Notes On the Twobody Density Code Structure: Units and Integrations

Harald W. Grießhammer

Institute for Nuclear Studies, Department of Physics, George Washington University, Washington DC 20052, USA

Based on hgrie's and Alex Long's research of the code, these notes are expanded from my notes in hgrie's notebook *[Few-N Processes With Densities, pp. 15-21]*. Equation and page numbers in *emphasis* refer to this notebook. In addition, there are comments in the code and emails Nov 2023.

Text in `texttt` refers to code variable, file names, routines, etc.

?? This is right this is added text

Version 26th March 2024

Contents

1 Purpose

An explanation of the twobody code structure with emphasis on integrations and units.

2 Units

The "mantle" code has base unit fm (but NOT the output file/`Result()`, see below!):

`p12, p12p`	momenta in fm^{-1}
`rho`	2N density in fm^3 (quantum numbers per volume momentum space)
`k`	photon omentum/energy still given in MeV

However, in `finalstatesums.twobodyvia2Ndensity.f`, the call

```
call Calculate2BIntegralI2(...,p12*HC,P12MAG(ip12p)*HC,...)
```

converts the momenta from fm^{-1} to MeV. That subroutine is defined in `calculate2BI2.f`. Therefore, that routine and the subsequent "kernel" parts of the code use base unit MeV:

`calculate2BI2.f`	
`2Bkernel.*.f`	part of "kernel", specific to process
`2Bspinsym.*.f`	part of "kernel", specific to process
`2Bspinasym.*.f`	part of "kernel", specific to process

Therefore, the twobody "kernel diagrams" in these files are all using base unit of MeV.

Output units: The code is constructed such that if the kernel is given in units of MeV^{-n}, then the variable `Result()` and its output-to-file is in $\mathrm{MeV}^{(3-n)}$.

Multiplying by powers of `HC` $= 197....$ MeVfm translates into output of final MEs `Result()` into powers of fm.

EXAMPLE: Compton has twobody kernel in MeV^{-4} ($n = 4$), so the resultant ME is in MeV^{-1}. To convert to fm, multiply by `HC`. In Compton, that multiplication by HC is not done in the fortran code, but later in the mathematica processing files.

EXAMPLE: Pion Photoproduction kernel has units MeV^{-2} if the output should be the twobody functions F_{TL}. Therefore, $n = -2$ and the `Results()` output is in MeV^1. But the F_{TL} output should be in fm^{-1}, so divide here in kernel by `HC` to get fm^{-1} units in `Results()`.

Routines which compute vectors, like `calcmomenta.f`, simply use the same base unit in and out, i.e. are "unit neutral".

Therefore, the overall units of the output are MeV^{3-n} for a kernel with base units MeV^{-n}!

The conversion $\frac{(\mathtt{HC})^3}{(2\pi)^3}$ above is programmed in `finalstatesums.twobodyvia2Ndensity.f`'s "fact=...") and converts between the fm units of the "mantle" and the MeV units of the "kernel".

It ALSO includes ONE of the Fourier volumes $\frac{1}{(2\pi)^3}$ as a factor of the twobody integration. There is no second Fourier volume (killed by phase space). This guarantees that onebody and twobody have the same size and can simply be summed to get the total amplitude:

$$\text{amplitude} \;=\; \text{onebody} \;+\; \text{twobody} \quad , \tag{2.1}$$

without any relative factors (provided both provide output in same base units). However, *to include this factor or not depends on your choice of twobody kernel.*

That means in pion photoproduction, one may instead define it as part of the prefactor `K2n` of a diagram. This implies that if you want the twobody `Result()` output to be F_{TL}, you must un-compensate it here by `*(2*Pi)**3`. On the other hand, if you want the twobody `Result()` output to be E_+ etc., so that you can simply add as in eq. (2.1) above, then the prefactor `K2n` shouldNOT contain the $\frac{1}{(2\pi)^3}$, i.e. multiply NOT with `*(2*Pi)**3/HC`, but with

```
K2n = sqrt(4*Pi*alpaEM)*gA*mpi**2/(16*Pi*fpi**3)*10**3          (2.2)
```

to get `Result()` in the canonical units of $10^{-3}\ m_{\pi+}^{-1}$.

Set your kernel up in `2Bkernel.*.f` so that your `Result()` has the desired units and factors of $(2\pi)^3$. Do NOT make unit changes outside this file!

3 Integration Coding

3.1 Overall Structure

The integral is symbolically (including the permutation factor already in the code itself, and denoting on the left the units of each line):

$$
\begin{aligned}
[\mathrm{MeV}]^{3-n} \quad & \binom{A}{2} \langle M' | O_{12} | M \rangle \equiv \\
[\mathrm{fm}]^{-6} \quad & \binom{A}{2} \sum_{\substack{\texttt{mt12,j12,s12,l12,m12} \\ \text{in } \texttt{main.twobody.f}}} \sum_{\substack{\texttt{mt12p=mt12} \\ \texttt{j12p,s12p,l12p} \\ \texttt{m12p,Mzp,Mz} \\ \text{in } \texttt{finalstatesums.*.f}}} \underbrace{\int \frac{\mathrm{d}p'_{12}\ p'^2_{12}}{(2\pi)^3} \int \mathrm{d}p_{12}\ p^2_{12}}_{\text{in } \texttt{fact=...} \text{ of } \texttt{finalstatesums.*.f}} \\
[\mathrm{MeVfm}]^3 \quad & \times \texttt{HC}^3 \qquad\qquad \text{in } \texttt{fact=...} \text{ of } \texttt{finalstatesums.*.f} \\
[\mathrm{fm}]^3 \quad & \times \rho^{M'M}_{\alpha'_{12}\alpha_{12}}(p'_{12}, p_{12}; \vec{q}) \qquad \text{in } \texttt{fact=...} \text{ of } \texttt{finalstatesums.*.f} \\
[\text{no units}] \quad & \times \sum_{\substack{\texttt{msp}\equiv m^{s'}_{12},\texttt{ms}\equiv m^{s}_{12} \\ \text{in } \texttt{calculate2BI2.f}}} \langle l'_{12} s'_{12} (m'_{12} - m^{s'}_{12}) m^{s'}_{12} | j'_{12} m'_{12} \rangle \langle l_{12} s_{12} (m_{12} - m^{s}_{12}) m^{s}_{12} | j_{12} m_{12} \rangle \\
[\text{no units}] \quad & \times \int_{\text{in } \texttt{calculate2BI2.f}} \mathrm{d}\hat{p}'_{12}\ \mathrm{d}\hat{p}_{12}\ Y^{\dagger}_{l'_{12}(m'_{12}-m^{s'}_{12})}(\hat{p}'_{12})\ Y_{l_{12}(m_{12}-m^{s}_{12})}(\hat{p}_{12}) \\
[\mathrm{MeV}]^{-n} \quad & \times \underbrace{O^{\alpha'_{12} m^{s'}_{12} \alpha_{12} m^{s}_{12}}_{12}(\vec{p}\,'_{12}, \vec{p}_{12}; \vec{q})}_{\text{in } \texttt{2Bkernel.*.f}} \ .
\end{aligned}
\tag{3.1}
$$

using the Clebsch-Gordan coefficients $\langle j_1 j_2 m_1 m_2 | j m \rangle$ in the convention of refs. [1, 2]. The assignation of quantum numbers between code and the quantities should be self-explaining except maybe `ms` $= m^s_{12}$, `msp` $= m^{s'}_{12}$. The isospin `t12` $= t'_{12}$ is determined by the Pauli

principle: $(-) = (-)^{s_{12}+l_{12}+t_{12}} = (-)^{s'_{12}+l'_{12}+t'_{12}}$, implemented in `main.twobody.f` for the `t12` $\in \{0;1\}$ case as `t12=(1-(-1)**(l12+s12+1))/2`.

Only the last line is coded in the kernel's `2Bkernel.*.f` and `spintricks*.f`, namely the twobody operator $\mathcal{O}_{12}$ projected onto orbital angular momenta via Clebsches.

All other lines are coded in the mantle.

3.2 Radial Integrations

The radial integration grids are set up in `main.twobody.f` via

`call TRNS(NP12A,NP12B,NP12,P12A,P12B,P12C,P12MAG,AP12MAG)`

They are passed down to `finalstatesums.*.f`, where they are performed:

`ip12` $= 1, \ldots,$ `NP12`	index of the `NP12` points of radial integration p_{12}
`P12MAG(ip12)`	momentum p_{12} of index `ip12`
`AP12MAG(ip12)`	Gaußian weight of momentum p_{12} of index `ip12`

The integration over p_{12} is done as sum over `ip12` in `main.twobody.f`, but the integrand (including weight p_{12}^2`AP12MAG`) is set up in `finalstatesums.*.f`.

The integration over p'_{12} is done inside `finalstatesums.*.f` as sum over `ip12p`:

`ip12p` $= 1, \ldots,$ `NP12`	index of the `NP12` points of radial integration p'_{12}
`P12MAG(ip12p)`	momentum p'_{12} of index `ip12p`
`AP12MAG(ip12p)`	Gaußian weight of momentum p'_{12} of index `ip12p`

The combined integrand plus permutation factor plus measure plus density plus $\hbar c$ factor is set up in `finalstatesums.*.f` as

$$\begin{aligned} \texttt{fact=}&\underbrace{\texttt{Anucl*(Anucl-1)/2}}_{\binom{A}{2}}\underbrace{\texttt{*p12**2*wp12}}_{\mathrm{d}p_{12}\,p_{12}^2} \\ &\underbrace{\texttt{*P12MAG(ip12p)**2*AP12MAG(ip12p)/(2*Pi)**3}}_{\frac{\mathrm{d}p'_{12}\,p'^2_{12}}{(2\pi)^3}} \\ &\underbrace{\texttt{*rho(ip12,ip12p,rindx)}}_{\rho^{M'M}_{\alpha'_{12}\alpha_{12}}(p'_{12},p_{12};\vec{q})}\underbrace{\texttt{*HC**3.d0}}_{(\hbar c)^3} \end{aligned} \tag{3.2}$$

3.3 Angular Integrations

The angular integration grids are set up in `main.twobody.f` via

`call Setquad12(th12,Nth12,phi12,Nphi12,...)`

and passed via `finalstatesums.*.f` to `calculate2BI2.f`, where they are performed, calling the kernel-specific file `2Bkernel.*.f`.

The integral is performed by adding iteratively to the variable `Int(extQnum,ml12p,ml12)` the following:

$$\underbrace{\texttt{Yl12pstar(ml12p)}}_{Y^{\dagger}_{l'_{12}(m'_{12}-m^{s'}_{12})}(\hat{p}'_{12})}\ \underbrace{\texttt{*Yl12(ml12)}}_{Y_{l_{12}(m_{12}-m^{s}_{12})}(\hat{p}_{12})}\ \underbrace{\texttt{*angweight12(ith,iphi)}}_{\mathrm{d}\hat{p}_{12}}\underbrace{\texttt{*angweight12(jth,jphi)}}_{\mathrm{d}\hat{p}'_{12}}$$

$$\underbrace{\texttt{*Kernel2B(extQnum,s12p,msp,s12,ms)}}_{O_{12}^{\alpha'_{12}m^{s'_{12}}_{12}\alpha_{12}m^{s}_{12}}(\vec{p}\,'_{12},\vec{p}_{12};\vec{q})} \tag{3.3}$$

[Strictly speaking, `Yl12pstar=Real(Yl12p(ml12p))-ci*Imag(Yl12p(ml12p))` is defined just before the summation `do extQnum=1,extQnumlimit` is done.]
[`ml12=m12-ms`, `ml12p=m12p-msp`. The other quantum numbers should be self-explanatory.]

The summation is again over indices:

`ith` $= 1,\ldots,$ `Nth12`	index of the `Nth12` points of angular integration θ_{12}
`iphi`	index of the `Nphi12` points of angular integration ϕ_{12}
	where `iphi=ith` when the Lebedev-Laikov method is used
	`iphi=1,...,Nphi12` when Gaußian integrations in θ and ϕ separately are used
`th12(ith)`	angle θ_{12} of index `ith`
`phi12(iphi)`	angle ϕ_{12} of index `iphi`
`angweight12(ith,iphi)`	weight of angle (θ_{12},ϕ_{12}) of index (`ith`,`iphi`).

For the primed variables:

`jth` $= 1,\ldots,$ `Nth12`	index of the `Nth12` points of angular integration θ'_{12}
`jphi`	index of the `Nphi12` points of angular integration ϕ'_{12}
	where `jphi=jth` when the Lebedev-Laikov method is used
	`jphi=1,...,Nphi12` when Gaußian integrations in θ and ϕ separately are used
`th12(jth)`	angle θ'_{12} of index `jth`
`phi12(jphi)`	angle ϕ'_{12} of index `jphi`
`angweight12(jth,jphi)`	weight of angle $(\theta'_{12},\phi'_{12})$ of index (`jth`,`jphi`).

4

4.1

5 Concluding Questions

Acknowledgements

As usual, we acknowledge no responsibility whatsoever.

References

[1] A. R. Edmonds, "Angular Momentum in Quantum Mechanics", Princeton University Press (1974).

[2] R. L. Workman *et al.* [Particle Data Group], PTEP **2022** (2022) 083C01 doi:10.1093/ptep/ptac097 and http://pdg.lbl.gov.