Chasing a quantum anisotropy - with GPUs

Denes Molnar, Purdue University & Wigner RCP

GPU Day 2019, ELTE, Budapest, Jul 11-12, 2019

continuation of DM, Greene, Wang, arXiv:1404.4119v2

Outline

I. General idea, prior estimates for "quantum v_2" anisotropy

II. Recalculation with fewer approximations $\rightarrow$ GPUs

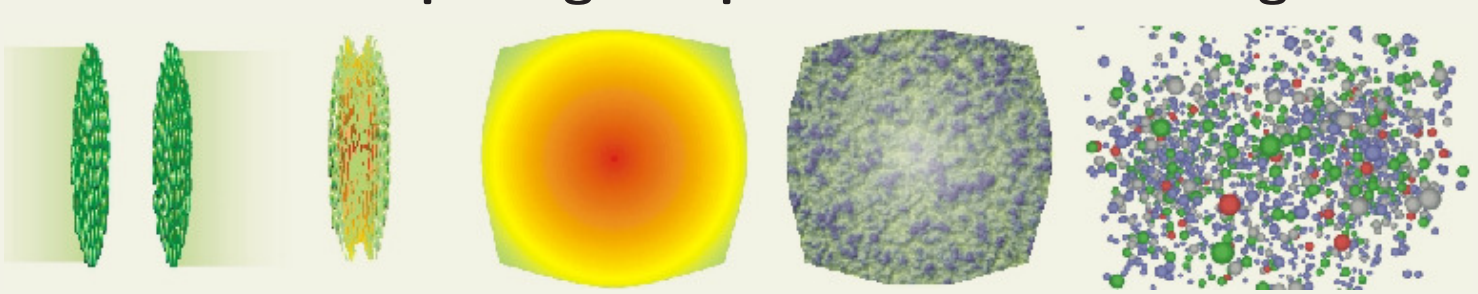
III. Summary and next steps

Heavy ion physics

bang two heavy nuclei together to study the quark-gluon plasma

e.g., at Large Hadron Collider (LHC) or Relativistic Heavy Ion Collider (RHIC)

initial nuclei **quark-gluon plasma** **hadron gas**



[image: S. Bass]

$\mathcal{O}(10\ \mathrm{fm})$

$<\!\!---\ \mathcal{O}(10\ \mathrm{fm}/c)\ ---\!\!>$

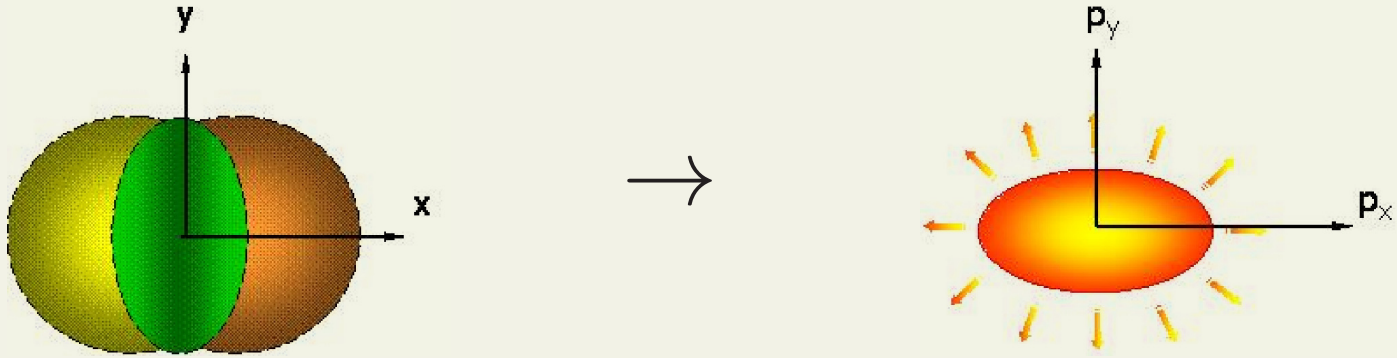
Initconds **Hydrodynamics** **Kinetic theory / flight to detectors**

RHIC/LHC data indicate that extremely hot ($\sim 10^{12} K$) matter is created that behaves hydrodynamically, and has very small specific shear viscosity.

e.g., Gale et al, IJMP A28 ('13)

Elliptic flow (v_2)

most important observable for hydrodynamic behavior



$$\varepsilon \equiv \frac{\langle y^2 - x^2 \rangle}{\langle x^2 + y^2 \rangle}$$

$$v_2 \equiv \frac{\langle p_x^2 - p_y^2 \rangle}{\langle p_x^2 + p_y^2 \rangle} \equiv \langle \cos 2\phi_p \rangle$$

initial spatial anisotropy (ε) converts to final momentum anisotropy (v_2)

Common interpretation: hydrodynamic evolution

- **initially, $\varepsilon > 0$ but $v_2 = 0$**
- **asymmetric pressure gradients subsequently create nonzero v_2**

$\rightarrow$ then compare with hydro to measure properties, such as shear viscosity

Romatschke & Luzum, PRC78 ('08), ...

but is hydrodynamics justified for fermi-scale systems??

v_2 from quantum mechanics

Back of envelope estimate:

$$v_2 \sim \frac{\langle p_x^2 - p_y^2 \rangle}{\langle p_x^2 + p_y^2 \rangle}$$

from uncertainty relation (for ground state, with $\hbar = 1$):

$$\langle p_x^2 \rangle \sim 1/R_x^2 \,, \quad \langle p_y^2 \rangle \sim 1/R_y^2$$

$$\Rightarrow v_2 \sim \frac{R_y^2 - R_x^2}{R_y^2 + R_x^2} = \varepsilon \quad (!)$$

as much v_2 as initial eccentricity, "for free" without any hydro (!)

Of course, at $T > 0$ excited states also enter

and subsequent expansion matters too (hydro + QM??)

Quantum v_2 at nonzero T

stat. physics for simple gas: $H = \sum_i H_1(\mathrm{p}_i, \mathrm{r}_i)\,,\ H_1(\mathrm{p},\mathrm{r}) = K(\mathrm{p}) + V(\mathrm{r})$

Classically, smooth integrals:

$$\frac{dN}{d\mathbf{p}} = N\frac{\int d\mathbf{r}\, e^{-H_1(\mathrm{p},\mathrm{r})/T}}{\int d\mathbf{r}\, d\mathbf{p}\, e^{-H_1(\mathrm{p},\mathrm{r})/T}} = N\frac{e^{-K(\mathrm{p})/T}}{\int d\mathbf{p}\, e^{-K(\mathrm{p})/T}} = isotropic \quad \Rightarrow \quad v_n \equiv 0$$

But in QM, level spacing matters:

$$f(\mathbf{p}) \equiv \frac{dN}{d\mathbf{p}} = \frac{1}{Z}\sum_j |\psi_j(\mathbf{p})|^2\, e^{-E_j/T} = \textbf{anisotropic}$$

for nonrelativistic particle, in 2D harmonic oscillator trap,

arXiv:1404.4119v2

$$v_2 \approx \frac{\hbar^2}{12 k_B T M \langle r_x^2\rangle}\cdot\frac{\varepsilon}{1+\varepsilon} = \frac{\hbar^2}{12 p_{th}^2 \langle r_x^2\rangle}\cdot\frac{\varepsilon}{1+\varepsilon} \qquad \textbf{Nonzero}$$

Vanishes only in the $T \to \infty$ or $M \to \infty$ or $size \to \infty$ limits.

Though $v_2 \neq 0$, there is no hydrodynamic flow anywhere.

$$\mathcal{L} = \frac{i\hbar}{2}\left(\psi^*\dot{\psi} - \dot{\psi}^*\psi\right) - \frac{\hbar^2}{2M}(\nabla\psi^*)(\nabla\psi) - V(\mathbf{r},t)\psi^*\psi$$

apply Noether's theorem:

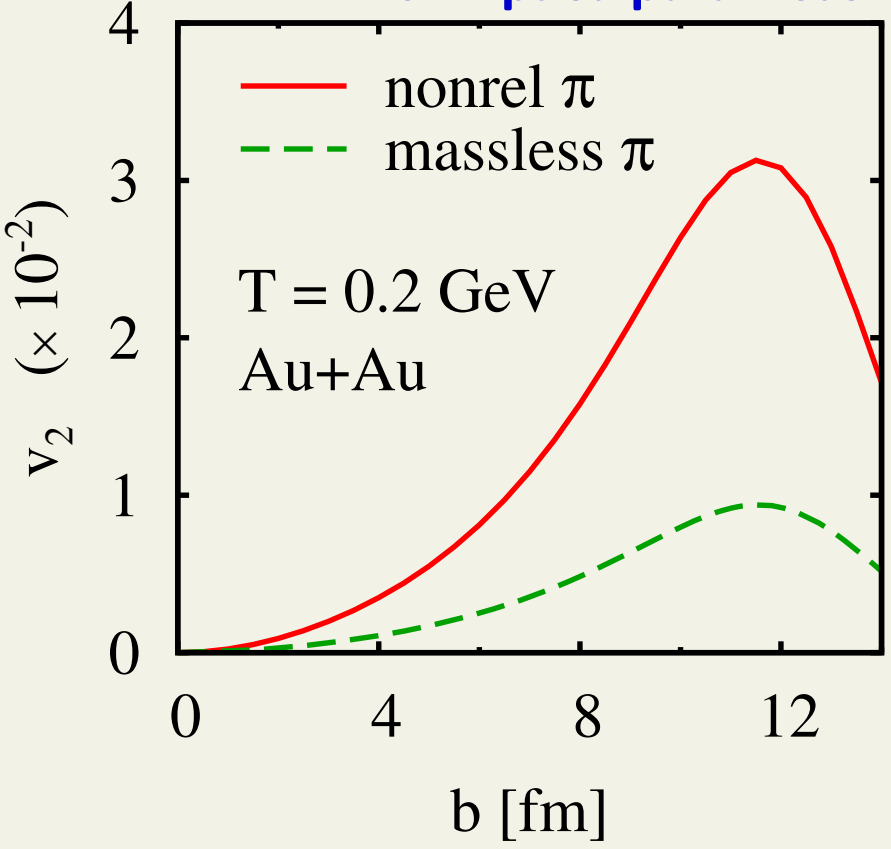
$$\begin{aligned}
T^{00} &= \frac{\hbar^2}{2M}(\nabla\psi^*)(\nabla\psi) + V(\mathbf{r},t)\psi^*\psi && (1)\\
T^{0i} &= \frac{i\hbar}{2}\left(\psi\nabla_i\psi^* - \psi^*\nabla_i\psi\right) && (2)\\
T^{i0} &= \frac{i\hbar}{2M}\left(\frac{\hbar^2}{2M}\Delta\psi - V\psi\right)(\nabla_i\psi^*) + c.c. && (3)\\
T^{ij} &= \frac{\hbar^2}{2M}\left\{(\nabla_i\psi^*)(\nabla_j\psi) - \frac{1}{2}\delta_{ij}\left[\psi^*\Delta\psi + (\nabla\psi^*)(\nabla\psi)\right]\right\} + c.c. && (4)
\end{aligned}$$

The HO wave functions are real $\Rightarrow$ $T^{0i} \equiv 0 \equiv T^{i0}$

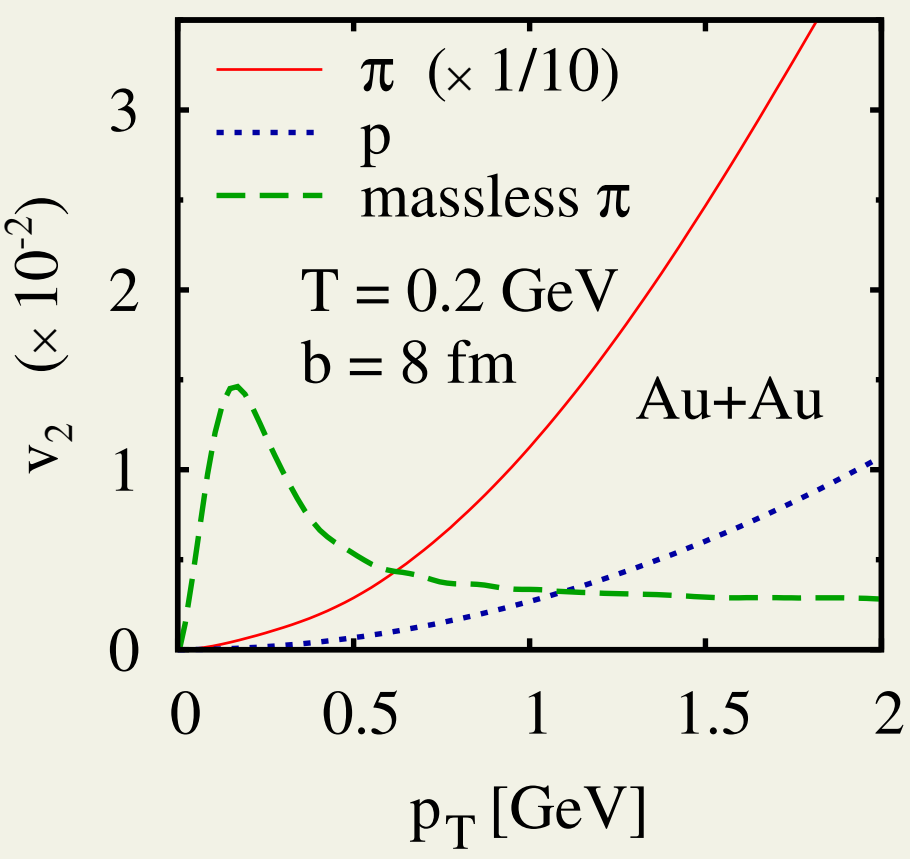
Estimate for ion collisions

Au+Au at RHIC: significant intrinsic v_2 for light hadrons 1404.4119v2

vs impact parameter



vs momentum



but notable caveats, such as: nonrelativistic pions vs massless pions

$\rightarrow$ question arises where v_2 for correct $\sqrt{p^2+M^2}$ would fall...

,

Relativistic calculation

Single-particle Hamiltonian:

$$H \equiv K + V = \sqrt{p_x^2 + p_y^2 + M^2} + \mu^3[(1+\alpha)r_x^2 + (1-\alpha)r_y^2]$$

[μ, α: trap parameters, M: particle mass]

Trick: swap p and r, and rescale

$$\bar{r}_{x,y} \equiv -\frac{p_{x,y}}{\mu\sqrt{1 \pm \alpha}} \ , \quad \bar{p}_{x,y} \equiv \mu\sqrt{1 \pm \alpha}\, r_{x,y} \ , \qquad H \equiv \mu \bar{H} \ , \qquad M \equiv \mu \bar{M}$$

preserves commutation relations $[\bar{r}_i, \bar{p}_j] = i\delta_{ij}$, and Hamiltonian becomes

$$\bar{H} = \bar{p}_x^2 + \bar{p}_y^2 + \sqrt{(1+\alpha)\bar{r}_x^2 + (1-\alpha)\bar{r}_y^2 + \bar{M}^2} \equiv \bar{K} + \bar{V}$$

same as a nonrelavistic particle in some nontrivial potential (!)

at end μ, α need to be dialed to get desired system size $\langle x^2 \rangle$ and $\langle y^2 \rangle$

Diagonalize in finite basis

expand over finite basis $|\psi_j\rangle = \sum_{n=1}^{N} c_{j,n}|\phi_n\rangle$

Schrödinger's equation becomes an $N \times N$ **generalized eigenvalue problem**

$$\sum_n \bar{H}_{mn} c_{j,n} = \bar{E}_j \sum_n O_{mn} c_{j,m}$$

with matrix elements and overlaps

$$\bar{H}_{mn} \equiv \langle \phi_m | \bar{H} | \phi_n \rangle \ , \ O_{nm} \equiv \langle \phi_m | \phi_n \rangle$$

Here, $\bar{H}_{mn}$ **includes**

$$\begin{aligned} \bar{V}_{mn} &= \int dxdy\, \phi_m(x,y)\, \phi_n(x,y) \sqrt{Ax^2 + By^2 + \bar{M}^2} \\ &= \int drrd\varphi\, \phi_m(r,\varphi)\, \phi_n(r,\varphi) \sqrt{r^2(C + D\cos 2\varphi) + \bar{M}^2} \end{aligned}$$

Need many eigenvectors and eigenvalues in order to construct momentum distribution $f(\mathrm{p}) \propto \sum_j |\psi_j(\mathrm{p})|^2\, e^{-E_j/T}$**, and from that** v_2

Naive cost: - **quadrature (matrix elements):** $\mathcal{O}(N^2)$

- **diagonalization:** $\mathcal{O}(N^3)$

in practice, need $N \sim 200^2 - 500^2$

but integration becomes $\mathcal{O}(N^{2+\delta})$ if ϕ_n involves iterations or integrands oscillate

try moving quadrature to GPUs

- **good part: minimal data** - few integers in, value & error out

- **bad part: adaptive routines hard on GPU (conditionals, iterative loops)**

$\rightarrow$ **doing quadrature well helps other physics calculations too**

e.g., 4D quadrature in kinetic theory DM & Wolff, PRC95 ('17)

V_{mn} on GPUs

First, some standard tricks:

- **factorized basis** $\phi_n(x,y) = X_{n_1}(x)\,Y_{n_2}(y)$ $\qquad (0 \le n_{1,2} < M = \sqrt{N})$
  or $\phi_n(r,\varphi) = R_{n_1}(r)\,F_{n_2}(\varphi)$
- **exploit parity:** $\bar{V}$ **even in both** x **and** $y \Rightarrow \phi_{n,m}$ **must match in parity**

  4 parity classes (++, +-, -+, --), each with $\approx N/4$ **states. E.g.,**

$$\begin{aligned}\bar{V}_{nm} &= 4\int_0^\infty dx\, X_{n_1}(x)\, X_{m_1}(x) \int_0^\infty dy\, Y_{n_2}(y)\, Y_{m_2}(y)\, \bar{V}(x,y) \\ &= \int_0^{\pi/2} d\varphi\, F_{n_2}(\varphi)\, F_{m_2}(\varphi) \int_0^\infty dr\, r\, R_{n_1}(r)\, R_{m_1}(r)\, \bar{V}(r,\varphi)\end{aligned}$$

Bases: - **product of 1D harmonic oscillators** $(X \times Y)$ $\rightarrow$ **slow** $\mathcal{O}(N^3)$ **or worse**

- **2D polar:** $F(\varphi) \propto \cos(k\varphi)$, $\sin(k\varphi)$**; localized splines for** $R(r)$

$[R(r) \propto r^k$ is ill-conditioned because large overlaps $O_{mn} \sim \mathcal{O}(1)]$

Method 1: just do the same as on CPU

2 nested integrals via adaptive 1D routines from GNU Scientific Lib (GSL)

Gauss-Kronrod quadrature (61 points):

$$\int_a^b dx\, f(x) \approx \sum_{i=0}^{60} w_i\, f(x_i)$$

error estimate: take only half the points $\int_a^b dx\, f(x) \approx \sum_{i\ even}^{60} w'_i\, f(x_i)$

If error large, bisect $[a, b]$ and its bisections, until total error small enough

$\rightarrow$ always bisect interval that has largest error next

sketch of iterative 1D quadrature code: $\int_a^b dt\, f(t)$

```
sections = alloc_iv(NMAX);   // storage for intervals
iv1 = [a,b];                 // initial interval
integrate_f(iv1);            // get integral and error
n = 0;
while (error_is_big && n < NMAX) {
    sections[n] = iv1;
    i = worst_section(sections);   // find interval to split
    iv1 = left_half(sections[i]);
    iv2 = right_half(sections[i]);
    integrate_f(iv1);    // integrate over both halves
    integrate_f(iv2);
    update_sum_and_error(iv1, iv2);   // track total and error
    sections[i] = iv2;
    n ++;
}
```

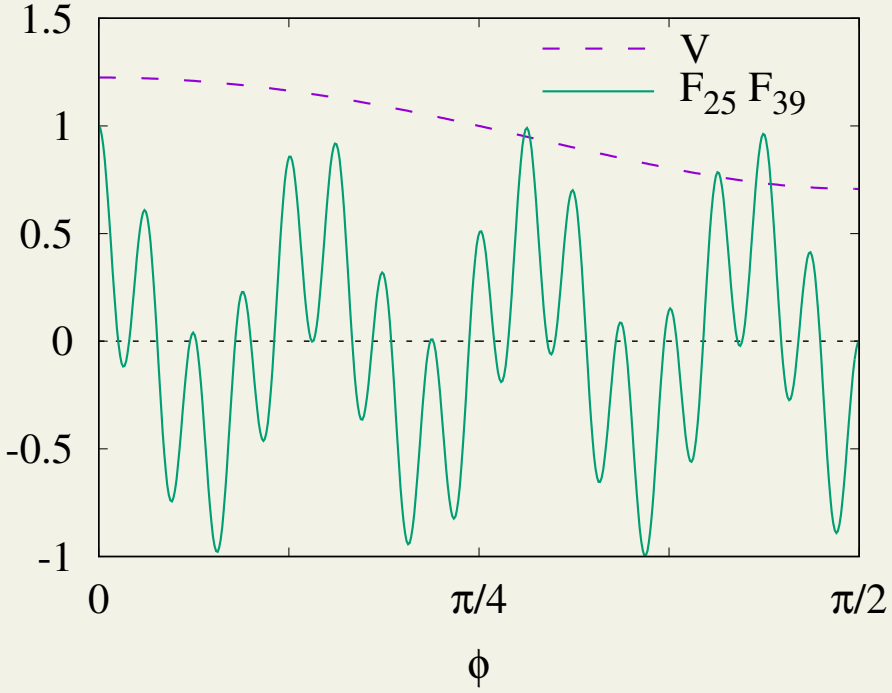
both loop and search involve conditionals - not that ideal for GPU

Method 2: improve inner loop

empirically: inner integration takes only 1 interval $(\approx 2/3$ of time$)$
or 2 intervals $(\approx 1/3$ of time$)$

it turns out to be faster to use 2 intervals all the time $\rightarrow$ no inner loop

unfortunately, cannot do same for outer loop because φ-integral is oscillatory



$$\int_0^{\pi/2} d\varphi \, \cos m_1\varphi \, \cos m_2\varphi \, \sqrt{C' + D' \cos 2\varphi}$$

$\Rightarrow$ **# of adaptive subdivisions in φ grows with m_1 and m_2**

Method 3: match threads to workgroups better → group by similar runtime

needed because OpenCL waits until all threads finish in workgroup

original code did $idx = n_2 + Mm_2 + M^2n_1 + M^3m_1$

```
for (m1 = 0; m1 < M; m1 ++)    // F_m1
   for (n1 = 0; n1 < M; n1 ++)    // R_n1
      for (m2 = 0; m2 < M; m2 ++)    // F_m2
         for (n2 = 0; n2 < M; n2 ++)    // R_n2
            start_task(n1, m1, n2, m2);
```

instead do $idx = n_2 + Mn_1 + M^2m_2 + M^3m_1$

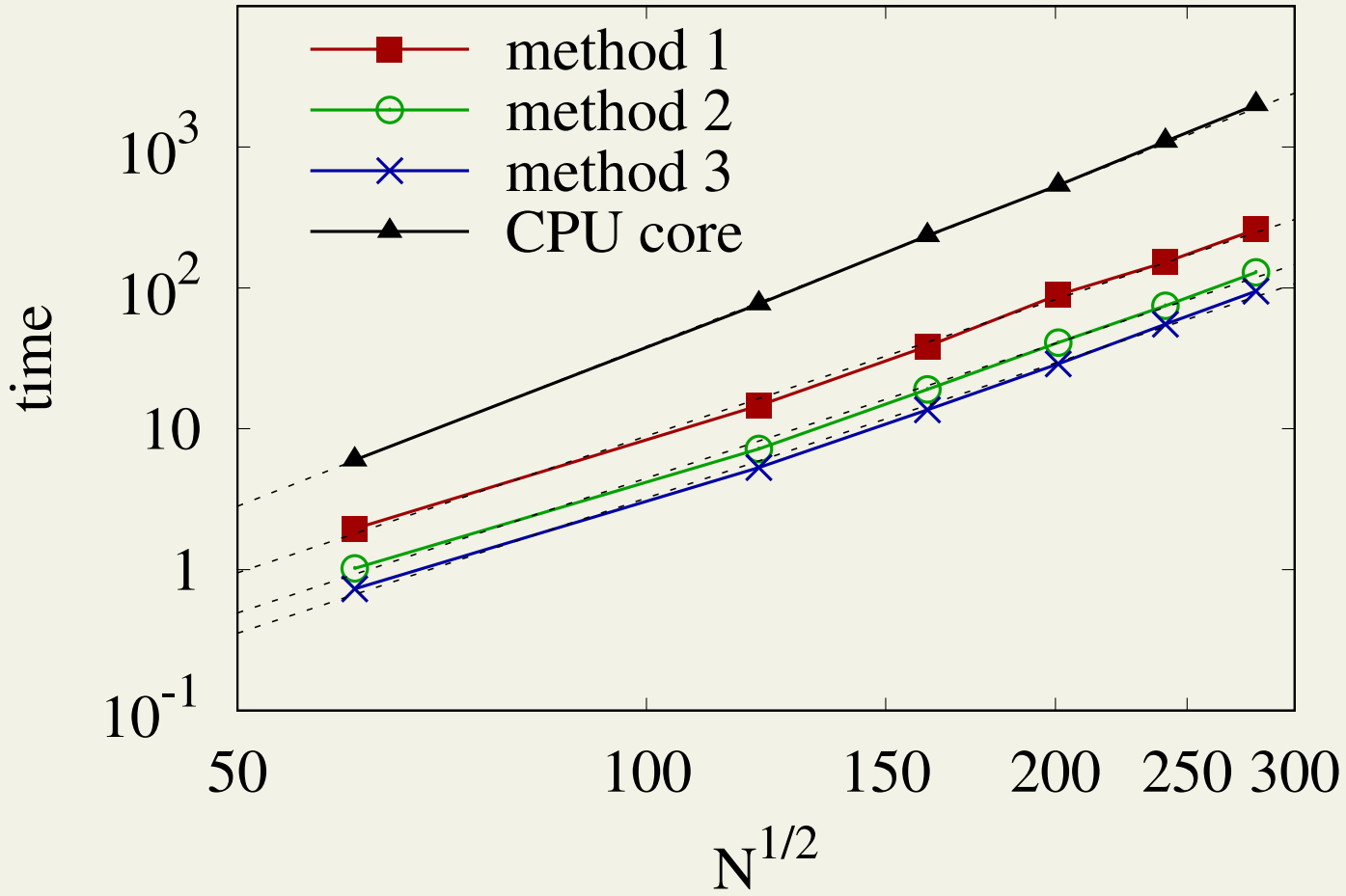
```
for (m1 = 0; m1 < M; m1 ++)    // F_m1
   for (m2 = 0; m2 < M; m2 ++)    // F_m2
      for (n1 = 0; n1 < M; n1 ++)    // R_n1
         for (n2 = 0; n2 < M; n2 ++)    // R_n2
            start_task(n1, m1, n2, m2);
```

⇒ **more threads with similar m_1, m_2 fall into same workgroup**

GPU vs CPU runtime

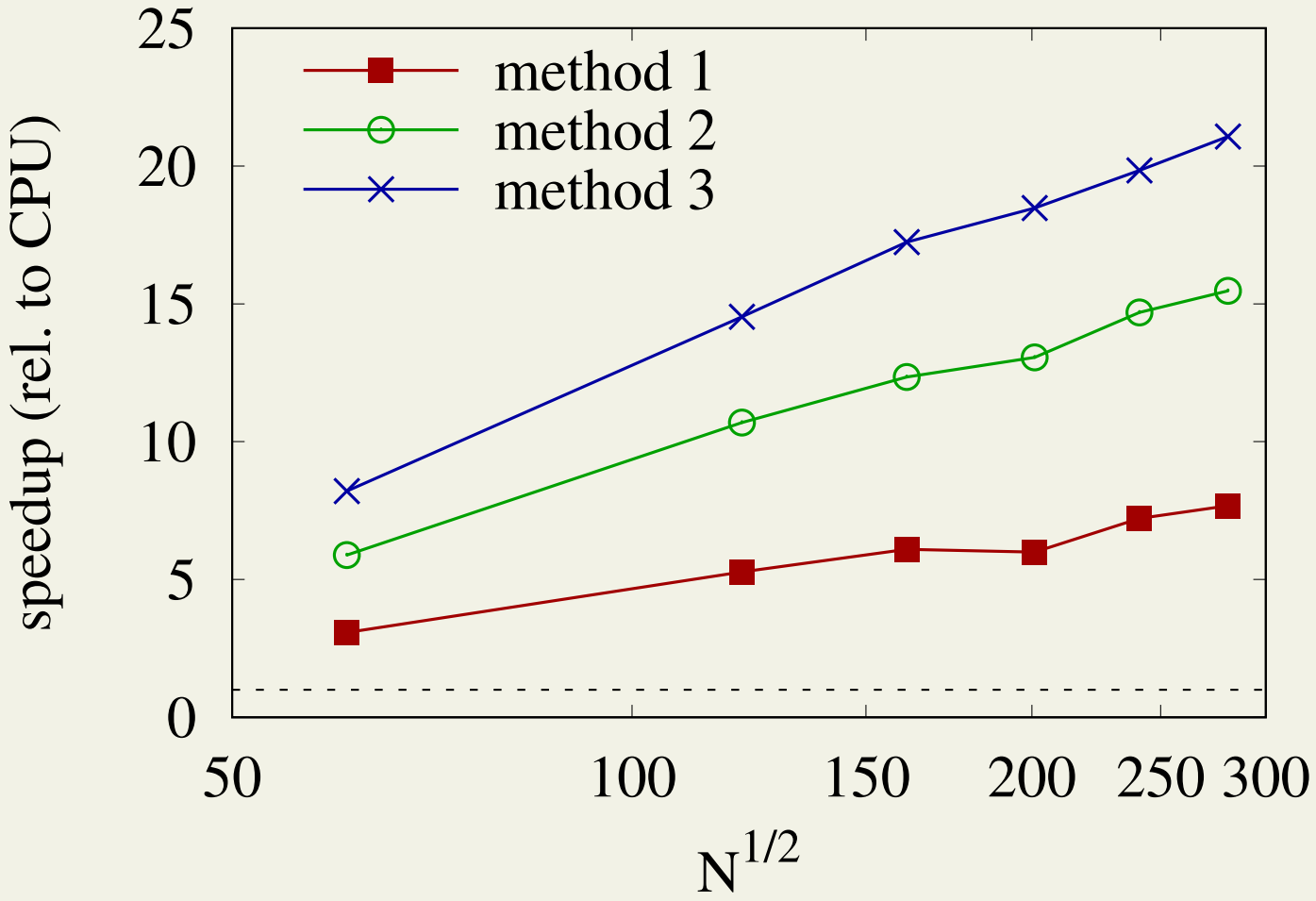
CPU: Xeon E5-2660 at RCAC Purdue, **GPU:** Vega 56 at Wigner GPU Lab

absolute time vs problem size $\rightarrow$ GPU looks useful



CPU: Xeon E5-2660 at RCAC Purdue, **GPU:** Vega 56 at Wigner GPU Lab

relative time vs problem size $\rightarrow$ **up to** $\sim 20\times$ **faster than 1 CPU core**



better WGs $\sim 3\times$

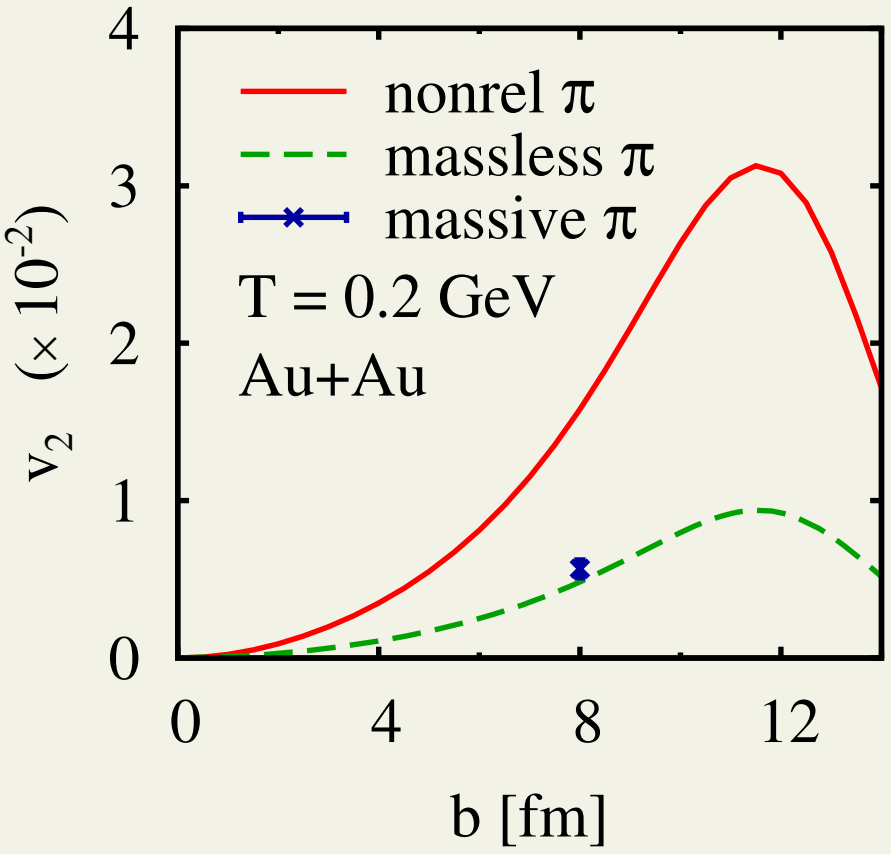
no inner loop $\sim 2\times$

GPU baseline

Preliminary results

quantum anisotropy in Au+Au at RHIC, now with massive, relativistic pions

polar $r - \varphi$ basis, $N = 281^2$ $\rightarrow$ will need $N \sim 400^2 - 500^2$ eventually



massive pion v_2 lies very close to massless case

More ideas / Next steps

- converge vs N, get massive pion v_2 vs impact param and momentum

- **oscillatory φ-integrals**:

$$\int d\varphi \begin{bmatrix} \cos k\varphi \, \cos l\varphi \\ \sin k\varphi \, \sin l\varphi \end{bmatrix} V(r,\varphi)$$

  - change product in integrand to single cosine via $[\cos(a+b) \pm \cos(a-b)]/2$

  - then integrate piecewise over each full period of the cosine

can help eliminate adaptive iteration in outer loop for V_{mn}

fewer basis fn combinations (m_1, m_2): $M^2 \to \mathcal{O}(M)$ in matrix elements

- **diagonalize with GPU**

  - at high N, diagonalization cost will dominate

  - cannot put it all on GPU (at $N \sim 500^2$, answer takes ~ 30 GB RAM)

    but worthwhile to look into GPU-accelerated linear algebra

Summary

Hydrodynamics is not the only source of momentum anisotropies. Quantum systems with coordinate space anisotropy have, in general, momentum anisotropy (Heisenberg uncertainty relation). Prior calculations for a gas trapped in 2D suggest a sizeable quantum anisotropy for pions in Au+Au collisions at RHIC. [DM, Greene, Wang, 1404.4119v2]

However, the estimates varied by a factor of 3 or more depending on whether nonrelativistic or massless pions were considered. We recalculate the problem with unapproximated $\sqrt{p^2+m^2}$ kinetic term, using matrix elements computed on GPU. Preliminary calculations for pions show an anisotropy that is very close to the massless result.

Most important lesson:

adapting calculations to GPUs requires one to rethink the problem, which can lead to better algorithms on CPUs as well