ジャイロ運動論的シミュレーションコードGKVを用いた微視的不安定性・乱流輸送解析

前山伸也

名大理

GKV講習会
2016年12月16日

Contents

以下、**ユーザに編集してもらう点は青字**、**注意点は赤字**で示します。

実習のための下準備

始めに /tmp/GKV_tutorial/gkvp_f0.48.tar.gz をホームディレクトリにコピー。

```
cp –r /tmp/GKV_tutorial/ ./
```

ソースコード・平衡データを展開して、サンプル平衡データを所定の場所にコピー。

```
cd GKV_tutorial/
tar xzvf gkvp_f0.48.tar.gz
tar xzvf equilibrium_sample.tar.gz
cp –r equilibrium_sample/igs_sample_nss21ntheta33/ gkvp_f0.48/run/input_eqdsk/
```

プラズマシミュレータでFFTWを有効化する。

```
module load fftw-fx
```

プラズマシミュレータでgnuplot5.0を有効化するために、**~/.cshrc の末尾にエイリアスを追加**。

```
alias gnuplot /usr/local/gnuplot/5.0.1/bin/gnuplot
```

※ユーザのシェル環境に依存すると思うので適宜対応。

GKVのコード構造(バージョンgkvp_f0.48)

gkvp_f0.48/
- README_for_namelist.txt　簡単な説明書き
- Version_memo.txt　最近の更新履歴
- src/　ソースファイル群
  - gkvp_f0.48_header.f90　解像度、MPIの設定モジュール
  - gkvp_f0.48_out.f90　　標準データ出力モジュール
- lib/　乱数・ベッセル関数ライブラリ呼び出しモジュール
- extra_tools/　ポスト処理ツール等
  - fig_stdout.tar.gz　アスキーデータのPDF化
  - v29diag.tar.gz　　バイナリデータの解析
- run/　コンパイルおよび計算実行
  - gkvp_f0.48_namelist　物理パラメータの設定
  - sub.q　バッチジョブ用スクリプト(計算機依存)
  - shoot　ジョブ投入スクリプト(計算機依存)
  - Makefile　コンパイル情報(計算機依存)
  - backup/　各計算機向けsub.q,shoot,Makefileのバックアップ
  - input_vmec/　VMEC平衡サンプル
  - input_eqdsk/　EQDSK平衡サンプル

GKVで扱える問題

ある平衡の下で、微視的不安定性や乱流揺動、粒子・熱輸送の局所解析。

1. 線形解析

- 線形モードの成長率、実周波数、揺動間のクロスフェーズなどを調べる。
- 線形モードの独立性から、特定の波数のみ解析するため計算は早い。
- 非線形飽和機構が入っていないので、振幅の絶対値は求まらない。

2. 非線形解析

- 揺動スペクトル、粒子・熱輸送、その他諸々の乱流揺動解析を行う。
- 乱流混合を扱うために多数のモード間の非線形結合を解く必要があり、計算に時間がかかる。要求解像度も問題に依るので、数値的健全性確保のためにエントロピーバランスやスペクトルの収束性確認が必要。

今日の実習の問題設定

実験の密度・温度・磁場計測などから、解析したいプラズマの平衡分布が求まっている(EQDSK形式)。この平衡下で微視的不安定性が存在するか線形解析したい。

0. 解析する半径位置を決めて、実験的な物理パラメータをGKVの入力パラメータに換算し、run/gkvp_f0.48_namelistに入力する。また、GKVで読み込めるように加工したMHD平衡データを用意する。
1. src/gkvp_f0.48_header.f90に計算格子数およびMPI並列数を入力する。
2. バッチジョブスクリプトsub.qを設定する。
3. ジョブ投入スクリプトshootにディレクトリの設定をする。
4. コンパイルし、計算を実行。
5. 出力データを解析する。(ポスト処理ツールの利用)

※非線形解析も計算タイプ”nonlinear”とし高解像度化する位で同様の手順。

0. 実験→GKV換算、namelistへの入力、平衡の加工

実験計測より、
- 局所パラメータを算出
- MHD平衡を構築

GKVで解析するため、
- 実験→GKVパラメータ換算（規格化）
- 対応する物理パラメータのnamelistへの入力
- MHD平衡からGKVで必要となるメトリックデータへの加工

先の仲田さんの説明でできるようになったはずなので、今日の実習では既に用意済み。
run/gkvp_f0.48_namelist　　物理パラメータのnamelist
run/input_eqdsk/METRIC_axi.OUT　　EQDSKから構築したメトリックデータ

0. 実験→GKV換算、namelistへの入力、平衡の加工

run/gkvp_f0.48_namelist

念のため計算実行にかかわる部分だけおさらいすると、

```
 &calct calc_type="linear",        計算タイプ linear / nonlinear
       z_bound="outflow",
       z_filt="off",
       z_calc="up5",
       art_diff=0.d0,
       num_triad_diag=0, &end
 &triad mxt = 0, myt = 0/
 &equib equib_type = "eqdsk", &end    平衡磁場モデル analytic / s-alpha /  vmec / eqdsk
…
&runlm e_limit = 60.d0, &end        計算実行の実時間[秒]
&times  tend = 200.d0,               シミュレーション上の上限時間
        dtout_fxv = 1.d0,                データ出力の時間間隔1
        dtout_ptn = 0.1d0,               データ出力の時間間隔2
        dtout_eng = 0.1d0,               データ出力の時間間隔3
        dtout_dtc = 0.01d0, &end         自動時間刻み幅の調整間隔
…
```

0. 実験→GKV換算、namelistへの入力、平衡の加工

run/gkvp_f0.48_namelist

```
…
&nperi  n_tht = 4,          磁力線方向ボックスサイズ（ポロイダル角で±n_tht*pi）
        kymin = 0.1d0,      磁力線ラベル方向ボックスサイズ ly = pi / kymin
        m_j   = 1,          半径方向ボックスサイズ lx = pi / kxmin
        del_c = 0.d0, &end                    kxmin = |2*pi*s_hat*kymin/m_j|
…
&igsp   s_input = 0.50,
        mc_type = 0,
        q_type = 1,
        nss = 21,                ※ nss = NPSI (in IGS)
        ntheta = 33,     &end    ※ ntheta = NCHI (in IGS) + 1 = 2*global_nz/n_tht +1
…
```

1.src/gkvp_f0.48_header.f90に計算格子数およびMPI並列数を入力する。

src/gkvp_f0.48_header.f90

```
!-----------------------------------
!  Dimension size (grid numbers)
!-----------------------------------
!  Global simulation domain
!  in  x, y,z,v,m (0:2*nxw-1,  0:2*nyw-1,-global_nz:global_nz-1,1:2*global_nv,0:global_nm)
!  in kx,ky,z,v,m (   -nx:nx,0:global_ny,-global_nz:global_nz-1,1:2*global_nv,0:global_nm)

  integer, parameter :: nxw = 2, nyw = 8
  integer, parameter :: nx = 0, global_ny = 5       ! 2/3 de-aliasing rule
  integer, parameter :: global_nz = 64, global_nv = 24, global_nm = 15

  integer, parameter :: nzb = 3, &  ! the number of ghost grids in z
                        nvb = 3     ! the number of ghost grids in v and m

!-----------------------------------
!  Data distribution for MPI
!-----------------------------------
  integer, parameter :: nprocw = 1, nprocz = 4, nprocv = 2, nprocm = 2, nprocs = 2
```

1.src/gkvp_f0.48_header.f90に計算格子数およびMPI並列数を入力する。

ここで、

nx | kxモード数– nx:nx
global_ny | kyモード数0:global_ny
（さらにnxw>nx*3/2, nyw>global_ny*3/2となるように設定。）
global_nz | 磁力線方向座標 –n_tht*pi < zz < n_tht*pi を –global_nz:global_nz-1で離散化
global_nv | 磁力線方向速度 -vmax<vl<vmaxを1:2*global_nvで離散化
global_nm | 磁気モーメント 0<mu<vmax^2/2を0:global_nmで離散化

nprocw, nprocz, nprocv, nprocm, nprocs はky,zz,vl,mu方向および粒子種のMPI領域分割数。

ただし、
※(global_ny+1)/nprocw, global_nz/nprocz,
global_nv/nprocv, (global_nm+1)/nprocmは整数。
※nprocsは扱う粒子種数と一致。

2.バッチジョブスクリプトsub.qを設定する。

run/sub.qでMPI/OpenMP並列数を指定する。

run/sub.q

```
#PJM -L "rscunit=fx"
#PJM -L "rscgrp=X24"
#PJM -L "node=8"
#PJM -L "elapse=00:20:00"
#PJM -j
#PJM --mpi “proc=32“      ※総MPIプロセス数はnprocw*nprocz*nprocv*nprocm*nprocs
#### --mpi "rank-map-hostfile=myrankmap"
#PJM -g 16391

## Note that Max. core num. per 1 node on PS is 32.

setenv  PARALLEL  8                  # Thread number for automatic parallelization
setenv  OMP_NUM_THREADS  8   # Thread number for Open MP
                ※プラズマシミュレータではproc*OMP_NUM_THREADS = node * 32 とする。
以下は不変。
```

3.ジョブ投入スクリプトshootにディレクトリの設定をする。

run/shoot

```
#### Environment setting
set DIR=/data/lng/maeyama/gkv_training/    実行後のデータが出力されるディレクトリ
set LDM=gkvp_mpifft.exe                ※maeyama→個別のユーザ名に書き換えてください。
set NL=gkvp_f0.48_namelist
set SC=pjsub
set JS=sub.q
## For VMEC, set VMCDIR including metric_boozer.bin.dat
#set VMCDIR=./input_vmec
## For IGS, set IGSDIR including METRIC_{axi,boz,ham}.OUT
set IGSDIR=./input_eqdsk
                        IGSで生成したMETRIC_{axi,boz,ham}.OUTが置いてあるディレクトリ
```

以下は不変。

4. コンパイルし、計算を実行

コンパイルする。

```
cd  run/
module  purge
module  load  tcsuite-fx  fftw-fx
make  clean
make
```

計算を実行する。以下の形式でステップジョブ実行される。

```
./shoot   START_NUM   END_NUM   (JOB_ID)
```

例）シングルジョブ投入（*.001）　　./shoot 1 1
　　シングルジョブ投入（*.002）　　./shoot 2 2
　ステップジョブ投入（*.003-*.005）./shoot 3 5
　継続ステップジョブ投入　　./shoot 6 7 11223
　（*.005まで計算するジョブJOB_ID=11223がキュー中にあり、
　　それに続けて*.006-*.007のジョブを実行させようとした。）

4. コンパイルし、計算を実行

正常に計算が実行されれば、run/shootで設定した出力ディレクトリ
（今回の例では /data/lng/maeyama/gkv_training/）
に以下のデータが書き出される。

log/ 計算ログ
cnt/ 継続計算用バイナリデータ
fxv/ 分布関数バイナリデータ(いくつかの磁力線方向座標位置で)
phi/ ポテンシャル、流体モーメント、エントロピーバランスに関するバイナリデータ
hst/ アスキー形式の標準出力
その他:実行環境バックアップのためのコピー

Appendix A. GKVの出力データー覧 にまとめた。
さらに詳細は、ソースコード src/gkvp_f0.48_out.f90 を参照。

5. 出力データを解析する。(ポスト処理ツールの利用)

出力データを解析するには、

5-a) 自力で何とかする。
- GKVの出力データは一覧にまとめてあるので、後は適当にポスト処理する。
- アスキー形式の標準出力くらいなら簡単。
- MPI領域分割されたバイナリデータを読み込むのは結構手間。

コードのオープン化にあたり、ポスト処理ツールとして以下の2つを提供。

5-b) hst/のアスキー標準出力を一括でPDF化するためのスクリプト fig_stdout

5-c) phi/などのバイナリデータを解析するためのポスト処理プログラム diag

5-b) hst/のアスキー標準出力を一括でPDF化するためのスクリプト fig_stdout

extra_tools/fig_stdout.tar.gz をGKV出力データのあるディレクトリ（今回は /data/lng/maeyama/gkv_training/）に展開すると、

fig_stdout/
- make_pdf.csh　PDF作成シェルスクリプト
- pdf/　PDFが格納されるディレクトリ
- eps/　PDF作成に用いられたepsが格納されるディレクトリ
- data/　eps作成に用いられた元データが格納されるディレクトリ
- src/　gnuplot用スクリプト等

ができるので、以下のコマンド

```
cd  fig_stdout/
./make_pdf.csh  clean
./make_pdf.csh
```

を実行すると、一覧のPDF（fig_stdout/pdf/fig.pdf）やeps、元データが格納される。
（※必ずしも図のスケール等が見やすいとは限らない。）

5-c) phi/などのバイナリデータを解析するためのポスト処理プログラム diag

extra_tools/v29diag.tar.gz を適当な場所に展開すると、

v29diag/

- Makefile
- go.diag_ps_fx バッチジョブ用スクリプト
- backup/
- plotfile/ gnuplot用スクリプトのサンプル
- src/ ソースコード
  - diag_header.f90 解像度等の設定
  - diag_main.f90 解析モジュールの呼び出し
  - diag_rb.f90 GKV出力バイナリデータ読み込みモジュール
  - diag_********.f90 その他各種設定
  - …
  - out_********.f90 特定の図を描くための専用モジュール群
  - …

5-c) phi/などのバイナリデータを解析するためのポスト処理プログラム diag

diagで何ができるのか。

- diag_rbモジュールを利用することで、MPI分割されたGKVの出力データから、「解析したい時間ステップの解析したい物理量」を読み込める。(Appendix Bも参照)
- 読み込まれた変数は、領域分割されていない変数として再構成されるので、GKVのMPI分割に習熟していないユーザでも比較的簡便に扱える。
- 何か解析したいと思ったときは、その解析を行うためのモジュールout_********.f90を作成し、変数などは外から見えないようにする。主プログラムdiag_main.f90はモジュールを呼び出すだけなので、複数人が独立にモジュール作成しても競合しない。

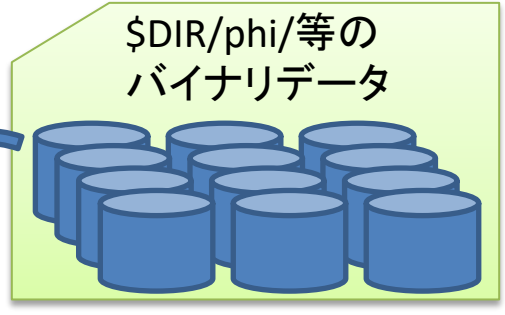
```
PROGRAM diag_main
  use out_sample
   …

  call out_sample_sub
   …

END PROGRAM diag_main
```

Call

```
MODULE out_sample
  use diag_rb
  private
  public :: out_sample_sub
 CONTAINS
SUBROUTINE out_sample_sub
   …
END SUBROUTINE out_sample_sub
END MODULE out_sample
```



$DIR/post/dataに解析結果を出力する。

5-c) phi/などのバイナリデータを解析するためのポスト処理プログラム diag

【diagの使い方】

1. v29diag/src/diag_header.f90で、!%%% DIAG parameters %%% と !%%% GKV parameters %%% 内のパラメータを設定する。
2. src/diag_main.f90で解析したいモジュールを呼び出す。
3. go.diagで、解析したいデータのあるディレクトリを DIR に設定する。
4. コンパイル。（make）
5. 実行。（pjsub go.diag_ps_fx）
6. ${DIR}/post 下にポスト処理されたデータが書き出される。

解析例①　あるモードk の静電ポテンシャルの磁力線方向z分布
解析例②　静電ポテンシャルのx,y平面分布(悪い曲率領域 z=0 での断面図)

まとめ

GKVの利用方法をハンズオン形式で説明した。

<u>要約</u>

GKVの物理モデル・数値モデル → 渡邉資料

を踏まえた上で、

実験→GKVパラメータ換算

MHD平衡データの加工 → 仲田資料

の準備をしてから、

src/gkvp_f0.48_header.f90	解像度、MPIの設定
run/gkvp_f0.48_namelist	物理・数値パラメータの設定
run/sub.q	MPI・OpenMPの設定
run/shoot	平衡データ・出力ディレクトリの設定

の後にコンパイル、実行。

hst/のアスキー標準出力を一括でPDF化するためのスクリプト fig_stdout

phi/などのバイナリデータを解析するためのポスト処理プログラム diag

などを利用して、結果を解析する。

Appendix A. GKVの出力データー覧

cnt/*cnt*
fxv/*fxv*
phi/*phi*, *Al*, *mom*, *trn*, (非線形の場合のみ *tri*)
hst/*bln*, *ges*, *gem*, *qes*, *qem*, *wes*, *wem*,
eng, *men*, *dtc*, *mtr*, (線形の場合のみ *frq*, *dsp*)
log/*log*

cnt/gkvp_f0.47.(MPIランク6桁).cnt.(ラン数3桁)

- ファイル形式： バイナリ
- 出力間隔： ランの終了時
- 出力を行うMPIランク： すべて
- 総ファイル数： nprocw*nprocz*nprocv*nprocm*nprocs * (総ラン数)
- GKVコード中の出力ユニット： ocnt
- 格納データ:
  time, ff(-nx:nx,0:ny,-nz:nz-1,1:2*nv,0:nm)

ここで、

time: 時刻（倍精度実数）

ff: 揺動ジャイロ中心分布関数（倍精度複素数）

fxv/gkvp_f0.47.(MPIランク6桁).(粒子種1桁).fxv.(ラン数3桁)

- ファイル形式： バイナリ
- 出力間隔： dtout_fxv
- 出力を行うMPIランク： すべて
- 総ファイル数： nprocw*nprocz*nprocv*nprocm*nprocs * (総ラン数)
- GKVコード中の出力ユニット： ofxv
- 格納データ:
  time, ff(-nx:nx,0:ny,1:2*nv,0:nm)

ここで、

time: 時刻（倍精度実数）
ff: 揺動ジャイロ中心分布関数（倍精度複素数）at iz=-nz（z方向MPIランクrankzに依存して、書き出す磁力線方向位置は異なる。）

phi/gkvp_f0.47.(MPIランク6桁).0.phi.(ラン数3桁)

- ファイル形式： バイナリ
- 出力間隔： dtout_ptn
- 出力を行うMPIランク： ranks == 0 .and. vel_rank == 0
- 総ファイル数： nprocw*nprocz * (総ラン数)
- GKVコード中の出力ユニット： ophi
- 格納データ：
  time, phi(-nx:nx,0:ny,-nz:nz-1)

ここで、

time: 時刻（倍精度実数）

phi: 揺動静電ポテンシャル（倍精度複素数）

phi/gkvp_f0.47.(MPIランク6桁).0.Al.(ラン数3桁)

- ファイル形式： バイナリ
- 出力間隔： dtout_ptn
- 出力を行うMPIランク： ranks == 0 .and. vel_rank == 0
- 総ファイル数： nprocw*nprocz * (総ラン数)
- GKVコード中の出力ユニット： oAl
- 格納データ：
  time, Al(-nx:nx,0:ny,-nz:nz-1)

ここで、

time: 時刻（倍精度実数）
Al: 揺動ベクトルポテンシャル（倍精度複素数）

phi/gkvp_f0.47.(MPIランク6桁).(粒子種1桁).mom.(ラン数3桁)

- ファイル形式： バイナリ
- 出力間隔： dtout_ptn
- 出力を行うMPIランク： vel_rank == 0
- 総ファイル数： nprocw*nprocz*nprocs * (総ラン数)
- GKVコード中の出力ユニット： omom
- 格納データ：
  time, mom(-nx:nx,0:ny,-nz:nz-1,0:nmom-1)

ここで、

time: 時刻（倍精度実数）

mom: 揺動流体モーメント（倍精度複素数）

現状、nmom＝6として以下の6つの流体量を順に出力。

$$\tilde{n}_{sk} = \int dv^3 J_{0sk} \tilde{f}_{sk}, \qquad \tilde{u}_{\parallel sk} = \int dv^3 v_\parallel J_{0sk} \tilde{f}_{sk}, \qquad \tilde{p}_{\parallel sk} = \int dv^3 \frac{v_\parallel^2}{2} J_{0sk} \tilde{f}_{sk},$$

$$\tilde{p}_{\perp sk} = \int dv^3 \mu B J_{0sk} \tilde{f}_{sk}, \tilde{q}_{\parallel\parallel sk} = \int dv^3 v_\parallel \frac{v_\parallel^2}{2} J_{0sk} \tilde{f}_{sk}, \tilde{q}_{\parallel\perp sk} = \int dv^3 v_\parallel \mu B J_{0sk} \tilde{f}_{sk}$$

phi/gkvp_f0.47.(MPIランク6桁).(粒子種1桁).trn.(ラン数3桁)

- ファイル形式： バイナリ
- 出力間隔： dtout_eng
- 出力を行うMPIランク： zsp_rank == 0 .and. vel_rank == 0
- 総ファイル数： nprocw*nprocs * (総ラン数)
- GKVコード中の出力ユニット： otrn
- 格納データ:

  time, $S_{sk}, W_{Ek}, W_{Mk}, R_{sEk}, R_{sMk}, I_{sEk}, I_{sMk}, D_{sk}, \Gamma_{sEk}, \Gamma_{sMk}, Q_{sEk}, Q_{sMk}$

ここで、

time: 時刻（倍精度実数）
他はすべてサイズ(-nx:nx,0:ny)の倍精度実数配列で、左から順に、ジャイロ中心揺動エントロピー、静電揺動エネルギー（イオン分極項含む）、磁場揺動エネルギー、波粒子相互作用（$W_E \rightarrow S_s$）、波粒子相互作用（$W_M \rightarrow S_s$）、ExB流による非線形エントロピー伝達、磁場揺動による非線形エントロピー伝達、衝突散逸、ExB流による粒子輸送フラックス、磁場揺動による粒子輸送フラックス、ExB流によるエネルギー輸送フラックス、磁場揺動によるエネルギー輸送フラックス

phi/gkvp_f0.47.s(粒子種1桁)mx(mxt4桁)my(myt4桁).tri.(ラン数3桁)

※mxt,mytはnamelistで指定したもの。

- ファイル形式： バイナリ
- 出力間隔： dtout_ptn（calc_type==“nonlinear” .and. num_triad_diag>0）
- 出力を行うMPIランク： rank == 0
- 総ファイル数： nprocs * num_triad_diag * (総ラン数)
- GKVコード中の出力ユニット： otri
- 格納データ:

  time, $J_{sE\boldsymbol{k}}^{\boldsymbol{p},\boldsymbol{q}}$, $J_{sE\boldsymbol{p}}^{\boldsymbol{q},\boldsymbol{k}}$, $J_{sE\boldsymbol{q}}^{\boldsymbol{k},\boldsymbol{p}}$, $J_{sM\boldsymbol{k}}^{\boldsymbol{p},\boldsymbol{q}}$, $J_{sM\boldsymbol{p}}^{\boldsymbol{q},\boldsymbol{k}}$, $J_{sM\boldsymbol{q}}^{\boldsymbol{k},\boldsymbol{p}}$

ここで、

time: 時刻（倍精度実数）
他はすべてサイズ(-nx:nx,-global_ny:global_ny)の倍精度実数配列で、モードk=(mxt,myt)に固定して、p=(px,py)の関数として表したもの（qは-k-pで求まる）。先の3つは、ExB流の非線形性によるp,qからkへのエントロピー伝達とそのcyclicな入れ替え、後の3つは、磁場揺動の非線形性によるp,qからkへのエントロピー伝達とそのcyclicな入れ替え。

hst/gkvp_f0.47.bln.(粒子種1桁).(ラン数3桁)

- ファイル形式： アスキー
- 出力間隔： dtout_eng
- 出力を行うMPIランク： rank == 0
- 総ファイル数： nprocs * (総ラン数)
- GKVコード中の出力ユニット： obln
- 格納データ:

$$\mathrm{time}, S_s, W_E, W_M, R_{sE}, R_{sM}, I_{sE}, I_{sM}, D_s, \frac{T_s \Gamma_{sE}}{L_{ps}}, \frac{T_s \Gamma_{sM}}{L_{ps}}, \frac{\Theta_{sE}}{L_{Ts}}, \frac{\Theta_{sM}}{L_{Ts}}$$

ここで、

time: 時刻（実数）

S_sからD_sまではサイズ(2)の実数配列（配列要素1,2はそれぞれky/=0成分とky==0成分）で、左から順に、ジャイロ中心揺動エントロピー、静電揺動エネルギー（イオン分極項含む）、磁場揺動エネルギー、波粒子相互作用（$W_E \rightarrow S_s$）、波粒子相互作用（$W_M \rightarrow S_s$）、ExB流による非線形エントロピー伝達、磁場揺動による非線形エントロピー伝達、衝突散逸。残り4つは実数で、エントロピーバランス方程式における、粒子輸送項（ExB流、磁場揺動）、エネルギー輸送項（ExB流、磁場揺動）

hst/gkvp_f0.47.ges.(粒子種1桁).(ラン数3桁)

- ファイル形式： アスキー
- 出力間隔： dtout_eng
- 出力を行うMPIランク： rank == 0
- 総ファイル数： nprocs * (総ラン数)
- GKVコード中の出力ユニット： oges
- 格納データ:
  time, Γ_{sE}, Γ_{sEk_y}(0:global_ny)

ここで、

time: 時刻（実数）
Γ_{sE}: ExB流による粒子輸送フラックス（実数）
Γ_{sEk_y}: ExB流による粒子輸送フラックスのy方向波数スペクトル（実数配列）

hst/gkvp_f0.47.gem.(粒子種1桁).(ラン数3桁)

- ファイル形式： アスキー
- 出力間隔： dtout_eng
- 出力を行うMPIランク： rank == 0
- 総ファイル数： nprocs * (総ラン数)
- GKVコード中の出力ユニット： ogem
- 格納データ:
  time, Γ_{sM}, Γ_{sMk_y}(0:global_ny)

ここで、

time: 時刻（実数）
Γ_{sM}: 磁場揺動による粒子輸送フラックス（実数）
Γ_{sMk_y}: 磁場揺動による粒子輸送フラックスのy方向波数スペクトル（実数配列）

hst/gkvp_f0.47.qes.(粒子種1桁).(ラン数3桁)

- ファイル形式： アスキー
- 出力間隔： dtout_eng
- 出力を行うMPIランク： rank == 0
- 総ファイル数： nprocs * (総ラン数)
- GKVコード中の出力ユニット： oqes
- 格納データ:
  time, Q_{sE}, Q_{sEk_y}(0:global_ny)

ここで、

time: 時刻（実数）
Q_{sE}: ExB流によるエネルギー輸送フラックス（実数）
Q_{sEk_y}: ExB流によるエネルギー輸送フラックスのy方向波数スペクトル（実数配列）

hst/gkvp_f0.47.qem.(粒子種1桁).(ラン数3桁)

- ファイル形式： アスキー
- 出力間隔： dtout_eng
- 出力を行うMPIランク： rank == 0
- 総ファイル数： nprocs * (総ラン数)
- GKVコード中の出力ユニット： oqem
- 格納データ:
  time, Q_{sM}, Q_{sMk_y}(0:global_ny)

ここで、

time: 時刻（実数）
Q_{sM}: 磁場揺動によるエネルギー輸送フラックス（実数）
Q_{sMk_y}: 磁場揺動によるエネルギー輸送フラックスのy方向波数スペクトル（実数配列）

hst/gkvp_f0.47.wes.(ラン数3桁)

- ファイル形式： アスキー
- 出力間隔： dtout_eng
- 出力を行うMPIランク： rankg == 0
- 総ファイル数： (総ラン数)
- GKVコード中の出力ユニット： owes
- 格納データ:
  time, W_E, W_{Ek_y}(0:global_ny)

ここで、

time: 時刻（実数）
W_E： 静電揺動エネルギー（実数）
W_{Ek_y}： 静電揺動エネルギーのy方向波数スペクトル（実数配列）

hst/gkvp_f0.47.wem.(ラン数3桁)

- ファイル形式： アスキー
- 出力間隔： dtout_eng
- 出力を行うMPIランク： rankg == 0
- 総ファイル数： (総ラン数)
- GKVコード中の出力ユニット： owem
- 格納データ:
  time,W_M, W_{Mk_y}(0:global_ny)

ここで、

time: 時刻(実数)
W_M: 磁場揺動エネルギー(実数)
W_{Mk_y}: 磁場揺動エネルギーのy方向波数スペクトル(実数配列)

hst/gkvp_f0.47.eng.(ラン数3桁)

- ファイル形式： アスキー
- 出力間隔： dtout_eng
- 出力を行うMPIランク： rankg == 0
- 総ファイル数： (総ラン数)
- GKVコード中の出力ユニット： oeng
- 格納データ:

  time,$\sum_{k_x,k_y}\langle|\tilde{\phi}_{\boldsymbol{k}}|^2\rangle$, $\sum_{k_x}\langle|\tilde{\phi}_{\boldsymbol{k}}|^2\rangle$(0:global_ny)

ここで、

time: 時刻（実数）

$\sum_{k_x,k_y}\langle|\tilde{\phi}_{\boldsymbol{k}}|^2\rangle$: 揺動静電ポテンシャル二乗振幅（実数）

$\sum_{k_x}\langle|\tilde{\phi}_{\boldsymbol{k}}|^2\rangle$: 揺動静電ポテンシャル二乗振幅のy方向波数スペクトル（実数配列）

hst/gkvp_f0.47.men.(ラン数3桁)

- ファイル形式： アスキー
- 出力間隔： dtout_eng
- 出力を行うMPIランク： rankg == 0
- 総ファイル数： (総ラン数)
- GKVコード中の出力ユニット： omen
- 格納データ:

  time,$\sum_{k_x,k_y}\langle|\tilde{A}_{\parallel \boldsymbol{k}}|^2\rangle$, $\sum_{k_x}\langle|\tilde{A}_{\parallel \boldsymbol{k}}|^2\rangle$(0:global_ny)

ここで、

time: 時刻（実数）

$\sum_{k_x,k_y}\langle|\tilde{A}_{\parallel \boldsymbol{k}}|^2\rangle$: 揺動ベクトルポテンシャル二乗振幅（実数）

$\sum_{k_x}\langle|\tilde{A}_{\parallel \boldsymbol{k}}|^2\rangle$: 揺動ベクトルポテンシャル二乗振幅のy方向波数スペクトル（実数配列）

hst/gkvp_f0.47.dtc.(ラン数3桁)

- ファイル形式： アスキー
- 出力間隔： dtout_eng
- 出力を行うMPIランク： rankg == 0
- 総ファイル数： (総ラン数)
- GKVコード中の出力ユニット： odtc
- 格納データ:
  time, dt, dt_limit, dt_nl

ここで、

time: 時刻(実数)
dt: 時間刻み幅(実数)
dt_limit: 時間刻み幅の見積もり(実数)
dt_nl: 非線形移流速度から算出した数値安定な時間刻み幅の見積もり(実数)

hst/gkvp_f0.47.mtr.(ラン数3桁)

- ファイル形式： アスキー
- 出力間隔： ランの開始時
- 出力を行うMPIランク： rankg == 0
- 総ファイル数： (総ラン数)
- GKVコード中の出力ユニット： omtr
- 格納データ:

$$z, \theta(\text{または}\varphi), B, \frac{\partial B}{\partial x}, \frac{\partial B}{\partial y}, \frac{\partial B}{\partial z}, g^{xx}, g^{xy}, g^{xz}, g^{yy}, g^{yz}, g^{zz}, \sqrt{g}$$

ここで、データはすべて実数で、左から順に
磁力線方向座標、ポロイダル角（ただしequib_type==vmecの時はトロイダル角）、
磁場強度、磁場強度の微分3つ、メトリックテンソルの要素6つ、Jacobian。

hst/gkvp_f0.47.frq.(ラン数3桁)

- ファイル形式： アスキー
- 出力間隔： dtout_eng (calc_type == linear .or. calc_type == lin_freq)
- 出力を行うMPIランク： rankg == 0
- 総ファイル数： (総ラン数)
- GKVコード中の出力ユニット： ofrq
- 格納データ:
  time, omega(1:global_ny)

ここで、

time: 時刻（実数）

omega: 線形複素周波数（複素数）[＝（実周波数，成長率）]のy方向波数スペクトル

hst/gkvp_f0.47.dsp.(ラン数3桁)

- ファイル形式： アスキー
- 出力間隔： ランの終了時 (calc_type == linear .or. calc_type == lin_freq)
- 出力を行うMPIランク： rankg == 0
- 総ファイル数： (総ラン数)
- GKVコード中の出力ユニット： odsp
- 格納データ:
  ky, omega, diff, 1-ineq

ここで、

ky: y方向波数(実数)
omega: 線形複素周波数(複素数)[＝(実周波数，成長率)]
diff: 収束の相対誤差[omega(t)-omega(t-dt)]/omega(t) (複素数)
1-ineq: Schwartzの不等式で評価した収束誤差(実数)

log/gkvp_f0.47.(MPIランク6桁).(粒子種1桁).log.(ラン数3桁)

- ファイル形式： アスキー
- 出力間隔： 随時
- 出力を行うMPIランク： すべて
- 総ファイル数： nprocw*nprocz*nprocv*nprocm*nprocs * (総ラン数)
- GKVコード中の出力ユニット： olog
- 格納データ：
  シミュレーションに関するログ

Appendix B. ポスト処理プログラムdiagにおけるバイナリデータの読み込み

diag_rbモジュールを利用してバイナリデータを読み込む

ポスト処理プログラムdiag中でGKV出力バイナリデータを読み込むには、diag_rbモジュールで用意されているサブルーチンをcallする。

例）use diag_rb, only : rb_phi_loop
complex(kind=DP) :: phi(-nx:nx,0:global_ny,-global_nz:global_nz-1)
integer :: loop = 100
call rb_phi_loop(loop, phi)
※出力レコード loop = 100 番目の静電ポテンシャルphiを読み込む。
※phiの場合は time = dtout_ptn * loop の時刻のデータに相当。

以下、diag_rbの構造と、主要なサブルーチンについて一覧を示す。

v29diag/src/diag_rb.f90の構造

```
MODULE diag_rb
  public
  integer, dimension(1:enum) :: loop_phi_sta, loop_phi_end

 CONTAINS

SUBROUTINE  rb_phi_gettime( loop, time )
   …
END SUBROUTINE rb_phi_gettime

SUBROUTINE  rb_phi_loop( loop, phi )
   …
END SUBROUTINE rb_phi_loop

…

END MODULE
```

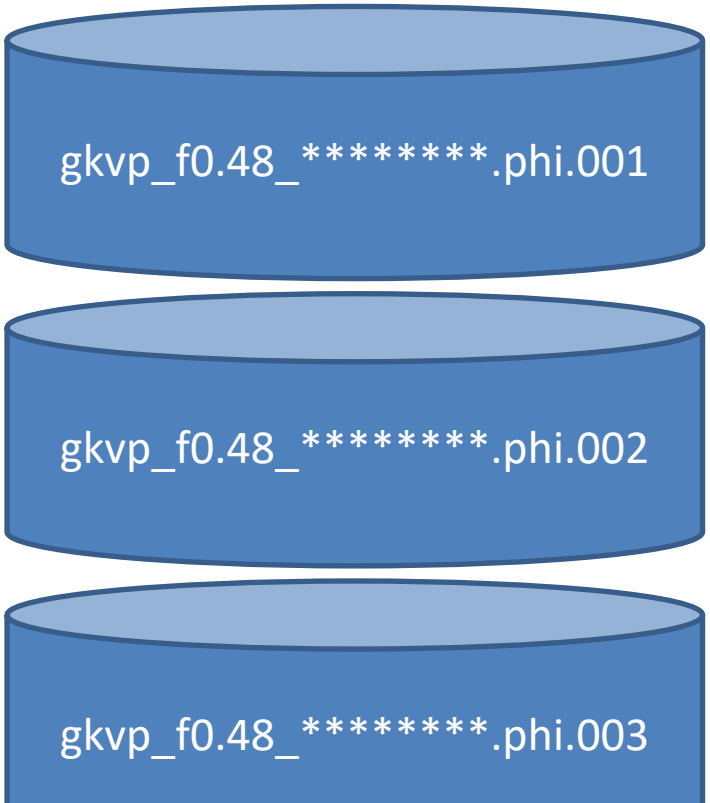
※phiのバイナリデータのレコード数を格納しておく変数

※レコード数に対応するtimeを読み込むサブルーチン

※レコード数に対応するphiを読み込むサブルーチン

バイナリデータのレコード数

integer, dimension(1:enum) :: loop_phi_sta, loop_phi_end



loop_phi_sta(001)
…
…
Loop_phi_end(001)

Loop_phi_sta(002)
…
…
Loop_phi_end(002)

Loop_phi_sta(003)
…
…

バイナリデータのサイズから各ランにおけるレコード数を算出できる。

全レコード数は
loop_phi_sta(001)から
loop_phi_end(enum)まで。

※ラン数snumからenumまでしか解析しないとしても、データはラン数001から置いておく必要あり。

rb_phi_gettime(loop, time)

【引数の型、入出力】
integer, intent(in) :: loop
real(kind=DP), intent(out) :: time

【役割】
静電ポテンシャルのバイナリデータ
　phi/gkvp_f0.47_(MPIランク6桁).0.phi.(ラン数3桁)
について、

レコード数 loop の時刻に対応する時刻を time に読み込む。

rb_Al_gettime(loop, time)

【引数の型、入出力】
integer, intent(in) :: loop
real(kind=DP), intent(out) :: time

【役割】
ベクトルポテンシャルのバイナリデータ
　phi/gkvp_f0.47_(MPIランク6桁).0.Al.(ラン数3桁)
について、

レコード数 loop の時刻に対応する時刻を time に読み込む。

rb_mom_gettime(loop, time)

【引数の型、入出力】
integer, intent(in) :: loop
real(kind=DP), intent(out) :: time

【役割】
流体モーメントのバイナリデータ
　phi/gkvp_f0.47_(MPIランク6桁).(粒子種1桁).mom.(ラン数3桁)
について、

レコード数 loop の時刻に対応する時刻を time に読み込む。

rb_trn_gettime(loop, time)

【引数の型、入出力】
integer, intent(in) :: loop
real(kind=DP), intent(out) :: time

【役割】
エントロピーバランスに関する諸量のバイナリデータ
　phi/gkvp_f0.47_(MPIランク6桁).(粒子種1桁).trn.(ラン数3桁)
について、

レコード数 loop の時刻に対応する時刻を time に読み込む。

rb_phi_loop(loop, phi)

【引数の型、入出力】
integer, intent(in) :: loop
complex(kind=DP), dimension(-nx:nx,0:global_ny,-global_nz:global_nz-1), intent(out) :: phi

【役割】
静電ポテンシャルのバイナリデータ
　phi/gkvp_f0.47_(MPIランク6桁).0.phi.(ラン数3桁)
について、

レコード数 loop の時刻に対応する静電ポテンシャルを phi に読み込む。

rb_Al_loop(loop, Al)

【引数の型、入出力】
integer, intent(in) :: loop
complex(kind=DP), dimension(-nx:nx,0:global_ny,-global_nz:global_nz-1), intent(out) :: Al

【役割】
ベクトルポテンシャルのバイナリデータ
　phi/gkvp_f0.47_(MPIランク6桁).0.Al.(ラン数3桁)
について、

レコード数 loop の時刻に対応するベクトルポテンシャルを Al に読み込む。

rb_mom_imomisloop(imom, is, loop, mom)

【引数の型、入出力】
integer, intent(in) :: imom, is, loop
complex(kind=DP), dimension(-nx:nx,0:global_ny,-global_nz:global_nz-1), intent(out) :: mom

【役割】
流体モーメントのバイナリデータ
　phi/gkvp_f0.47_(MPIランク6桁).(粒子種1桁).mom.(ラン数3桁)
について、

レコード数 loop の時刻に対応する流体モーメントを mom に読み込む。
ここで、isは粒子種番号で、imom=0～5はそれぞれ順に以下の流体モーメントに対応する。

$$\tilde{n}_{s\boldsymbol{k}} = \int dv^3\, J_{0s\boldsymbol{k}} \tilde{f}_{s\boldsymbol{k}}, \qquad \tilde{u}_{\parallel s\boldsymbol{k}} = \int dv^3\, v_\parallel J_{0s\boldsymbol{k}} \tilde{f}_{s\boldsymbol{k}}, \qquad \tilde{p}_{\parallel s\boldsymbol{k}} = \int dv^3\, \frac{v_\parallel^2}{2} J_{0s\boldsymbol{k}} \tilde{f}_{s\boldsymbol{k}},$$

$$\tilde{p}_{\perp s\boldsymbol{k}} = \int dv^3\, \mu B J_{0s\boldsymbol{k}} \tilde{f}_{s\boldsymbol{k}}, \tilde{q}_{\parallel\parallel s\boldsymbol{k}} = \int dv^3\, v_\parallel \frac{v_\parallel^2}{2} J_{0s\boldsymbol{k}} \tilde{f}_{s\boldsymbol{k}}, \tilde{q}_{\parallel\perp s\boldsymbol{k}} = \int dv^3\, v_\parallel \mu B J_{0s\boldsymbol{k}} \tilde{f}_{s\boldsymbol{k}}$$

rb_trn_itrnisloop(itrn, is, loop, mom)

【引数の型、入出力】
integer, intent(in) :: itrn, is, loop
complex(kind=DP), dimension(-nx:nx,0:global_ny), intent(out) :: trn

【役割】
エントロピーバランスに関する諸量のバイナリデータ
　phi/gkvp_f0.47_(MPIランク6桁).(粒子種1桁).trn.(ラン数3桁)
について、

レコード数 loop の時刻に対応するエントロピーバランスに関する諸量を trn に読み込む。
ここで、isは粒子種番号で、itrn=0～11はそれぞれ順に、
ジャイロ中心揺動エントロピー、静電揺動エネルギー（イオン分極項含む）、磁場揺動エネルギー、波粒子相互作用（$W_E \rightarrow S_s$）、波粒子相互作用（$W_M \rightarrow S_s$）、ExB流による非線形エントロピー伝達、磁場揺動による非線形エントロピー伝達、衝突散逸、ExB流による粒子輸送フラックス、磁場揺動による粒子輸送フラックス、ExB流によるエネルギー輸送フラックス、磁場揺動によるエネルギー輸送フラックス。
※繰り返しになるが、itrn＝8,9,10,11が粒子・熱輸送束の波数空間スペクトルに対応。