cdots|v\rangle = \langle u_+|\,\sigma_{\mu\nu\ldots}\,|v_\mp\rangle\,.$$

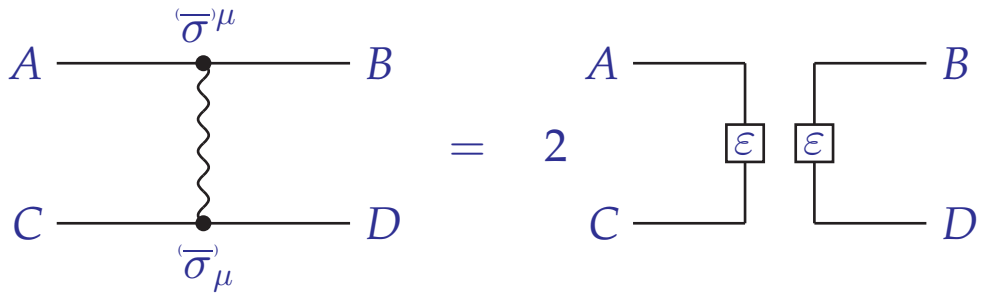
Fierz Identities

With the Fierz identities for sigma matrices it is possible to remove all Lorentz contractions between sigma chains:

$$\langle A|\,\sigma_\mu\,|B\rangle\;\langle C|\,\overline{\sigma}^\mu\,|D\rangle \;=\; 2\;\langle A|D\rangle\;\langle C|B\rangle$$



$$\langle A|\,\overline{\sigma}_\mu\,|B\rangle\;\langle C|\,\overline{\sigma}^\mu\,|D\rangle \;=\; 2\;\langle A|\,\varepsilon\,|C\rangle\;\langle B|\,\varepsilon\,|D\rangle^{\ddagger}$$



$^{\ddagger}\,\varepsilon = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$ is the spinor metric.

Implementation

We can go over to the Weyl representation completely and express all other kinematical objects through sigma chains:

$$g_{\mu\nu} = \frac{1}{2}\operatorname{Tr}\sigma_{\mu\nu}$$

$$\varepsilon_{\lambda\mu\nu\rho} = \frac{1}{2}\operatorname{Tr}\sigma_{\lambda\mu\nu\rho} - \frac{1}{4}\left(\operatorname{Tr}\sigma_{\lambda\mu}\operatorname{Tr}\sigma_{\nu\rho} - \operatorname{Tr}\sigma_{\lambda\nu}\operatorname{Tr}\sigma_{\mu\rho} + \operatorname{Tr}\sigma_{\lambda\rho}\operatorname{Tr}\sigma_{\mu\nu}\right)$$

$$\sigma_\mu k^\mu = \begin{pmatrix} k^0 + k^3 & k^1 - \mathrm{i}k^2 \\ k^1 + \mathrm{i}k^2 & k^0 - k^3 \end{pmatrix} =: \hat{k}$$

$$\overline{\sigma}_\mu k^\mu = \begin{pmatrix} k^0 - k^3 & -k^1 + \mathrm{i}k^2 \\ -k^1 - \mathrm{i}k^2 & k^0 + k^3 \end{pmatrix} =: \hat{\bar{k}}$$

For the numerical evaluation, we then need to implement:

- objects: (arrays)

$$|u_\pm\rangle \sim \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \qquad \text{(for all spinors)}$$

$$\hat{k},\ \hat{\bar{k}} \sim \begin{pmatrix} a & b \\ c & d \end{pmatrix} \qquad \text{(for all vectors)}$$

- operations: (functions)

$$\langle u|v\rangle \sim (u_1\ u_2)\cdot\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \qquad \text{“ss”}$$

$$\hat{k}\,|v\rangle \sim \begin{pmatrix} a & b \\ c & d \end{pmatrix}\cdot\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \qquad \text{“ms”}$$

$$\hat{k}_1\hat{k}_2 \sim \begin{pmatrix} a_1 & b_1 \\ c_1 & d_1 \end{pmatrix}\cdot\begin{pmatrix} a_2 & b_2 \\ c_2 & d_2 \end{pmatrix} \qquad \text{“mm”}$$

These functions are sufficient to compute all sigma chains if an Accumulator (“register” for intermediate results) is used:

$$\langle u|\,\hat{k}_1\hat{\bar{k}}_2\hat{k}_3\,|v\rangle = \texttt{ss}(u,\ \texttt{ms}(\hat{k}_1,\ \texttt{ms}(\hat{\bar{k}}_2,\ \texttt{ms}(\hat{k}_3,\ v))))$$

Conclusions

- The number of fermionic structures that have to be computed can be reduced from $\mathcal{O}(n_F^2)$, with the trace technique, to $\mathcal{O}(n_F)$ when calculating the fermion chains directly.
- The amplitude $\mathcal{M}$ then becomes a handy complex number, rather than a "bunch of form factors," $\{c_i F_i\}$. For example, we can compute $|\mathcal{M}|^2$ by simply squaring, we don't have to sum up.
- Using the Weyl representation, the computation of the fermion chains is broken down from a 4-dim. to a 2-dim. problem. Lorentz contractions between fermion chains can be completely disentangled.
- The implementation is simple, even more so if one systematically expresses all kinematic objects through sigma chains, also non-fermionic ones like $g_{\mu\nu}$, $\varepsilon_{\lambda\mu\nu\rho}$, k_μ.