Eulerian Algorithms for the Discretization of (Plasma) Kinetic Equations

James Juno[1], Jason TenBarge(advisor)[2], Ammar Hakim(advisor)[2], William Dorland(advisor)[1]

Association for Computing Machinery Supercomputing Conference 2018

1: University of Maryland College Park, 2: Princeton Plasma Physics Lab

Vlasov-Maxwell System

- Weakly collisional plasmas are modeled by the Vlasov-Maxwell system

$$\frac{\partial f_s}{\partial t} + \nabla_{\mathbf{z}} \cdot (\boldsymbol{\alpha}_s f_s) = C[f]$$

$$\nabla_{\mathbf{z}} = (\nabla_{\mathbf{x}}, \nabla_{\mathbf{v}}), \quad \boldsymbol{\alpha}_s = \left(\mathbf{v}, \frac{q_s}{m_s}(\mathbf{E} + \mathbf{v} \times \mathbf{B})\right)$$

$$\frac{\partial \mathbf{B}}{\partial t} + \nabla \times \mathbf{E} = 0, \quad \nabla \cdot \mathbf{B} = 0,$$

$$\epsilon_0 \mu_0 \frac{\partial \mathbf{E}}{\partial t} - \nabla \times \mathbf{B} = -\mu_0 \mathbf{J}, \quad \nabla \cdot \mathbf{E} = \frac{\varrho_c}{\epsilon_0}$$

$$\rho_c = \sum_s q_s \int_{-\infty}^{\infty} f_s \, d\mathbf{v}, \quad \mathbf{J} = \sum_s q_s \int_{-\infty}^{\infty} \mathbf{v} f_s \, d\mathbf{v}$$

Discontinuous Galerkin Method

- Discontinuous Galerkin (DG) algorithms combine power of finite element methods (high-order accuracy, geometry), with advantages of finite volume methods (limiters, locality of data)
- Vlasov spatial discretization

$$\int_{K_j} w \frac{\partial f_h}{\partial t} \, d\mathbf{z} + \oint_{\partial K_j} w^- \mathbf{n} \cdot \hat{\mathbf{F}} \, dS - \int_{K_j} \nabla_z w \cdot \alpha_h f_h \, d\mathbf{z} = 0$$

- And for Maxwell's equations

$$\int_{\Omega_j} \varphi \frac{\partial \mathbf{B}_h}{\partial t} \, d\mathbf{x} + \oint_{\partial \Omega_j} d\mathbf{s} \times (\varphi^- \hat{\mathbf{E}}_h) - \int_{\Omega_j} \nabla \varphi \times \mathbf{E}_h \, d\mathbf{x} = 0$$

$$\epsilon_0 \mu_0 \int_{\Omega_j} \varphi \frac{\partial \mathbf{E}_h}{\partial t} \, d\mathbf{x} - \oint_{\partial \Omega_j} d\mathbf{s} \times (\varphi^- \hat{\mathbf{B}}_h) + \int_{\Omega_j} \nabla \varphi \times \mathbf{B}_h \, d\mathbf{x} = -\mu_0 \int_{\Omega_j} \varphi \mathbf{J}_h \, d\mathbf{x}$$

- We solve the Riemann problem for the surface terms with upwinding. For Maxwell's equations this looks schematically like

$$\hat{E}_2 = [\![E_2]\!] - c\{B_3\} \quad \hat{E}_3 = [\![E_3]\!] + c\{B_2\}$$

$$\hat{B}_2 = [\![B_2]\!] + \{E_3\}/c \quad \hat{B}_3 = [\![B_3]\!] - \{E_2\}/c$$

$$\{g\} \equiv (g^+ - g^-)/2 \quad [\![g]\!] \equiv (g^+ + g^-)/2$$

- For the Vlasov equation we use a combination of upwinding and a relaxed global Lax-Friedrichs flux

$$\mathbf{n} \cdot \hat{\mathbf{F}} = \frac{1}{2}\mathbf{n} \cdot \left(\boldsymbol{\alpha}_h^+ (f_h^+ + f_h^-) - \boldsymbol{\tau}(f^+ - f^-)\right)$$

$$\text{where } \tau = \max_{\mathcal{T}} \left(\frac{q}{m}\mathbf{E}_h + \frac{q}{m}\mathbf{v} \times \mathbf{B}_h\right)$$

- **NOTE:** Conservation relations in a kinetic equation are implicit

$$\frac{d}{dt} \sum_j \sum_s \int_{K_j} \frac{1}{2} m |\mathbf{v}|^2 f_h \, d\mathbf{z} + \frac{d}{dt} \sum_j \int_{\Omega_j} \left(\frac{\epsilon_0}{2}|\mathbf{E}_h|^2 + \frac{1}{2\mu_0}|\mathbf{B}_h|^2\right) d^3\mathbf{x} = 0$$

- Algorithm can conserve implicit quantities like energy with proper choice of basis functions

Algorithmic Improvements: Pre-Generated Kernels

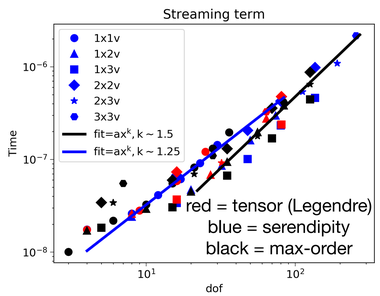
- **CRITICALLY,** since conservation relations implicit, we must evaluate the integrals in our algorithm exactly
  - **No aliasing errors!**
- Numerical quadrature scales poorly because equation is nonlinear → "Curse of dimensionality"

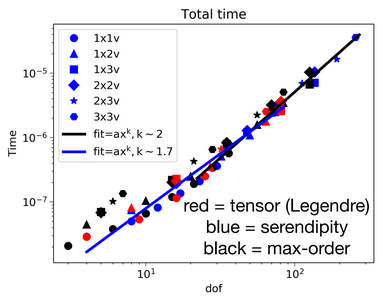
$$\int_{K_j} w \frac{\partial f_h}{\partial t} \, d\mathbf{z} + \oint_{\partial K_j} w^- \mathbf{n} \cdot \hat{\mathbf{F}} \, dS - \int_{K_j} \nabla_{\mathbf{z}} w \cdot \boldsymbol{\alpha}_h f_h \, d\mathbf{z} = 0 \quad \rightarrow \quad \mathcal{O}(N_q N_p)$$

	Base Polynomial Order	1	2	3	4	5
Dimension	$((3p+1)/2)^{X_{dim}} \times ((2p+2)/2)^3$					
1X3V		16	108	320	875	1728
2X3V		32	432	1600	6125	13824
3X3V		64	1728	8000	42875	110592

- Integrals can also be pre-computed, dominant operation becomes tensor-tensor multiplication
  - For example, volume force integral becomes $\mathcal{C}_{ijk} = \int_{K_l} \nabla_{\mathbf{v}} w_i(\mathbf{z}) \cdot \frac{q}{m} \left(\mathbf{E}_j(t)\phi_j(\mathbf{x}) + \mathbf{v} \times \mathbf{B}_j(t)\phi_j(\mathbf{x})\right) f_k(t) w_k(\mathbf{z}) \, d\mathbf{z}$
- If basis functions are orthonormal, resulting tensors are sparse!
  - We use a computer algebra system, Maxima, to evaluate these tensor-tensor multiplications

```
double VlasovVol1x1vTensor92(const double *w, const double *dxv, const double *EM, const double *f, double *out)
{
// w[NDIM]: Cell-center coordinates. dxv[NDIM]: Cell spacing. EM/f: Input EM-field/distribution function. out: Incremented output
  double dv0dx0 = dxv[1]/dxv[0];
  double wv0dx0 = w[1]/dxv[0];
  const double dv10 = 2/dxv[1];
  const double *E0 = &EM[0];
  const double dv1 = dxv[1], wv1 = w[1];

  double abar0[3];

  abar0[0] = E0[0];
  abar0[1] = E0[1];
  abar0[2] = E0[2];

  double incr1[9];

  for(unsigned int i=0; i<9; ++i){

    incr1[i]=0.0;

  };

  const double amid1 = 0.7071067811865475*abar0[0]-0.7905694150420947*abar0[2];
  incr1[2] = 1.224744871391589*abar0[2]*f[4]+1.224744871391589*abar0[1]*f[1]+1.224744871391589*abar0[0]*f[0];
  incr1[3] = abar0[1]*(1.895445511501832*f[6]+1.224744871391589*f[0])+1.095445115010332*f[1]*abar0[2]+1.224744871391589*abar0[0]*f[1];
  incr1[5] = 2.738612787525831*abar0[2]*f[6]+2.738612787525831*abar0[1]*f[3]+2.738612787525831*abar0[0]*f[2];
  incr1[6] = 1.224744871391589*abar0[0]*f[4]+abar0[2]*(0.7826087964395517*f[4]+1.224744871391589*f[0])+1.095445115010332*abar0[1]*f[1];
  incr1[7] = abar0[1]*(2.449489742783178*f[6]+2.738612787525831*f[2])+2.449489742783178*abar0[2]*f[3]+2.738612787525831*abar0[0]*f[3];
  incr1[8] = abar0[2]*(1.749635538559413*f[6]+2.738612787525831*f[2])+2.738612787525831*abar0[0]*f[6]+2.449489742783178*abar0[1]*f[3];

  out[1] += 3.464101615137754*f[0]*wv0dx0*f[2]+dv0dx0;
  out[2] += incr1[2]*dv10;
  out[3] += 3.464101615137754*f[2]*wv0dx0*incr1[3]+dv10*(0.8944271909999159*f[5]+f[0])*dv0dx0;
  out[4] += 7.745966692414834*f[1]*wv0dx0*2.236067977499790*f[3]*dv0dx0;
  out[5] += incr1[5]*dv10;
  out[6] += 7.745966692414834*f[3]*wv0dx0*incr1[6]*dv10+(2.0*f[7]+2.236067977499790*f[1])*dv0dx0;
  out[7] += 3.464101615137755*f[5]*wv0dx0*incr1[7]*dv10+0.8944271909999161*f[2]*dv0dx0;
  out[8] += 7.745966692414834*f[7]*wv0dx0*incr1[8]*dv10+2.0*f[3]*dv0dx0;
return std::abs(w0dx0)*dv0dx0/2+std::abs(amid1)/dv1;
}
```

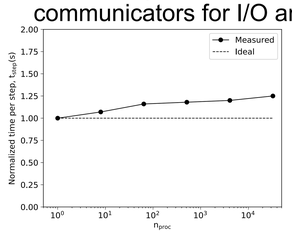


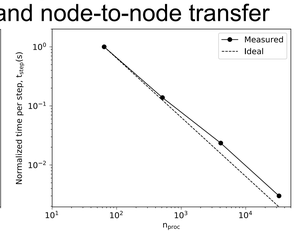


- **Key takeaways**
  - The tensor-tensor convolution is very sparse, ~300 multiplications with quadrature, ~70 now in 1X1V
  - Scaling shown is for **full** update, e.g. one 6D volume integral and 12 5D surface integrals
    - Other DG algorithms, such as those described in canonical textbooks (Hesthaven and Warburton 2008) have, at best, quadratic scaling in number of degrees and freedom, and that's with aliasing errors!
  - Scaling plots derived on **Intel Xeon Platinum 8160 ("Skylake") chips**,
  - Optimization flags: **"-O3,-axCORE-AVX512"**, Efficiency per node: **~10% of peak** (300 GFlops vs. 3 TFlops)

Algorithmic Improvements: MPI-3

- MPI-3 provides a means of an MPI-only implementation of multi-threaded parallelization
- Provides some advantages over MPI-OpenMP hybrid, such as reducing thread overhead/latency
- Current implementation only uses rank 0 of shared communicators for I/O and node-to-node transfer



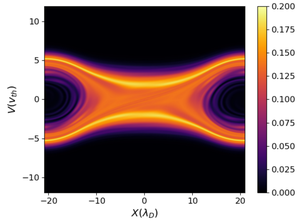


Summary and Future Work

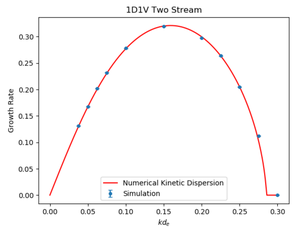
- Orthonormal basis functions make the tensor-tensor convolutions in a DG algorithm sparse, and unrolling them with Maxima produces well optimized code
- Further improvements in MPI-3 implementation: hyper-threading and better vectorization strategy
  - Also should explore multiple shared processes contributing to I/O and inter-node communication
- Application of this work extends beyond plasmas to general kinetic equations: Boltzmann, self-gravitating systems, neutrino transport
- Source code: https://bitbucket.org/ammarhakim/gkeyll/src/default/
- Documentation: https://gkyl.readthedocs.io/en/latest/

Benchmarks

- **Two stream instability**
  - Two counter-propagating beams are unstable to density perturbations
  - This instability grows an electric field in the drift direction and leads to strong particle trapping
  - Nonlinear phase characterized by phase space vortices

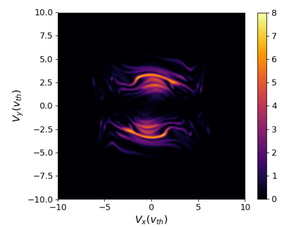


Nx = 96, Nv = 192,
p = 3, Serendipity

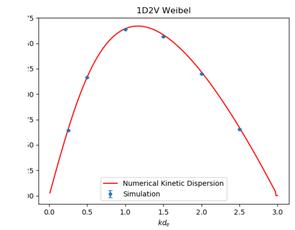


Nx = 16, Nv = 16,
p = 2, Serendipity

- **Weibel instability**
  - Explosive instability for growth of magnetic field in the presence of drifts

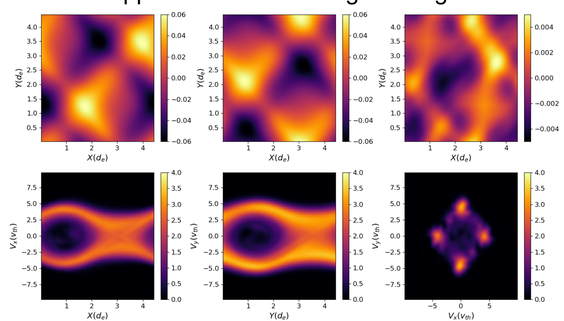


Nx = 96, Nvx, Nvy = 128,
p = 3, Serendipity



Nx = 16, Nvx, Nvy = 16,
p = 2, Serendipity

- **A combined two stream and Weibel instability**
  - The k vector associated with two stream is aligned with the drifts, Weibel is orthogonal
  - Spectrum of intermediate range of angles could have application to cosmological magnetic field



Acknowledgements: This work is supported by a NASA Earth and Space Science Fellowship, grant no. 80NSSC17K0428, and NSF Grant No. AGS-1338944