JAMを用いたシミュレーション
インストールから実行まで

金築俊輔

2015年5月12日

概 要

JAMのインストールと実行の方法について簡単な説明を書いておく。断面積に直す際の換算式や、簡単なfortranの文法もメモとして書いておく。

少しずつ書き足しています。何か変なところなどあれば教えて下さい。

目 次

1 イントロダクション

JAM[1]はハドロンカスケード模型を用いた原子核反応計算コードの一種である。これはRHICなどで実験が行われている重イオン衝突の解析によく用いられるものであるが、私はJ-PARC E05実験およびS-2Sスペクトロメータの設計のためのシミュレーションに用いている。具体的には、JAMで生成した散乱粒子分布を用いてトリガー検出器の構成や要求性能を見積もったり、エネルギー較正に使えそうな反応過程について調べたりしている。20 MeV – 200 GeVくらいの幅広いエネルギー領域の核反応シミュレーションに

[1]JAMはJet AA Microscopic transport modelの略だが、ソースコードには”This is JAM (Judy And Mary)”と書いてある。

使えると言われているので、我々のようなハイパー核分光実験くらいのエネルギーでも使って良いと思う[2]。このレポートでは JAM 本体でどのような模型によって計算が行われているかには触れず、とにかく自分のマシンにインストールして動くようにするまでの手順と、使う際に必要になるさまざまなスイッチの意味や、実際の反応率に直す際の計算方法をまとめておく。

2 インストール

2.1 インストールの手順

まず、奈良さんのウェブサイトからソースコードをダウンロードする。基本的に一番新しいバージョンを選べば良いと思う[3]。展開してできたディレクトリに入る。
http://www.aiu.ac.jp/~ynara/jam/

```
$ tar jxvf jam-1.362.tar.bz2
$ cd jam-1.362
```

そこに INSTALL というファイルがあり、インストール方法が次のように書かれている。

```
F77=gfortran ./configure --prefix=/home/myhome/
or write the following in your .bashrc
export F77="gfortran"
export FFLAGS="-O3 -g"
 aclocal
 automake -a
 autoconf
 libtoolize
```

説明が簡素すぎる気がするが、とにかく素直に従って実行して行く。

```
$ F77=gfortran ./configure --prefix=/home/kanatsuki/JAM/
$ aclocal
$ automake -a
$ autoconf
$ libtoolize（警告が出たら言われるがままに --force オプションを付ける。）
```

これで準備ができた。ここで今まで我慢してきた make; make install をする。prefix で指定したディレクトリ[4]に、今回ダウンロードしたソースから作った JAM のライブラリがインストールされる。

```
$ make
$ make install
$ ls /home/kanatsuki/JAM/lib/
libjam.a  libjam.la  libjam.so  libjam.so.0  libjam.so.0.0.0
```

[2] JAM がどれくらい正しい結果を返してくれるかについては、最初にがんばっていろいろ調べたので、別のレポートでまとめる。
[3] 1.16 あたりまでのバージョンにはバグがあり、私はかなり困ったので、特に K ビームを使う人は注意する必要がある。
[4] 絶対パスを書かないと怒られるので注意。

最後に、環境変数の設定をしておく。

```
export JAMLIB=/home/kanatsuki/JAM/lib
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$JAMLIB
export LD_RUN_PATH=$JAMLIB
export LD_AOUT_LIBRARY_PATH=$JAMLIB
```

これで JAM のインストールは完了である[5]。

2.2 動作確認

インストールが完了したら JAM を走らせてみる。書くのが面倒なので適当に./jamgo してみて下さい。走らせると JAMRUN.DAT というファイルができるので、それをチェックして思ったようなシミュレーションができているか確認しましょう。

3 プログラミング

main ディレクトリにたくさんのサンプルファイルがあるのでそれを参考にすれば良い[6]。概略をざっくり述べると、

1. ビーム条件の記述：粒子種、入射運動量（エネルギー)、衝突係数
2. その他の条件の記述：標的核種、座標系、イベント数
3. 最低限のメソッドの実行：初期条件 jaminit()、イベント生成 jamevt()、終了処理 jamfin

ということになる。これにほしい情報の出力等の処理を加えて行く。

3.1 粒子の ID 番号

これはマニュアルに書いてあるが、いままで使ってきたものをまとめておく。

粒子	番号	粒子	番号	粒子	番号	粒子	番号
p	2212	Λ	3122	π^-	-211	K^-	-321
n	2112	Σ^-	3112	π^0	111	K^0	311
Δ^-	1114	Σ^0	3212	π^+	211	$\bar{K}^0$	-311
		Σ^+	3222	η	221	K^+	321
		Ξ^-	3312	γ	22	K_S	310
		Ξ^0	3322			K_L	130
		Ξ^{*-}	3314				
		Ξ^{*0}	3324				

[5]私は初期に環境変数の設定を忘れていたが、JAM が実行できてしまい、そのせいで大変無駄な苦労をした。おそらくワークサーバに既に共有ソフトとして JAM がインストールされていて、そちらのライブラリを参照できてしまったのだと思う。

[6]ver. 1.21 くらいまでは私の mainkk.f もパッケージに入っていたが、いまは消えている。まあ見せられたものでもないが...。

3.2 いろいろな情報のとり方

これも使う頻度の高いもののみ書いておく。驚くほど少なかった。

関数	返り値	関数	返り値
k(2,i)	粒子 ID	p(1-3,i)	運動量の x-z 成分
k(7,i)	衝突回数		

3.3 反応計算の制御

用意されたスイッチを用いて、さまざまな物理過程計算を制御することができる。わかりやすいもので言えば、ある粒子の崩壊をストップさせたり、非弾性散乱のみ起こさせたり、ということができる。私はあまり理解が深くないので、頻度が高いものは以下の数種類しかない。

```
mstc(17)=0          ! 非弾性散乱のみ起こさせるスイッチが OFF になっている。
mstc(43)=1          ! 原子核中の核子のフェルミ運動を ON にしている。
parc(10)=0.25       ! s チャネル反応で生成したレゾナンスが崩壊するときの
                    ! u チャネル-t チャネルの混合比
kc=jamcomp(3312)    ! Xi-を
mdcy(kc,1)=0        ! 崩壊させない
```

N^* やハイペロンの励起状態などの崩壊を止めてやると、計算中でまずレゾナンスが生成されるのがわかる。たとえば Ξ は、まず Λ^* や Σ^* が生成され、それが崩壊してできるというように計算されているのがわかる。(直接の 2 体の反応が入っていない。)

他にも多数あるので、マニュアルを見つつ遊んでみれば良いと思う。

4 結果の利用法

あくまで私の個人的なやり方だが、JAM でほしい情報をファイルに書き出した後は、それを root ファイルに直してやり、そこから root で解析したり、Geant4 に食わせたりして使っている。ここでは、出力されたイベント数と断面積の関係や、散乱角についての注意をまとめておく。

4.1 断面積

あるビーム強度を与えたときのバックグラウンドのレートが知りたいといった場合にどうすれば良いか書いておく。まず必要な数字を書き出すと、

1. ビームの数（$N_{beam\mathrm{JAM}}$）
2. ビームの衝突係数の範囲（b_{min}〜b_{max}）
3. 注目する反応のイベント数（$N_{reaction\mathrm{JAM}}$）
4. 想定する標的原子核の数（N_{target}）

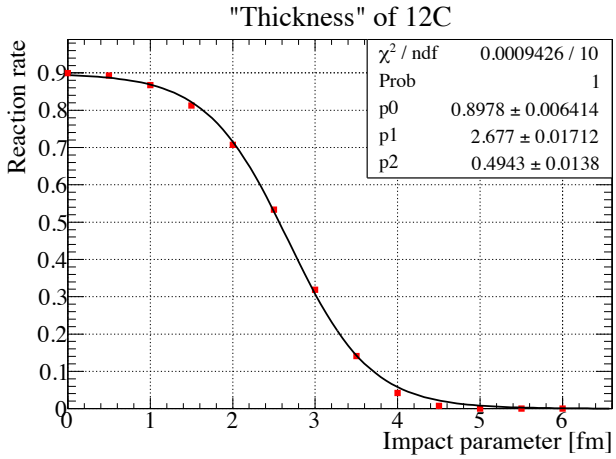

図 1: ^{12}C との反応確率と衝突係数。衝突係数を b=0.00–0.01, 0.50–0.51, 1.00–1.01,... として、10^6個 の K^- を 1.8 GeV/c で ^{12}C に投げたときの少なくとも 1 回以上何かしらの衝突が起こったイベントの割合。$b > 5$ fm ではほぼ反応が起こっていない。Woods-Saxon 型にフィットした分布になっている。

である。1,2 は JAM への入力値であり、3 は出力結果から自分で解析したエントリー数、4 はシミュレーションしたい標的の厚さから計算する。ビームを平面に一様に分布させたいので、$b_{min} = 0$ とし、$b_{max} = b$ は扱う原子核の大きさ以上の適切な値を用いる[7]。^{12}C の場合、5 fm あれば十分である（図 1）。まず、見たいイベントの（JAM 中の）断面積 $d\sigma$ に相当する量は以下の式で書ける。

$$\frac{d\sigma}{\text{ビームの面積}} = \frac{\text{注目する反応のイベント数}}{\text{ビームの数}} \tag{1}$$

$$d\sigma = \frac{N_{reaction\mathrm{JAM}}}{N_{beam\mathrm{JAM}}} \times \pi b^2[\mathrm{fm}^2] \tag{2}$$

この関係を使うと、ビーム数 $N_{beam\mathrm{JAM}}$、標的原子核数 N_{target} を想定したときの注目する反応のイベント数 $Yield$ は

$$Yield = N_{beam} \times N_{target}[/\mathrm{cm}^2] \times d\sigma[\mathrm{fm}^2] \tag{3}$$

$$= N_{reaction\mathrm{JAM}} \times \frac{N_{beam}}{N_{beam\mathrm{JAM}}} \times \frac{N_{target}[/\mathrm{cm}^2]}{1/\pi b^2[/\mathrm{fm}^2]} \tag{4}$$

というふうに計算できる。式（4）は、起きた反応に対して、ビームレートの比と、標的密度の比を掛けてやれば良いということを示している[8]。

自分がよく使うものとして CH_2 標的（9.6 g/cm^2）、C 標的（3 g/cm^2）の場合について具体的な数値を入れて計算しておく。b は 5 fm とした。CH_2 の場合、^{12}C が 8.23 g、H が 1.37 g あるので、これに応じた標的数を計算する。

$$\text{Const.} = \frac{N_{target}[/\mathrm{cm}^2]}{1/\pi b^2[/\mathrm{fm}^2]} = \frac{M[\mathrm{g/cm}^2] \times 6.02 \times 10^{23}}{A[\mathrm{g}]} \times \pi b^2[\mathrm{fm}^2] \tag{5}$$

$$= \begin{cases} 0.1182 & (^{12}\mathrm{C}\ 3\ \mathrm{g/cm}^2, b = 5) \\ 0.3243 & (^{12}\mathrm{C}\ 8.23\ \mathrm{g/cm}^2, b = 5) \\ 0.02638 & (\mathrm{H}\ 1.37\ \mathrm{g/cm}^2, b = 1.009) \end{cases} \tag{6}$$

[7]平面上で一様にさせたいときには b_{max} を負にする。正の値だと 1 次元的に一様乱数になる。
[8]JAM の中では標的原子核が 1 個あるとしているので、密度は $1/5^2\pi[/\mathrm{fm}^2]$ になる。

ここで、p 標的に関しては、$b = 1.009$ fm となっている。これは、一粒子対一粒子のシミュレーションでは、衝突係数を自分でどのように与えていても、K^-p の全断面積 32 mb に対応した値に強制的に直されるという JAM の仕様によるものである。したがって、p 標的についての計算では全イベントで必ず何かしらの衝突が起こることになる。

まとめると、JAM で生成したイベントを飛ばし、最終的なカウント数にビームのファクター $N_{beam}/N_{beam\mathrm{JAM}}$ と標的密度の換算係数（^{12}C なら 0.3243、H なら 0.02638）を掛けてスケールすれば、自分の見たいビーム粒子数と標的数を使ったときのある反応の計数率を知ることができる。

4.2 角分布に関する注意

JAM の出力結果について気をつけるべきことのひとつとして、散乱粒子の角度分布がある。どうやらデフォルトではビームは xz 平面上に分布するらしく（つまりビーム軌道は $(x, y) = (b_{min}\text{–}b_{max}, 0)$ の直線になる）、その結果、散乱粒子の方位角 ϕ は非対称な分布をする[9]。よって、たとえば ϕ について一様な結果を得るためには、プログラム中でビーム入射位置分布を (x, y) で一様にするか、得られた結果を ϕ について乱数で振ってやる必要がある。

5 fortran のメモ

fortran はたまによく見かけるけどマジメに勉強したことがない。必要最低限の条件式とかだけ覚えて済ませてしまっている。この節を独立したメモにしようかと思ったが、JAM をやるとき以外に fortran を自分で書くことは多分ないので、ここに簡単に書いておく[10]。

5.1 変数の宣言と型の名前

fortran ではどうやら変数の宣言が必須ではなく、暗黙の型宣言という便利なのかやっかいなのかわからないものがある[11]。基本的に最初に宣言しておく。

整数型	integer
実数型	real
倍精度実数型	double precision
複素数型	complex
倍精度複素数型	complex(kind(0d0))
論理型	logical
文字型	character

5.2 条件式

簡単な if 文の書き方は以下の通り。処理が 1 文のときは「if (条件) 処理」でも良い。

[9] ビーム入射位置分布設定に依存するものであり、物理的な微分断面積とは別の、シミュレーション上の問題である。

[10] 他に fortran プログラムを見かける機会と言えば、輸送行列を計算する Orbit くらいだろうか。transport や turtle も fortran のようだが、自分が文法を知っている必要はない。

[11] 個人的には好きではない（このせいでバグを生んだから）。

```
if (条件 1) then
  条件 1 が真の場合の処理
else if (条件 2) then
  条件 1 が偽で条件 2 が真の場合の処理
 ...
else if (条件 n) then
  条件 1 から条件 n-1 が偽で条件 n が真の場合の処理
else
  条件 1 から条件 n がすべて偽の場合の処理
end if
```

論理条件の書き方は以下のようなものがある。

比較演算子	例	論理演算子	例
.lt. または <	a.lt.b または a<b	.not.	.not. a==b
.le. または <=	a.le.10 または a<=10	.and.	a==b .and. b==c
.eq. または ==	a.eq.100 または a==100	.or.	a==b .or. b==c
.ne. または ¯	a.ne.0 または a$\bar{0}$	.neqv.	a<0 .neqv. b<0 (a と b の符号が異なる)
.gt. または >	a.gt.30 または a>30	.eqv.	a<0 .eqv. b<0 (a と b の符号が同じ)
.ge. または >=	a.ge.30 または a>=30		

5.3 ファイル入出力

よく理解していないが、fortran では write、read という関数を使って入出力を行う。この関数への第一引数は装置番号であり、どこへ write するのかという情報を与える。6 番は予約されており、標準出力を意味する。たとえば write(6,*) ”hogehoge” とすると、コマンドラインに hogehoge と表示される。ファイルに書き出したい場合は、まず出力用のファイルを開き、そのときに宣言した装置番号に対して write すれば良い。

```
open(46, file='foo.dat', status='replace')
write(46,*) "hoge1",23
write(6,*) "hoge4",56
```

こうすると、foo.dat というファイルと標準出力に、それぞれ

```
hoge1 23
hoge4 56
```

と出力される。

このような書き方だと、ascii で書き出される。よって、目視確認用に書き出す、またはイベント数が小さい場合はこれでも良いが、書き出すデータ量が大きい場合には、出力ファイルサイズが大きくて扱いづらくなるので、バイナリ出力をする方が良い。