Hands-on exercises: Transport code

N.M. Li[1,2], X.Q. Xu[2] and the BOUT++ team

[1]Dalian university of Technology, Dalian, Liaoning, 116024, China
[2]Lawrence Livermore National Laboratory, Livermore, CA 94550, USA

Presented at BOUT++ workshop
August 14-17, 2018
Livermore, CA, USA

This work was performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under Contract No. DE-AC52-07NA27344 and the CSC (No. 201606060097). LLNL-PRES-756618

Outline

- Transport Code introduction(Physical model)
- Hands-on exercise for transport code
  - ✓ Install the BOUT++ and download the transport code
  - ✓ Radial diffusion coefficient calculation by the IDL script
  - ✓ Setting-up the code
  - ✓ Running the code and Data analysis for a simple case

Plasma transport model with drifts

- **Based on Braginskii equations, the ion density, parallel ion velocity and temperature of ions and electrons are described with cross-filed drifts**

$$\frac{\partial N_i}{\partial t} = -\nabla_\parallel (V_{\parallel i} N_i) + \nabla_\perp \cdot (D_\perp \nabla_\perp N_i) + S_I^p - S_{rec}^p - \boldsymbol{\nabla} \cdot (N_i \boldsymbol{V}_{E\times B}) - \boldsymbol{\nabla} \cdot (N_i \boldsymbol{V}_{dia}^i)$$

$$\frac{\partial T_e}{\partial t} = -V_{\parallel e}\nabla_\parallel T_e - \frac{2}{3}T_e \nabla_\parallel V_{\parallel e} + \frac{2}{3N_i}\nabla_\parallel (k_{\parallel e}\nabla_\parallel T_e) + \frac{2}{3N_i}\nabla_\perp (N_i \chi_{\perp e}\nabla_\perp T_e)$$
$$+ \frac{2T_e}{3}\nabla_\perp \cdot \left(\frac{D_\perp}{N_i}\nabla_\perp N_i\right) + \frac{D_\perp}{N_i}\nabla_\perp N_i \cdot \nabla_\perp T_e + \nu_{rec} W_{rec} - \nu_I \left(T_e + \frac{2}{3}W_I\right) - \left(\frac{2m_e}{M_i}\right)\frac{T_e - T_i}{\tau_e}$$
$$-\boldsymbol{V}_{E\times B} \cdot \nabla T_e - \frac{2}{3}T_e \boldsymbol{\nabla} \cdot \boldsymbol{V}_{E\times B} - \frac{2}{3}T_e \boldsymbol{\nabla} \cdot \boldsymbol{V}_{dia}^e + \frac{5T_e}{3e}\boldsymbol{\nabla} \cdot \left(\frac{1}{B_0}\boldsymbol{b}_0 \times \nabla T_e\right) + \frac{2\eta_\parallel}{3N_i}{j_\parallel}^2 + 0.71\frac{2T_e}{3N_i e}\nabla_\parallel j_\parallel$$

$$\frac{\partial T_i}{\partial t} = -V_{\parallel i}\nabla_\parallel T_i - \frac{2}{3}T_i \nabla_\parallel V_{\parallel i} + \frac{2}{3N_i}\nabla_\parallel (k_{\parallel i}\nabla_\parallel T_i) + \frac{2}{3N_i}\nabla_\perp (N_i \chi_{\perp i}\nabla_\perp T_i)$$
$$+ \frac{2T_i}{3}\nabla_\perp \cdot \left(\frac{D_\perp}{N_i}\nabla_\perp N_i\right) + \frac{D_\perp}{N_i}\nabla_\perp N_i \cdot \nabla_\perp T_i + (\nu_{rec} - \nu_I)T_i + \left(\frac{2m_e}{M_i}\right)\frac{T_e - T_i}{\tau_e}$$
$$-\boldsymbol{V}_{E\times B} \cdot \nabla T_i - \frac{2}{3}T_i \boldsymbol{\nabla} \cdot \boldsymbol{V}_{E\times B} - \frac{2}{3}T_i \boldsymbol{\nabla} \cdot \boldsymbol{V}_{dia}^i - \frac{5T_i}{3Z_i e}\boldsymbol{\nabla} \cdot \left(\frac{1}{B_0}\boldsymbol{b}_0 \times \nabla T_i\right)$$

$$\frac{\partial V_{\parallel i}}{\partial t} = -V_{\parallel i}\nabla_\parallel V_{\parallel i} + \frac{4}{3N_i M_i}\nabla_\parallel (\eta_i \nabla_\parallel V_{\parallel i}) + \frac{D_\perp}{N_i}\nabla_\perp N_i \cdot \nabla V_{\parallel i} - \frac{\nabla_\parallel P}{N_i M_i} - (\nu_{CX} + \nu_I)(V_{\parallel i} - V_{\parallel a}) - \boldsymbol{V}_{ExB} \cdot \nabla V_{\parallel i}$$

Energy exchange
Source term
magnetic term
energy flux
ExB term

- **Er calculation by quasi-neutral constraint using vorticity formulation[1]**

$$\frac{\partial \varpi}{\partial t} = B_0 \nabla_\parallel J_\parallel + \mu_{i,\perp}\nabla_\perp^2 \varpi + \mu_{i,\parallel}\nabla_\parallel^2 \varpi + \nabla \cdot (2P\nabla \times \boldsymbol{b}) - \boldsymbol{V}_{E\times B} \cdot \nabla \varpi$$

$$\varpi = \frac{N_i M_i}{B_0}\left(\nabla_\perp^2 \phi + \frac{1}{Z_i e N_i}\nabla_\perp^2 P_i + \frac{1}{N_i}\nabla_\perp \phi \nabla_\perp N_i\right)$$

$$J_\parallel = -\frac{1}{\eta}\nabla_\parallel \phi + \frac{1}{\eta e N_e}\nabla_\parallel P_e + \frac{0.71 k_B}{\eta e}\nabla_\parallel T_e$$

[1] N.M. Li, Comput. Phys. Comm.(2018)

Neutral Particle equation

$$\frac{\partial N_a}{\partial t} - \nabla_{\parallel}\left(D_{\parallel a}^{c}\nabla_{\parallel}N_a\right) - D_{\perp a}^{c}\nabla_{\perp}^{2}N_i = -S_I^p + S_{rec}^p$$

$D_{\perp a}$ and $D_{\parallel a}$ are perpendicular (to the magnetic field) and parallel atom diffusion coefficients. In the atom diffusion process, the charge exchange dominates because of its larger rate coefficient than the recombination. Both parallel and perpendicular atom diffusion coefficients are simply calculated from atom force balance as:

$$D_{\perp a}^{c} = D_{\parallel a}^{c} = D_a = T_a/(M_a\nu_{CX}^{a}), \qquad \nu_{CX}^{a} = N_i\langle\sigma_{CX}V_{th,i}\rangle$$

Thermal velocity: $V_{th,i} = \sqrt{kT_i/M_i}$

Similar with the plasma flux limited coefficients, the atom diffusions coefficients are also flux limited and calculated as:

$$D_{\perp a}^{cl} = D_{\parallel a}^{cl} = D_{\parallel a}^{c}D_{\parallel a}^{fl}/\left(D_{\parallel a}^{c} + D_{\parallel a}^{fl}\right), \qquad D_{\parallel a}^{fl} = V_{th,a}L_a,$$

$$V_{th,a} = \sqrt{kT_a/M_a}$$

L_a is the gradient length at the steepest gradient region.

The Sheath Boundary Condition

- **Quasi-neutral condition assumption: ($ZeN_i = eN_e$);**
- **At the divertor plates, the sheath boundary conditions(SBC) are applied for plasma density, temperature, velocity and potential;**

- **The outgoing sheath particle flux is $\Gamma_i = N_e C_s$, where $C_s = \sqrt{\frac{k(T_e+T_i)}{m_i}}$.**
- **The total transmission heat fluxes are $Q_{e,i} = \gamma_{e,i} k T_{e,i} \Gamma_i$ where $\gamma_e \approx 4.8$ and $\gamma_i \approx 2.5$ are electron and ion sheath heat transmission factors, respectively.**
- **Therefor, for the transport equations the SBC are:**

$$V_{\parallel i}|_{\theta=0} = -C_s|_{\theta=0};\ V_{\parallel i}|_{\theta=2\pi} = C_s|_{\theta=2\pi}$$

$$\nabla_{\parallel} T_{e,i}|_{\theta=0} = Q_{e,i}/\left(k\kappa_{\parallel e,i}\right)|_{\theta=0};\ \nabla_{\parallel} T_{e,i}|_{\theta=2\pi} = -Q_{e,i}/\left(k\kappa_{\parallel e,i}\right)|_{\theta=2\pi}$$

- **Sheath potential**:

$$\phi_{sh} = \phi_B = \frac{T_e}{e} ln\left(\frac{v_{T_e}}{2\sqrt{\pi}C_s}\right) = \frac{T_e}{e}\frac{1}{2} ln\left[\frac{m_i}{m_e}\frac{2T_e}{T_i+T_e}\frac{1}{4\pi}\right]$$

Particle recycling boundary conditions

- **Wall particle recycling boundary conditions:**

The particle recycling boundary condition at the wall or the outermost boundary flux surface is:

$$\frac{\partial ln(N_i)}{\partial x}|_{wall} = -\frac{1}{\sqrt{g^{11}}L_{ni}^{w}}$$

$$\frac{\partial N_a}{\partial x}|_{wall} = \frac{\Gamma_a^w}{\sqrt{g^{11}}D_{\perp a}^c}$$

L_{ni}^{w} is the plasma density gradient length at the wall. R_{cyc} is the recycling coefficient.

Recycling atom particle flux at the wall: $\Gamma_a^w = R_{cyc}\Gamma_i^w = R_{cyc}D_{\perp i}^c N_i/L_{ni}^w$

- **Divertor particle recycling boundary condition:**

According to sheath boundary conditions of plasma density and ion velocity, the plasma particle flux onto the divertor plates is $\Gamma_i^d = N_i c_{se}$. Recycling atom particle flux from divertor plates is $\Gamma_a^d = R_{cyc}\Gamma_i^d$.

$$\nabla_{\parallel} N_a|_{\theta=0} = -\Gamma_a^d/D_{\parallel a}^c|_{y=0}; \quad \nabla_{\parallel} N_a|_{y=-1} = -\Gamma_a^d/D_{\parallel a}^c|_{y=-1}$$

Hands-on exercise for transport code

- Install the BOUT++ and download the transport code
- Radial diffusion coefficient calculation by the IDL script
- Setting-up the code
- Running the code and Data analysis for a simple case

Before exercise: login Edison

login Edison with your training account username and password

$ ssh -XY trainXX@edison.nersc.gov

$ password

Before exercise: download and install BOUT++ code

- **Install the BOUT++**

```
# To clone an individual branch(merge-github) using ssh with your nersc username and password, do the following:
$ cd
$ mkdir BOUT++
$ cd BOUT++
$ git clone -b merge-github ssh://train38@portal-auth.nersc.gov/project/projectdirs/bout/www/git/bout.git merge-github
$ cd merge-github/                    #cd to your bout
$ module swap PrgEnv-intel PrgEnv-gnu
$ ./configure MPICC=cc MPICXX=CC --with-netcdf=/global/u2/c/chma/edison/local --with-fftw=/global/u2/c/chma/edison/local          #configure bout
$ make
```

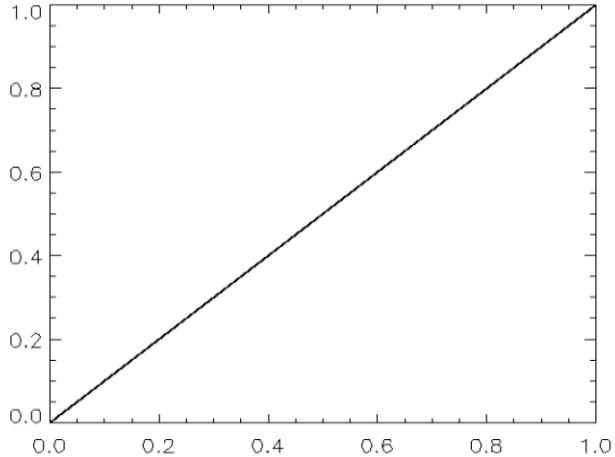
Before exercise: setting IDL

- **Module load IDL and Setting the IDL path**

```
$ cp  ~train38/PUBLIC/BOUT++_Workshop_2018/.idl_startup   ~/
$ cp  ~train38/PUBLIC/BOUT++_Workshop_2018/.bashrc.ext  ~/
$ source ~/.bashrc.ext
# using pcollect to collect data faster by adding pcollect.pro into your bout/tools/idllib
$ cp  ~train38/PUBLIC/BOUT++_Workshop_2018/pcollect.pro  to/your/bout/tools/idllib
```

- **Test the Idl**

```
$ cd
$ idl
IDL> plot,[0,1],chars=2,thick=2
IDL> exit
```



Before exercise: download transport code

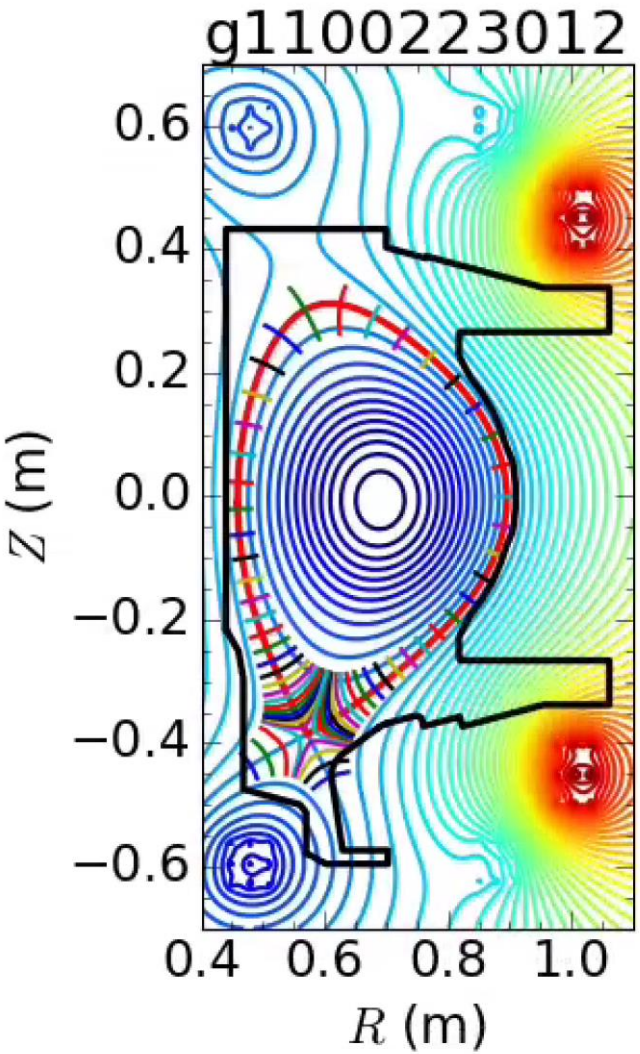
- **Download the transport code**

```
$ cd $SCRATCH/
$ cp -r ~train38/PUBLIC/BOUT++_Workshop_2018/Transport_Code  ./
$ cd Transport_Code/code
```

Generate the grid file

- Use kinetic EFIT equilibrium to generate grids
  - ✓ g-file to nc-flie by using hypnotoad
  - ✓ Grid resolution: NX = 2^n+4(radial), NY= 2^n(poloidal)
- The profiles from the p-file are from experimental measurement
  - ✓ Read p-file data by using pfile2Grid.pro
  - ✓ Fit and smooth the profile by using prof_fit.pro
  - ✓ Writing the data into the nc-file: Ni2Gridall_im.pro

Detail please see the Back up slides



An interpretive approach is used to infer the transport coefficients

- **Based on the experimentally measured plasma profiles inside the separatrix, the effective particle and heat diffusivities can be determined from transport equations(without drifts) with sources from inner radial boundary:**

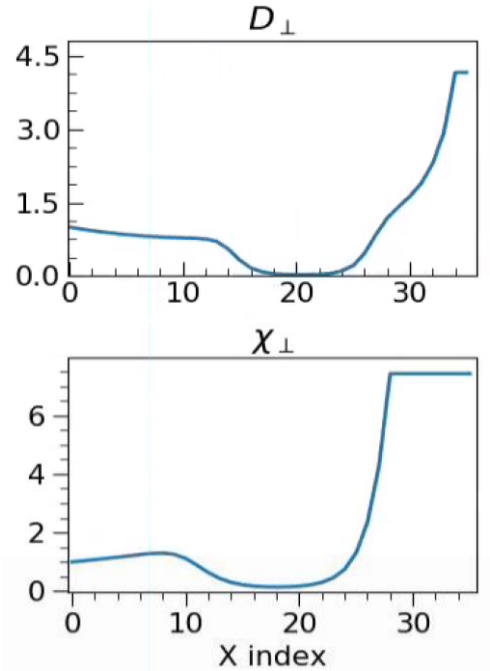
$$D_{\perp}\frac{\partial N_i}{\partial x} = -\Gamma|_{core}$$

$$N_i\chi_{\perp i}\frac{\partial T_i}{\partial x} = -\int_0^x N_i T_i \frac{\partial}{\partial x}\left(\frac{D_{\perp}}{N_i}\frac{\partial N_i}{\partial x}\right)dx - \int_0^x \frac{3D_{\perp}}{2}\frac{\partial N_i}{\partial x}\frac{\partial T_i}{\partial x}dx - Q|_{core}$$

- **In the SOL, the value is consistent for $\chi_{\perp}$**

```
$ idl
IDL> @radial_cofficient_v1.pro
```

```
Boundary values     Core      separatrix        Edge
ni :        2.54245   ---      0.742213   ---      0.623001
ti :        30.8755   ---       4.14614   ---       3.22443
te :        30.8755   ---       4.14614   ---       3.22443
Gradient     Core      separatrix        Edge
ni :       -131.791   ---      -1206.22   ---      -16.0596
ti_no_source :       -5845.23   ---      -9194.09   ---      705.656
te_no_source :       -5845.23   ---      -9194.09   ---      705.656
coefficient  separatrix
diff0:      0.0981367
chi:       0.571038
che:       0.762367
% Compiled module: FILE_WRITE.
```



Running the code

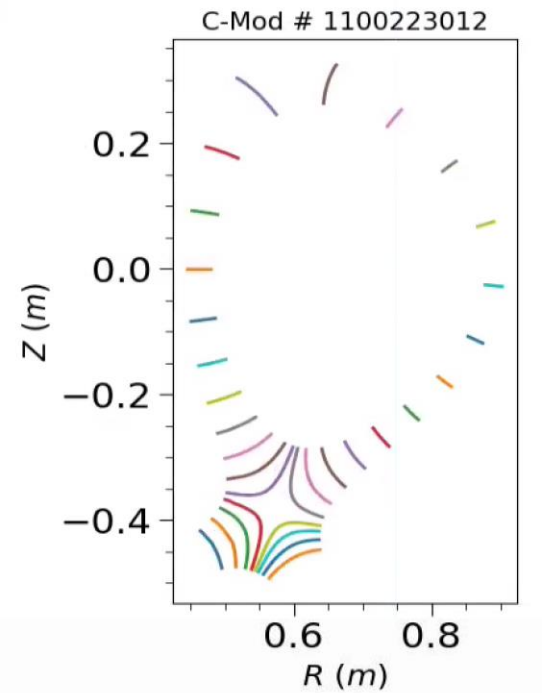
- **Exercise for a simple case:**
  - The C-Mod discharge are used with low grid resolution 36(radial)*32(poloidal)
  - Drifts and neutral are not included in the example case

```
# Compile the code
$ make                          # change BOUT_TOP = path/bout in the makefile
# Submit the job on edison
$ sbatch job_submit.sh
$ sqs                           # check your job
```

Geometry used in the simulation

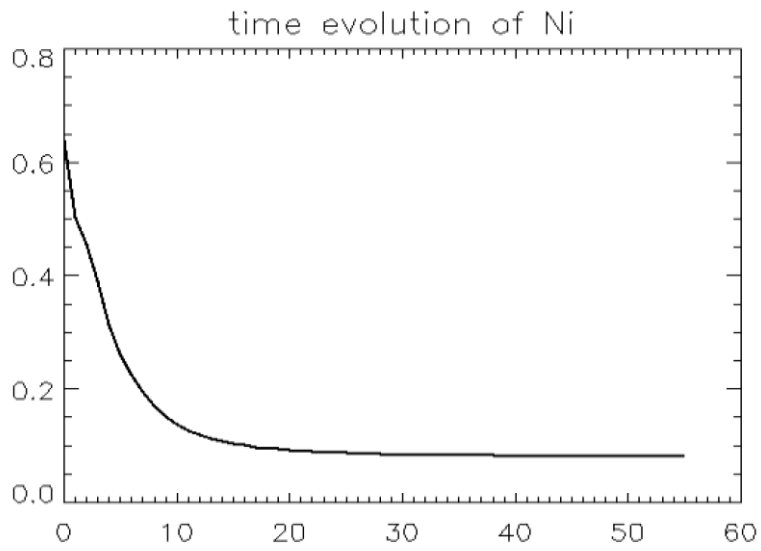
- **The grid-file:**
  - cmod_08MA_1100223012_1150_36x32y_0.9psi1.05_v1.bout.nc

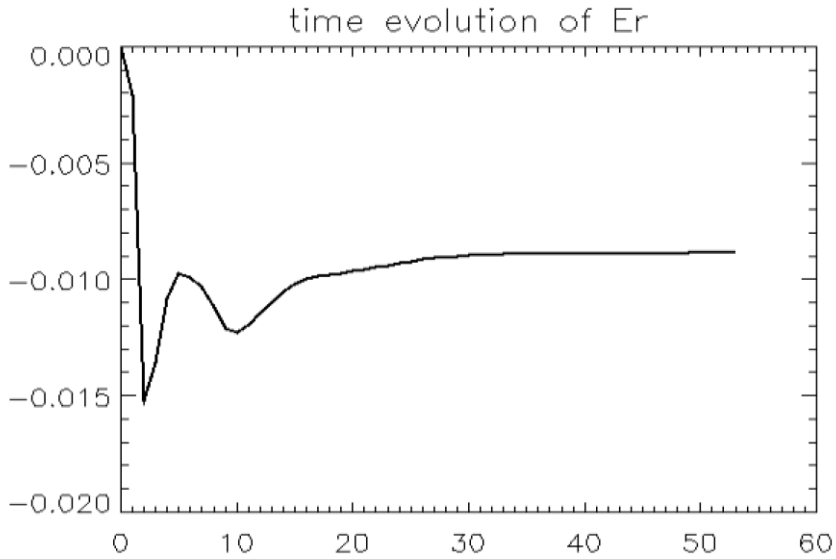
```
$ idl
IDL> g=file_import('cmod_08MA_1100223012_1150_36x32y_0.9psi1.05_v1.bout.nc')
IDL> plot,g.Rxy,g.Zxy,psym=1,chars=2,thick=2
IDL> help,g
```




```
IDL> help,g
** Structure <e5ab78>, 58 tags, length=175416, data length=175402, refs=1:
   CHI_E           FLOAT     Array[36, 32]
   CHI_I           FLOAT     Array[36, 32]
   DIFF            FLOAT     Array[36, 32]
   NIXEXP          FLOAT           2.54245
   PHI_0           FLOAT     Array[36, 32]
   E_R             FLOAT     Array[36, 32]
   N_IMP           FLOAT     Array[36, 32]
   PRESSURE_S      FLOAT     Array[36, 32]
   TEEXP           FLOAT     Array[36, 32]
   TIEXP           FLOAT     Array[36, 32]
   NEEXP           FLOAT     Array[36, 32]
   NIEXP           FLOAT     Array[36, 32]
   PSI_BNDRY       FLOAT      -2.38419e-07
   PSI_AXIS        FLOAT         -0.100966
   BXCVZ           DOUBLE    Array[36, 32]
   BXCVY           DOUBLE    Array[36, 32]
   BXCVX           DOUBLE    Array[36, 32]
   RMAG            FLOAT          0.902322
   BMAG            FLOAT           8.01822
   TI_X            FLOAT           1000.00
   TE_X            FLOAT           1000.00
   NI_X            FLOAT          0.784452
   TI0             FLOAT     Array[36, 32]
   TE0             FLOAT     Array[36, 32]
   NI0             FLOAT     Array[36, 32]
   JPAR0           DOUBLE    Array[36, 32]
```

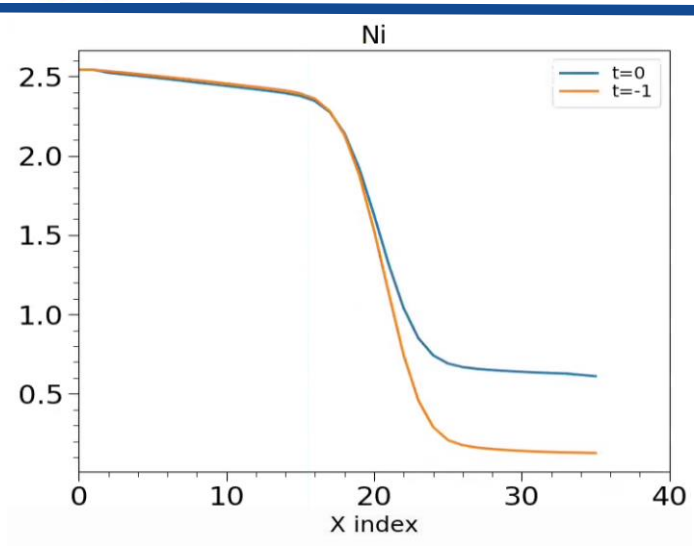
Steady state solution of profiles



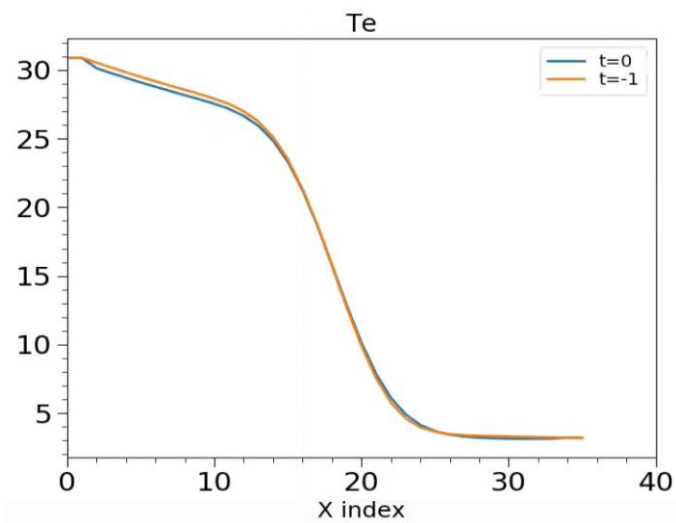



```
IDL> ni=pcollect(path='data',var='Ni')
IDL> er=pcollect(path='data',var='er00x')
IDL> plot,ni[30,20,0,*],chars=2,thick=2,title='time evolution of Ni'
IDL> plot,er[30,20,0,*],chars=2,thick=2,title='time evolution of Er'
```

Steady state solutions of density and temperature



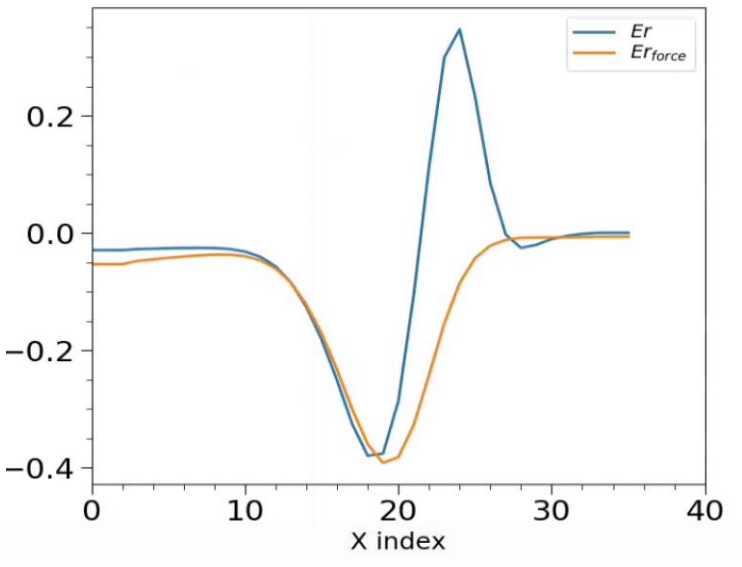
- **Inside the separatrix, we fixed the plasma density and temperature profiles to the experimental profiles with the calculated transport coefficient**
- **While in the SOL, we calculate them with sheath boundary conditions using the transport equations by extending the pedestal transport coefficients into the SOL**




```
IDL> te=pcollect(path='data',var='Te')
IDL> er=pcollect(path='data',var='er00x')
IDL> er_force=pcollect(path='data',var='er0x')
IDL> plot,ni[*,20,0,0],chars=2,thick=2
IDL> oplot,ni[*,20,0,-1],color=2,thick=2
```

```
IDL> plot,te[*,20,0,0],chars=2,thick=2
IDL> oplot,te[*,20,0,-1],color=2,thick=2
```

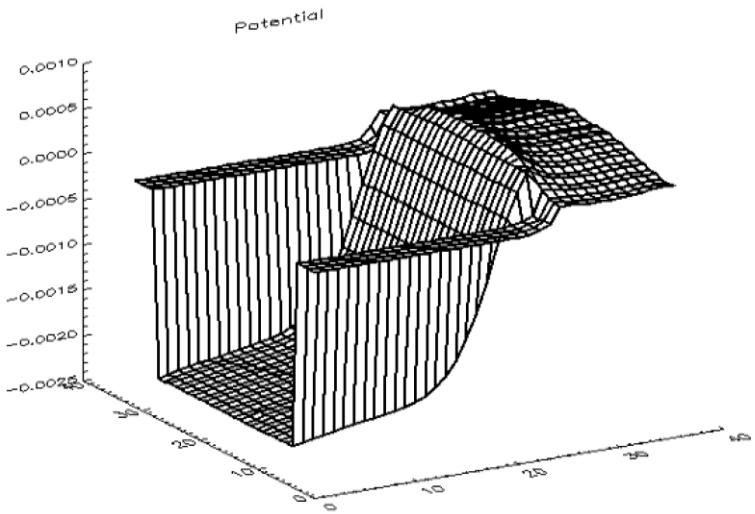
Steady state solutions of radial electric field

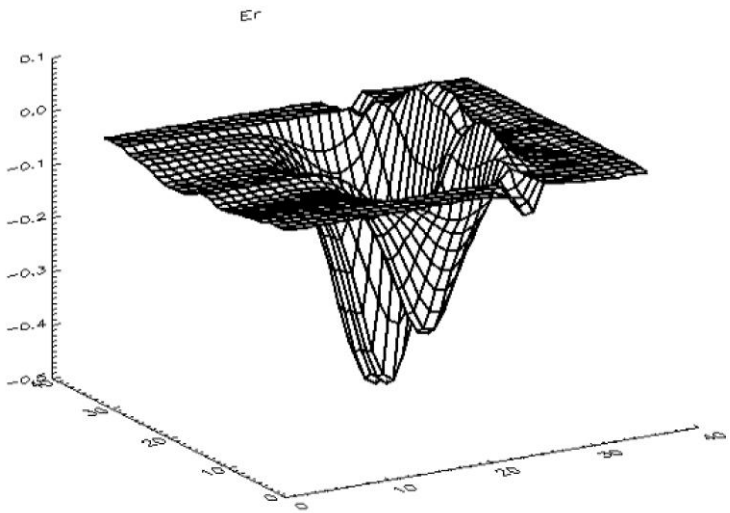



```
IDL> er=pcollect(path='data',var='er00x')
IDL> er_force=pcollect(path='data',var='er0x')
IDL> plot,er[*,20,0,0],chars=2,thick=2
IDL> oplot,er_force[*,20,0,0],color=2,thick=2
```

- **Inside the separatrix, the radial electric field is determined by the force balance with no net flow**
- **In the SOL, the thermal sheath potential is formed on the divertor plates, and it affects the electric field far from the sheath region**

surface plot of potential and radial electric field

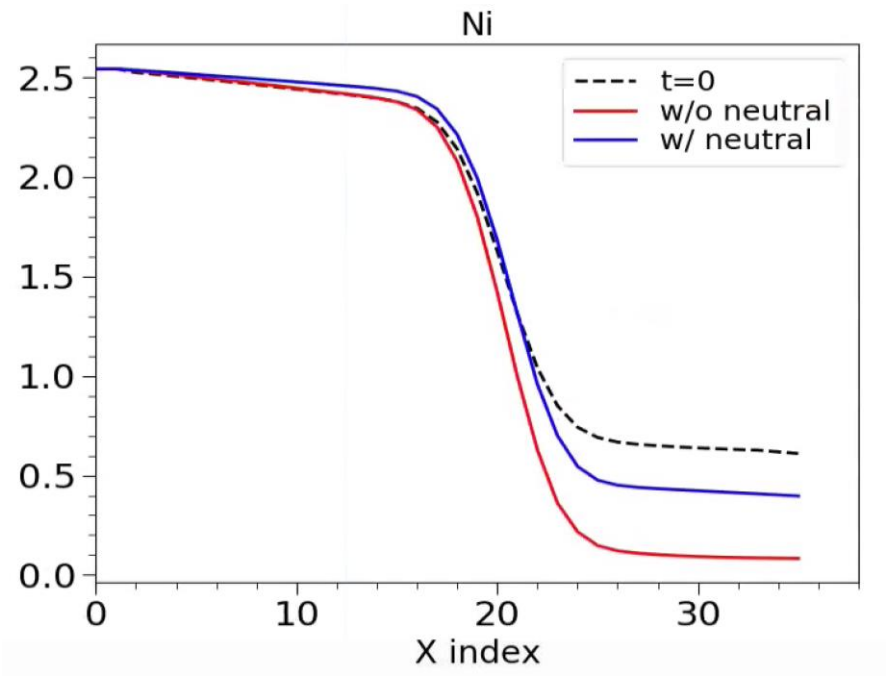





```
IDL> er=pcollect(path='data',var='er00x')
IDL> phi=pcollect(path='data',var='phi00')
IDL> surface,phi[*,*,0,-1],chars=2,thick=2,title='Potential'
IDL> surface,er[*,*,0,-1],chars=2,thick=2,title='Er'
```

Running options with neutral and particle recycling

```
Solving_Eq_Nn = true        # if true, particle recycling are used at the wall and divertor target
SBC_particle_recycle = true        # if true, particle recycling at Sheath BC (or divertor plates)
Wall_particle_recycle = true        # if true, particle recycling at wall
Rate_recycle_wall = 1.        # wall recycling rate; value in range [0,1]
Rate_recycle_div = 1.        # divertor recycling rate; value in range [0,1]
```



Running options with drifts

```
terms_cross = true              # if true, include cross terms in the potential equation
terms_exb_u00 = true            # include curvature drift term in vorticity equation
include_curvature = true        # include curvature drift term in vorticity equation

If include ExB drift, turn on the options:
terms_exb = true
curvature_phi = true            # Include grad(Vexb) effects

If include curvature drift, turn on the options:
include_curvature = true        # include curvature drift term
diamag = true                   # Include diamagnetic effects
energy_flux = true              # Include energy flux effects
```

More physics have been done by using BOUT++ transport module

- **The steady state solution of radial electric file and the effect of drifts on radial electric filed. This work has been publish by:**
  - [1]N.M. Li, Comput. Phys. Comm.(2018)
- **Divertor heat flux widths can been calculated by using BOUT++ transport code with cross-field drifts. C-Mod, EAST, CFETR and ITER have been simulated and the results will be shown during the workshop**
  - Guozhong Deng, EAST Simulation of divertor heat flux widths (Poster Session I)
  - Nami Li, Simulations of divertor heat flux width on C-Mod (Poster Session II)
  - Zeyu Li, Prediction of Divertor Heat Flux width on ITER and CFETR (Poster Session II and Friday talk)

Hands-on exercises: Transport code

BOUT++
Boundary Plasma Turbulence Code

Thanks for your attention!

This work was performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under Contract No. DE-AC52-07NA27344 and the CSC (No. 201606060097).

Back up

Setting IDL and Python path

1. Module load IDL and Python

```
$ module load idl/8.3            # for convenience, one can put this into ~/.bashrc.ext
$ module load python/2.7-anaconda-4.4
# BOUT++ merge-github branch  already include the IDL and Python library in your
bout/tools, you also can use the new python library build by JianGuo Chen
$ cd to/your/bout/tools
# download Python lib build by JianGuo Chen
$ git clone https://gitlab.com/conderls/pylib.git

# setting the startup for IDL and Python
$ cp  ~namili/PUBLIC/BOUT++_Workshop_2018/.idl_startup   ~/
$ cp  ~namili/PUBLIC/BOUT++_Workshop_2018/pythonrc.py
~/.ipython/profile_default/startup/
# using pcollect to collect data faster by adding pcollect.pro into your bout/tools/idllib
$ cp  ~namili/PUBLIC/BOUT++_Workshop_2018/pcollect.pro  your/bout/tools/idllib
```

Setting IDL and Python path

2. Setting the IDL and Python path

```
# add the IDL and Python path into the ~/.bashrc.ext
$ vi ~/.bashrc.ext
# setting the IDL path by yourself
$ export IDL_PATH=$(pwd)/idllib:$IDL_PATH
$ export IDL_PATH=$(pwd)/tokamak_grids/all:$IDL_PATH
$ export IDL_STARTUP=$HOME/.idl_startup
# setting the Python path by yourself
$ export PATH =$(pwd)/pylib /bin:$PATH
$ export PYTHONPATH=$(pwd)/pylib:$PYTHONPATH
$ source ~/.bashrc.ext
# install python dependencies
$ pip install -r  $(pwd)/pylib/pip-requirements     # if the command above failed, try:
$ pip install --user -r  $(pwd)/pylib/pip-requirements
```

Download and install GNU lib

NB: compile BOUT++ transport code with GNU Scientific Library(gsl)

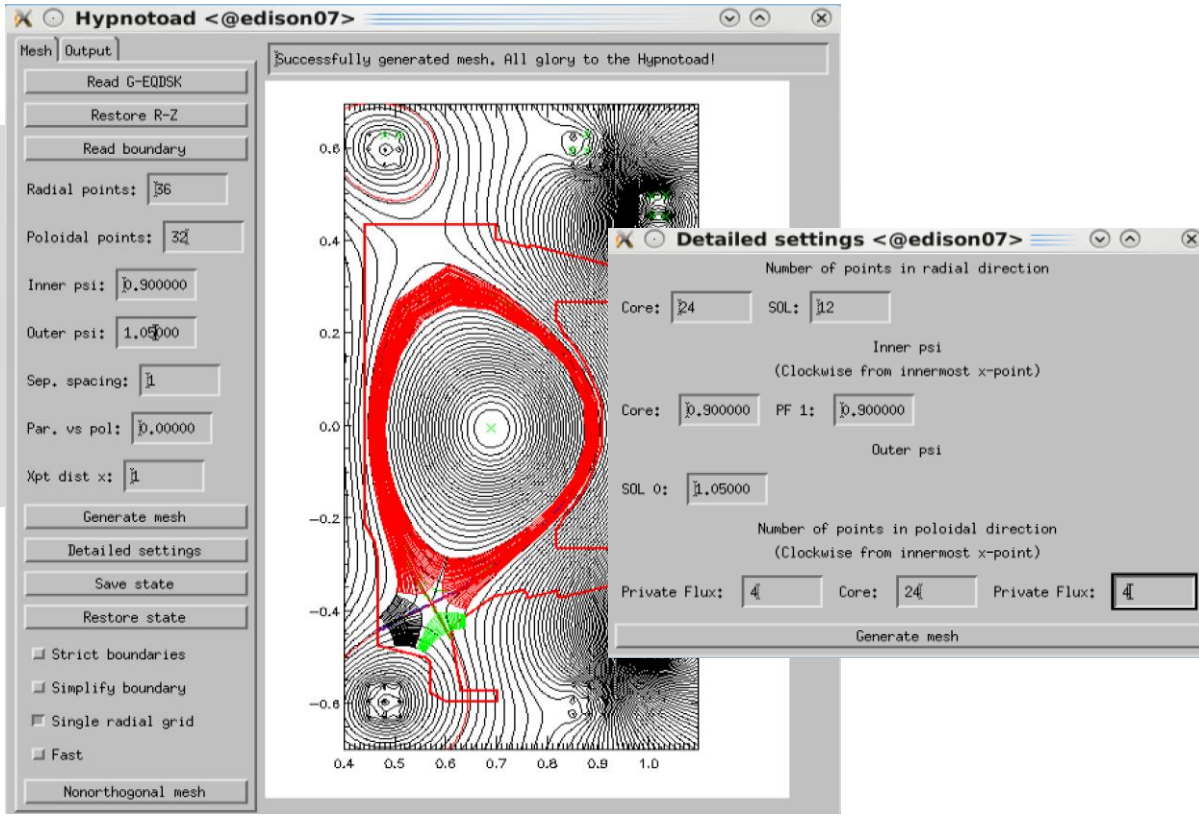
```
$ module load gsl                    #  load gsl module
$ cd /path/to/your/bout              # modify file make.config
$ cp make.config make.config_gsl
# add gsl environment to compile BOUT++ make.config_gsl
$ echo $GSL
# -I/usr/common/software/gsl/1.16/intel/include -
L/usr/common/software/gsl/1.16/intel/lib -lgsl -lgslcblas

# line 42,43 in make.config_gsl:
# append '-I/usr/common/software/gsl/1.16/gnu/include' to EXTRA_INCS
# append -L/usr/common/software/gsl/1.16/gnu/lib -lgsl -lgslcblas to EXTRA_LINBS
```

Generate grid-file for BOUT++

- **Do this work by using IDL scripts**
  - ✓ **First read a g-file equilibrium by using hypnotoad**
    - **G-file is a community standard equilibrium file format, originally from EFIT**
  - ✓ **Grid resolution: NX = 2^n+4(radial), NY= 2^n(poloidal)**

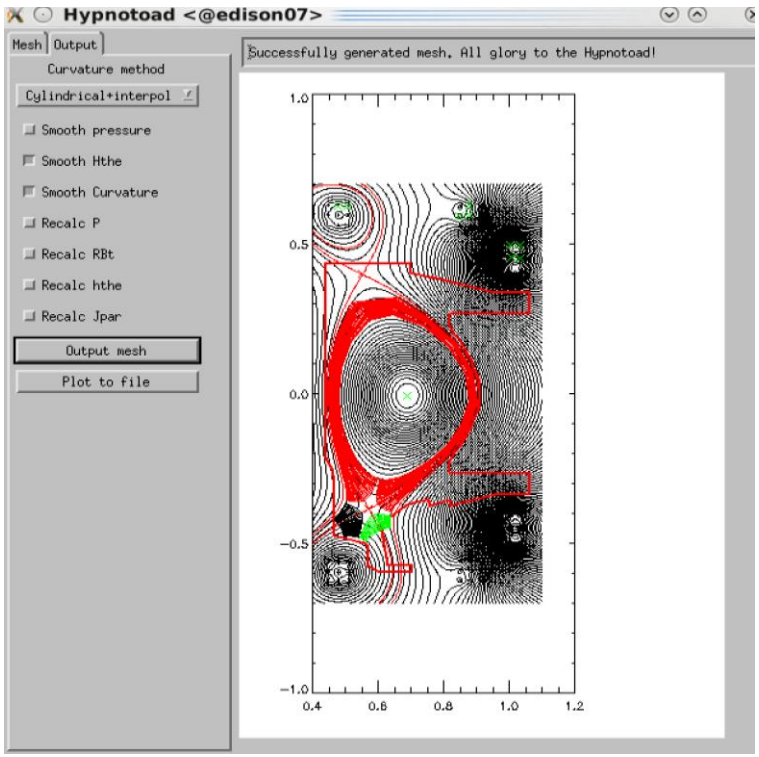
```
% cd BOUT++_Workshop_2018/gridgen
% idl
IDL> .r hypnotoad
IDL> hypnotoad
Read G-EQDSK -> Generate mesh ->Detailed settings
```

```
cp ~train38/PUBLIC/BOUT++_Workshop_2018/gridgen ./ -r
```



Generate grid-file for BOUT++

Output -> output mesh
Generating plasma profiles:
  1. Flat temperature profile
  2. Flat density profile
  3. Te proportional to density
% Compiled module: GET_INTEGER.
Profile option:1
Setting flat temperature profile
% Compiled module: GET_FLOAT.
Temperature (eV):1000
Maximum density (10^20 m^-3): 0.784452
% Compiled module: GET_YESNO.
Is this ok?y
Setting rmag = 0.902322
Setting bmag = 8.01822
Writing grid to file /global/u2/n/namili/grid/cmod-08MA.grid.nc
% Compiled module: FILE_OPEN.
% Compiled module: NCDF_EXISTS.
% Loaded DLM: NCDF.
% Compiled module: FILE_WRITE.
% Compiled module: REVERSE_INDS.
% Compiled module: FILE_CLOSE.
DONE



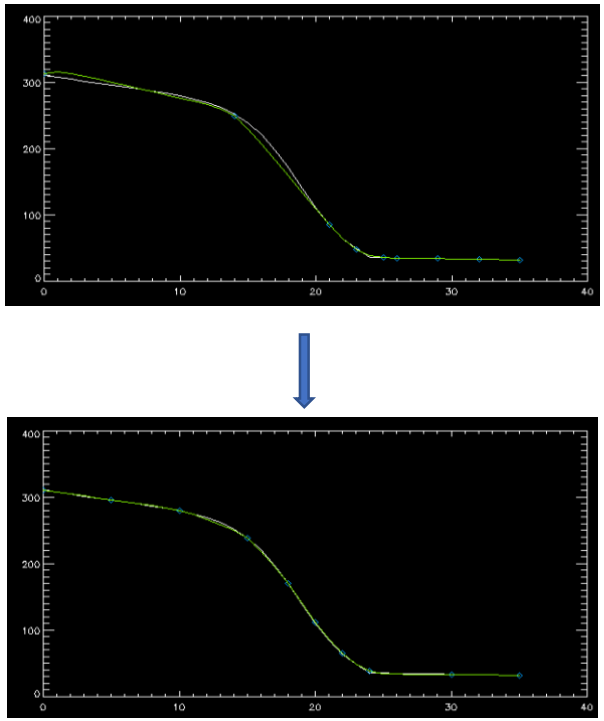
save

cmod-08MA.grid.nc

Read the experiment data to the grid-file

- **The profiles from the p-file are from experimental measurement**
  - ✓ **read experiment data from p-file by using pfile2Grid.pro**

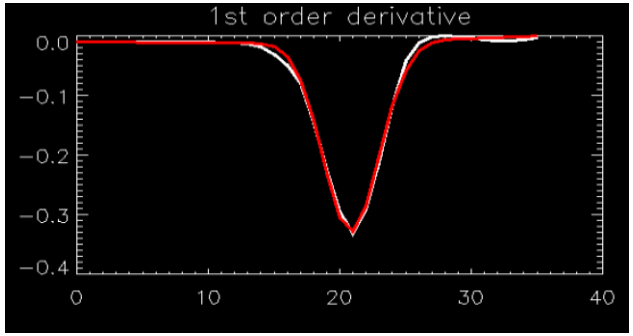
```
% cd ../grid
% idl
IDL> .r pfile2Grid.pro
IDL> pfile2grid,'p1100223012.01149_261','cmod-08MA.grid.nc',/spline
Is Er start from Line 4113 in pfile?(y/n):y
Is Omgeb start from Line 2828 in pfile?(y/n):y
Is this smooth OK?n
Using input x index for spline-nodes:
% Compiled module: GET_INTEGER.
Please enter the total number of input x index:10
The x index for position 1 is set to be:       0
Please enter x index for position      2 :
:5
……
Is this smooth OK?y
```

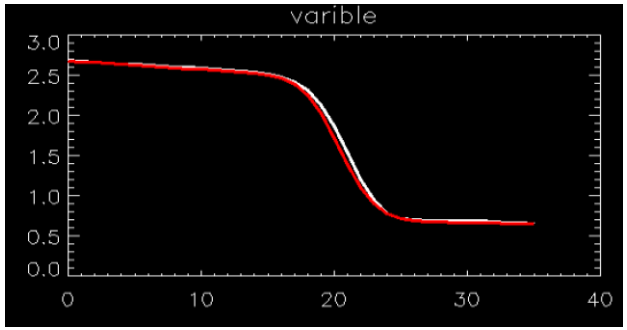


Read the experiment data to the grid-file

- **The profiles from the p-file are from experimental measurement**
  - ✓ read experiment data from p-file by using pfile2Grid.pro
  - ✓ **Smooth the profiles by using prof_fit.pro**

```
IDL> .r prof_fit.pro
% Compiled module: PROF_FIT.
IDL> restore,'ne.sav'
IDL> nd=prof_fit(nee,nterms=6)
No input grid, use index for x coordinate.
% Compiled module: DERIV.
% Compiled module: GAUSSFIT.
% Compiled module: POLY_FIT.
% Compiled module: CURVEFIT.
The coefficients for Gaussian function are:   -0.319874      20.8644
2.15467
  -0.00947844 -0.000369907  1.80779e-05
% Compiled module: GET_YESNO.
Is this fitting OK?y
IDL> nee=nd
IDL> save,nee,f='ne.sav'
……
```





Read the experiment data to the grid-file

- **The profiles from the p-file are from experimental measurement**
  - ✓ read experiment data from p-file by using pfile2Grid.pro
  - ✓ Smooth the profile by using prof_fit.pro
  - ✓ **Writing the data into the grid-file: Ni2Gridall_im.pro**

```
IDL> .r Ni2Gridall_im.pro

IDL> Ni2gridall_im,'p0.sav','ne.sav','ni.sav','te.sav','ti.sav','er.sav','cmod-08MA.grid.nc'

Is the 6th file Er?(y/n):y

% Compiled module: FILE_WRITE.
```

```
IDL> g=file_import('cmod-08MA.grid.nc')

IDL> help,g
```

```
PHI_0           FLOAT     Array[36, 32]
E_R             FLOAT     Array[36, 32]
N_IMP           FLOAT     Array[36, 32]
PRESSURE_S      FLOAT     Array[36, 32]
TEEXP           FLOAT     Array[36, 32]
TIEXP           FLOAT     Array[36, 32]
NEEXP           FLOAT     Array[36, 32]
NIEXP           FLOAT     Array[36, 32]
```

Setting-up the code input

```
# the grid file will be used
grid = "cmod_08MA_1100223012_1150_36x32y_0.9psi1.05_v1.bout.nc"

##!NB: NXPE must be divided by (NX-4), while 4 is the guard cells
NXPE = 16                # X parallelised if > 1, NXPE*NYPE=mppwidth

##### Solve equations ##############################
Solving_Eq_Ni = true
Solving_Eq_Ti = true
Solving_Eq_Te = true
Solving_Eq_Vi = true
Solving_Eq_Nn = false     # if true, particle recycling are used at the wall and divertor targets
Solving_Eq_phi01 = true
Solving_Eq_U00 = true

# read experiment profiles Te Ti Ni from grid file
load_experiment_profiles =true
# read diffusion transport coefficient from grid file
load_grid_trans = true
```

Setting-up the code input

```
# Apply Sheath Boundary condition
Sheath_BC = true
SBC_particle_recycle = true      # if true, particle recycling at Sheath BC (or divertor plates)
Wall_particle_recycle = true     # if true, particle recycling at wall
Rate_recycle_wall = 1.           # wall recycling rate; value in range [0,1]
Rate_recycle_div = 1.            # divertor recycling rate; value in range [0,1]

# cross-field drifts
terms_cross = true               # if true, include cross terms in the potential equation

If include ExB drift, turn on the options:
terms_exb = true
terms_exb_u00 = true
curvature_phi = true             # Include grad(Vexb) effects

If include curvature drift, turn on the options:
include_curvature = true         # include curvature drift term
diamag = true                    # Include diamagnetic effects
energy_flux = true               # Include energy flux effects
```

Setting-up the code input

```
# Setting Boundary condition for Ni, Ti and Te
[Ni]
# bndry_xin  = neumann(-126.484)          # same as dNidx_xin_au
bndry_xin  = relax(dirichlet(2.54245))    # same as Ni_core/Ni_x
bndry_pf  = neumann                       # at private flux region
bndry_xout = neumann                      # not necessary to turn off when use particle recycling

[Te]
# bndry_xin = neumann(-4778.38)           # NB: same as dTedx_xin_au
bndry_xin  = relax(dirichlet(30.88))      # same as Te_core/Te_x
bndry_pf  = neumann
bndry_xout  = neumann

[Te]
# bndry_xin = neumann(-4778.38)           # NB: same as dTedx_xin_au
bndry_xin  = relax(dirichlet(30.88))      # same as Ti_core/Ti_x
bndry_pf  = neumann
bndry_xout  = neumann
```

```
field3d.cxx(2585): warning #3180: unrecognized OpenMP #pragma
  #pragma omp critical
          ^

field3d.cxx(2640): warning #3180: unrecognized OpenMP #pragma
  #pragma omp critical (alloc)
          ^
```