BSQ Hydrodynamics and Challenges with a 4D equation of state

Collaborators:
Dekrayat ***Almaalol***, *Travis* ***Dore***,
Debora ***Mroczek***,
Jordi ***Salinas San Martin***,
Lydia ***Spychalla***, *Patrick* ***Carzon***,
Matthew ***Sievert***, *and*
Jacquelyn ***Noronha-Hostler***

Christopher Plumberg
Pepperdine University

INT Workshop

Dense Nuclear Matter Equation of State from Heavy-Ion Collisions

December 5-9, 2022

Goals for this talk

- Introduce **new** hydrodynamics code for propagating conserved charges in heavy-ion collisions:
  CCAKE (**C**onserved **C**h**A**rges with hydrodynami**K** **E**volution)
- Discuss several challenges to doing this from the thermodynamic side when using multiple conserved charges
- Explore some possible solutions and some open questions
- **Discussion focuses** (foci?):
  - *What improvements on the constraints on the EOS can we expect from future heavy-ion experiments?*
  - *What development is necessary for transport codes to address the above questions?*

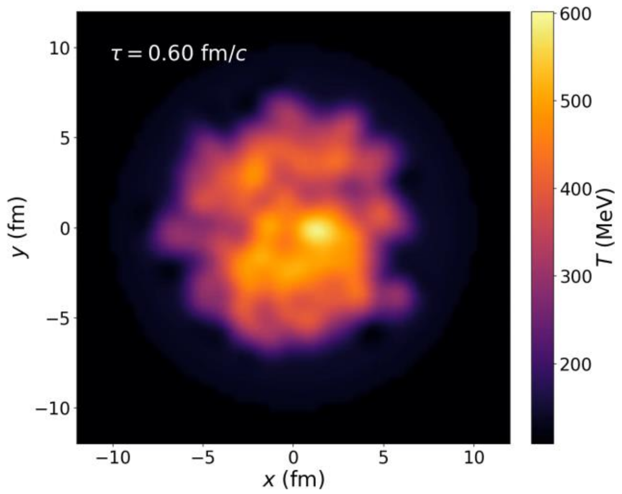
Formalism

BSQ Simulations

– **Equations of motion**:

$\nabla_\mu T^{\mu\nu} = 0$

$(\textit{energy-momentum conservation})$



– **Propagate densities**:

e (*energy density*) or

s (*entropy density*)

– Equations of motion closed by **equation of state**: $P = P(T)$

– **Need to know** T coordinate for a **given** e/s point

BSQ Simulations

- **Equations of motion**:

  $$\nabla_\mu T^{\mu\nu} = 0$$

  (*energy-momentum conservation*)

  $$\nabla_\mu J_i^\mu = 0 \quad (i = B, S, Q)$$

  (*charge conservation*)

- **Propagate densities**:

  e (*energy density*) or

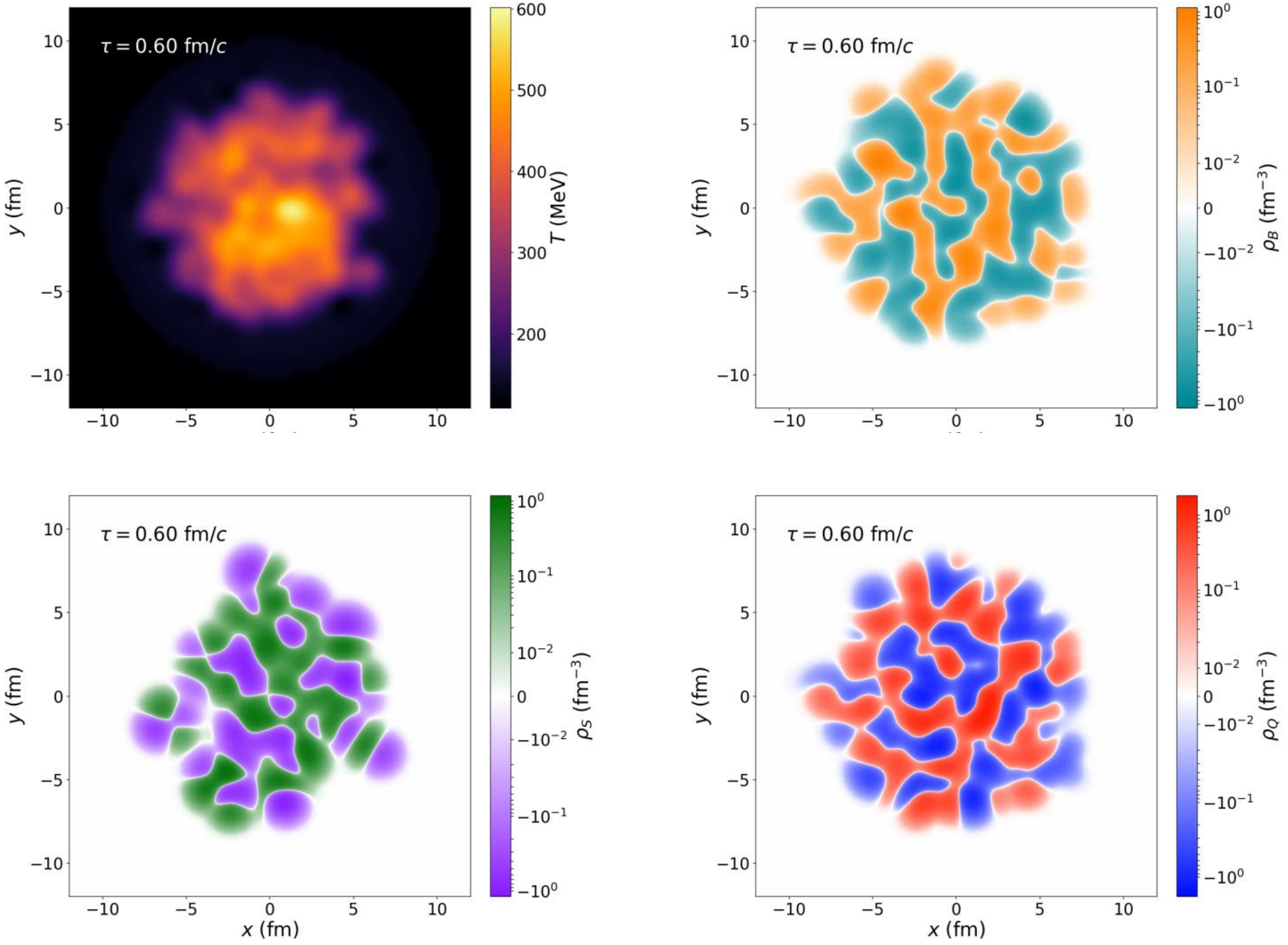
  s (*entropy density*)

  ρ_B (*baryon density*)

  ρ_S (*net strangeness density*)

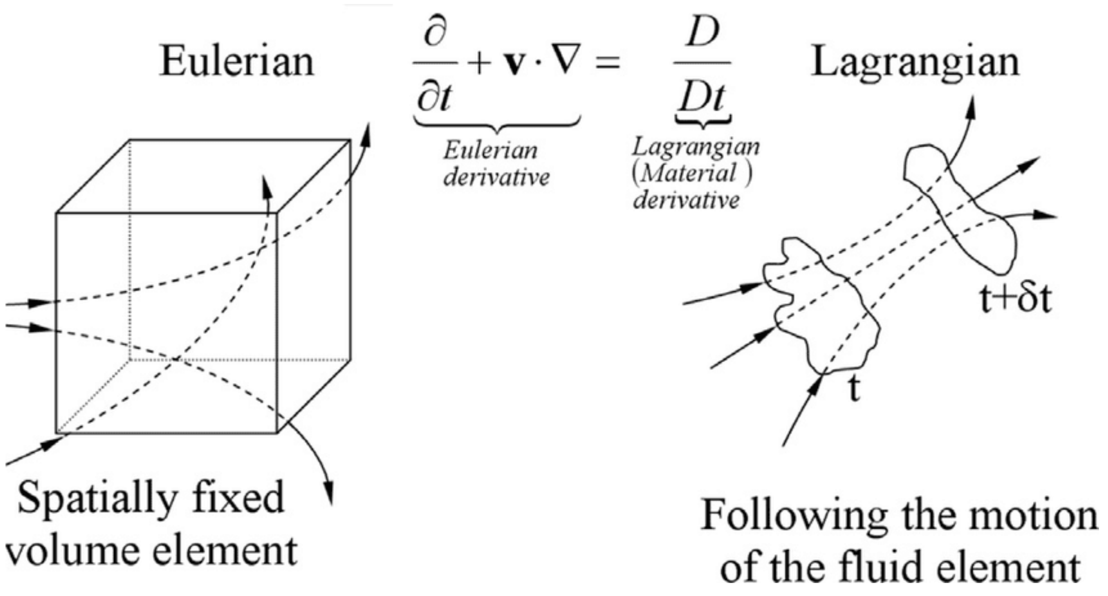
  ρ_Q (*electric charge density*)

- Equations of motion closed by **equation of state**: $P = P(T, \{\mu_i\})$



- **Need to know** (T, μ_B, μ_S, μ_Q) coordinates for a **given** $(e/s, \rho_B, \rho_S, \rho_Q)$ point

Smoothed Particle Hydrodynamics (SPH)

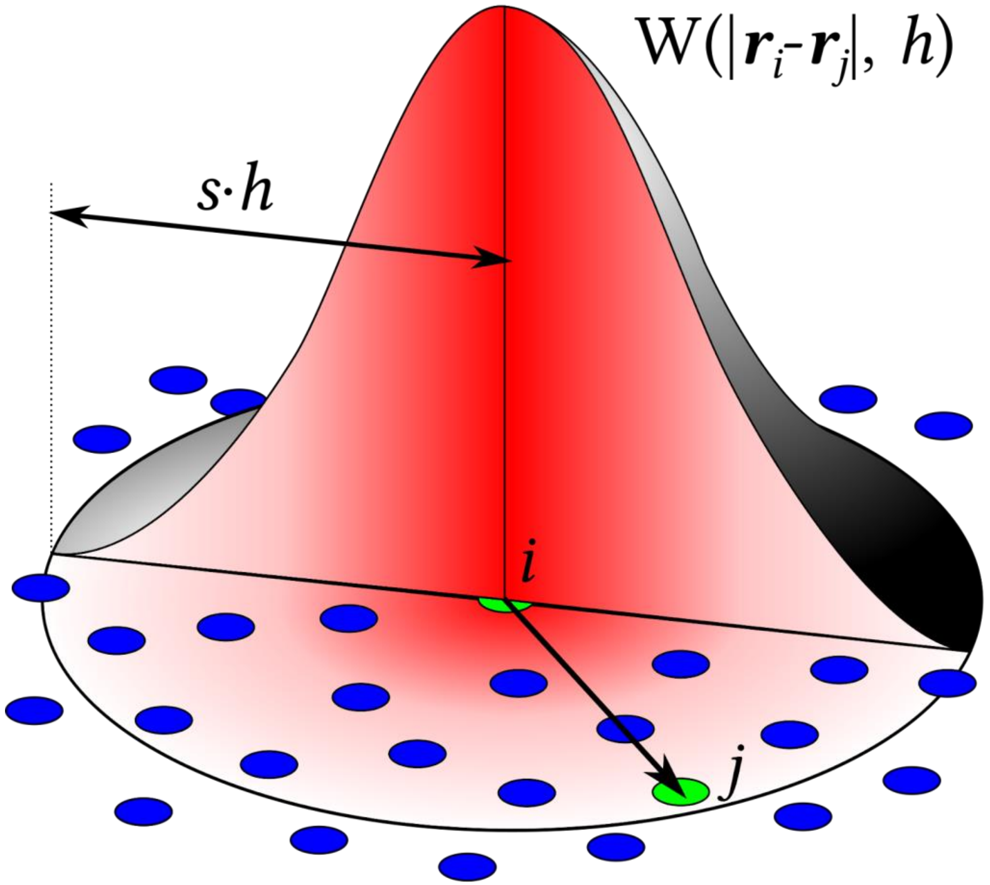


Grid-based hydrodynamics

Smoothed particle hydrodynamics

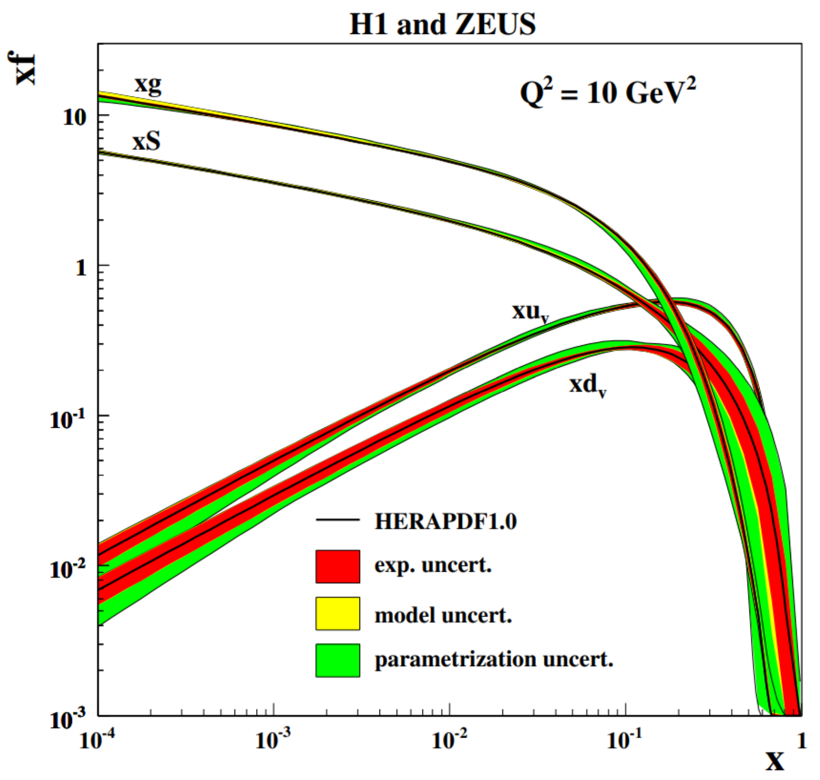
Conservation laws built-in by construction

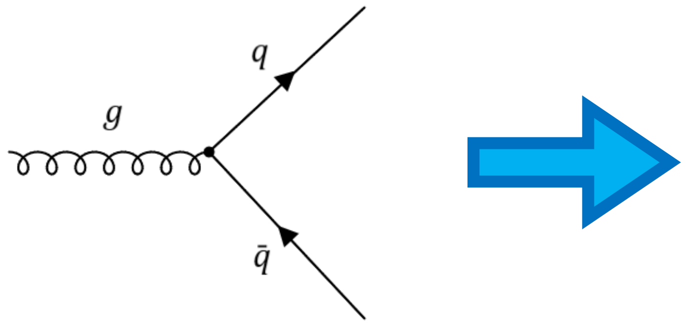
Kernel function W imposes coarse-graining onto set of fictitious 'SPH particles'

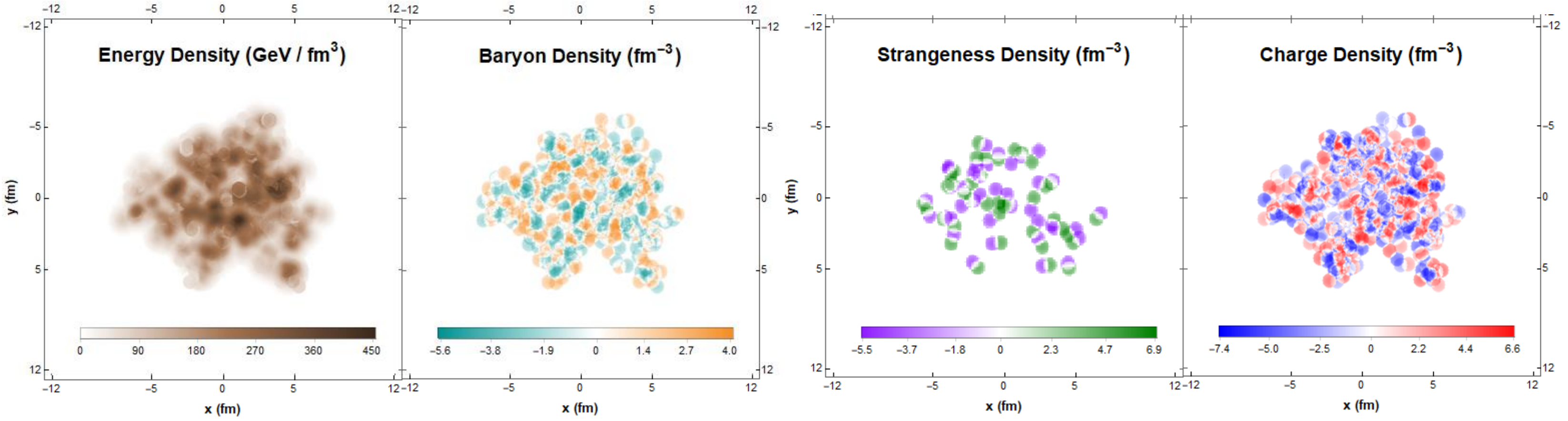


BSQ Initial Conditions

- CCAKE accepts **ICCING** initial conditions (Initial Conserved Charges In Nuclear Geometry)
- ICCING relies on the gluon-saturated initial state at mid-rapidity to determine probabilities for gluon splitting to quark pairs
- Use color-glass condensate (CGC) framework to generate *local charge fluctuations*







[arXiv:1911.12454 [nucl-th]]

BSQ Evolution

Israel-Stewart fluid dynamics

Dekrayat Almaalol, Travis Dore, Jacquelyn Noronha-Hostler [arXiv:2209.11210 [hep-th]]

$$S^\mu = \underbrace{s u^\mu}_{\text{equilibrium}} - \underbrace{\sum_q^{B,S,Q} \alpha_q n_q^\mu}_{\text{1st-order term}} - \frac{1}{2} u^\mu \underbrace{\left(\beta_\Pi \Pi^2 + \beta_\pi \pi^{\mu\nu}\pi_{\mu\nu} + \sum_q^{B,S,Q} \beta_n^{qq'} n_q^\mu n_\mu^{q'} \right)}_{\text{2nd-order terms}} - \underbrace{\sum_q^{B,S,Q} \left(\gamma_{n\Pi}^q n_q^\mu \Pi + \gamma_{n\pi}^q n_q^\nu \pi_\nu^\mu \right) - \frac{1}{2}\left(u^\nu \beta_{\Pi\pi} \Pi \pi_{\mu\nu} \right)}_{\text{2nd-order coupling terms}}$$

Second law of thermodynamics

$$\partial_\mu S^\mu = \frac{\beta_0}{2\eta} \pi_{\mu\nu}\pi^{\mu\nu} + \frac{\beta_0}{\zeta}\Pi^2 + \frac{1}{\kappa_{qq'}} n_\mu^q n_{q'}^\mu \geq 0$$

Hydrodynamic modeling with multiple conserved charges introduces a host of **new transport coefficients** characterizing charge diffusion, shear-diffusion couplings, etc.

NS Transport coefficients

$\eta, \zeta, \kappa_{qq'}$

2nd order Transport coefficients

$\beta_\Pi, \beta_\pi, \gamma_{n\Pi}, \gamma_{n\pi}, \beta_{qq'}$

Fotakis et al, 2203.11549 [nucl-th]

Slide credit: Dekrayat Almaalol

BSQ Thermodynamics

This work: lattice QCD EoS given by Taylor expansion of pressure in powers of chemical potentials

$$\frac{P(T,\mu_B,\mu_Q,\mu_S)}{T^4} = \sum_{i,j,k} \frac{1}{i!j!k!}\chi^{BQS}_{i,j,k}\left(\frac{\mu_B}{T}\right)^i\left(\frac{\mu_Q}{T}\right)^j\left(\frac{\mu_S}{T}\right)^k$$

Claudia Ratti 2018 *Rep. Prog. Phys.* **81** 084301

J. Noronha-Hostler, P. Parotto, C. Ratti, J. Stafford PRC 100 (2019),
A. Monnai et al., PRC 100 (2019)

"Susceptibilities" $\chi^{BQS}_{i,j,k}$ functions of temperature; matched to lattice QCD at high T and hadron resonance gas at low T

Charge/entropy densities obtained by taking derivatives w.r.t. P
Energy density obtained using Gibbs' relation

How do we invert given set of densities for corresponding phase diagram coordinates?

Challenges

Strategy #1: root-finding

Goal: obtain $(T_0, \mu_{B,0}, \mu_{S,0}, \mu_{Q,0})$ from $(e_0, \rho_{B,0}, \rho_{S,0}, \rho_{Q,0})$

Given:

$$\begin{aligned} e &= e(T, \vec{\mu}) \\ \rho_B &= \rho_B(T, \vec{\mu}) \\ \rho_S &= \rho_S(T, \vec{\mu}) \\ \rho_Q &= \rho_Q(T, \vec{\mu}) \end{aligned}$$

Solve:

$$\begin{aligned} e_0 &= e(T_0, \vec{\mu}_0) \\ \rho_{B,0} &= \rho_B(T_0, \vec{\mu}_0) \\ \rho_{S,0} &= \rho_S(T_0, \vec{\mu}_0) \\ \rho_{Q,0} &= \rho_Q(T_0, \vec{\mu}_0) \end{aligned}$$

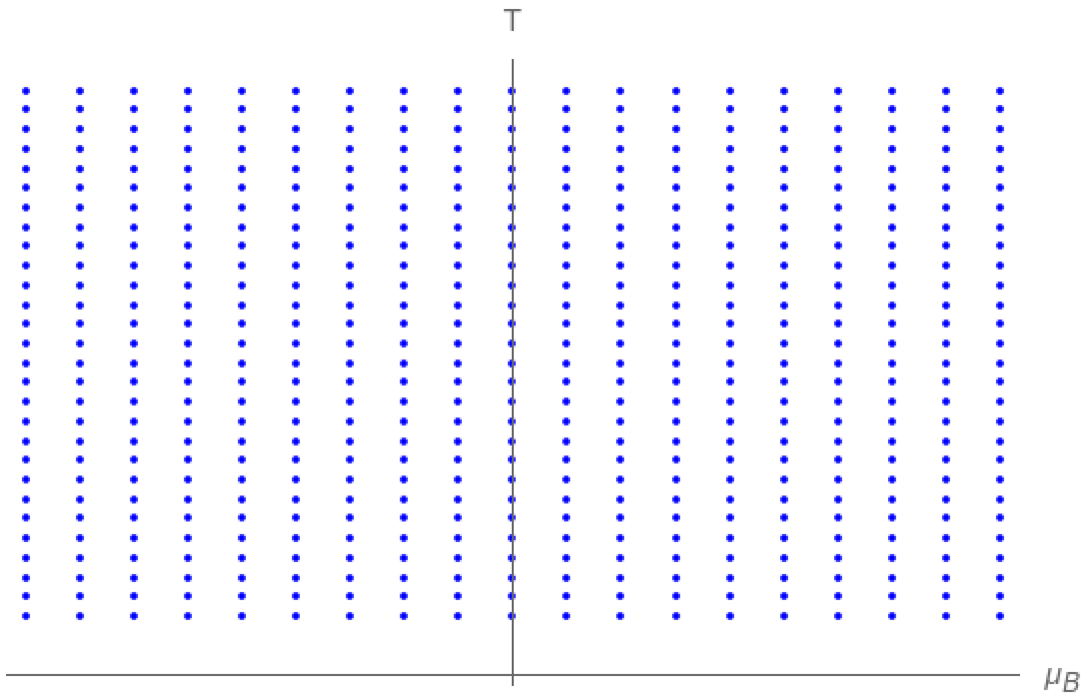
Construct interpolants from table of equation-of-state (e.g., LQCD) data

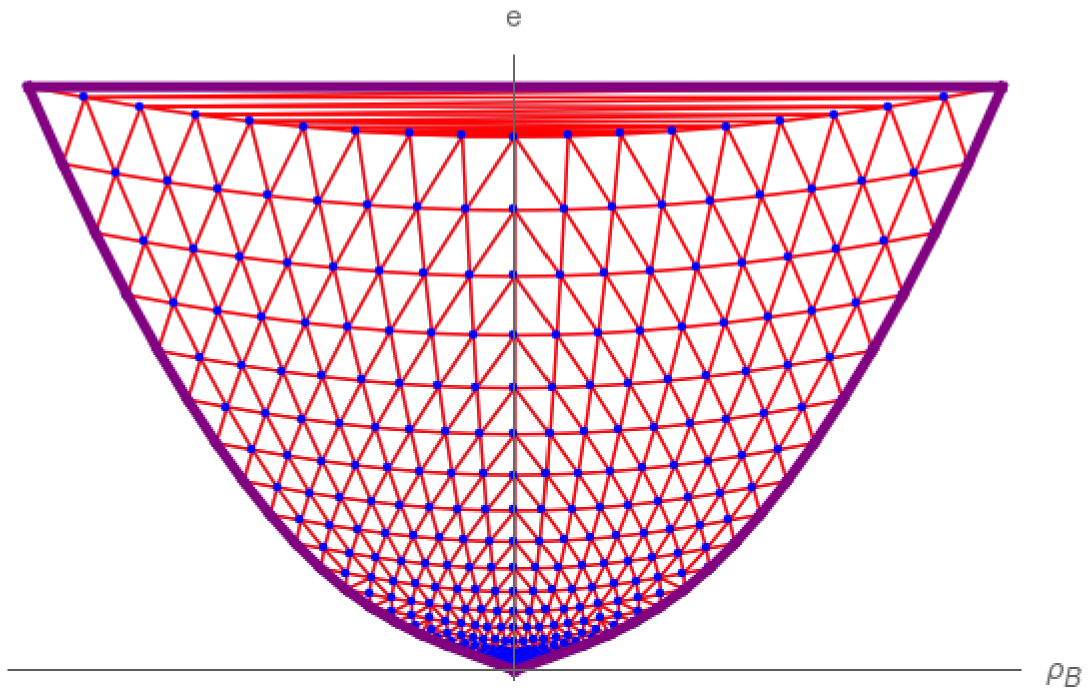
Couple to multi-dimensional rootfinder (e.g., via GSL library)

Current default functionality of `CCAKE`

Alternative: can we construct functions $T(e, \vec{\rho})$, $\vec{\mu}(e, \vec{\rho})$?

Strategy #2: Delaunay interpolation



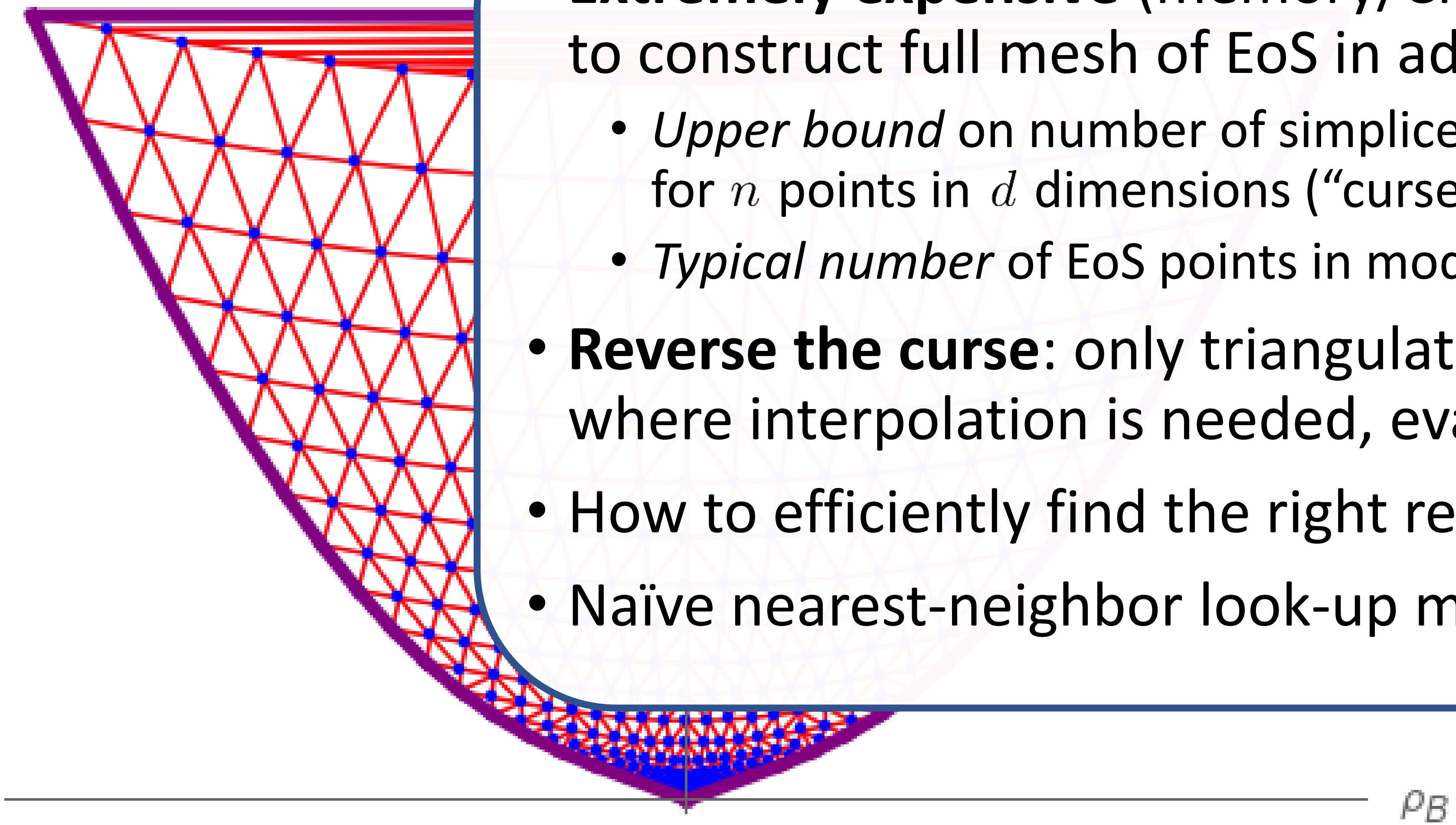


$$\text{Uniform } T\text{–}\mu_B \text{ grid} \quad \not\Rightarrow \quad \text{uniform } e\text{–}\rho_B \text{ grid}$$

– Perform linear interpolation on Delaunay triangulation of scattered density points

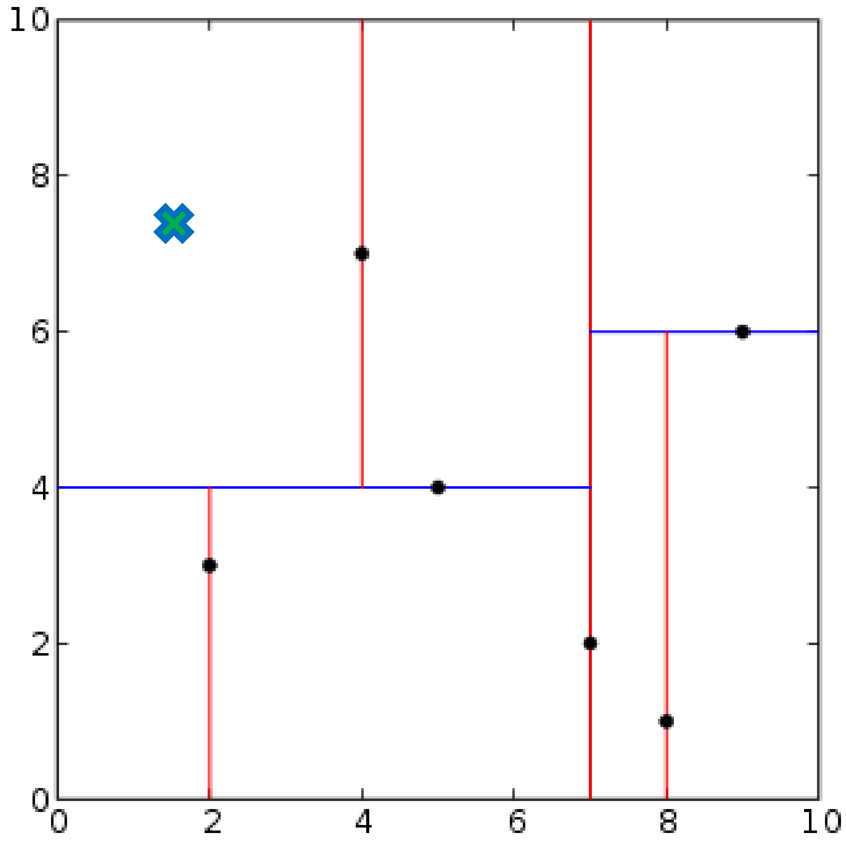
– Only defined inside **convex hull** (bold line)

Constructing the Delaunay mesh

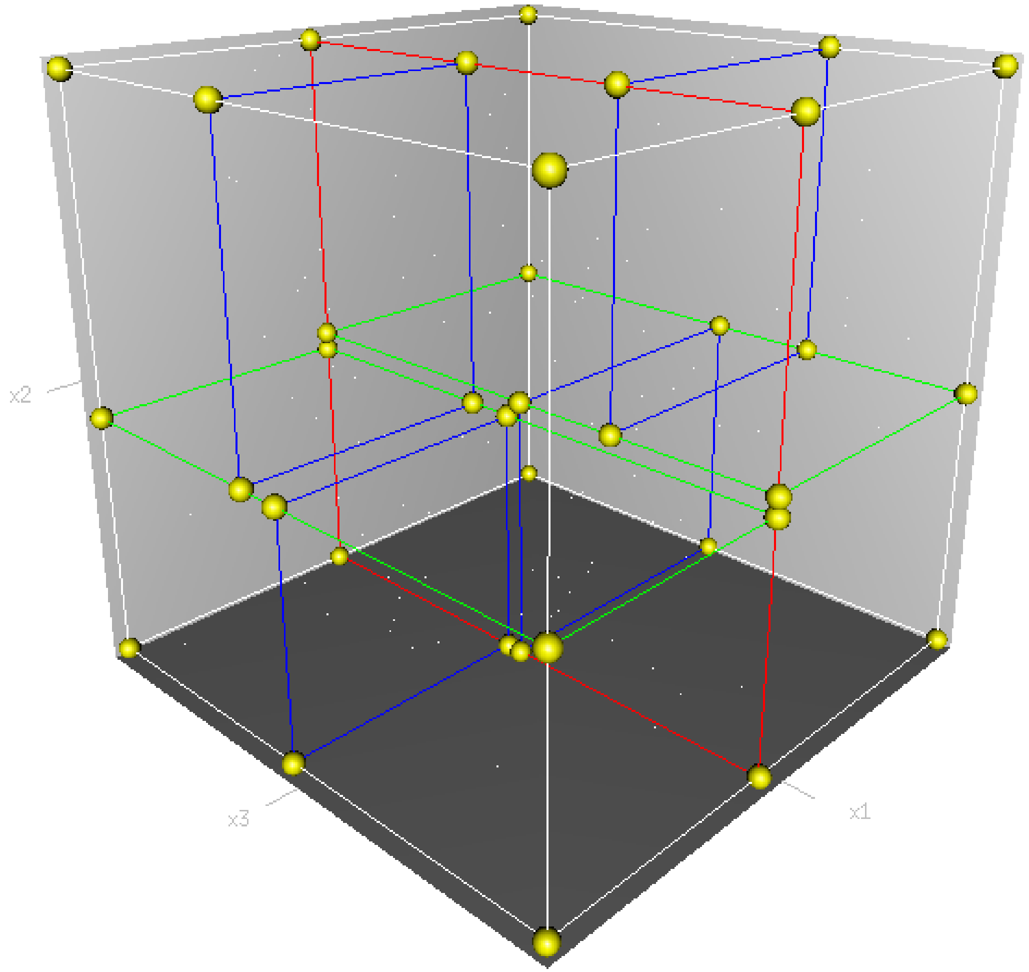


- **Extremely expensive** (memory/CPU time) to construct full mesh of EoS in advance
  - *Upper bound* on number of simplices grows like $O\left(n^{\lceil d/2 \rceil}\right)$, for n points in d dimensions ("curse of dimensionality")
  - *Typical number* of EoS points in modest grid: $O\left(10^5 - 10^7\right)$ in 4D
- **Reverse the curse**: only triangulate the region where interpolation is needed, evaluated at runtime
- How to efficiently find the right region to triangulate?
- Naïve nearest-neighbor look-up may be *very* inefficient

Finding closest simplex efficiently: *k*-d trees



2D example



3D example

A rough algorithm

Compute (T, μ_B, μ_S, μ_Q) distributions on scattered $(e, \rho_B, \rho_S, \rho_Q)$ grids

Identify convex hull inside of which density interpolation is defined

Build k-d trees of density grids

For given densities $(e_0, \rho_{B,0}, \rho_{S,0}, \rho_{Q,0})$:

– *locate* containing / neighboring simplices

– *construct* Delaunay triangulation

– *evaluate* unique linear interpolant at input densities

Download: `https://github.com/astrophysicist87/eos_delaunay_demo` (see backup slides)

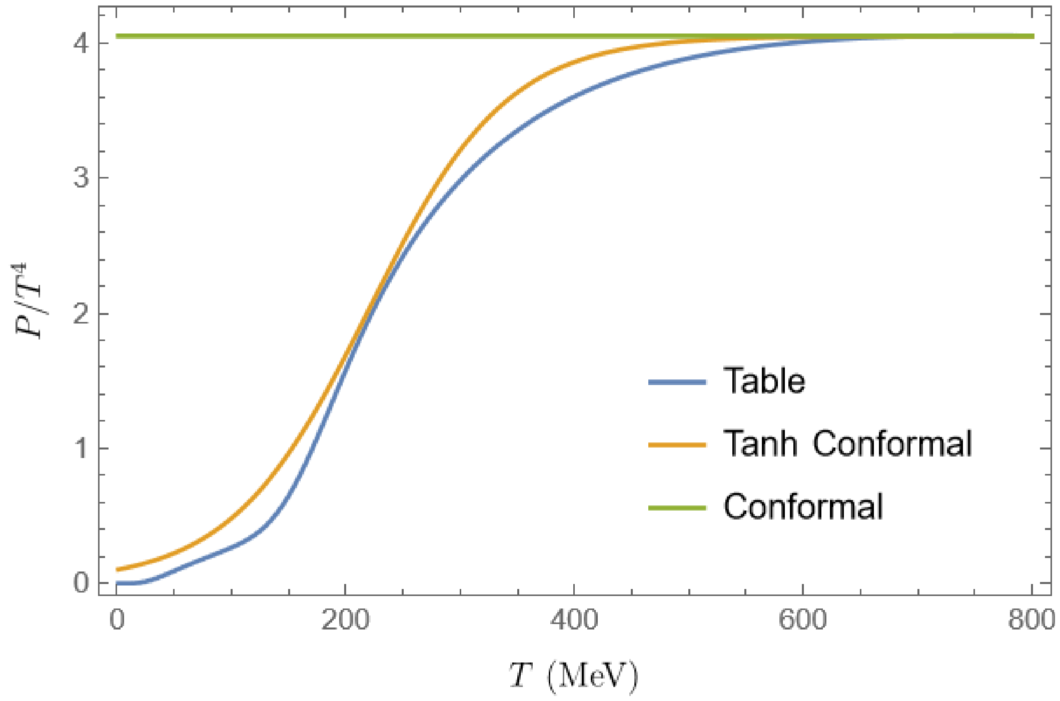
What if the search fails?

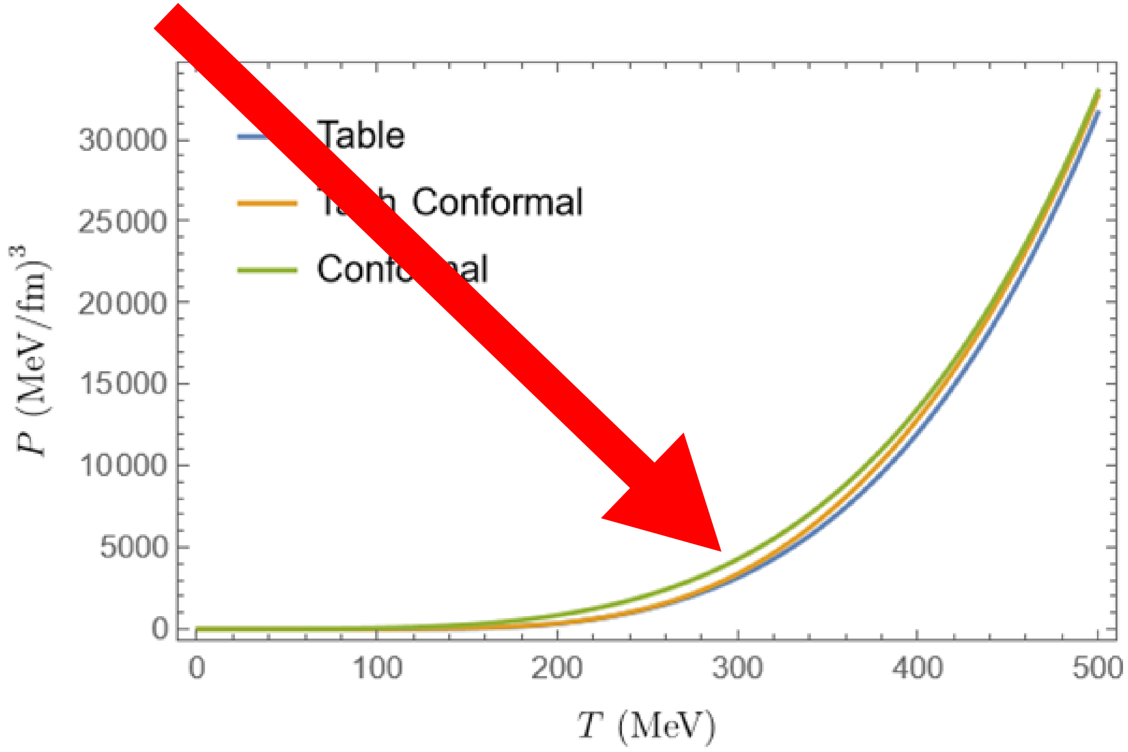
- There is a troubling possibility that I have not addressed yet: *what if the search fails?*
- Obviously the ideal is that this never happens
- This can happen for any of a number of reasons:
  - The true solution **may exist outside** the convex hull of the current grid
  - There may not be ***any*** **solution** for the chosen equation of state
  - There may be **multiple "correct" solutions**
- When this happens in hydrodynamics,
  how should we close the equations of motion?
  - Crash the code
  - Use a "default value"
  - Discard charge densities
  - Supplement with a different "back-up" equation of state

“Back-up” Equations of State

- If the preferred (read: *tabulated lattice QCD*) EoS fails to yield a unique solution, then “fall back” to an alternative EoS which can provide a solution
- Available back-ups:
  - “Tanh-conformal” EoS provides better approximation to lattice at mu = 0
  - Conformal
  - Conformal-diagonal
- Explicit parametrizations in backup slides

Comparison: lattice vs. "back-up" EoSs



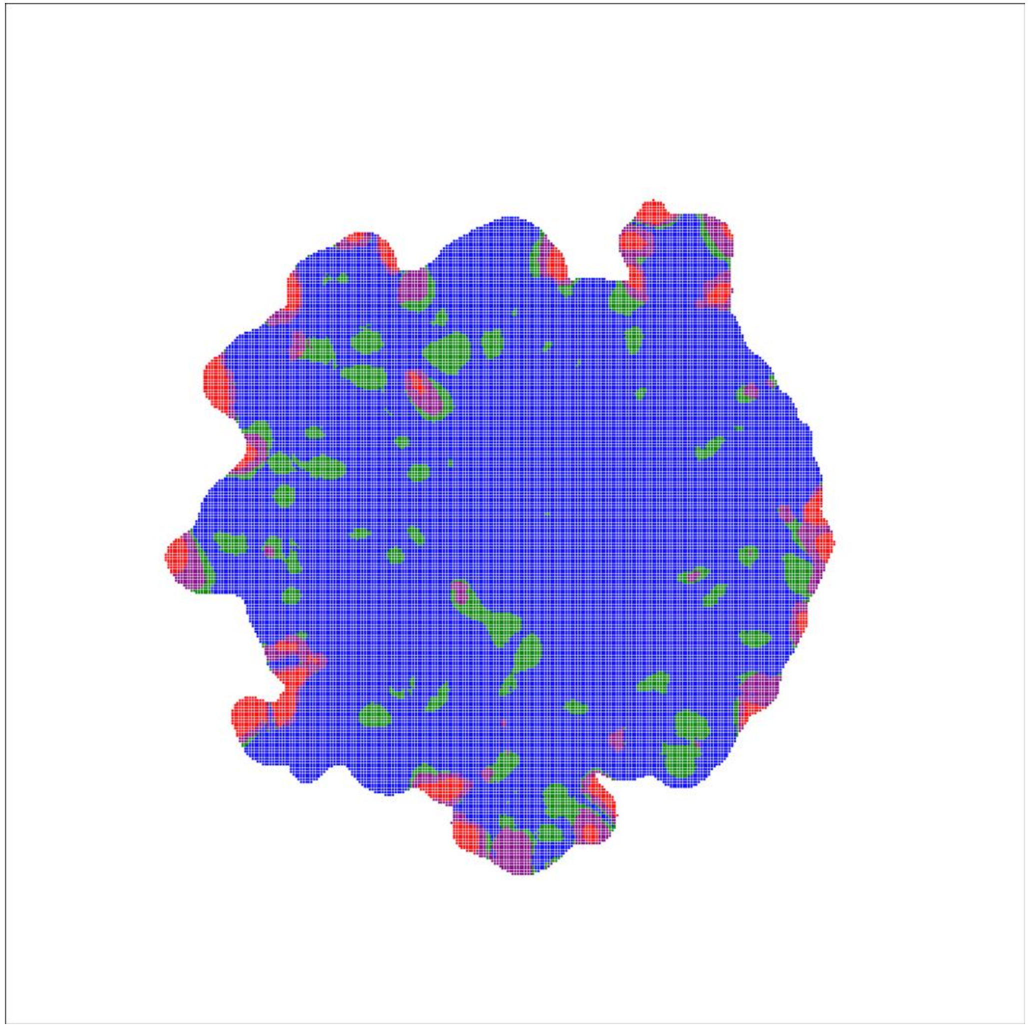


- Conformal-diagonal reduces to Conformal when $\mu_{\mathrm{B,S,Q}} = 0$
- Total energy depends on both energy and pressure
- Total integrated violations below ~0.5%

Animation of different types in hydro

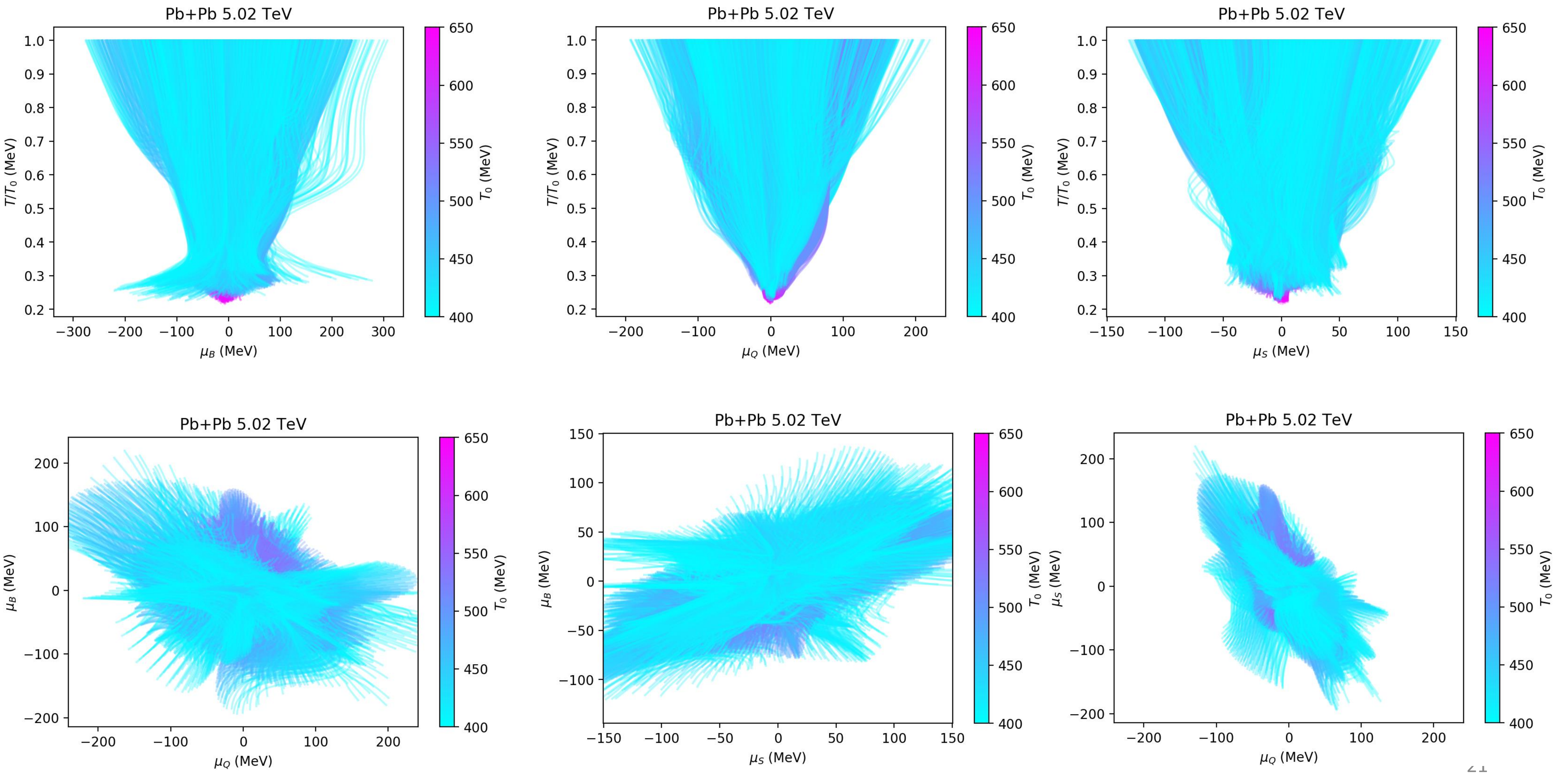
- Typical (central) Pb+Pb event showing EoS for each fluid cell

- **Blue: Table**
- **Green: Tanh-conformal**
- **Purple: Conformal**
- **Red: Conformal-diagonal**

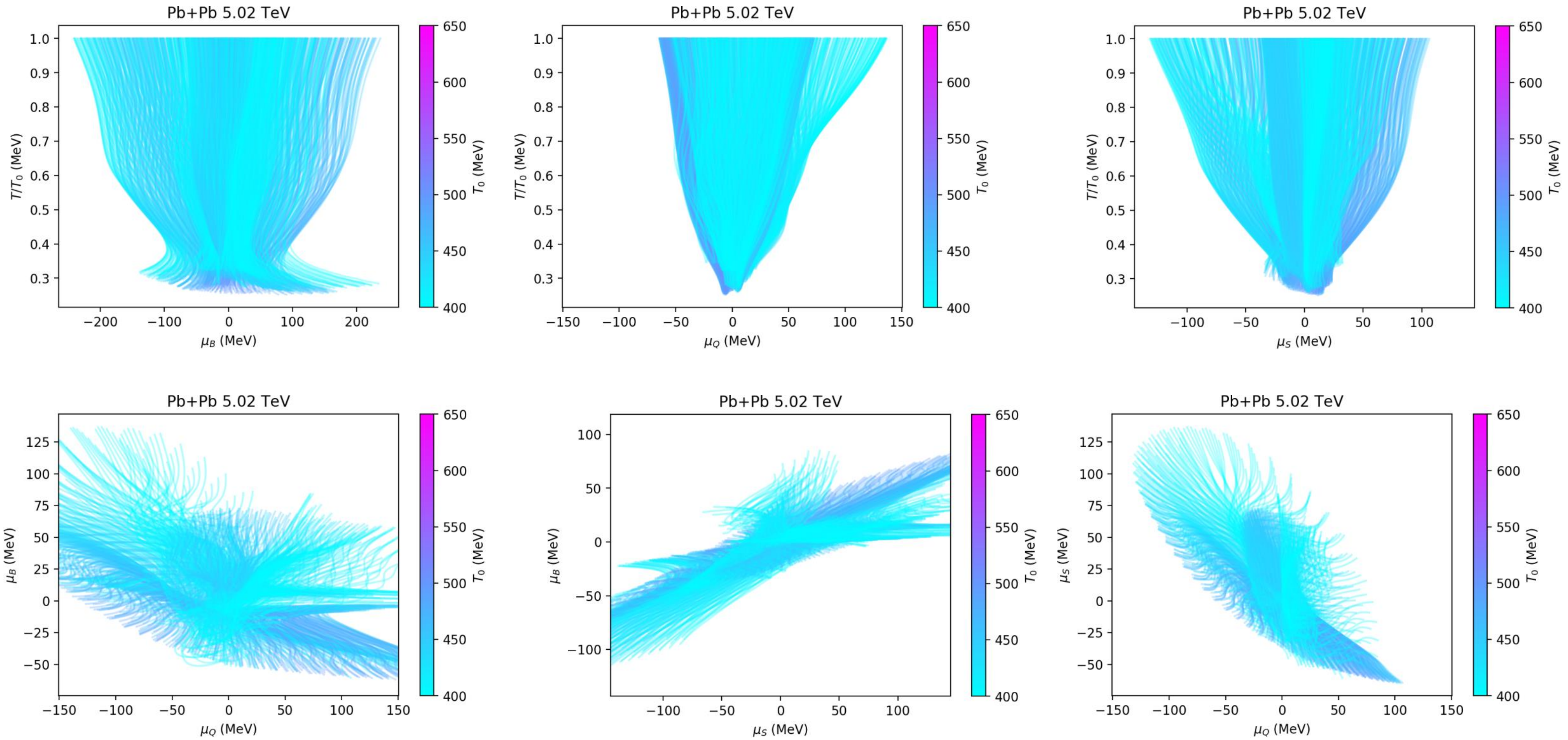


How well do HICs probe the QCD phase diagram?

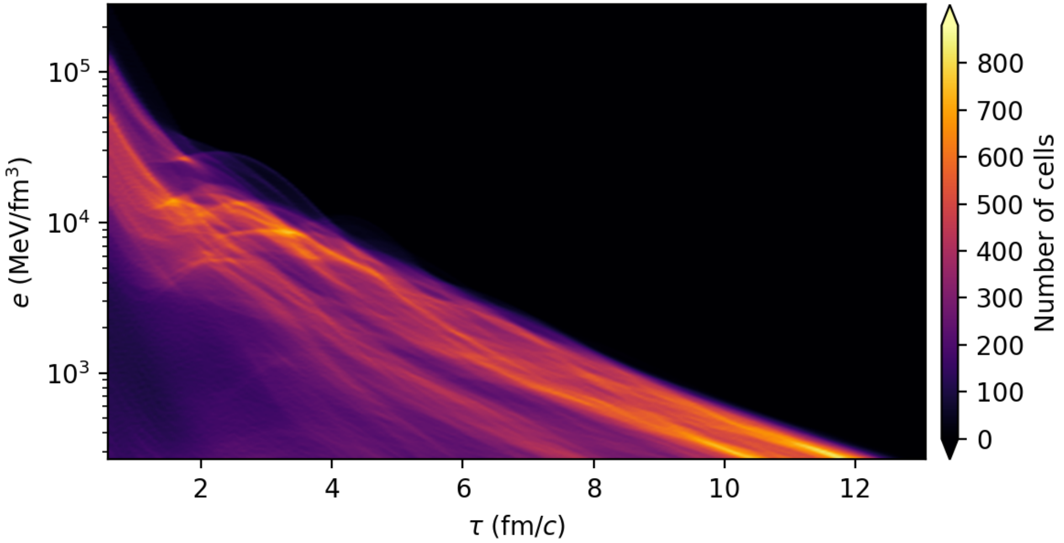
Phase diagram trajectories (0-5%)

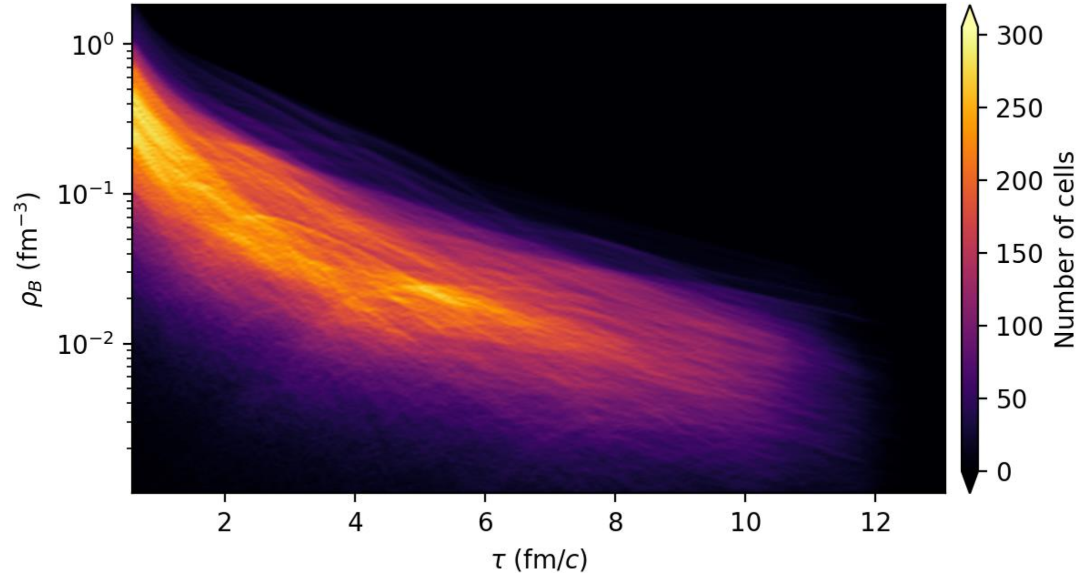


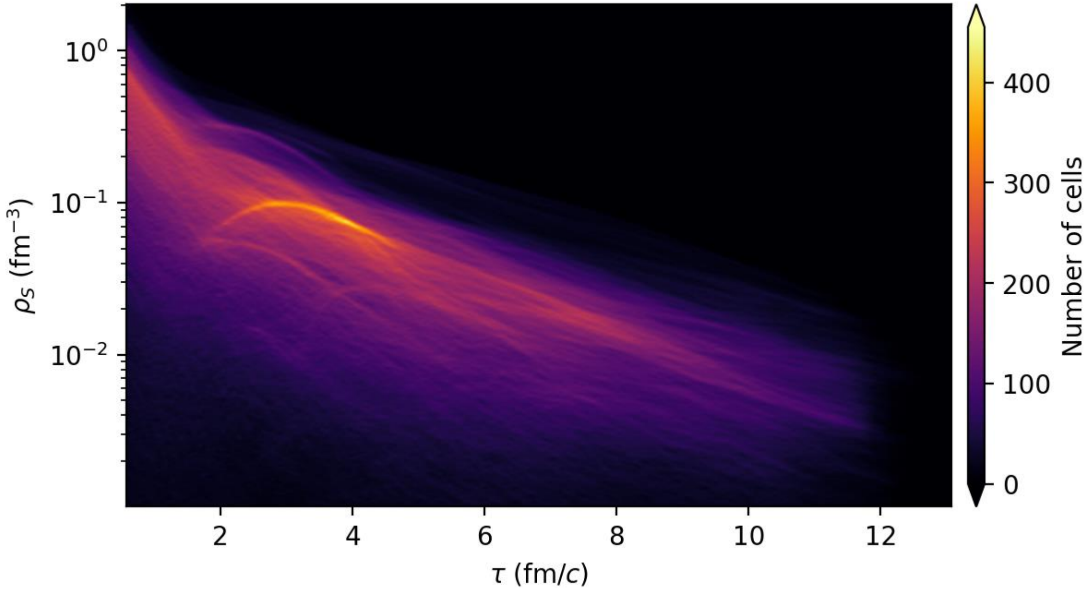
Phase diagram trajectories (20-30%)

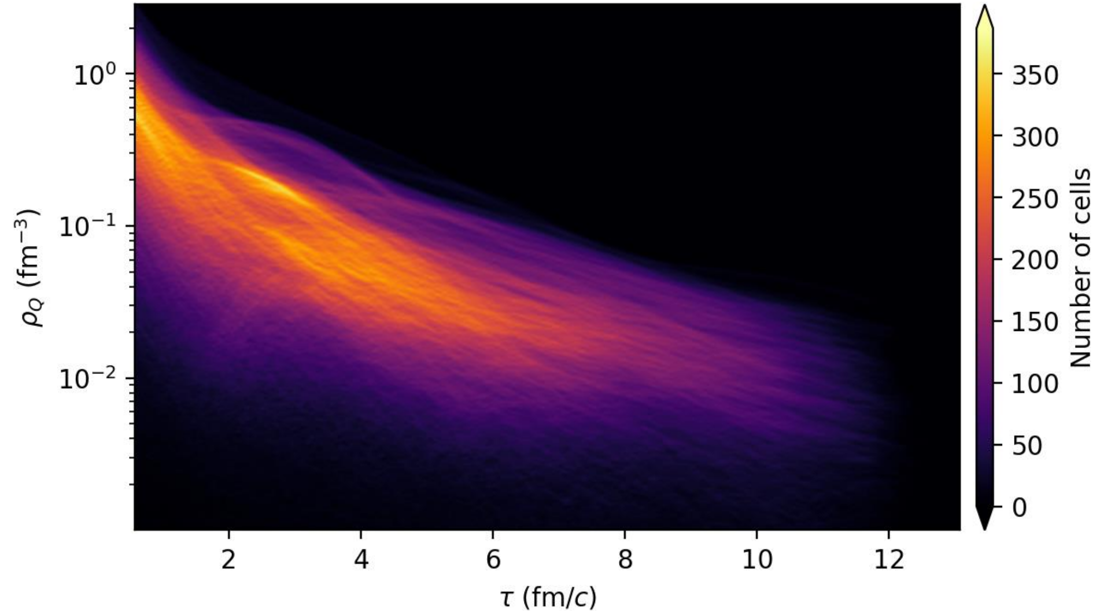


Density distributions (0-5)%

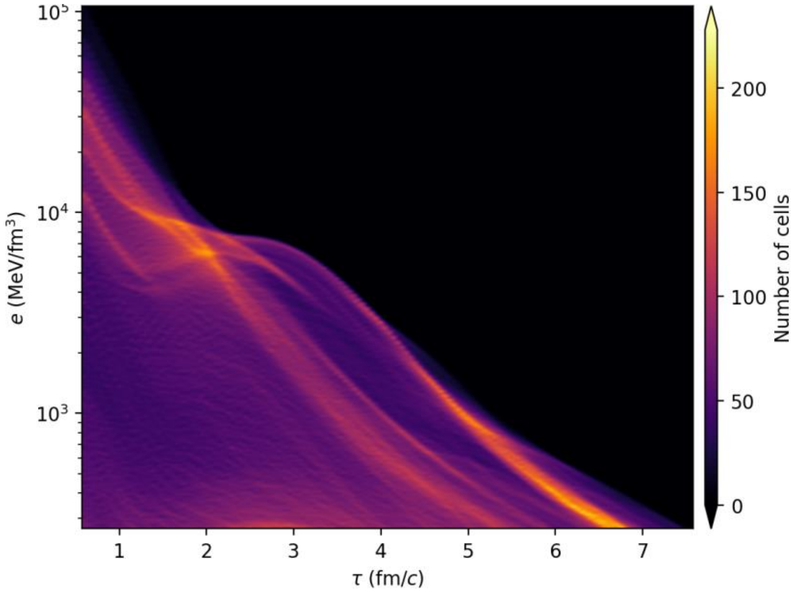




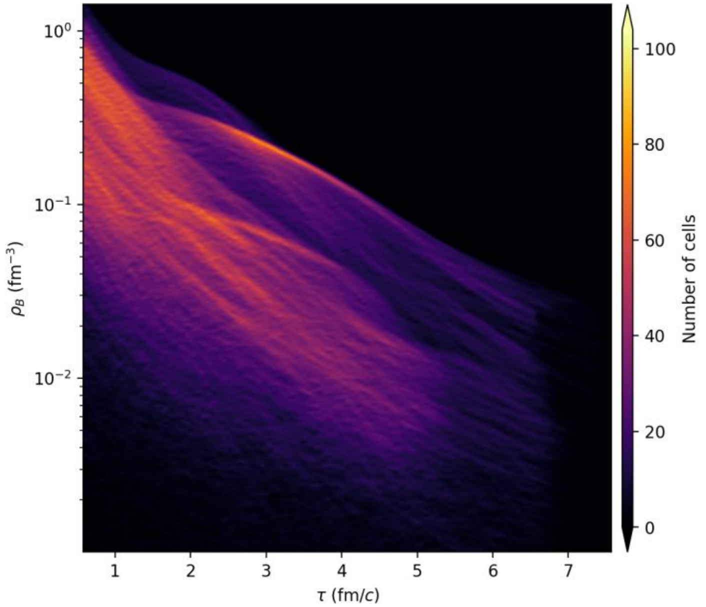


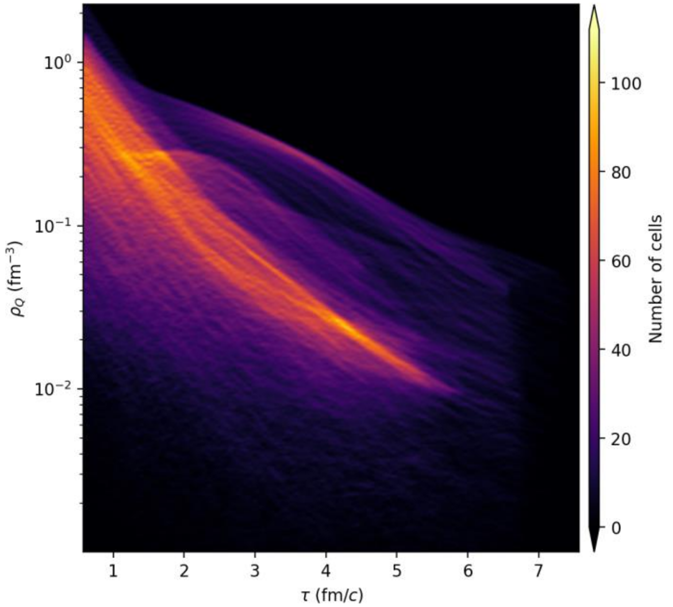


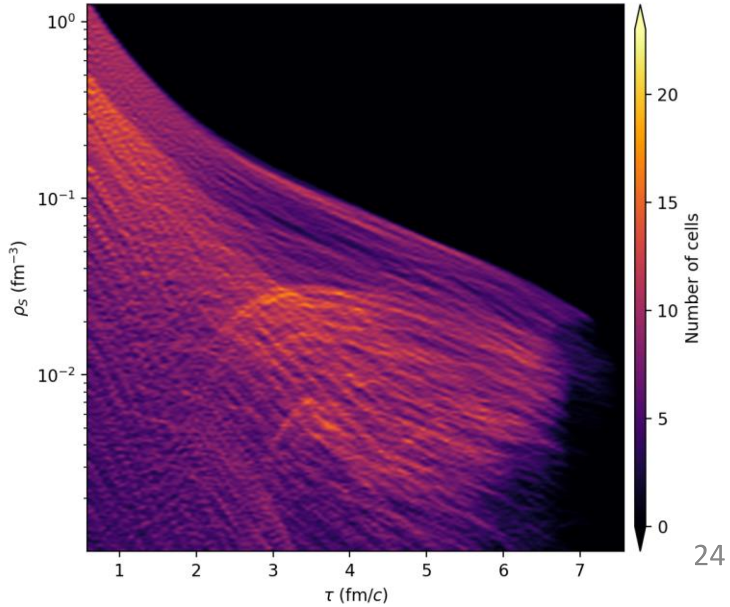
Density distributions (20-30)%



- Particle trajectories sample wide range of chemical potentials and densities, even at LHC energies
- Holds for central and mid-central collisions
- Prospect of constraining wide swath of QCD phase diagram using current (and future) HI experiments







Summary

- Several challenges facing transport/hydrodynamics codes with multiple conserved charges
  - Multiple charges requires knowledge of multi-dimensional (>=4D) EoS
  - Charge fluctuations can reach large values even at LHC energies, requiring more complete coverage of multi-dimensional space
  - Need to find fast approaches to inverting EoS
  - Inversion of EoS for given input densities may be an ill-posed problem
    - Possibility of *multiple* solutions
    - Possibility of *no* solutions
  - Development needed for BSQ initial conditions, transport coefficients, etc.
- Some possible solutions
  - Delaunay interpolation + *k*-d trees
  - Back-up equations of state
    (*how to minimize e-conservation violations?*)

Thank you!

Backup slides

“Back-up” EoS #1: “Tanh-conformal”

- Definition:

$$p_{\rm tc}(T,\mu_B,\mu_S,\mu_Q) = c\tanh\left(\frac{T-T_c}{T_s}\right)\left(\left(\frac{T}{T_0}\right)^2+\left(\frac{\mu_B}{\mu_{B,0}}\right)^2+\left(\frac{\mu_S}{\mu_{S,0}}\right)^2+\left(\frac{\mu_Q}{\mu_{Q,0}}\right)^2\right)^2$$

- Scale parameters determined to mimic tabulated EoS at high T as closely as possible:

$$\begin{aligned}
c &\equiv p_{\rm table}(T_{\rm max},0,0,0)/T_{\rm max}^4\\
T_0 &\equiv 1\\
p_{\rm c}(T_{\rm max},\mu_{B,{\rm max}},0,0;\mu_{B,0}) &\equiv p_{\rm table}(T_{\rm max},\mu_{B,{\rm max}},0,0)\\
p_{\rm c}(T_{\rm max},0,\mu_{S,{\rm max}},0;\mu_{S,0}) &\equiv p_{\rm table}(T_{\rm max},0,\mu_{S,{\rm max}},0)\\
p_{\rm c}(T_{\rm max},0,0,\mu_{Q,{\rm max}};\mu_{Q,0}) &\equiv p_{\rm table}(T_{\rm max},0,0,\mu_{Q,{\rm max}})
\end{aligned}$$

- Two additional parameters in tanh() chosen to mimic transition to HRG:

$$T_c = 220\ \text{MeV},\ T_s = 120\ \text{MeV}$$

“Back-up” EoS #2: “Conformal”

- Definition:

$$p_{\mathrm{c}}(T, \mu_B, \mu_S, \mu_Q) = c\left(\left(\frac{T}{T_0}\right)^2 + \left(\frac{\mu_B}{\mu_{B,0}}\right)^2 + \left(\frac{\mu_S}{\mu_{S,0}}\right)^2 + \left(\frac{\mu_Q}{\mu_{Q,0}}\right)^2\right)^2,$$

- Not the most general (any quartic combinations are acceptable)
- Scale parameters determined as in “Tanh-conformal”
- Overall factor c determined by

$$c \equiv \frac{\pi^2}{90}\left(2\left(N_c^2 - 1\right) + \frac{7}{2}N_c N_f\right)$$

where

$$N_c = 3 \text{ and } N_f = 2.5$$

“Back-up” EoS #3: “Conformal-diagonal”

- Definition:

$$p_{\mathrm{cd}}(T,\mu_B,\mu_S,\mu_Q)=c\left(\left(\frac{T}{T_0}\right)^4+\left(\frac{\mu_B}{\mu_{B,0}}\right)^4+\left(\frac{\mu_S}{\mu_{S,0}}\right)^4+\left(\frac{\mu_Q}{\mu_{Q,0}}\right)^4\right),$$

- Scale parameters determined as in “Tanh-conformal” and “Conformal”
- Overall factor c same as “Conformal”
- One can prove

$$e\geq e_{\min}\left(\vec{\rho}\right)=\frac{3}{4\cdot 2^{2/3}c^{1/3}}\left(\left(\mu_{B,0}\left|\rho_B\right|\right)^{4/3}+\left(\mu_{S,0}\left|\rho_S\right|\right)^{4/3}+\left(\mu_{Q,0}\left|\rho_Q\right|\right)^{4/3}\right)$$

  is a necessary and sufficient condition for given set of (e,ρ_B,ρ_S,ρ_Q) to have a real solution
- If one propagates (s,ρ_B,ρ_S,ρ_Q), then a real solution is always guaranteed

Code demo: Delaunay $(e, \rho_B, \rho_S, \rho_Q)$ interpolator

```
//==========================================
// read path to input file from command line
string path_to_file = string(argv[1]);

//==========================================
// set up EoS object
cout << "Initializing equation of state "
             "interpolator:" << endl;
cout << "  --> reading in equation of state "
             "table from: " << path_to_file << endl;
eos_delaunay EoS( path_to_file );

//==========================================
// vectors to store input densities and
// interpolated result for (T,muB,muQ,muS)
vector<double> result(4, 0.0);
vector<double> point({ 5754.35,  0.00231029,
                       0.351709, 0.378919   }); // just an example
```

Code demo: Delaunay $(e, \rho_B, \rho_S, \rho_Q)$ interpolator

```
//=========================================
// call the interpolator
const size_t n_repeat = 1000;
cout << "Calling the interpolator "
     << n_repeat << " times for test point: \n"
                    "  --> {e,B,S,Q} = {"
     << point[0] << " MeV/fm^3," << point[1] << " 1/fm^3, "
     << point[2] << " 1/fm^3, "  << point[3] << " 1/fm^3}" << endl;

// multiple calls to improve timing estimate
for (size_t i = 0; i < n_repeat; i++)
  EoS.interpolate(point, result);
```

Invocation: `$ ./interpolate_ebsq eos.dat`

```
================================================================================
= Code:     Equation of state interpolator
= Purpose:  Performance and closure tests of Delaunay interpolator
= Author:   Christopher Plumberg
= Contact:  plumberg@illinois.edu
= Date:     April 28, 2022
================================================================================

Initializing equation of state interpolator:
  --> reading in equation of state table from: eos.dat
       - read in 1000000 lines.
       - read in 2000000 lines.
       - read in 3000000 lines.
       - read in 4000000 lines.
       - read in 5000000 lines.
       - read in 6000000 lines.
       - read in 7000000 lines.
  --> check minima and maxima:
       - e:     0.0151087     729992      [MeV/fm^3]
       - B:     -25.56        25.56       [1/fm^3]
       - S:     -65.1234      65.1234     [1/fm^3]
       - Q:     -37.5927      37.5927     [1/fm^3]
  --> setting up kd-trees: finished in 7.63744 seconds!

Calling the interpolator 1000 times for test point:
  --> {e,B,S,Q} = {5754.35 MeV/fm^3,0.00231029 1/fm^3, 0.351709 1/fm^3, 0.378919 1/fm^3}
  --> exact result (units MeV):          252.5    52.5    52.5    52.5
  --> interpolated result (units MeV):   252.448    52.5715    52.571    52.5597

Average time to interpolate  0.00512986 s.
Total runtime: 56.6377 s.
```

Moderate grid size

Reasonable accuracy

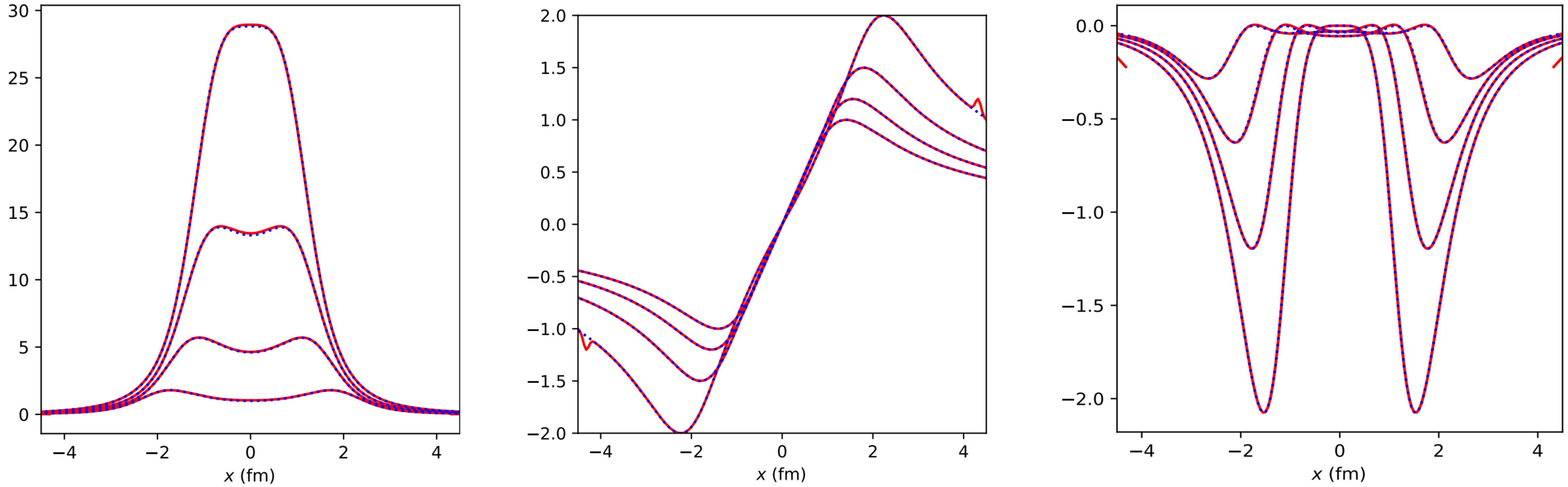
~ 200 solutions$/s$

Gubser checks

Blue (dotted): exact

Red (solid): CCAKE

$$\tau = 1.0,\ 1.2,\ 1.5,\ 2.0\ \mathrm{fm}/c$$



Energy density: $e\ (\mathrm{fm}^{-4})$ | Flow velocity: u^r | Shear stress: $\pi^{xx}\ (\mathrm{fm}^{-4})$