June 23rd, 2022
Eurofusion & IFERC WK on GPU Programming
Online

GPU acceleration of 5D full-*f* gyrokinetic code GKNET using OpenACC

Contents

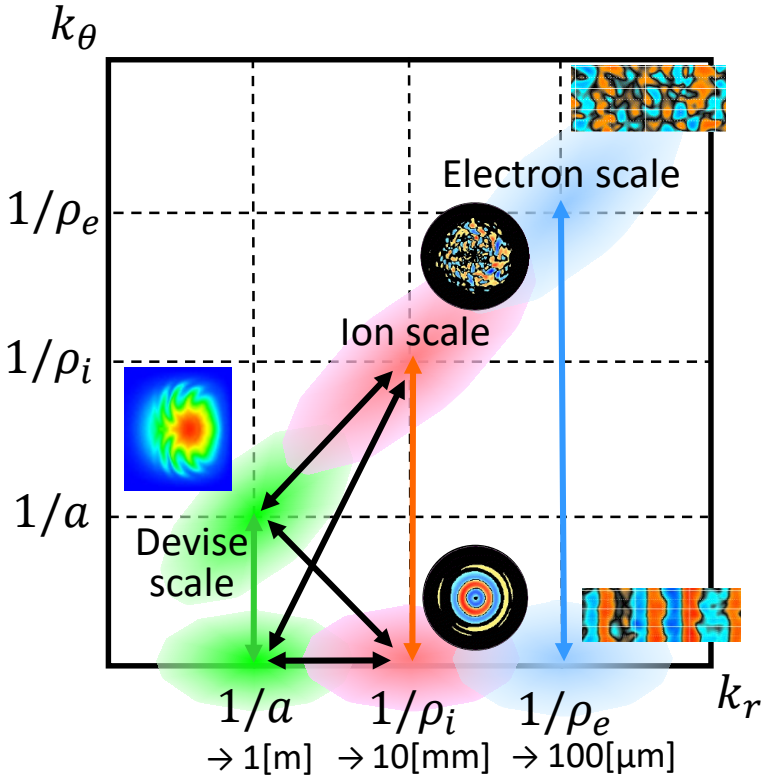
Kenji IMADERA **Shuhei GENKO** **Shuhei OKUDA**

Graduate School of Energy Science, Kyoto University, Japan

Acknowledgement

M. Yagi (QST), N. Miyato (QST), H. Seto (QST), A. Naruse (NVIDIA Japan)

Multi-scales in magnetically confined plasmas



Device size scale
Profile evolution, MHD
Spatial scale～1[m]
→ Transport/Fluid simulation

Ion gyro scale
Ion-scale turbulence/flow
Spatial scale～10[mm]
→ Gyrokinetic simulation

Electron gyro scale
Electron-scale turbulence/flow
Spatial scale~100[μm]
→ Gyrokinetic simulation

- ✓ Basic approach is scale separation (Reduction to elements).
- ✓ Our purpose is to do direct numerical multi-scale simulation for both devise-scale profile evolution and ion-scale turbulence to clarify their hierarchical interactions.

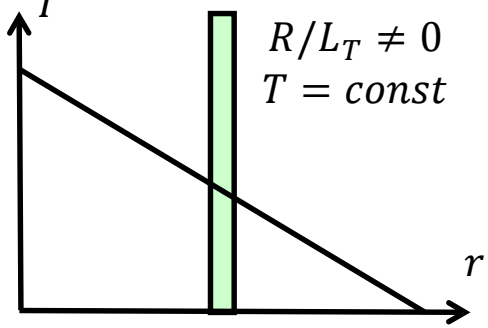
Global/Local gyrokinetics

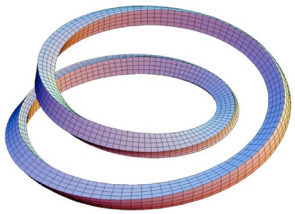
Local δf approach

$$\partial_t f_{eq} - [H, f_{eq}] = C(f_{eq}) + S$$

$$\partial_t \delta f - [H, \delta f] - [\delta H, f_{eq}] - [\delta H, \delta f] = C(\delta f)$$

Gradient-driven





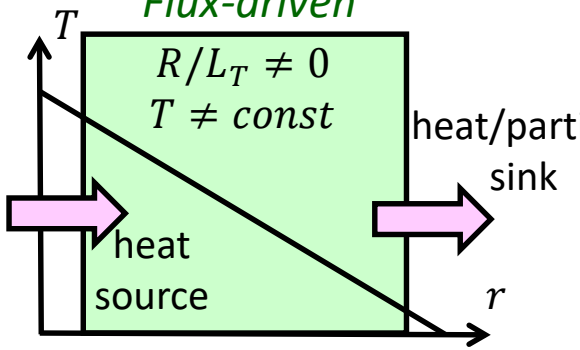
- Very powerful tool to estimate turbulent transport process
- Computationally efficient
  -> multi(ion/electron)-scale simulation

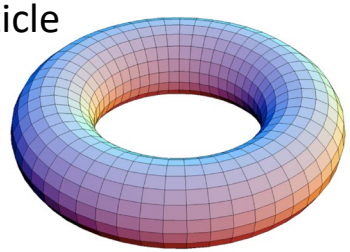
Global full-f approach

$$\partial_t f_{eq} - [H, f_{eq}] = C(f_{eq}) + S$$

$$\partial_t \delta f - [H, \delta f] - [\delta H, f_{eq}] - [\delta H, \delta f] = C(\delta f)$$

Flux-driven





- Multi(profile/ion)-scale simulation
- Mean E_r is self-consistently determined
  -> Internal Transport Barrier (ITB)
- Both neoclassical & turbulent transport process can be traced

Global full-*f* gyrokinetic code GKNET

[Ishizawa, PoP-2019]

GKNET
Polar
Electrostatic
Adiabatic Electron
full-*f*

GKNET-KE
Polar
Electrostatic
Kinetic Electron
global delta-*f*

GKNET-KE-EM
Polar
Electromagnetic
Kinetic Electron
global delta-*f*

GKNET-(R, Z)
Rectangular
Electrostatic
Adiabatic Electron
full-*f*

[Obrejan, PFR-2015]
[Obrejan, CPC-2017]

GKNET-HE
Polar
Electrostatic
(Hybrid) Kinetic Electron
[Lanti, JP-2018]
full-*f*

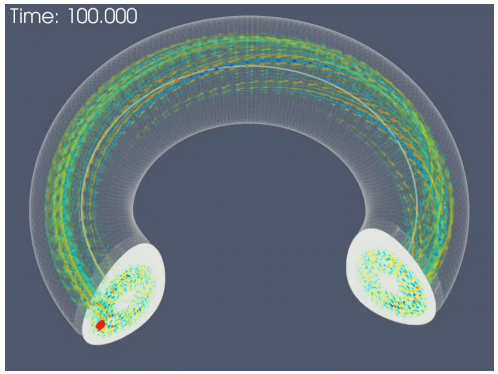
[Imadera, IAEA-2020]

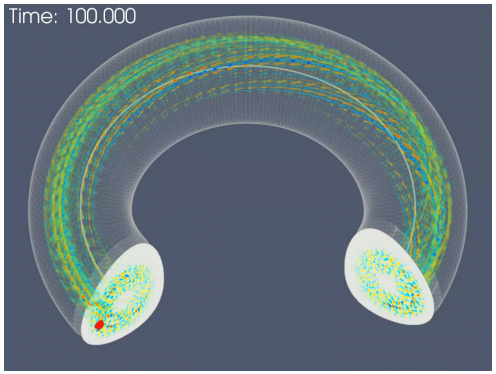
✓ Original GKNET is a full-*f* electrostatic gyrokinetic code with adiabatic electron, which is extended to

(1) Rectangular coordinate version

(2) Electromagnetic delta-*f* version

(3) Hybrid electron full-*f* version





Animation of 3D electrostatic potential and trapped ion (left) and trapped electron (right) obtained by GKNET-HE

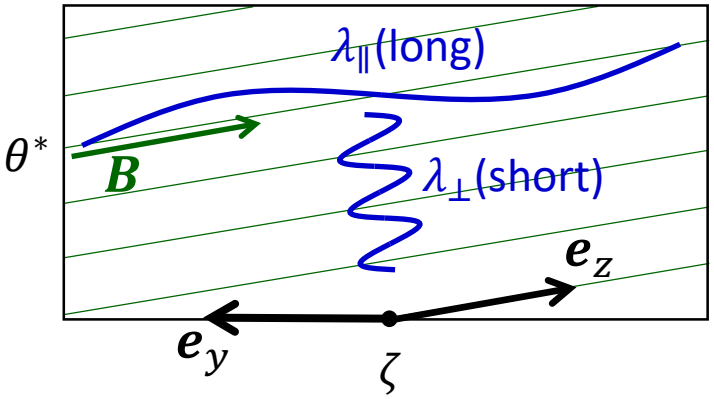
Targets of this talk

✓ However, the numerical cost becomes huge once we treat kinetic electron dynamics.

A) Implementation of field aligned coordinate

✓ To reduce the number of the simulation grid and the resultant calculation time, we introduced field aligned coordinate given by geometrical toroidal angle ζ and straight field-line poloidal angle θ^* as;

$$\begin{cases} x = \rho \\ y = q(\rho)\theta^* - \zeta \\ z = \theta^* \end{cases}$$



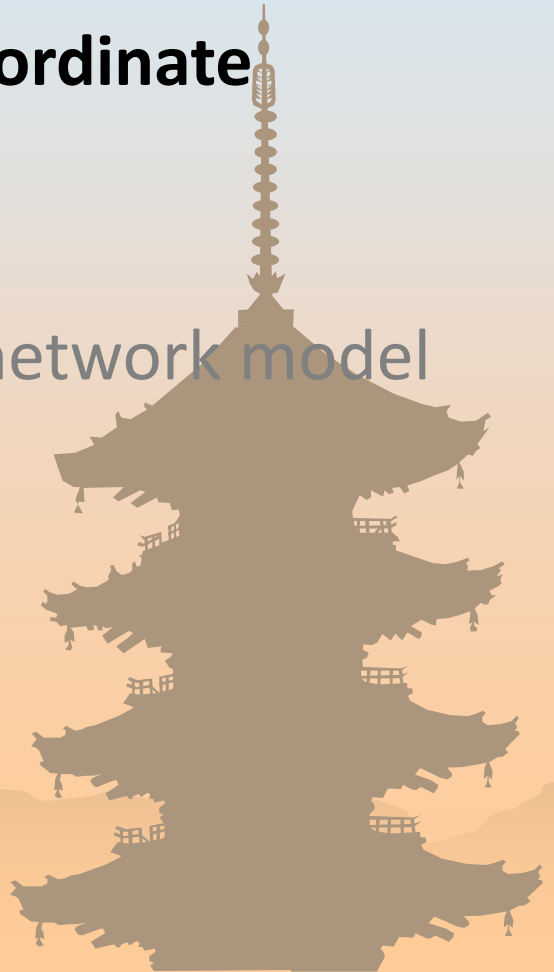
B) GPU acceleration by OpenACC

✓ We also tried the GPU acceleration by using OpenACC directives and then verified its efficiency on MARCONI 100 (CINECA, Italy).

C) Evaluation of globality based on neural network model

✓ We evaluated the globality of heat transport from obtained 1D data of temperature and heat flux by utilizing the neural network model. (GPU acceleration is still going on…)

Contents



Implementation of field aligned coordinate - 1

Circular concentric magnetic field in field aligned coordinate

$$\boldsymbol{B} = \frac{B_0}{\bar{q}(r)R}\boldsymbol{e}_\theta + \frac{B_0 R_0}{R^2}\boldsymbol{e}_\varphi$$

Toroidal coordinate (r, θ, φ)

straight field-line poloidal angle

$$\theta^* = \frac{1}{q(r)}\int_0^\theta \frac{B\cdot\nabla\varphi}{B\cdot\nabla\theta'}d\theta'$$

$$\begin{cases} \rho = r \\ \theta^* = 2\tan^{-1}\left(\sqrt{\dfrac{R_0 - r}{R_0 + r}}\tan\dfrac{\theta}{2}\right) \\ \zeta = \varphi \end{cases}$$

$$\boldsymbol{B} = \frac{B_0}{q(r)R^2}\boldsymbol{e}_{\theta^*} + \frac{B_0 R_0}{R^2}\boldsymbol{e}_\varphi$$
$$= g(\rho)\delta(\rho,\theta^*)\boldsymbol{e}^\rho + I(\rho)\boldsymbol{e}^{\theta^*} + g(\rho)\boldsymbol{e}^\zeta$$

Toroidal coordinate with straight field-line poloidal angle (ρ, θ^*, ζ)

$$\begin{cases} x = \rho \\ y = q(\rho)\theta^* - \zeta \\ z = \theta^* \end{cases}$$

$$\boldsymbol{B} = \frac{B_0 R_0}{q(r)R^2}\boldsymbol{e}_z$$
$$= [q'(x)z + \delta(x,z)]g(x)\boldsymbol{e}^x - g(x)\boldsymbol{e}^y + [q(x)g(x) + I(x)]\boldsymbol{e}^z$$

Field aligned coordinate (x, y, z)

Implementation of field aligned coordinate - 2

Gyrokinetic equation of motion in field aligned coordinate

$$\frac{dx}{dt} = \underbrace{-\varepsilon\mu\frac{g}{eD}\partial_z B}_{\text{Grad } B \text{ drift}} \underbrace{-\varepsilon\frac{m}{e}v_\parallel^2\frac{g}{DB}\partial_z B}_{\text{Curvature drift}} \underbrace{-\varepsilon\frac{qg+I}{D}\partial_y\phi - \varepsilon\frac{g}{D}\partial_z\phi}_{E\times B \text{ drift}}$$

$$\frac{dy}{dt} = \varepsilon\mu\frac{qg+I}{eD}\partial_x B - \varepsilon\mu\frac{(q'z+\delta)g}{eD}\partial_z B + \varepsilon\frac{m}{e}v_\parallel^2\frac{1}{D}\left[g\partial_z\delta - I' + \frac{(qg+I)\partial_x B - (q'z+\delta)g\partial_z B}{B}\right]$$
$$+\varepsilon\frac{qg+I}{D}\partial_x\phi - \varepsilon\frac{(q'z+\delta)g}{D}\partial_z\phi$$

$$\frac{dz}{dt} = \underbrace{v_\parallel\frac{B}{D}\frac{d\psi}{dr}}_{\text{Parallel streaming}} + \varepsilon\mu\frac{g}{eD}\partial_x B + \varepsilon\frac{m}{e}v_\parallel^2\frac{g}{DB}\partial_x B + \varepsilon\frac{g}{D}\partial_x\phi + \varepsilon\frac{(q'z+\delta)g}{D}\partial_y\phi$$

$$\frac{dv_\parallel}{dt} = \varepsilon\mu\frac{v_\parallel g\partial_z B}{eDB}\partial_x B - \frac{\mu B}{mD}\left[\frac{d\psi}{dr} + \varepsilon\frac{m}{e}v_\parallel\frac{g\partial_x B}{B^2}\right]\partial_z B$$
$$+\varepsilon\frac{v_\parallel g\partial_z B}{DB}\partial_x\phi - \varepsilon\frac{v_\parallel}{D}\left[g\partial_z\delta - I' + \frac{(qg+I)\partial_x B - (q'z+\delta)g\partial_z B}{B}\right]\partial_y\phi - \frac{eB}{mD}\left[\frac{d\psi}{dr} + \varepsilon\frac{m}{e}v_\parallel\frac{g\partial_x B}{B^2}\right]\partial_z\phi$$

- ✓ Advection term along the magnetic field line appears only in dz/dt.
- ✓ These equations are derived from the gyrokinetic one-form so that the phase space conservation is rigorously satisfied. -> Morinishi scheme can be applied

Implementation of field aligned coordinate - 3

Gyrokinetic quasi-neutrality condition in field aligned coordinate

$$\nabla\cdot\left(\frac{m_i n(x)}{B(x,z)^2}\nabla_\perp\phi\right)-\frac{e^2 n}{T_e}\left[\phi-\langle\phi\rangle_f\right]=-2\pi e\iint\langle\delta f_i\rangle\frac{B_\parallel^*}{m_i}dv_\parallel d\mu$$

$$\Rightarrow\ (L_0+L_1)\phi(x,y,z)=s(x,y,z)$$

$$L_0=c_1(x,z)\frac{\partial^2}{\partial x^2}+c_2(x,z)\frac{\partial^2}{\partial y^2}+c_3(x,z)\frac{\partial}{\partial x}\frac{\partial}{\partial y}+c_4(x,z)\frac{\partial}{\partial x}+c_5(x,z)\frac{\partial}{\partial y}+c_6(x)$$

$$L_1=l_1(x,z)\frac{\partial}{\partial x}\frac{\partial}{\partial z}+l_2(x,z)\frac{\partial}{\partial y}\frac{\partial}{\partial z}+l_3(x,z)\frac{\partial^2}{\partial z^2}+l_4(x,z)\frac{\partial}{\partial z}$$

Step-1 : FFT along the y direction ← because all the coefficients are independent to y

Step-2 : Set the initial guess $\hat{\phi}_n^{(0)}(x,z)$, and then solve $\hat{L}_0\hat{\phi}_n^{(1)}(x,z)+\hat{L}_{1,D}\hat{\phi}_n^{(1)}(x,z)=\hat{s}_n(x,z)-\hat{L}_{1,ND}\hat{\phi}_n^{(0)}(x,z)$ by using the 1D matrix solver

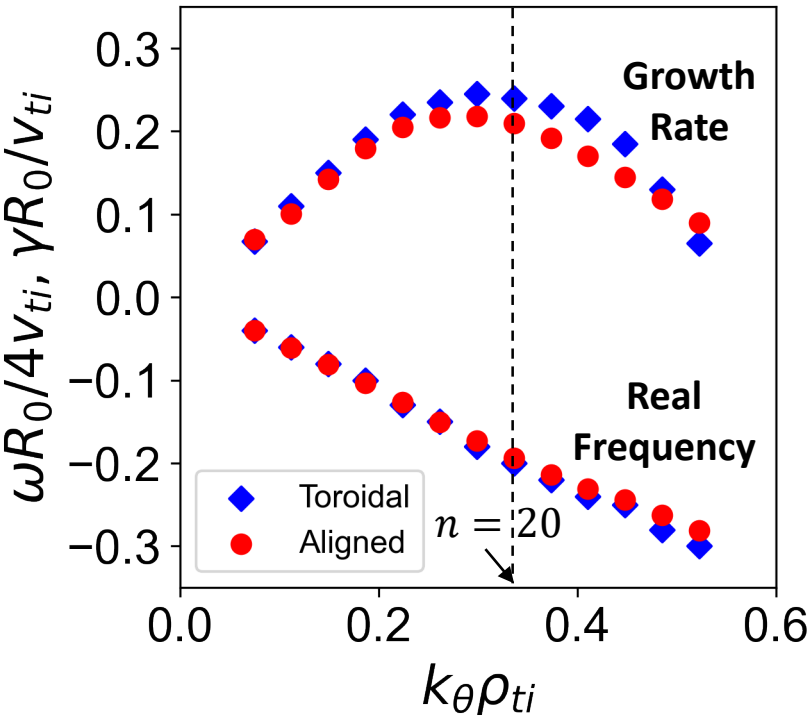
Step-3 : By repeating Step-2 (=Jacobi method), get the conveged solution $\hat{\phi}_n$

← because $\frac{\partial\phi}{\partial z}$ is higher order, a few iterations are enough for the convergence

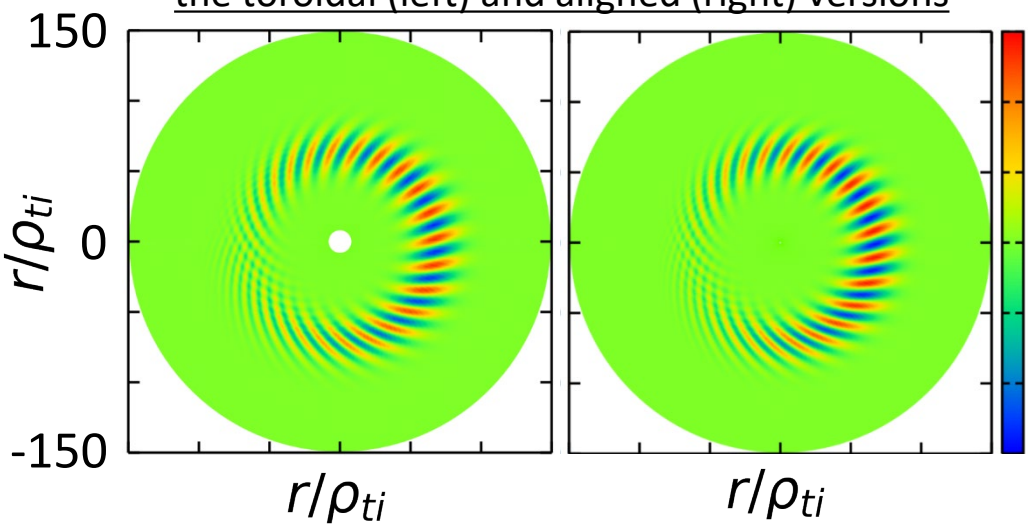
Linear benchmark test - 1

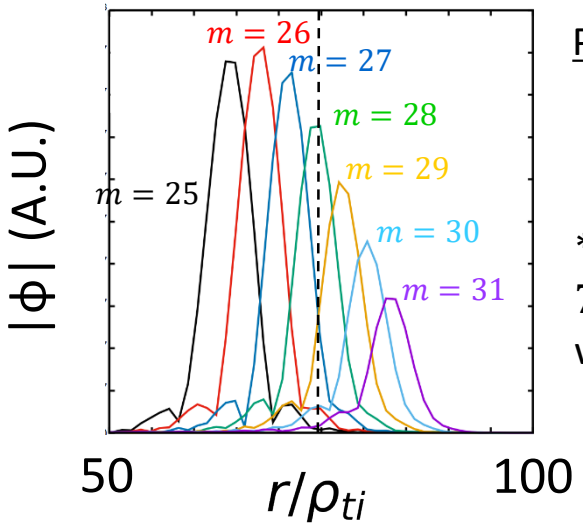
Parameter	Value
a_0/ρ_i	150
a_0/R_0	0.36
$(R_0/L_n)_{r=a_0/2}$	2.22
$\left(R_0/L_{T_{i,e}}\right)_{r=a_0/2}$	6.92

Dispersion relation of ITG mode



Poloidal eigenfunction with $n = 20$ obtained by the toroidal (left) and aligned (right) versions



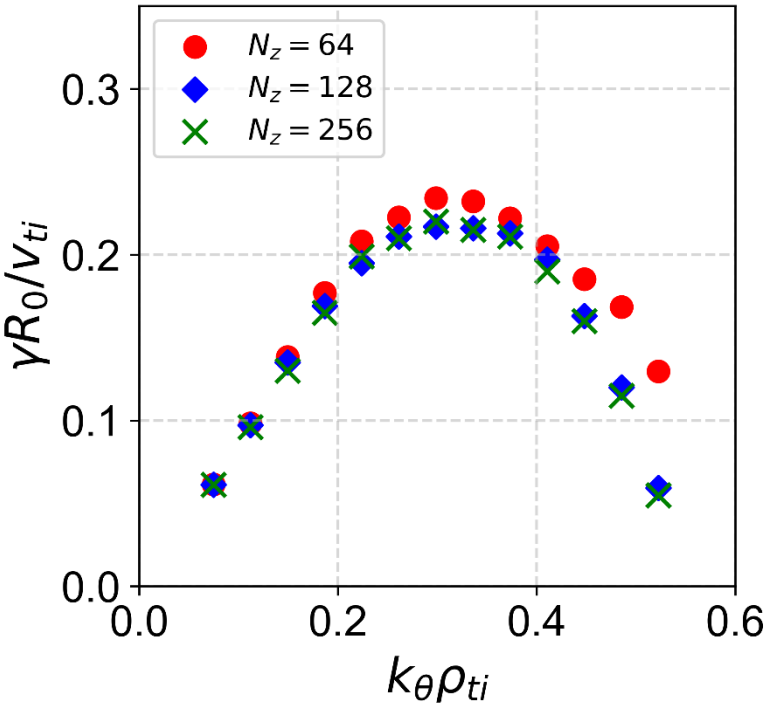


Poloidal harmonics of $n = 20$ obtained by the aligned version

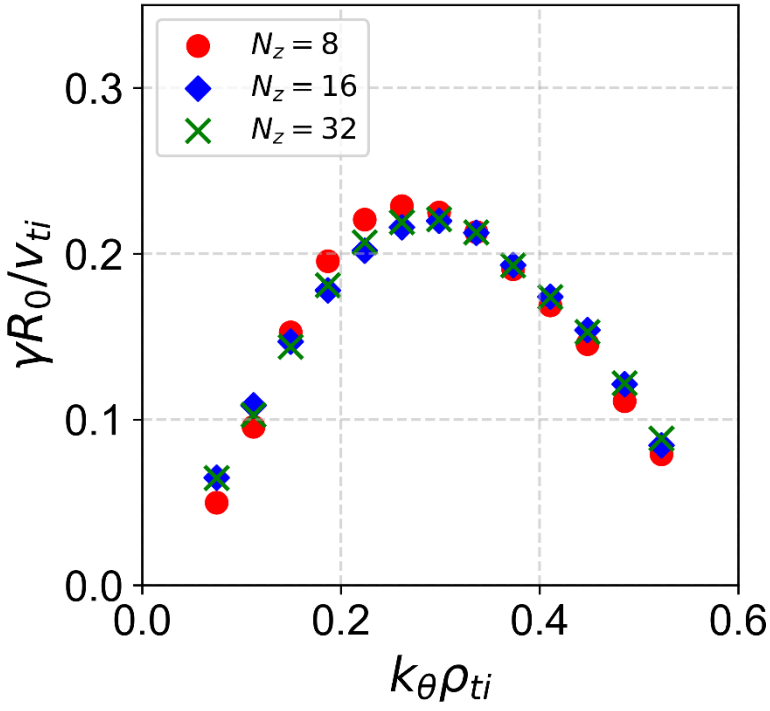
*Since $q = 1.4$ at $r = 75\rho_{ti}$, it coincides with $m/n = 28/20$.

Linear benchmark test - 2

Dispersion relation of ITG mode (toroidal)



Dispersion relation of ITG mode (aligned)

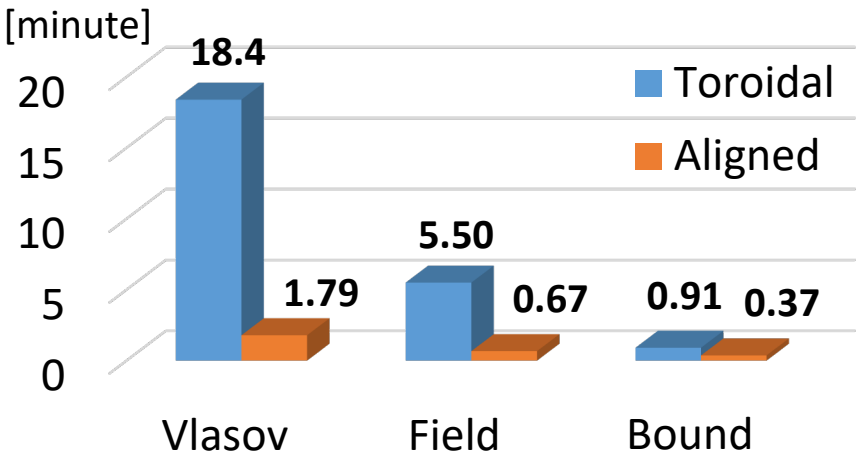


L_x	L_y	L_z	L_v	L_μ	N_x	N_y	$N_{v_\parallel}$	N_μ
135	2π	2π	10	12.5	128	216	80	16

✓ In the standard positive magnetic shar case around $\hat{s} = 0.78$, $N_z = 16$ is enough for the convergence in the aligned version, while $N_z = 128$ is required in the toroidal version.

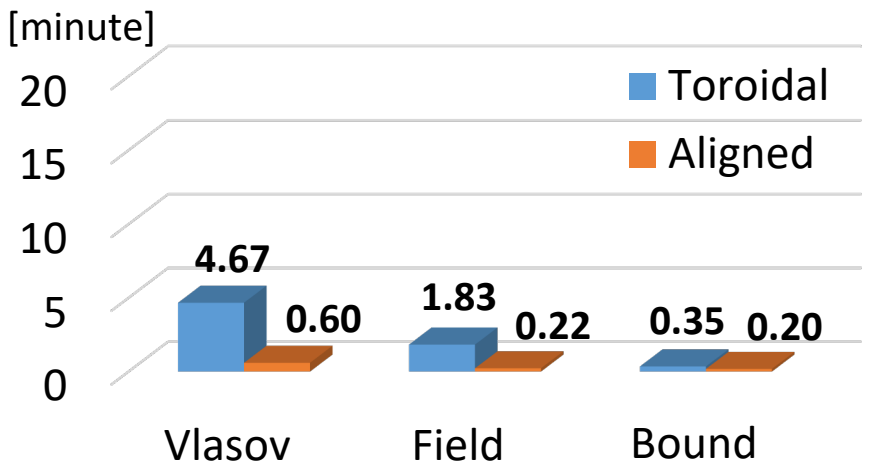
Linear benchmark test - 3

Computation time of the toroidal and aligned versions (256[core] on JFRS-1)



N_x	N_y	N_z	$N_{v_\parallel}$	N_μ	N_t
128	128	128	80	16	1600

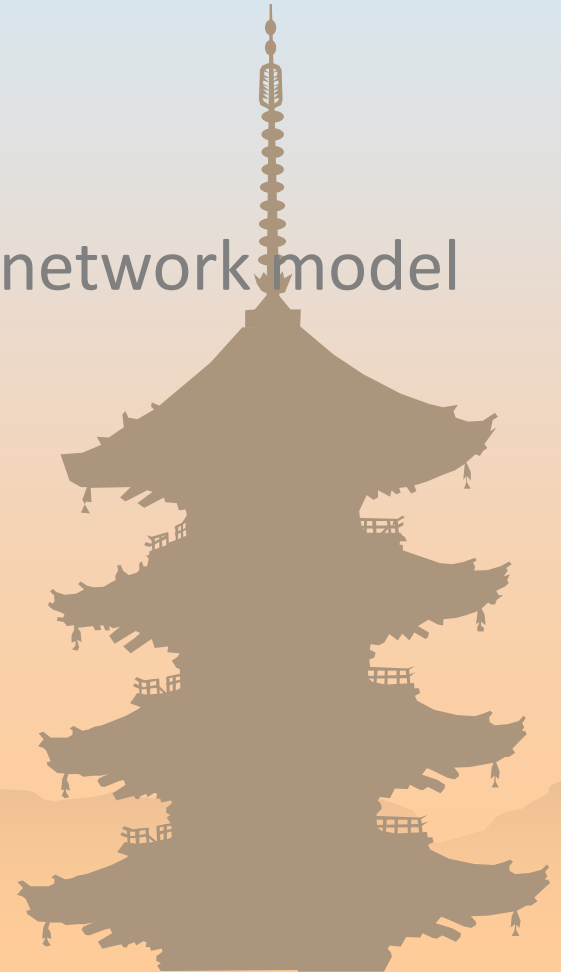
Computation time of the toroidal and aligned versions (1024[core] on JFRS-1)



N_x	N_y	N_z	$N_{v_\parallel}$	N_μ	N_t
128	128	16	80	16	800

- ✓ Total computation time is reduced by $2.83[min]/24.81[min]\sim0.11$ in 256 cores, while $1.02[min]/6.85[min]\sim0.15$ in 1024 cores.
- ✓ However, the scaling of the aliened version is worse than that of the toroidal one. Especially, the boundary setting which consists of 1D FFT and MPI_ISEND/I_RECV possibly becomes the bottleneck in larger-scale simulations.
  → Optimization of MPI domain decomposition & Hybrid MPI-OpenMP parallelization

Contents



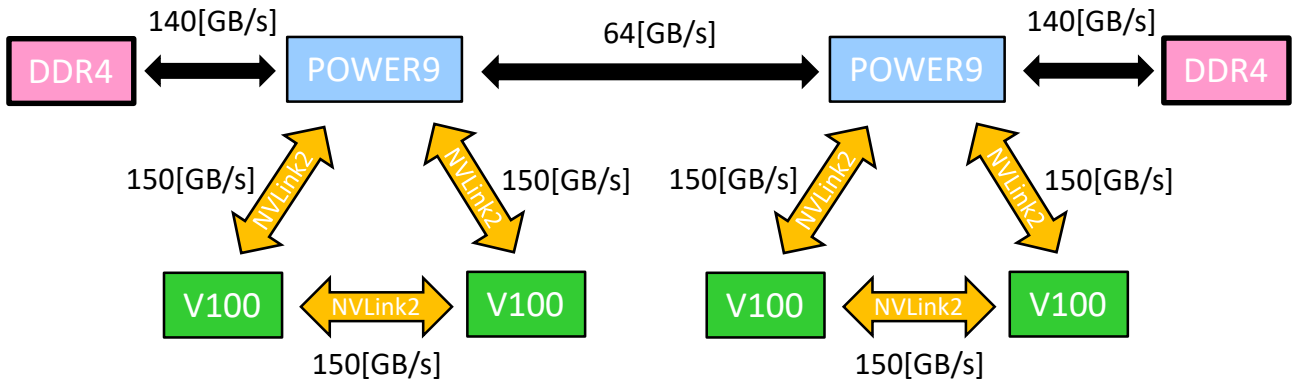
GPU acceleration by OpenACC - 1

- ✓ To improve the calculation speed and the scaling, we also introduced the OpenACC directives which enables us to utilize GPU parallelization.
- ✓ Then we benchmarked the efficiency of MPI+OpenACC parallelization on MARCONI 100 (CINECA, Italy).

MARCONI 100 at CINECA (Italy)

CPU	IBM POWER9 AC922 (3.1[GHz], 16[core]) × 2
GPU	NVIDIA Volta V100 × 4, NVLink v2.0
Node performance	32.653 [Tflops]
Node memory size	256[GB] (16GB DDR4 DIMM × 16)

(18th in TOP500)



[https://www.hpc.cineca.it/]

- ✓ Node performance is 10 times faster than that of JFRS-1.
- ✓ The memory band width is relatively wider between GPU-GPU.

GPU acceleration by OpenACC - 2

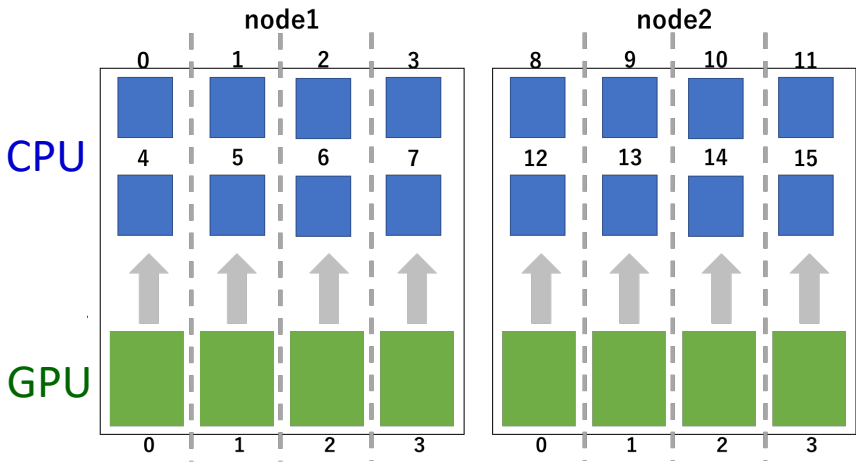
(1) Vlasov: loop collapse

```
def = acc_get_num_devices
(acc_devise_nvidia)
gpuid = mod(rank, def)
call acc_set_device_num(gpuid,
acc_device_defaut)

!$acc data copy(…) &
!$acc& copyin(...) &
!$acc& create(…)

!$acc wait
!$acc kernels
!$acc loop collapse(4) gang vector
DO x_i = 3, N_x_p+2
  DO y_i = 3, N_y_p+2
    DO z_i = 3, N_z_p+2
      DO v_i = 3, N_v+2
        DO u_i = 3, N_u+3
          Heavy calculation
        END DO
      END DO
    END DO
  END DO
END DO
!$acc end kernels
```

Image of GPU distribution to CPUs



- ✓ In the Vlasov part, the most heavy 5D loops ($\sim 10^8$ times) are collapsed to one loop and then distributed to each GPU.
- ✓ Each CPU is explicitly linked to the GPU in same node.
- ✓ The OpenACC data directives (copy, copyin, etc.) are utilized for CPU-GPU data transfer.

GPU acceleration by OpenACC - 3

(2) Collision: calculation and communication hiding by asynchronous optimization

```
!$acc kernels async(0)
!$acc loop collapse(3) gang vector
DO x_i = 3, N_x_p+2
 DO y_i = 3, N_y_p+2
  DO z_i = 3, N_z_p+2
  !$acc loop seq
   DO v_i = 4, N_v+3
    DO u_i = 3, N_u+3
     moment_local(z_i, y_i, x_i, 0) = ・・・
   END DO
  END DO
 END DO
END DO
!$acc end kernels
!$acc update self(moment_local(:, :, :, 0)) async(0)

!$acc kernels async(1)
!$acc loop collapse(3) gang vector
DO x_i = 3, N_x_p+2
 DO y_i = 3, N_y_p+2
  DO z_i = 3, N_z_p+2
  !$acc loop seq
   DO v_i = 4, N_v+3
    DO u_i = 3, N_u+3
     moment_local(z_i, y_i, x_i, 1) = ・・・
   END DO
  END DO
 END DO
END DO
!$acc end kernels
!$acc update self(moment_local(:, :, :, 1)) async(1)
```

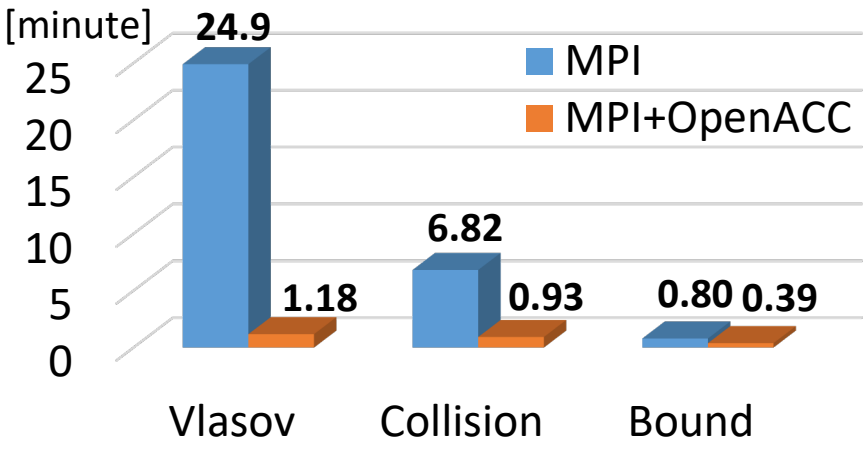
```
!$acc kernels async(2)
!$acc loop collapse(3) gang vector
DO x_i = 3, N_x_p+2
 DO y_i = 3, N_y_p+2
  DO z_i = 3, N_z_p+2
  !$acc loop seq
   DO v_i = 4, N_v+3
    DO u_i = 3, N_u+3
     moment_local(z_i, y_i, x_i, 2) = ・・・
   END DO
  END DO
 END DO
END DO
!$acc end kernels
!$acc update self(moment_local(:, :, :, 2)) async(2)

!$acc wait(0)
CALL MPI_ALLREDUCE(moment_local(:, :, :, 0))
!$acc wait(1)
CALL MPI_ALLREDUCE(moment_local(:, :, :, 1))
!$acc wait(2)
CALL MPI_ALLREDUCE(moment_local(:, :, :, 2))
```

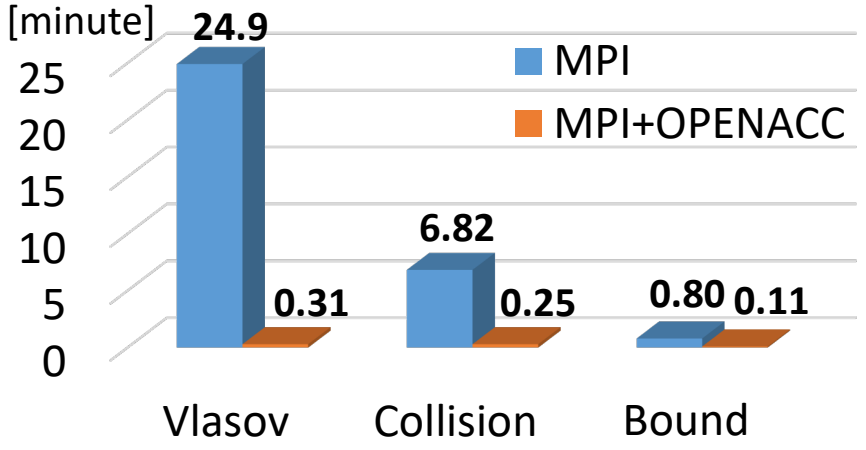
- ✓ By using the fact that “moment_local” is independent with each other, the asynchronous execution is utilized to hide the calculation and communication. (Same technique is also applied to boundary data communication)

GPU acceleration by OpenACC - 4

Computation time for the time-integration of f with 16 nodes(256[core], 16[GPU])

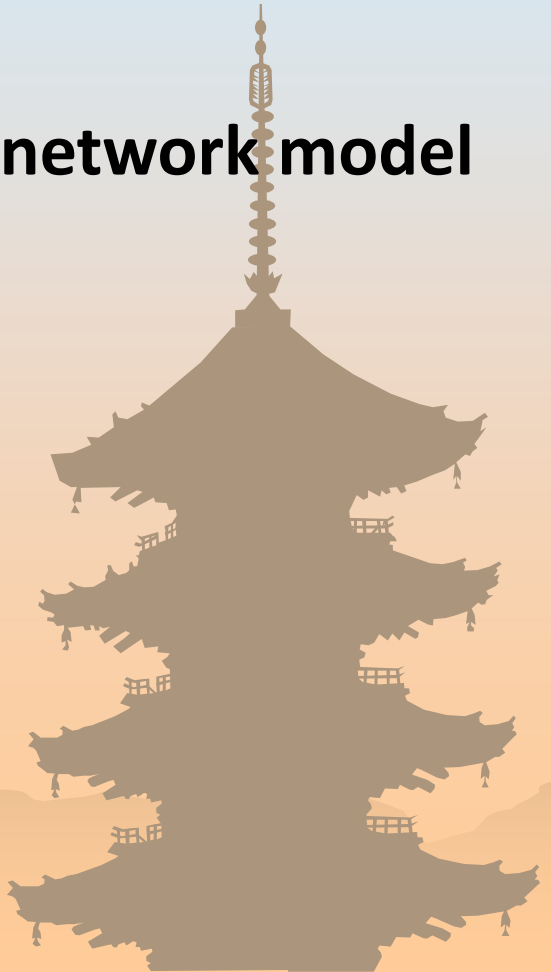


Computation time for the time-integration of f with 16 nodes(256[core], 64[GPU])



- ✓ By using 1[GPU] on each 1[node], the calculation is accelerated by 13 times (left).
- ✓ By using 4[GPU] on each 1[node] (the maximum number on MARCONI 100), the accelerated rate becomes 48 times (right).
- ✓ However, FFT part is still under the development.

Contents



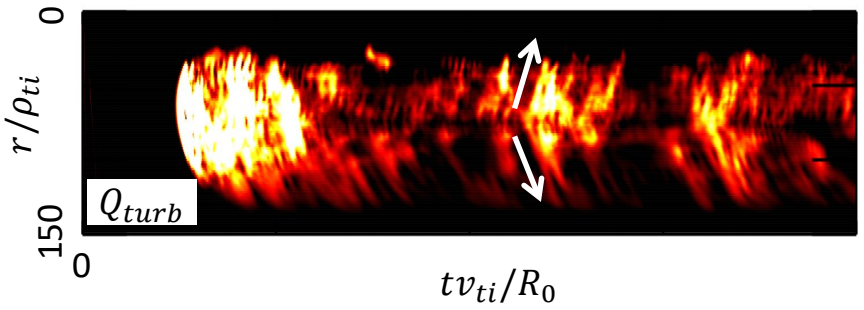
Background: "Globality" of turbulent transport - 1

Global turbulent transport in flux-driven ITG simulation

- ✓ In flux-driven simulation based on full-f gyrokinetic model, we often observe global turbulent transport such as avalanches and burst phenomenon.

  [Hahm and Diamond, JKPS-2018]

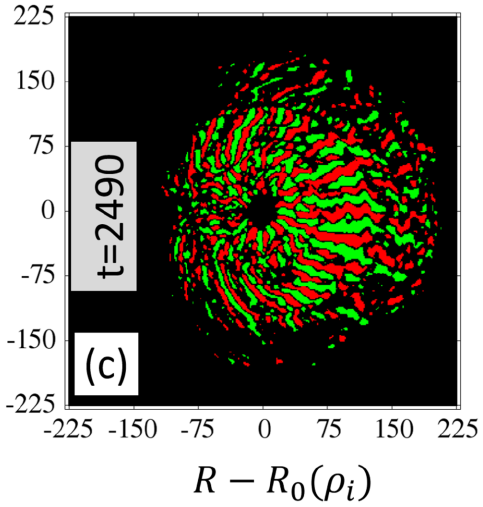
Time-spatial evolution of turbulent heat flux in GKNET simulation

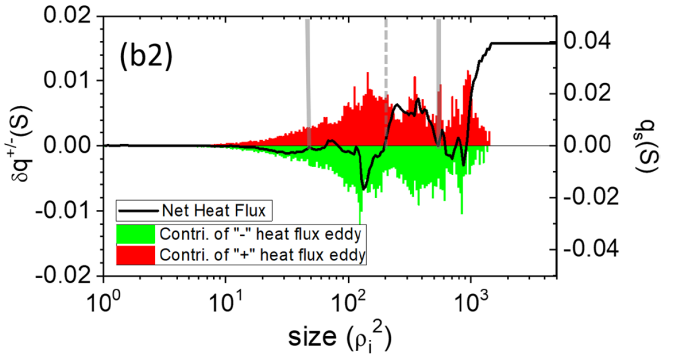


[Imadera, IAEA-2014]

- ✓ In our GKNET simulations, we identified that radially extended structures can drive the global burst of turbulent transport.

Transport is determined locally or globally?





[Wang, NF-2020]

Background: "Globality" of turbulent transport - 2

Evaluation of the Kernel of turbulent heat transport coefficient

Local model (real space)

$$Q(r) = \chi(r)\nabla T(r)$$

Global model (real space)

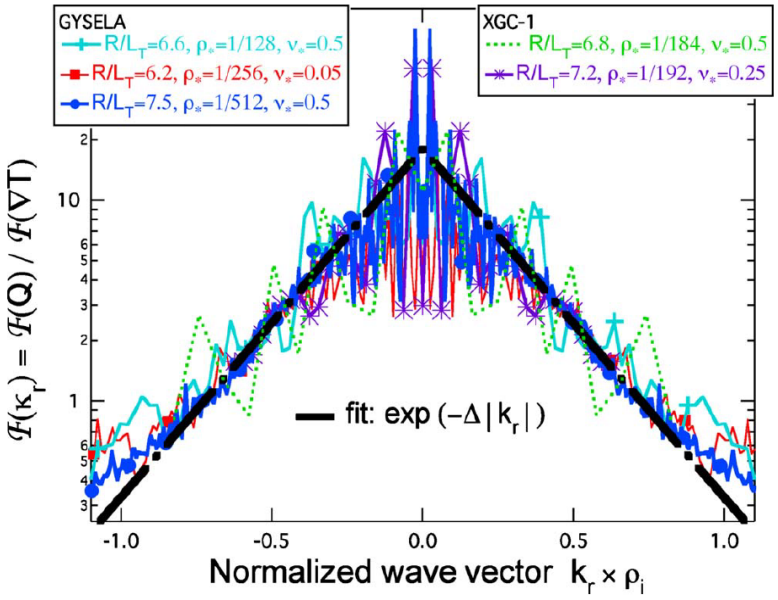
$$Q(r) = \int \chi(r,\bar{r})\nabla T(\bar{r})d\bar{r}$$

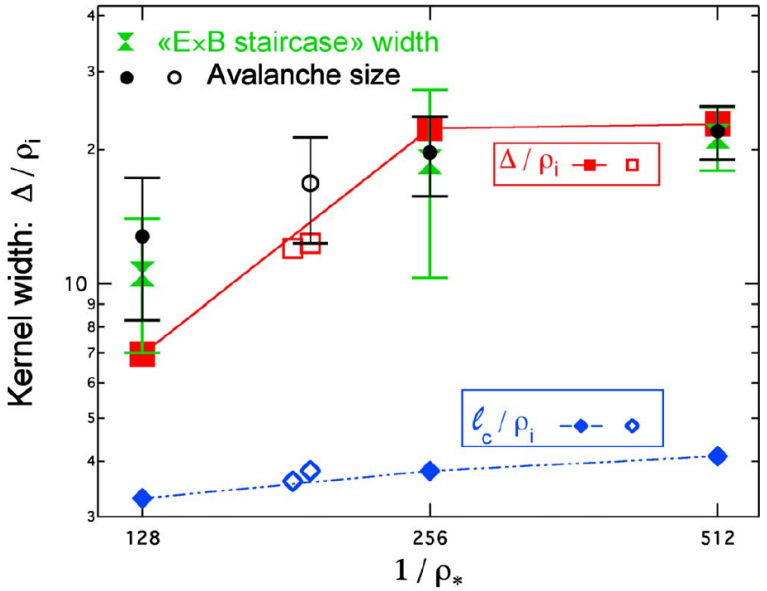
convolution

Global model (k space)

$$\hat{Q}(k) = \hat{\chi}(k)\,\widehat{\nabla T}\,(k)$$

[Dif-Pradalier, PRE-2010]





✓ According to the GYSELA simulations, the typical scale length of turbulent heat transport coefficient is evaluated as $10\rho_{ti}$~$20\rho_{ti}$, which is longer than the correlation length of turbulence ($3\rho_{ti}$~$4\rho_{ti}$).

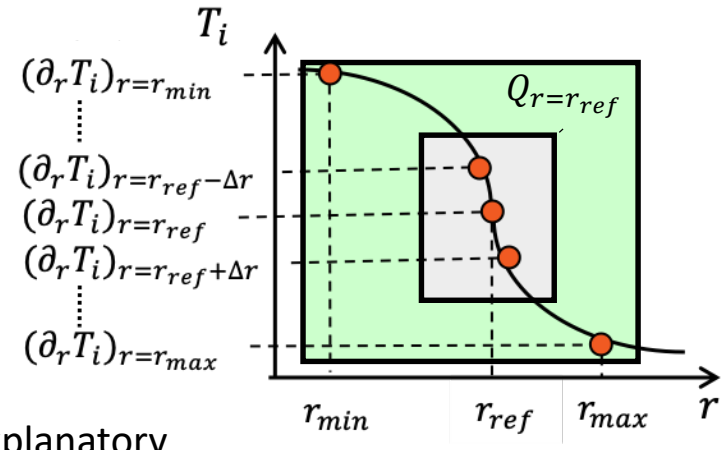
Purpose of this work

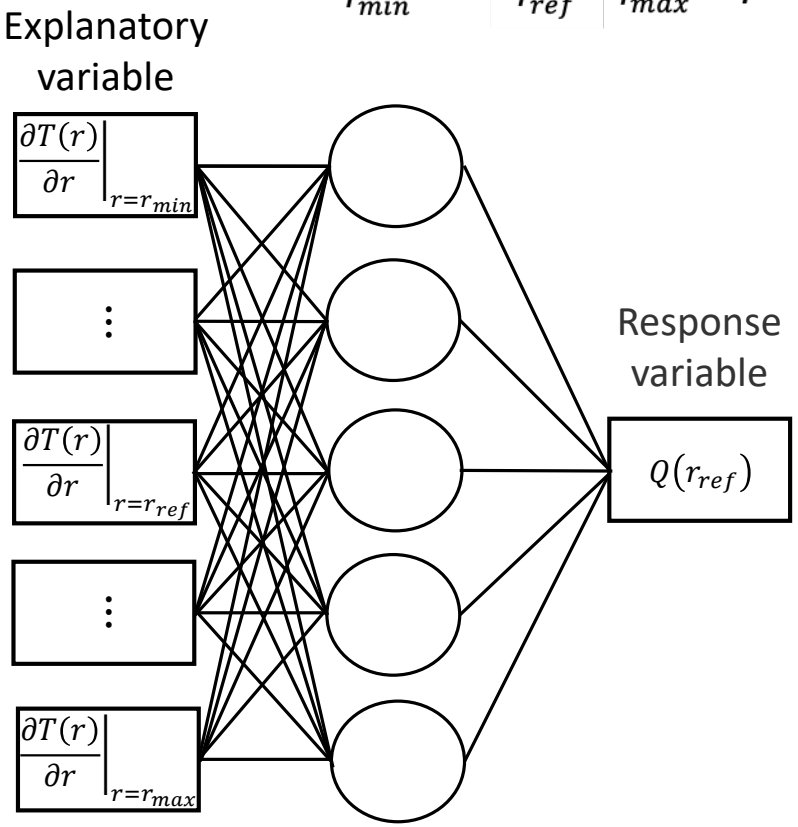
Purpose of this work

$$Q(r_{ref}) = \int \chi(r_{ref}, r)\frac{\partial T(r)}{\partial r}dr \cong \sum_{n=1}^{N} w_n \frac{\partial T(r)}{\partial r}\Big|_{r=r_n}$$

- ✓ By setting the temperature gradients at each radius as explanatory variable and heat flux as response variable, we have developed the neural network model. -> Virtual global transport model in real space
- ✓ Based on this model, we evaluated the typical scale length of heat transport by utilizing Accumulation Local Effect (ALE).

[Daniel, J. Roy. Stat. Soc.-2020]





Accumulation Local Effect

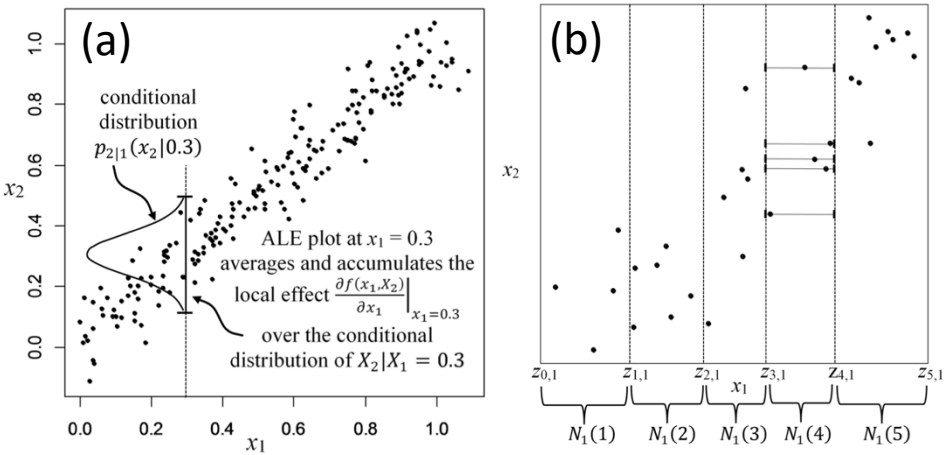
Accumulation Local Effect [Daniel, J. Roy. Stat. Soc.-2020]

✓ Even if the variables have strong correlation with each other, this method can extract the linear relationship between explanatory and response ones.

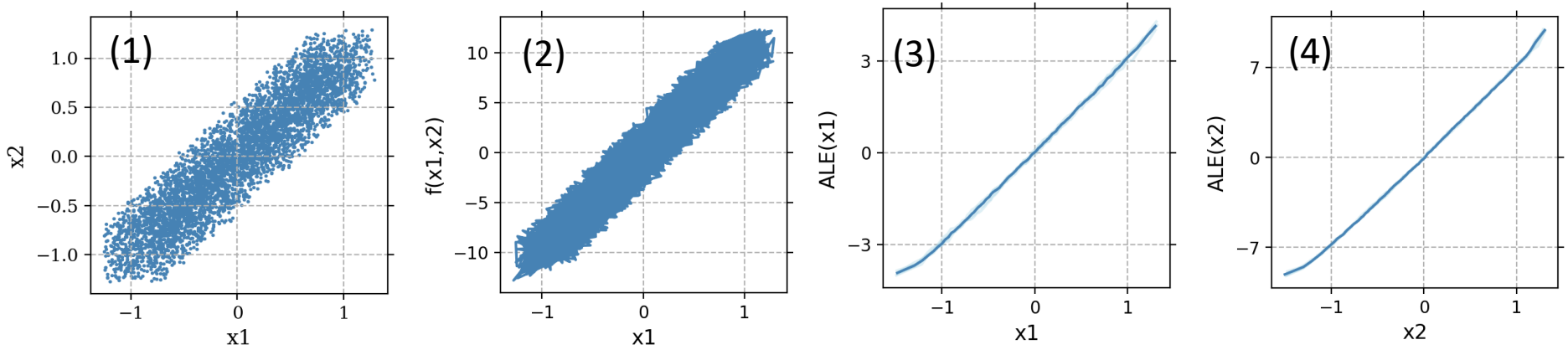
$$\hat{f}_{x_s,ALE}(x_s) = \int_{z_{0,1}}^{x_s} \int_{x_c} \frac{\delta \hat{f}(x_s, x_c)}{\delta x_s} P(x_c|z_s) dx_c dz_s - const$$

(f : developed NN model, x_s : target, x_c : the others)

Calculation step of ALE



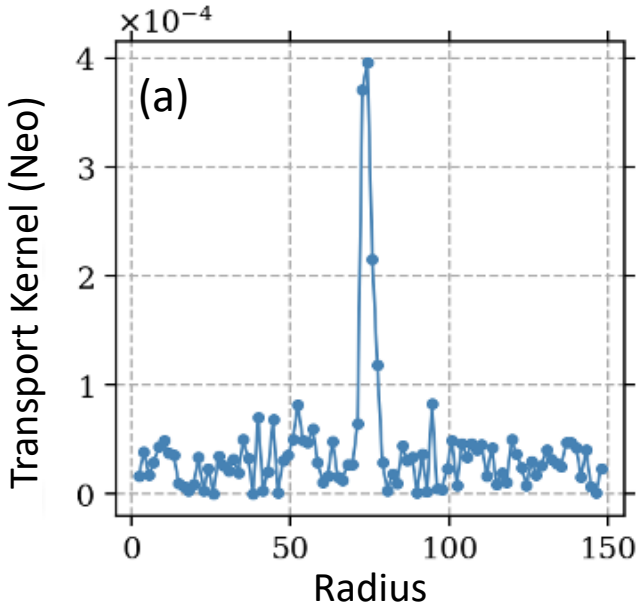
Ex. $f(x_1, x_2) = 3x_1 + 7x_2$

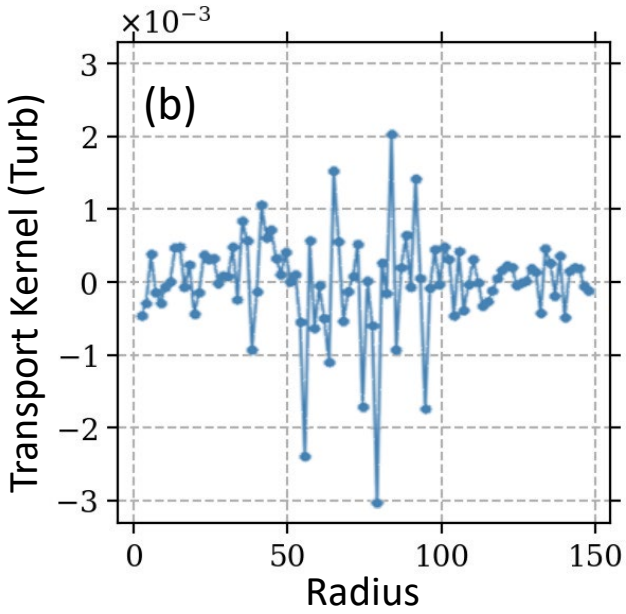


Evaluation of heat transport kernel - 1

Evaluation of heat transport kernel by ALE

✓ By applying ALE to our global neural network model, we calculated the heat transport kernel for $Q_i(r = 0.5a_0)$.





✓ Neoclassical heat transport kernel becomes δ function-like. -> Transport is local.

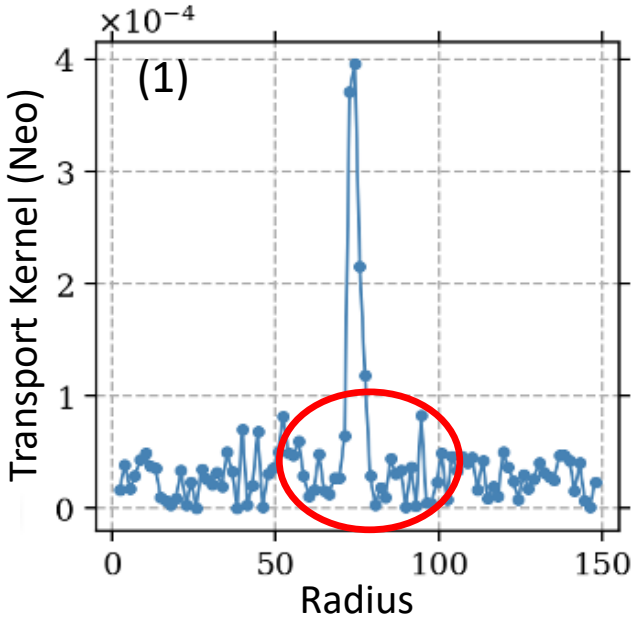
✓ Turbulent heat transport kernel shows no clear correlation.

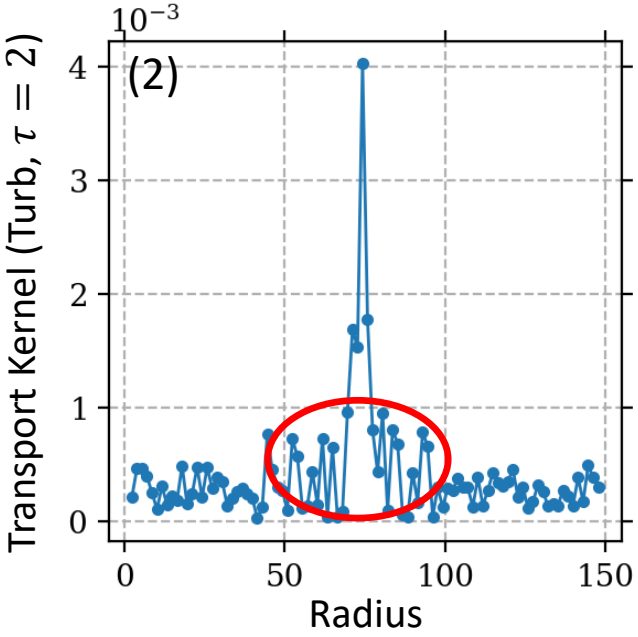
$$Q(r) = \int \chi_r(r')\delta(r' - r)\partial_r T(r, r')dr' = \chi_r(r)\partial_r T(r)$$

Evaluation of heat transport kernel - 2

Evaluation of heat transport kernel by ALE with a finite time delay

- ✓ We re-calculated the heat transport kernel by considering a finite time delay like $Q(r,t) = \int \chi(r,r')\partial_r T(r', t-\tau)dr'$.
- ✓ As the result, we found that temperature gradient and turbulent transport have a strong correlation in the case with $\tau = 2R_0/v_{ti}$.





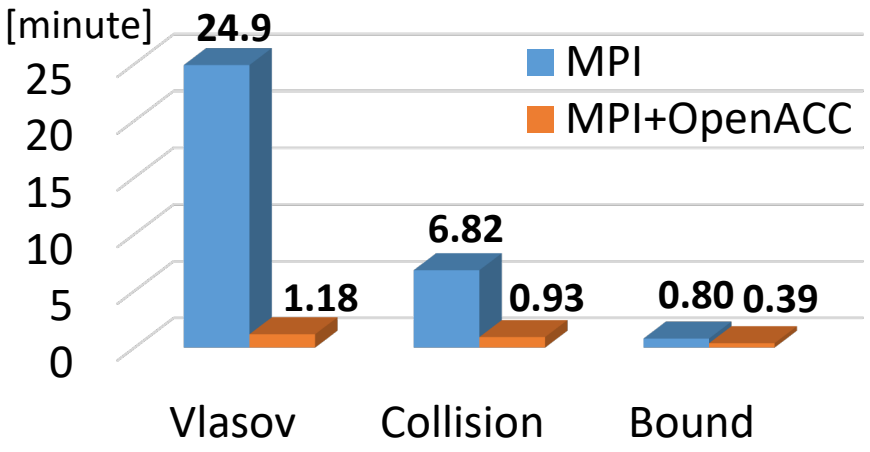
- ✓ Turbulent heat transport kernel with a finite time delay is also localized but its typical scale length is relatively longer.

Summary

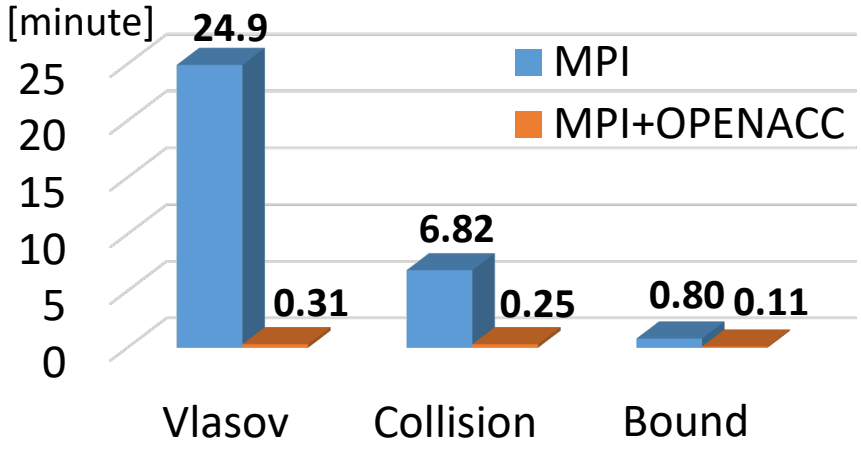
Summary

- ✓ By introducing field aligned coordinate, the grid number is reduced by $1/8$ and the resultant calculation time also becomes $1/8$.
- ✓ By utilizing OpenACC directives, the calculation speed to time-integrate the distribution function is accelerated by 48 times (below).
- ✓ By utilizing the machine learning, we found that turbulent heat transport kernel with a finite time delay $\tau = 2R_0/v_{ti}$ shows relatively longer correlation.

Computation time for the time-integration of f with 16 nodes(256[core], 16[GPU])



Computation time for the time-integration of f with 16 nodes(256[core], 64[GPU])



Future Plans (related to GPU acceleration)

(1) Direct data transfer between GPU-GPU

- ✓ Now we are considering MPI communication without backing the date to host by using "CUDA_aware_MPI" (almost done).
- ✓ In addition, we are introducing "acc host_data" instead of "acc_update" for direct data transfer between GPU-GPU (on going).

(2) GPU Acceleration to Python program for neural network model

- ✓ We are trying GPU acceleration to the Python program for neural network model. But the acceleration rate is still low. (Problem size? The type of NN model?)