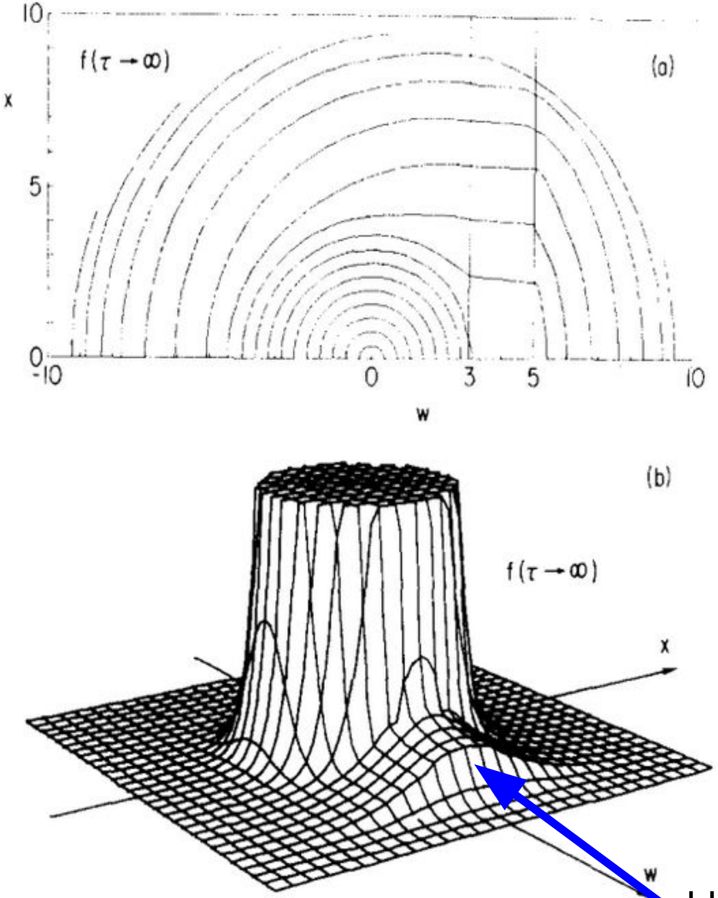
Building a Fokker-Planck Solver using the MFEM Finite Element Library

Ben Antognetti (Univ. New Hampshire)
Syun'ichi Shiraiwa, Nicola Bertelli (PPPL)

Motivation and Goal

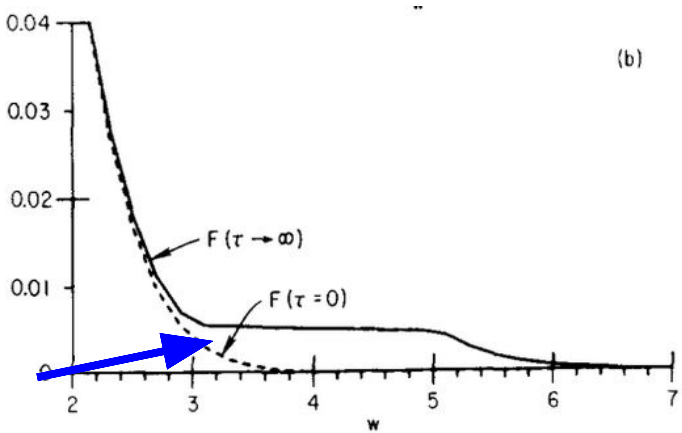
Evaluating current drive requires solving the Fokker-planck (FP) equation for the distribution function

In the tokamak integrated simulation, it is required to solve FP equation many times, and an efficient FP solver is desirable



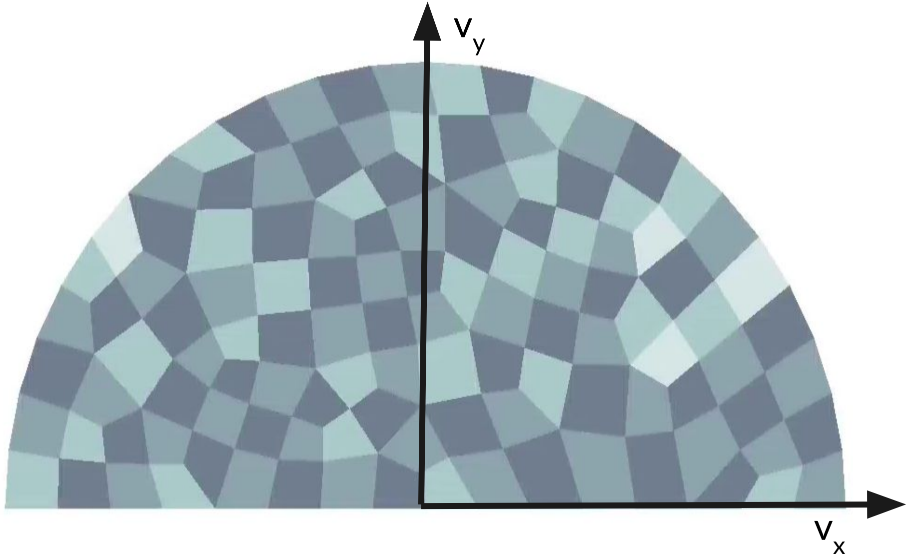
C.F.F. Karney and N.J. Fisch (1979)

LH driven current



MFEM

- C++ finite element method library developed at LLNL
  - Assembles the FEM linear system using the weak form
  - Supports for parallelization
  - Released the initial GPU implementation in 2019



2D quadrilateral mesh made using PiScope

In this study, we focus on the non-relativistic FP equation in the uniform plasma and develop a FP solver on GPU

FP Equation in the Cartesian coordinate system

- FP equation is usually given in the spherical coordinate system. In order to avoid multiplicity at v=0, we transform it to the cartesian coords.
- Apply chain rule to convert partials
- Include Jacobian and change of infinitesimal operator

$$\left(\frac{\partial f_a}{\partial t}\right)_c^b = \frac{\Gamma_a}{v^2}\left(\frac{\partial}{\partial v}\left(A_a^b f_a + B_a^b \frac{\partial f_a}{\partial v}\right) + \underbrace{\frac{1}{\sin\theta}\frac{\partial}{\partial \theta}\left(F_a^b \frac{\partial f_a}{\partial \theta}\right)}_{\text{Collision operator}}\right)$$

Chain Rule +

$$v_x = v\cos\theta$$

$$v_y = v\sin\theta$$

$$\left(\frac{\partial f_a}{\partial t}\right)_c^b = c f_a + \vec{d}\cdot\nabla f_a + \nabla^T\cdot(E\nabla f_a)$$

We used A, B, and F in Eq. 2.4.20 in Killeen, Kerbel, McCoy, and Mirin (1986)

Implementation using MFEM

- Add terms with integrators
  - MFEM team (Dylan) added the matrix coefficient support (PR1665)
- Application of boundary conditions
  - f = 1 at $v = v_{max}$ (essential BC)
  - $df/dv_y = 0$ at $v_y=0$ (natural BC)
- Looking at Steady State Solution
- Nested Krylov
  - Outer FGMRES Solver
  - Inner GMRES Solver as a pre-conditioner

MFEM Features Examples Documentation Gallery Download

Square Operators

These integrators are designed to be used with the BilinearForm object to assemble square linear operators.

Class Name	Spaces	Coef.	Operator	Continuous Op.	Dimension
MassIntegrator	H1, L2	S	$(\lambda u, v)$	λu	1D, 2D, 3D
DiffusionIntegrator	H1	S, M	$(\lambda \nabla u, \nabla v)$	$-\nabla \cdot (\lambda \nabla u)$	1D, 2D, 3D

MFEM Features Examples Documentation Gallery Download

Other Scalar Operators

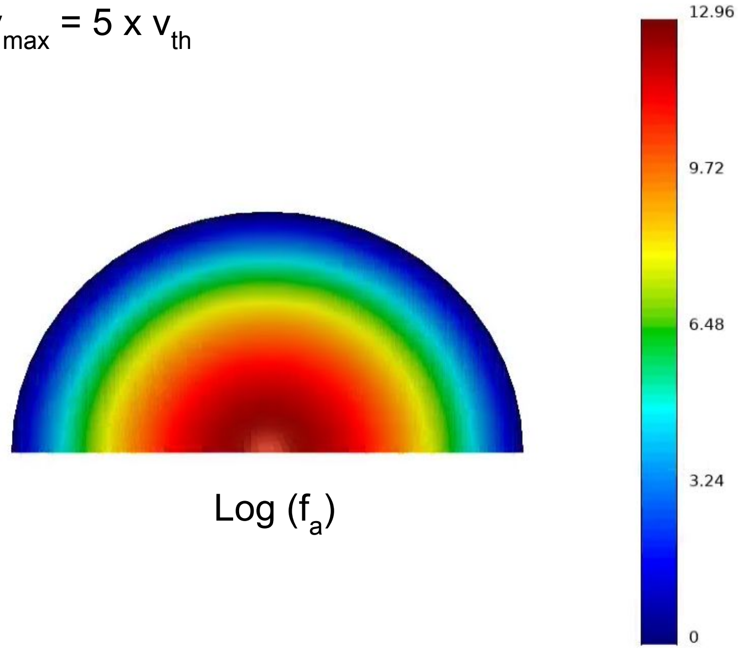
Class Name	Domain	Range	Coef.	Dimension	Operator	Notes
DerivativeIntegrator	H1, L2	H1, L2	S	1D, 2D, 3D	$(\lambda \frac{\partial u}{\partial x_i}, v)$	The direction i See MixedDirecti for a more gen
ConvectionIntegrator	H1	H1	V	1D, 2D, 3D	$(\vec{\lambda} \cdot \nabla u, v)$	This is designe BilinearForm See MixedDirecti for a rectangul
GroupConvectionIntegrator	H1	H1	V	1D, 2D, 3D	$(\alpha\vec{\lambda} \cdot \nabla u, v)$	Uses the "grou formulation fo
BoundaryMassIntegrator	H1, L2	H1, L2	S	1D, 2D, 3D	$(\lambda\, u, v)$	Computes a m faces of a dom above for a mc

https://mfem.org/bilininteg/

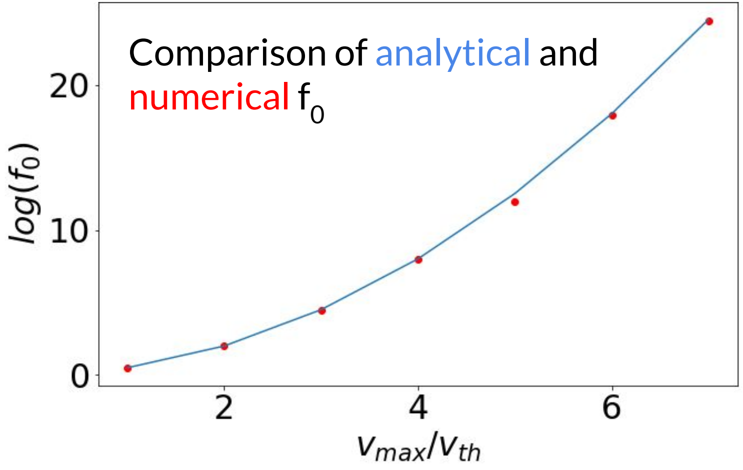
Solution of Base FP Equation

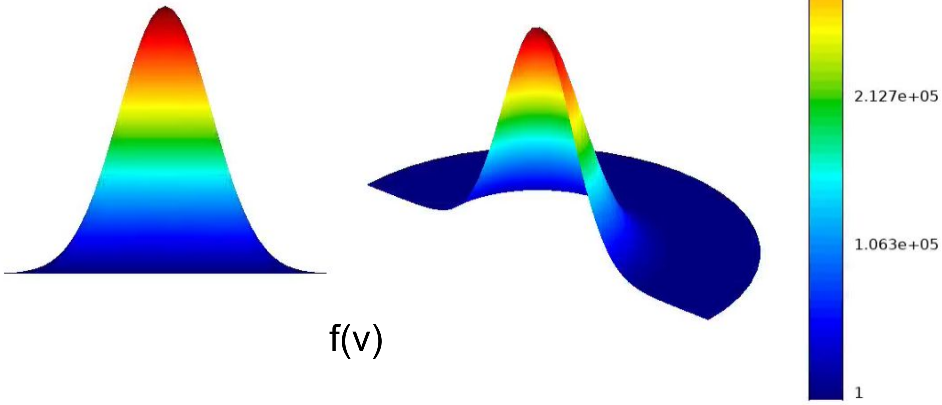
$$\left(\frac{\partial f_a}{\partial t}\right)_c^b = cf_a + \vec{d}\cdot\nabla f_a + \nabla^T\cdot(E\nabla f_a)$$

$v_{max} = 5 \times v_{th}$



Log (f_a)



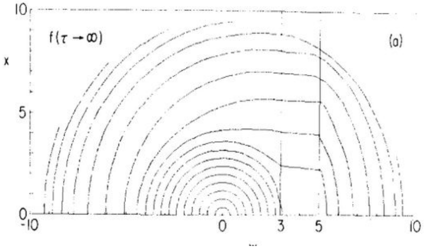


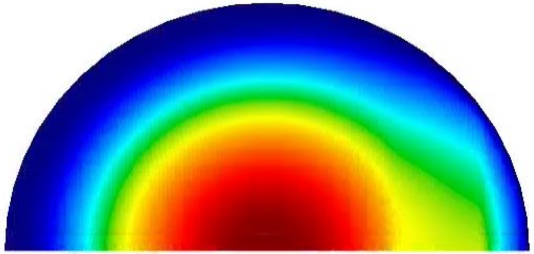
f(v)

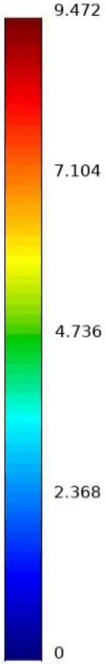
Distribution looks to follow a Maxwellian as expected

Solution of FP Equation with Lower Hybrid Term

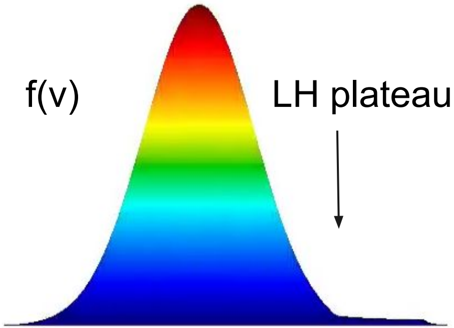
$$\left(\frac{\partial f_a}{\partial t}\right)_c^b = c f_a + \vec{d} \cdot \nabla f_a + \nabla^T \cdot (E \nabla f_a) + h(v_x) \frac{\partial^2 f_a}{\partial v_x^2}$$

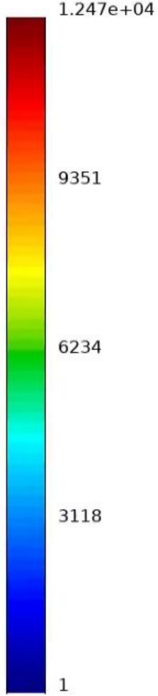






Log (f)





Comparison of Serial/Parallel and CPU/GPU

$v_{max}/v_{th} = 5$

P = element order
H = mesh refinement count

		P=2 H=1	P=2 H=2	P=2 H=3	P=3 H=1	P =3 H = 2	P =3 H = 3
Serial	CPU	3.468s	13.49s	53.52s	9.044s	35.43s	150.5s
	GPU	13.88s	5.081s	8.993s	14.41s	13.50s	7.005s
Parallel (np=2)	CPU	7.677s	126.3s	>1h	20.42s	401.8s	>1h
	GPU	115.4s	139.2s	>1h	115.8s	171.6s	>1h

Average run times for 5 runs

On PPPL TRAVERSE cluster

- In serial code, GPU out-performs CPU by a large degree.
- Working on understanding parallel performance

Summary

Implemented a 2D Fokker-Planck solver for non-relativistic uniform plasmas

- Implemented using the Cartesian coordinate system
- Added LH terms as an auxiliary diffusion
- Compared performance with different configuration and problem size
- Will compare current drive efficiency with the Karney(1979) paper

Issues to use GPUs from MFEM

- Mesh needs to be quadrilaterals
- GPU does not have many preconditioner options yet
- Need to understand performance in parallel