closures-2d

Tim Shaffer[1]

August 5, 2015

[1] Mentor: C. Kristopher Garrett

Goals

We are releasing software that tests angular approximations in kinetic transport simulations

- Based on code used for a previous publication (citation)
- Open source so that others can use/collaborate
- Makes it easy to implement new features
- Implements S_N, P_N, FP_N, M_N, PP_N, D_N (experimental)

[pictures]

Improvements

Algorithmic changes:

- Added experimental Lebedev quadrature
- Removed γ factor for ansatz correction in `momopt`
- Changed `momopt`'s regularization in case of bad condition number

Software-related improvements:

- Cross-platform build system
- Automated testing
- Better MPI communication
- Improved documentation
- Improved interface
- Bugfixes
- Profiling and optimization

Release

We are releasing this code as open source software.

Now uses SCons, a Python-based build system, rather than Makefiles

- Cross platform
- Sets up library search paths
- Intelligent compilation

Depends on:

- GSL
- BLAS
- LAPACK
- OpenMP (optional)
- Open MPI (optional)
- PAPI (optional)

Testing

We implemented several automated tests for the software. The testing code is written in Python and integrated with the build system.

The testing code carries out

- Regression tests – compare output to reference data included with the software
- Mass Conservation tests – check that total density remains the same
- Convergence tests – measure the effect of decreasing the cell size on the precision of the output

```
convergence tests:
 - moment(sspline)
 -- dx / 16                       35.922s
 -- dx / 8                         4.580s
 -- dx / 4                         0.603s          2.340759
 -- dx / 2                         0.092s          2.082828
 -- dx / 1                         0.022s          1.954163
```

Performance

Added timing and profiling-related measurement capabilities

Tested additional optimizations

- Spatial blocking (reduce scheduling overhead, cache misses)
- CPU pinning (ccNUMA)
- Memory alignment

The Importance of Caching

Blocking

What Are Kinetic Equations?

Macroscopic

- $\rho(\mathbf{x}, t)$ – Density
- $\mathbf{u}(\mathbf{x}, t)$ – Velocity
- $E(\mathbf{x}, t)$ – Kinetic Energy

Discretize $\mathbf{x}$, t into 100 values: 4GB memory requirement

Mesoscopic

- $f(\mathbf{x}, \mathbf{v}, t)$ – Density with respect to space *and velocity*

Discretize $\mathbf{x}$, $\mathbf{v}$, t into 100 pieces: 800TB memory requirement

Macroscopic can be derived from mesoscopic

- $\rho(\mathbf{x}, t) = \int_{\mathbb{R}^3} f \, d\mathbf{v}$
- $\mathbf{u}(\mathbf{x}, t) = \frac{1}{\rho} \int_{\mathbb{R}^3} \mathbf{v} f \, d\mathbf{v}$
- $E(\mathbf{x}, t) = \frac{1}{2} \int_{\mathbb{R}^3} \|\mathbf{v} - \mathbf{u}\|^2 \, d\mathbf{v}$

What Are Kinetic Equations?

First used for rarefied gas dynamics (e.g. high altitude gases where collisions do not dominate the physics)

$$\partial_t f + \mathbf{v} \cdot \nabla_{\mathbf{x}} = C(f)$$

where $\int C(f)\, \mathrm{d}\mathbf{v} = 0$.

Integrate against $\mathbf{v}$ to get First Euler/Navier-Stokes equation

$$\partial_t \rho + \nabla_{\mathbf{x}} \cdot (\rho \mathbf{u}) = 0$$

Other areas:

- Radiation transport
- Plasma simulations

Kinetic Problem

Unit Speed, Isotropic Scattering

$$\partial_t f + \Omega \cdot \nabla_{\mathbf{x}} = \frac{\sigma}{4\pi} \langle f \rangle - \sigma f$$

where $x \in \mathbb{R}^3$, $\Omega \in S^2$, σ is the scattering cross section, and $\langle \cdot \rangle = \int_{S^2} \cdot \, \mathrm{d}\Omega$.

Solvers

kinetic

- Implements S_N
- Easy to compute
- Permits negative densities
- Suffers from ray effects at low order

[picture]

moment

- Implements P_N
- Somewhat easy to compute
- Permits negative densities
- Suffers from oscillatory artifacts
- Filters can improve performance

[picture]

Solvers

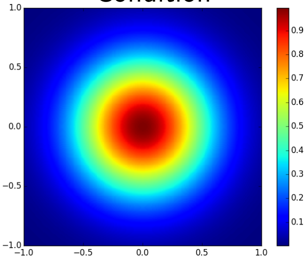
momopt

- Implements M_N and PP_N
- Difficult to compute
- Ensures positivity
- Requires additional optimization procedure

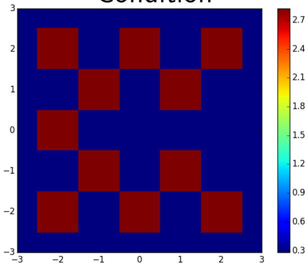
[picture]
[picture]

Initial Conditions

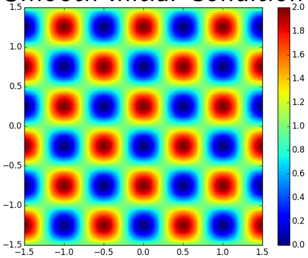
Gaussian Initial Condition



σ_T for Lattice Initial Condition



Smooth Initial Condition



Time Stepping

CFL Condition

A small enough time step is necessary for convergence. In this case,

$$\frac{\Delta t}{\Delta x} + \frac{\Delta t}{\Delta y} \leq 1$$

Heun's Method

To compute a first estimate, carry out two Euler steps. Now use the average of the initial state and the estimate.

- Explicit, two-stage Runge-Kutta method
- Second order accurate
- Strong stability preserving

Quadratures

Currently uses Chebyshev-Legendre quadrature.

- Constructed from n Gauss-Legendre rule
- Optimized for symmetry
- Exactly integrates to order $2n-1$ moments

Added (experimental) Lebedev quadrature

- Constructed based on octahedral symmetry group
- Optimal with respect to number of points
- Structure does not lend itself to symmetry optimizations
- Negative weights

Optimization Problem

`momopt` uses nonlinear spectral methods, so updates entail solving an optimiztion problem for a given **u** and moments **m**.

$$\min_{\alpha} \left\langle \exp(\alpha^{\mathsf{T}}\mathbf{m}) \right\rangle - \alpha^{\mathsf{T}}\mathbf{u}$$

To use a Newton solver, we need

- An objective funtion $F(\alpha) = \left\langle \exp(\alpha^{\mathsf{T}}\mathbf{m}) \right\rangle - \alpha^{\mathsf{T}}\mathbf{u}$
- Gradient $g(\alpha)$ of $F(\alpha)$
- Hessian $H(\alpha)$ of $F(\alpha)$

Now the estimated $\alpha' = \alpha + t\mathbf{d}$ where $\mathbf{d} = -H(\alpha)^{-1}g(\alpha)$.

γ factor

`momopt` computes ansatz for the optimization, yielding an *approximate* solution $\mathbf{u} \approx \left\langle \exp(\alpha^\mathsf{T}) \right\rangle$.

Needed to estimate error and try to correct via γ ratio.

Required fine mesh and low tolerance on optimization to prevent non-realizability.

Instead, compute $\hat{\mathbf{u}}$ such that $\hat{\mathbf{u}} = \left\langle \exp\left(\alpha^\mathsf{T}\right) \right\rangle$.

Fixed regularization

In case of bad condition number (i.e. the system is too sensitive to errors) fall back to isotropic α to allow the algorithm to complete successfully

Source Code

The source code is available on Github

`https://github.com/ckrisgarrett/closures-2d`

Bibliography

cite that presentation, linesource paper