An Introduction to Computational Fluctuating Hydrodynamics

Alejandro L. Garcia[1], John B. Bell[2], Andrew Nonaka[2], and Changho Kim[3]

[1]San Jose State University
[2]Lawrence Berkeley National Laboratory
[3]University of California Merced

June 7, 2024

These notes are an introduction to fluctuating hydrodynamics (FHD) and the formulation of numerical schemes for the resulting stochastic partial differential equations. Fluctuating hydrodynamics was originally introduced by Landau and Lifshitz [1] as a way to put thermal fluctuations into a continuum framework by including a stochastic forcing to each dissipative transport process (e.g., heat flux). While FHD has been useful in modeling transport and fluid dynamics at the microscopic scale, theoretical calculations have been feasible only with simplifying assumptions. As such there is great interest in numerical schemes for Computational Fluctuating Hydrodynamics (CFHD). There are a variety of algorithms (e.g., spectral, finite element, lattice Boltzmann) but in this introduction we focus on finite volume schemes. Accompanying these notes is a demonstration program in Python available on GitHub.

Stochastic Heat Equation

Our first example is the stochastic heat equation (SHE) for the transport of internal energy by thermal diffusion. This section presents a short derivation, for details see [2]. We start with the continuity equation for energy

$$\frac{\partial}{\partial t}(\rho e) = -\nabla \cdot \boldsymbol{Q} \tag{1}$$

where ρ is the (constant) mass density and e is the specific internal energy. The heat flux, $\boldsymbol{Q}$, can be separated into deterministic and stochastic contributions and is written in Onsager form as

$$\boldsymbol{Q} = \overline{\boldsymbol{Q}} + \widetilde{\boldsymbol{Q}} = L\boldsymbol{X} + \widetilde{\boldsymbol{Q}} \tag{2}$$

where L is the Onsager coefficient and $\boldsymbol{X}$ is the thermodynamic force. Fluctuation-dissipation gives that the stochastic flux has zero mean and correlation

$$\langle \widetilde{\boldsymbol{Q}}(\mathbf{r},t)\widetilde{\boldsymbol{Q}}(\mathbf{r}',t')\rangle = 2k_B L\ \delta(\mathbf{r}-\mathbf{r}')\ \delta(t-t') \tag{3}$$

Readers familiar with the Langevin equation for Brownian motion [3] will notice the similarities between that stochastic ODE and the above stochastic PDE.

From non-equilibrium thermodynamics [2, 4], the deterministic total rate of entropy change in a region Ω is

$$\frac{dS}{dt} = \int_\Omega \frac{ds}{dt}\, d\mathbf{r} = \int_\Omega \boldsymbol{X} \cdot \boldsymbol{J}\, d\mathbf{r} - \left[\frac{\overline{\boldsymbol{Q}}}{T}\right]_{\partial\Omega} \tag{4}$$

where $\boldsymbol{J}$ is the thermodynamic flux. The first term is the internal entropy production and the second is the change due to heat flow at the boundary. Using the Gibbs equation, $du = \rho de = T ds$, gives

$$\int_\Omega \frac{ds}{dt}\, d\mathbf{r} = \int_\Omega \frac{\rho}{T}\frac{de}{dt}\, d\mathbf{r} = -\int_\Omega \frac{1}{T}\nabla \cdot \overline{\boldsymbol{Q}}\, d\mathbf{r} = \int_\Omega \frac{1}{T}\nabla \cdot (\lambda \nabla T)\, d\mathbf{r} \tag{5}$$

The last equality uses the phenomenological Fourier law for heat flow, $\overline{\boldsymbol{Q}} = -\lambda \nabla T$, where λ is the thermal conductivity. Integrating by parts gives

$$\int_\Omega \frac{1}{T}\nabla \cdot (\lambda \nabla T)\, d\mathbf{r} = \left[-\frac{\overline{\boldsymbol{Q}}}{T}\right]_{\partial\Omega} - \int_\Omega \nabla\frac{1}{T} \cdot (\lambda\nabla T)\, d\mathbf{r} \tag{6}$$

so the internal entropy production can be expressed as

$$\int_\Omega \boldsymbol{X} \cdot \boldsymbol{J}\, d\mathbf{r} = -\int_\Omega \nabla\frac{1}{T} \cdot (\lambda \nabla T)\, d\mathbf{r} = \int_\Omega \nabla\frac{1}{T} \cdot (\lambda T^2\, \nabla\frac{1}{T})\, d\mathbf{r} \tag{7}$$

From this equality, $\boldsymbol{X} \cdot \boldsymbol{J} = \boldsymbol{X} \cdot (L\boldsymbol{X})$, and Eq. (2), we can identity the thermodynamic force, the thermodynamics flux and the Onsager coefficient to be [4, 5]

$$\boldsymbol{X} = \nabla\frac{1}{T} \quad , \quad \boldsymbol{J} = \lambda T^2\, \nabla\frac{1}{T} = \overline{\boldsymbol{Q}} \quad \text{and} \quad L = \lambda T^2, \tag{8}$$

respectively. The noise correlation is then given by

$$\langle \widetilde{\boldsymbol{Q}}(\mathbf{r},t)\widetilde{\boldsymbol{Q}}(\mathbf{r}',t')\rangle = 2k_B\lambda T^2\ \delta(\mathbf{r}-\mathbf{r}')\ \delta(t-t') \tag{9}$$

Finally, writing $e = c_V T$ the stochastic heat equation is

$$\rho c_V \frac{\partial T}{\partial t} = \nabla \cdot (\lambda \nabla T + \sqrt{2k_B\lambda T^2}\ \widetilde{\boldsymbol{Z}}) \tag{10}$$

where $\widetilde{\mathbf{Z}}$ is a white noise vector uncorrelated in space and time, that is,

$$\langle \widetilde{\boldsymbol{Z}}_s(\mathbf{r},t)\widetilde{\boldsymbol{Z}}_{s'}(\mathbf{r}',t')\rangle = \ \delta(\mathbf{r}-\mathbf{r}')\ \delta(t-t')\ \delta_{s,s'} \tag{11}$$

where $s \in \{x, y, z\}$ is a Cartesian coordinate direction.

For simplicity we focus on the one-dimensional case (e.g., thin rod) and write the stochastic heat equation in compact form as

$$\begin{align} \partial_t T &= \partial_x(\kappa \partial_x T + \alpha T \tilde{Z}) \tag{12}\\ &= \kappa \partial_x^2 T + \alpha \partial_x T\tilde{Z} \tag{13}\end{align}$$

where $\kappa = \lambda/\rho c_V$ and $\alpha = \sqrt{2k_B\lambda}/\rho c_V$ are taken to be constants. In the one-dimensional case the noise variance is

$$\langle \tilde{Z}(x,t)\tilde{Z}(x',t')\rangle = \frac{1}{\mathcal{A}}\ \delta(x-x')\ \delta(t-t') \tag{14}$$

where $\mathcal{A}$ is the cross-sectional area of the system.

CFHD Schemes for SHE

Equation (13) can be numerically integrated in time to produce sample trajectories for $T(x,t)$. We discretize space and time as $x_i = i\Delta x$ and $t^n = n\Delta t$ so temperature is $T_i^n = T(x_i, t^n)$. Using forward difference in time

$$\partial_t f \Rightarrow \frac{f_i^{n+1} - f_i^n}{\Delta t} \tag{15}$$

and centered differences in space

$$\partial_x f \Rightarrow \frac{f^n_{i+\frac{1}{2}} - f^n_{i-\frac{1}{2}}}{\Delta x} \qquad \partial_x^2 f \Rightarrow \frac{f^n_{i+1} - 2f^n_i + f^n_{i-1}}{\Delta x^2} \tag{16}$$

For the spatial index the integer values are at cell centers while half-integer values are at cell faces.

With this we have the forward Euler (FE) scheme

$$\begin{aligned} T_i^{n+1} = T_i^n &+ \frac{\kappa \Delta t}{\Delta x^2}\left(T^n_{i+1} - 2T^n_i + T^n_{i-1}\right) \\ &+ \frac{\alpha \Delta t}{\Delta x}\left(T^n_{i+\frac{1}{2}} \tilde{Z}^n_{i+\frac{1}{2}} - T^n_{i-\frac{1}{2}} \tilde{Z}^n_{i-\frac{1}{2}}\right) \end{aligned} \tag{17}$$

For the temperature at a face we can use the arithmetic average, $T^n_{i+\frac{1}{2}} = (T^n_{i+1} + T^n_i)/2$. Similarly, the standard Predictor-Corrector (PC) scheme is

$$\begin{aligned} T_i^{*} = T_i^n &+ \frac{\kappa \Delta t}{\Delta x^2}\left(T^n_{i+1} - 2T^n_i + T^n_{i-1}\right) \\ &+ \frac{\alpha \Delta t}{\Delta x}\left(T^n_{i+\frac{1}{2}} \tilde{Z}^n_{i+\frac{1}{2}} - T^n_{i-\frac{1}{2}} \tilde{Z}^n_{i-\frac{1}{2}}\right) \end{aligned} \tag{18}$$

and

$$\begin{aligned} T_i^{n+1} = \frac{1}{2}\Big[T_i^n + T_i^* &+ \frac{\kappa \Delta t}{\Delta x^2}\left(T^*_{i+1} - 2T^*_i + T^*_{i-1}\right) \\ &+ \frac{\alpha \Delta t}{\Delta x}\left(T^*_{i+\frac{1}{2}} \tilde{Z}^n_{i+\frac{1}{2}} - T^*_{i-\frac{1}{2}} \tilde{Z}^n_{i-\frac{1}{2}}\right)\Big] \end{aligned} \tag{19}$$

where T_i^*, computed in the predictor step, is used in the corrector step.[1]

The discretized noise has variance

$$\langle \tilde{Z}^n_{i+\frac{1}{2}} \tilde{Z}^m_{j+\frac{1}{2}} \rangle = \frac{1}{\mathcal{A}} \frac{\delta_{i,j}}{\Delta x} \frac{\delta_{n,m}}{\Delta t} = \frac{\delta_{i,j}}{\Delta V} \frac{\delta_{n,m}}{\Delta t} \tag{20}$$

where ΔV is the volume of a grid point cell. Numerically this noise is generated as

$$\tilde{Z}^n_{i+\frac{1}{2}} = \frac{1}{\sqrt{\Delta V \Delta t}} \mathfrak{N}^n_{i+\frac{1}{2}} \tag{21}$$

where $\mathfrak{N}$ are independent, normal (Gaussian) distributed psuedo-random numbers.

[1]The random numbers used in the predictor and corrector steps may be the same or different; see [6].

CFHD example in Python

The Python program, `StochasticHeat`, illustrates the numerical methods described in the previous section for the stochastic heat equation. The program is listed in the appendix and can be downloaded as a Jupyter Notebook from `github.com/AlejGarcia/IntroFHD`. The simulation computes trajectories for the one-dimensional stochastic heat equation and analyzes their statistical properties (e.g., variance of temperature). The reader is encouraged to download and run the program since results are obtained in about 5 minutes on a laptop.

At the top of the program there are various global options:

- Periodic or Dirichlet boundary conditions
- Thermodynamic equilibrium or constant temperature gradient
- Initialize with or without temperature perturbations
- Run simulation with or without thermal fluctuations
- Use Forward Euler (FE) or Predictor-Corrector (PC) scheme
- Number of grid cells
- Number of time steps

Other parameters, such as L, A, Δt, etc., can also be changed from their default values within the program. Simulation results in this section are from runs using 32 grid cells, running for either 2×10^6 (equilibrium cases) or 2×10^7 (temperature gradient case) time steps. Default values were used for the other parameters (see program listing in Appendix).

At thermodynamic equilibrium the variance and static correlation of temperature fluctuations are predicted by statistical mechanics to be [7]

$$\langle \delta T_i^2 \rangle = \frac{k_B \langle T_i \rangle^2}{\rho c_V \Delta V} \qquad \text{and} \qquad \langle \delta T_i \delta T_j \rangle = \langle \delta T_i^2 \rangle \; \delta_{i,j} \tag{22}$$

with the average temperature, $\langle T_i \rangle$, equal to the equilibrium temperature, T_{eq}. Figures 1 and 2 show that the simulation results for Dirichlet boundaries (i.e., fixed temperature at the boundaries) are in good agreement with theory, especially for the Predictor-Corrector scheme.

Figure 3 shows that temperature correlations are slightly different for periodic boundary conditions. This is because, due to conservation of energy, $\sum_i \rho c_V \langle \delta T_i \rangle = 0$ so the correlation is shifted by an additive constant such that $\sum_i \langle \delta T_i \delta T_j \rangle = 0$. This is a common feature in systems with conserved variables, such as density fluctuations in closed systems.

The assessment of numerical schemes in CFHD is best done by measuring the static structure factor for fluctuations. For the SHE this is $S_k = \langle \hat{T}_k \hat{T}_k^* \rangle$ where

$$\hat{T}_k = \sum_{j=0}^{N-1} T_j \exp(-2\pi \mathrm{i}\, jk/N) \tag{23}$$

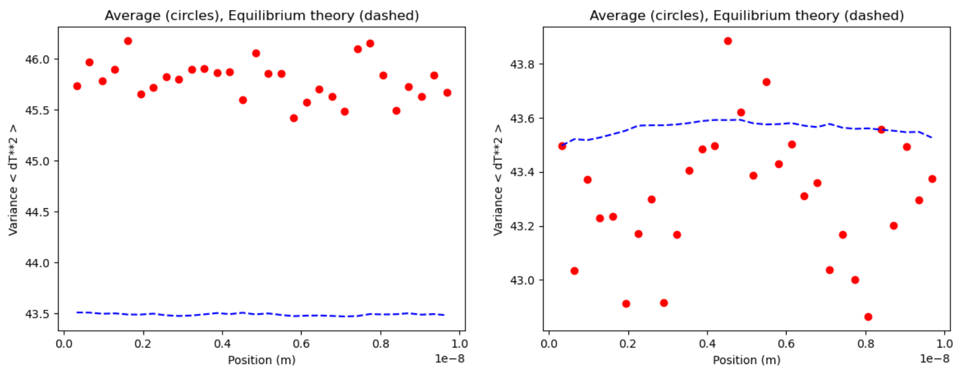

Figure 1: Temperature variance $\langle \delta T_i^2 \rangle$ versus x_i at equilibrium with Dirichlet boundary conditions for (left) FE scheme; (right) PC scheme. Theory line given by (22).

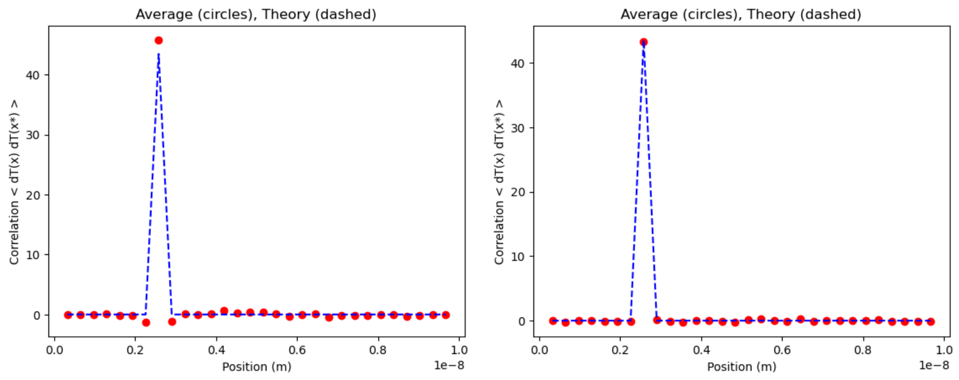

Figure 2: Temperature correlation $\langle \delta T_i \delta T_j \rangle$ versus x_i at equilibrium with Dirichlet boundary conditions with $x_j = L/4$ for (left) FE scheme; (right) PC scheme. Theory line given by (22).

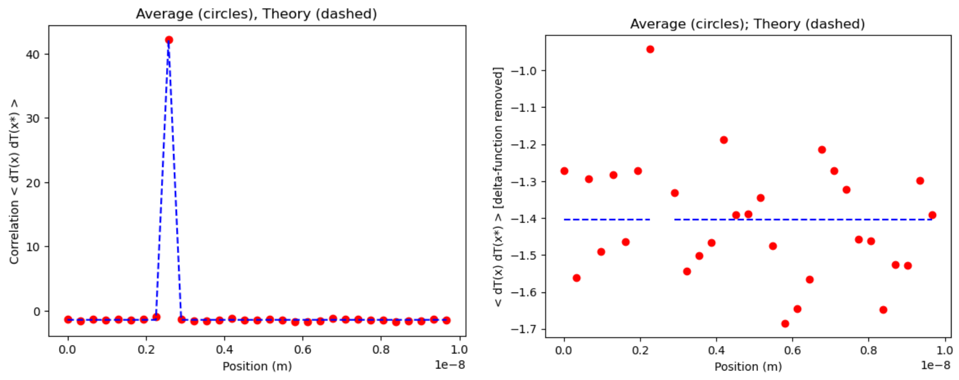

Figure 3: Temperature correlation $\langle \delta T_i \delta T_j \rangle$ versus x_i for periodic boundary conditions using the PC scheme. Results are plotted both with (left) and without (right) the point at $x_j = L/4$.

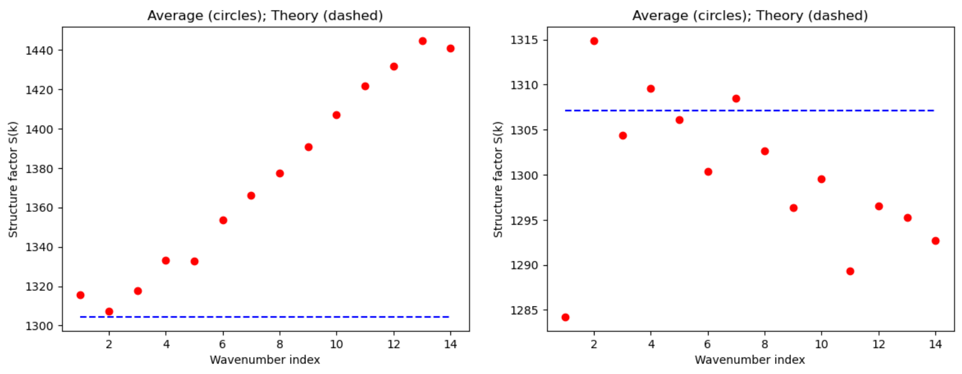

Figure 4: Static structure factor, S_k versus k at equilibrium for (left) FE scheme; (right) PC scheme. Theory line given by (24).

is the discrete Fourier transform with $k = 0, 1, \ldots, N-1$. At thermodynamic equilibrium the discretized structure factor is

$$S_k = \frac{k_B T_{\mathrm{eq}}^2}{\rho c_V} N \tag{24}$$

Figure 4 shows that the FE scheme has significant error for large wavenumber while the PC scheme is accurate for all k. An analysis of discretization errors for various numerical schemes is presented in [6]; for Forward Euler the error in S_k is $O(\kappa \Delta t\, k^2)$ while for Predictor-Corrector it is $O(\kappa^2 \Delta t^2 k^4)$.

Finally we consider a non-equilibrium scenario, specifically a linear temperature gradient imposed by Dirichlet boundary conditions. In this case there is a weak, long-range correlation of temperature fluctuations [8]

$$\langle \delta T_i \delta T_j \rangle = \frac{k_B T_{\mathrm{eq}}^2}{\rho c_V \Delta V} \delta_{i,j} + \frac{k_B (\nabla T)^2}{\rho c_V A L} \times \begin{cases} x_i (L - x_j) & \text{if } x_i < x_j \\ x_j (L - x_i) & \text{otherwise} \end{cases} \tag{25}$$

Figure 5 shows that the simulation results are in good agreement with this theoretical result.

Stochastic Species Diffusion

In fluctuating hydrodynamics there are several stochastic diffusion equations that are closely related to the stochastic heat equation. For example, the diffusion of species mass is described by a similar stochastic partial differential equation with the main difference being the functional form of the stochastic flux. As above we start with

$$\partial_t (\rho c) = -\nabla \cdot \boldsymbol{F} \tag{26}$$

where c is the species concentration and the species flux is

$$\boldsymbol{F} = \overline{\boldsymbol{F}} + \widetilde{\boldsymbol{F}} = \mathcal{L} \boldsymbol{\mathcal{X}} + \widetilde{\boldsymbol{F}} \tag{27}$$

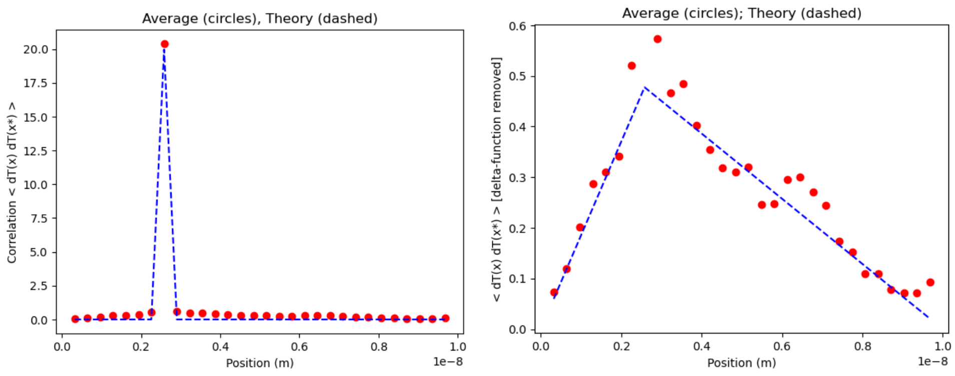

Figure 5: Temperature correlation $\langle \delta T_i \delta T_j \rangle$ versus x_i for linear temperature gradient imposed by Dirichlet boundary conditions using the PC scheme. Results are plotted both with (left) and without (right) the point at $x_j = L/4$. Theory line given by (25).

Assuming that the system is isothermal (i.e., no Soret effect) irreversible thermodynamics tells us that the thermodynamic force is [4, 5]

$$\boldsymbol{\mathcal{X}} = \nabla \left(\frac{\mu}{T} \right) = \frac{\nabla \mu}{T} \tag{28}$$

where μ is the chemical potential. For ideal solutions

$$\mu = \mu_0(T) + \frac{k_B T}{m} \ln c \qquad \text{so} \qquad \nabla \mu = \frac{k_B T}{mc} \, \nabla c. \tag{29}$$

where m is the particle mass. Since the phenomenological law for species flow is Fick's law, $\overline{\boldsymbol{F}} = -\rho D \nabla c$, then the Onsager coefficient is

$$\mathcal{L} = \frac{m \rho D \, c}{k_B} \tag{30}$$

The noise correlation is

$$\langle \widetilde{\boldsymbol{F}}(\mathbf{r}, t) \widetilde{\boldsymbol{F}}(\mathbf{r}', t') \rangle = 2 m \rho D \, c \; \delta(\mathbf{r} - \mathbf{r}') \; \delta(t - t') \tag{31}$$

so the stochastic species diffusion equation is

$$\partial_t \, c = \nabla \cdot \left(D \nabla c + \sqrt{\frac{2D \, c}{n_0}} \; \widetilde{\boldsymbol{Z}} \right) \tag{32}$$

where $n_0 = \rho/m$ is the number density of the pure state ($c = 1$). This may also be written in terms of species number density, $n = c \, n_0$, as

$$\partial_t \, n = \nabla \cdot (D \nabla n + \sqrt{2D \, n} \; \widetilde{\boldsymbol{Z}}) \tag{33}$$

This stochastic partial differential equation is also known as the Dean-Kawasaki equation. Recall that the stochastic heat equation is

$$\partial_t T = \nabla \cdot \left(\kappa \nabla T + \sqrt{\frac{2\kappa k_B T^2}{\rho c_V}}\, \widetilde{\boldsymbol{Z}} \right) \tag{34}$$

The deterministic parts of the two equations above are functionally similar but the stochastic parts are distinctly different.

The stochastic species diffusion equation can also be derived by coarse-graining the discrete random walk model for diffusion. Additional examples of discrete models that have associated stochastic diffusion equations are: the excluded random walk model; the "train model" for the diffusion of momentum [9, 10]; and the "Knudsen chain" model for gas effusion [11].

A Menagerie of Fluctuating Hydrodynamic Equations

The general form of the fluctuating hydrodynamic equations for a set of state variables $\boldsymbol{U}$ is

$$\partial_t \boldsymbol{U} = -\nabla \cdot (\boldsymbol{\mathcal{F}}_H(\boldsymbol{U}) + \boldsymbol{\mathcal{F}}_D(\boldsymbol{U}) + \boldsymbol{\mathcal{N}}(\boldsymbol{U})\widetilde{\boldsymbol{Z}}) \tag{35}$$

where $\boldsymbol{\mathcal{F}}_H$ and $\boldsymbol{\mathcal{F}}_D$ are the hyperbolic and diffusive fluxes with $\boldsymbol{\mathcal{N}}$ being the noise amplitude. For example, for the stochastic species diffusion equation $\boldsymbol{U} = n$, $\boldsymbol{\mathcal{F}}_H = 0$, $\boldsymbol{\mathcal{F}}_D = \nabla n$, and $\boldsymbol{\mathcal{N}} = \sqrt{2Dn}$. In general, the fluctuation-dissipation theorem relates $\boldsymbol{\mathcal{F}}_D$ to $\boldsymbol{\mathcal{N}}$ and tells us that the stochastic term is independent of the hyperbolic term.

This brief introduction to CFHD has focused on stochastic diffusion equations due to the pedagogical benefit of their simplicity. We close with an outline of other fluctuating hydrodynamic equations including references describing finite volume schemes for solving them.

- Stochastic Burger's equation [12]

  $$\partial_t c = \partial_x \left(ac(1-c) - D\,\partial_x c + \sqrt{2Dc(1-c)}\,\tilde{Z} \right) \tag{36}$$

  Note that for $c \ll 1$ this resembles the stochastic species diffusion equation with the addition of a hyperbolic flux, $\boldsymbol{F}_H = ac(1-c)$.

- Stochastic "train model" equations [9, 10]

  $$\partial_t \rho = -\partial_x (D \nabla \rho + \sqrt{2mD\rho}\,\tilde{Z}), \tag{37}$$

  $$\partial_t (\rho u) = -\partial_x (D \nabla (\rho u) + \sqrt{2mD\rho u^2}\,\tilde{Z}) \tag{38}$$

  These equations are a simple model for the transport of momentum density, ρu, in a fluid.

- Reaction-diffusion systems [13]
  The reaction-diffusion system has N_s species diffusing and undergoing N_r reactions. By denoting the number density of species s by n_s, the equations of fluctuating reaction-diffusion can be written formally as the stochastic PDEs

$$\frac{\partial n_s}{\partial t} = \nabla \cdot \left(D_s \nabla n_s + \sqrt{2 D_s n_s} \boldsymbol{\mathcal{Z}}_s^{(D)} \right) + \sum_{r=1}^{N_r} \nu_{sr} \left(a_r(\mathbf{n}) + \sqrt{a_r(\mathbf{n})} \mathcal{Z}_r^{(R)} \right), \quad (39)$$

  where D_s is the diffusion coefficient of species s, $a_r(\mathbf{n})$ is the propensity function indicating the rate of reaction r, and ν_{sr} is the stoichiometric coefficient of species s in reaction r.

- Compressible Navier-Stokes equations (single species fluid) [14, 15]

$$\partial_t \, \rho = -\nabla \cdot (\rho \boldsymbol{u}) \,, \quad (40)$$

$$\partial_t \, (\rho \boldsymbol{u}) = -\nabla \cdot \left[(\rho \boldsymbol{u} \otimes \boldsymbol{u} + p \mathbb{I}) + \overline{\Pi} + \widetilde{\Pi} \right], \quad (41)$$

$$\partial_t \, (\rho E) = -\nabla \cdot \left[(\rho E + p) \, \boldsymbol{u} + \overline{\boldsymbol{Q}} + \widetilde{\boldsymbol{Q}} + \left(\overline{\Pi} + \widetilde{\Pi} \right) \cdot \boldsymbol{u} \right] \quad (42)$$

  where $\boldsymbol{u}$, p, and $\Pi = \overline{\Pi} + \widetilde{\Pi}$ are fluid velocity, pressure, and stress tensor; total specific energy is $E = e + \frac{1}{2} u^2$. Note that (42) reduces to the stochastic heat equation when $\boldsymbol{u} = 0$. See references for explicit expression of the deterministic and stochastic fluxes.

- Compressible Navier-Stokes equations (multi-species fluid) [15, 16]
  Same as above but replace (40) with

$$\partial_t \, \rho_k = -\nabla \cdot (\rho_k \boldsymbol{u} + \overline{\boldsymbol{F}}_k + \widetilde{\boldsymbol{F}}_k) \quad (43)$$

  where ρ_k is the mass density for species k and $\rho = \sum_k \rho_k$. For ideal mixtures the diffusive flux, $\boldsymbol{F}_k$, is similar to that of the species diffusion equation described in the previous section.

- Hydrodynamics plus chemical reactions [17, 18]
  Add source terms to the RHS of (43) for the deterministic and stochastic rates of reactions, $\overline{\Omega}_k$ and $\widetilde{\Omega}_k$. These can be formulated from the chemical Langevin equation [19].

- Incompressible / Low Mach hydrodynamic equations [20, 21, 22, 23]
  The multi-species methodology can be extended to model isothermal mixtures of miscible incompressible liquids. Incompressibility of the fluids leads to a constrained evolution, similar to the incompressible Navier-Stokes equations, given by

$$\partial_t \, (\rho \boldsymbol{u}) = -\nabla \cdot \left[\rho \boldsymbol{u} \otimes \boldsymbol{u} + \overline{\Pi} + \widetilde{\Pi} \right] - \nabla \pi, \quad (44)$$

$$\partial_t \, \rho_k = -\nabla \cdot (\rho_k \boldsymbol{u} + \overline{\boldsymbol{F}}_k + \widetilde{\boldsymbol{F}}_k) \quad (45)$$

$$\nabla \cdot \boldsymbol{u} = -\nabla \cdot \left(\sum_k \frac{\overline{\boldsymbol{F}}_k + \widetilde{\boldsymbol{F}}_k}{\bar{\rho}_k} \right) \quad . \quad (46)$$

Here, π is a perturbational pressure and $\bar{\rho}_k$ is the pure component density for the k^{th} species. The resulting system retains in influence of fluctuations on mixing but is considerable more computationally efficient, particularly for liquids with a high sound speed.

- Stochastic Poisson-Nernst-Planck equations [24, 25]
  The incompressible version of CFHD for multi-species fluids can model electrolyte solutions by including charged species (ions). The chemical potential becomes the electrochemical potential with the electrical mobility given by the Nernst–Einstein relation and the diffusive flux by the Nernst-Planck equation. For scales comparable to the Debye length, the electric field is obtained by solving the Poisson equation. At scales larger than the Debye length, the fluid is electro-neutral. Electro-neutrality can be imposed as a constraint by solving a modified elliptic equation to compute the electric potential.

- Multi-phase fluids [26, 27, 28, 29]
  By adding non-local gradient terms to the free energy, one can incorporate interfacial tension in CFHD model to treat multiphase systems. Models can be created to both single component systems near the critical point and for multicomponent systems. For multicomponent systems, the low Mach number verison of the methodology can also be derived. These types of systems introduce additional higher order operators into the momentum equations and, for multicomponent systems, the species transport equations. See the references for specific forms of these equations.

Finally, `github.com/AMReX-FHD` is a repository of CFHD codes written using the `AMReX` framework.

References

[1] L. D. Landau and E. M. Lifshitz. *Fluid Mechanics, Course of Theoretical Physics, Vol. 6.* Pergamon Press, 1959.

[2] J. M. Ortiz de Zarate and J. V. Sengers. *Hydrodynamic Fluctuations in Fluids and Fluid Mixtures.* Elsevier Science, 2007.

[3] Crispin W Gardiner et al. *Handbook of stochastic methods*, volume 3. springer Berlin, 1985.

[4] S. R. DeGroot and P. Mazur. *Non-Equilibrium Thermodynamics.* North-Holland Publishing Company, Amsterdam, 1963.

[5] A. L. Garcia. *Essentials of Modern Thermodynamics.* Amazon, 2022.

[6] Aleksandar Donev, Eric Vanden-Eijnden, Alejandro Garcia, and John Bell. On the accuracy of finite-volume schemes for fluctuating hydrodynamics. *Communications in Applied Mathematics and Computational Science*, 5(2):149–197, 2010.

[7] Raj Kumar Pathria. *Statistical mechanics.* Elsevier, 2016.

[8] Alejandro L Garcia, M Malek Mansour, George C Lie, and Enrico Clementi. Numerical integration of the fluctuating hydrodynamic equations. *Journal of statistical physics*, 47:209–228, 1987.

[9] Alejandro Garcia, F Baras, and M Malek Mansour. A simple model for nonequilibrium fluctuations in a fluid. *American Journal of Physics*, 64:1488, 1996.

[10] Francis J Alexander, Alejandro L Garcia, and Daniel M Tartakovsky. Algorithm refinement for stochastic partial differential equations: Ii. correlated systems. *Journal of Computational Physics*, 207(2):769–787, 2005.

[11] Alejandro L. Garcia. Thermal fluctuations in a Knudsen flow system. *Physics Letters A*, 119(8):379–382, January 1987.

[12] John B Bell, Jasmine Foo, and Alejandro L Garcia. Algorithm refinement for the stochastic burgers' equation. *Journal of Computational Physics*, 223(1):451–468, 2007.

[13] Changho Kim, Andy Nonaka, John B Bell, Alejandro L Garcia, and Aleksandar Donev. Stochastic simulation of reaction-diffusion systems: A fluctuating-hydrodynamics approach. *The Journal of chemical physics*, 146(12), 2017.

[14] John B Bell, Alejandro L Garcia, and Sarah A Williams. Numerical methods for the stochastic landau-lifshitz navier-stokes equations. *Physical Review E*, 76(1):016708, 2007.

[15] Ishan Srivastava, Daniel R Ladiges, Andy J Nonaka, Alejandro L Garcia, and John B Bell. Staggered scheme for the compressible fluctuating hydrodynamics of multispecies fluid mixtures. *Physical Review E*, 107(1):015305, 2023.

[16] Kaushik Balakrishnan, Alejandro L Garcia, Aleksandar Donev, and John B Bell. Fluctuating hydrodynamics of multispecies nonreactive mixtures. *Physical Review E*, 89(1):013017, 2014.

[17] Amit Kumar Bhattacharjee, Kaushik Balakrishnan, Alejandro L Garcia, John B Bell, and Aleksandar Donev. Fluctuating hydrodynamics of multispecies reactive mixtures. *The Journal of chemical physics*, 142(22), 2015.

[18] Changho Kim, Andy Nonaka, John B Bell, Alejandro L Garcia, and Aleksandar Donev. Fluctuating hydrodynamics of reactive liquid mixtures. *The Journal of chemical physics*, 149(8), 2018.

[19] Daniel T Gillespie. The chemical langevin equation. *The Journal of Chemical Physics*, 113(1):297–306, 2000.

[20] Aleksandar Donev, Andy Nonaka, Yifei Sun, Thomas Fai, Alejandro Garcia, and John Bell. Low mach number fluctuating hydrodynamics of diffusively mixing fluids. *Communications in Applied Mathematics and Computational Science*, 9(1):47–105, 2014.

[21] Aleksandar Donev, Andy Nonaka, Amit Kumar Bhattacharjee, Alejandro L Garcia, and John B Bell. Low mach number fluctuating hydrodynamics of multispecies liquid mixtures. *Physics of Fluids*, 27(3), 2015.

[22] Andrew Nonaka, Yifei Sun, John Bell, and Aleksandar Donev. Low mach number fluctuating hydrodynamics of binary liquid mixtures. *Communications in Applied Mathematics and Computational Science*, 10(2):163–204, 2015.

[23] Florencio Balboa, John B Bell, Rafael Delgado-Buscalioni, Aleksandar Donev, Thomas G Fai, Boyce E Griffith, and Charles S Peskin. Staggered schemes for fluctuating hydrodynamics. *Multiscale Modeling & Simulation*, 10(4):1369–1408, 2012.

[24] Jean-Philippe Péraud, Andy Nonaka, Anuj Chaudhri, John B Bell, Aleksandar Donev, and Alejandro L Garcia. Low mach number fluctuating hydrodynamics for electrolytes. *Physical Review Fluids*, 1(7):074103, 2016.

[25] Aleksandar Donev, Andrew J Nonaka, Changho Kim, Alejandro L Garcia, and John B Bell. Fluctuating hydrodynamics of electrolytes at electroneutral scales. *Physical Review Fluids*, 4(4):043701, 2019.

[26] Barry Z Shang, Nikolaos K Voulgarakis, and Jhih-Wei Chu. Fluctuating hydrodynamics for multiscale simulation of inhomogeneous fluids: Mapping all-atom molecular dynamics to capillary waves. *The Journal of chemical physics*, 135(4), 2011.

[27] Anuj Chaudhri, John B Bell, Alejandro L Garcia, and Aleksandar Donev. Modeling multiphase flow using fluctuating hydrodynamics. *Physical Review E*, 90(3):033014, 2014.

[28] Katherine Klymko, Andrew Nonaka, John B Bell, Sean P Carney, and Alejandro L Garcia. Low mach number fluctuating hydrodynamics model for ionic liquids. *Physical Review Fluids*, 5(9):093701, 2020.

[29] Bryn Barker, John B Bell, and Alejandro L Garcia. Fluctuating hydrodynamics and the Rayleigh–Plateau instability. *Proceedings of the National Academy of Sciences*, 120(30):e2306088120, 2023.

APPENDIX: Python program StochHeat

```
# StochasticHeat - Program to calculate trajectories for the stochastic heat equation

# Set up configuration options and special features
import numpy as np                # Use numerical python library
import matplotlib.pyplot as plt    # Use plotting python library
# Display plots in notebook window
get_ipython().run_line_magic('matplotlib', 'inline')


# In[ ]:


#* Set option flags and key parameters for the simulation
BC_FLAG = 0                  # Boundary condition flag (0: Periodic; 1: Dirichlet)
NONEQ_FLAG = 0               # Non-equilibrium flag (0: Thermodynamic equilibrium; 1: Temperature gradient)
if( BC_FLAG == 0 and NONEQ_FLAG == 1 ):
    print('ERROR: Periodic boundary conditions are only for thermodynamic equilibrium')
    NONEQ_FLAG = 0           # Reset the flag to equilibrium
PERTURB_FLAG = 1             # Start from perturbed initial condition (0: No; 1: Yes)
STOCH_FLAG = 1               # Thermal fluctuations? (0: No, deterministic; 1: Yes, stochastic)
SCHEME_FLAG = 1              # Numerical scheme (0: Forward Euler; 1: Predictor-corrector)

Ncells = 32                  # Number of cells (including boundary cells '0' and 'Ncells-1' )
MegaSteps = 2                # Number of simulation timesteps in millions
# Recommend at least 2 million steps for equilibrium systems, 20 million for non-equilibrium systems


# In[ ]:


#* Set physical parameters for the system (iron bar)
kB = 1.38e-23       # Boltzmann constant (J/K)
mAtom = 9.27e-26    # Mass of iron atom (kg)
rho = 7870.         # Mass density of iron (kg/m^3)
c_V = 450.          # Specific heat capacity of iron (J/(kg K))
ThCond = 70.        # Thermal conductivity of iron (W/(m K))
Length = 1.0e-8     # System length (m)
Area = (5.0e-9)**2  # System cross-sectional area (m^2)


# In[ ]:


#* Set numerical parameters (time step, grid spacing, etc.).
if( BC_FLAG == 0 ):
    cellLo = 0;  cellHi = Ncells-1;     # For periodic BC, range of cells = { 0, 1, ..., Ncells-2 }
    faceLo = 0;  faceHi = Ncells-1;     # Face ii is left side of cell ii
else:
    cellLo = 1;  cellHi = Ncells-1;     # For Dirichlet BC, range of cells = { 1, 2, ..., Ncells-2 }
    faceLo = 1;  faceHi = Ncells-1;     # Face ii is left side of cell ii

dx = Length/(Ncells-1)     # Grid size (m)
dV = Area*dx               # Volume of grid cell (m^3)
xx = np.zeros(Ncells)
for ii in range(Ncells):
    xx[ii] = ii * dx       # Cell center positions (m)

kappa = ThCond / (rho * c_V )                # Coefficient in deterministic heat equation
# Coefficient in stochastic heat equation
if( STOCH_FLAG == 1 ):
    alpha = np.sqrt( 2 * kB * kappa / (rho * c_V) )
else:
    alpha = 0.  # Set alpha = 0 to turn off thermal fluctuations

stabilityFactor = 0.1                         # Numerical stability if stabilityFactor < 1.
dt = stabilityFactor * dx**2 /( 2. * kappa )  # Timestep (s)


# In[ ]:


#* Set initial conditions for temperature
Tref = 300.      # Reference temperature (K)
if( NONEQ_FLAG == 0 ):
    T_Left = Tref; T_Right = Tref
else:
    Tdiff = 400.   # Temperature difference across the system
    T_Left = Tref - Tdiff/2.; T_Right = Tref + Tdiff/2.

# Standard deviation of temperature in a cell at the reference temperature
Tref_SD = np.sqrt(kB*Tref**2 / (rho * c_V * dV))

# Set initial temperature and its deterministic steady-state value
T = np.zeros(Ncells); T0 = np.zeros(Ncells)
```

```
for ii in range(cellLo,cellHi):
    T0[ii] = T_Left + (T_Right-T_Left) * ii/(Ncells-1)      # Linear profile
    T[ii] = T0[ii]
    if( PERTURB_FLAG == 1 ):
        T[ii] += Tref_SD*np.random.randn()  # Add random perturbation

if( BC_FLAG == 0 ):
    T0[cellHi] = T0[0]       # Copy first cell into last cell for periodic BCs
    T[cellHi] = T[0]
else:
    T0[0] = T_Left;   T[0] = T_Left      # Set values in first and last cells
    T0[-1] = T_Right; T[-1] = T_Right


# In[ ]:


#* Summarize system parameters and initial state
print( 'System is iron bar with about ', int(rho*Length*Area/mAtom), ' atoms' )
print( 'System length = ',Length*1.0e9,' (nm); volume = ',
      (Length*Area)*1.0e27,' (nm**3)' )
print( 'Number of timesteps = ', MegaSteps,' million' )
print( 'Time step = ', dt*1.e15, ' (fs)' )
print( 'Number of cells = ', Ncells )
if( BC_FLAG == 0 ):
    print('** PERIODIC boundary conditions **')
else:
    print('++ DIRICHLET boundary conditions ++')

if( NONEQ_FLAG == 0 ):
    print('Equilibrium temperature = ',Tref,' Kelvin')
else:
    print('Fixed temperatures (K) = ',T_Left,' left; ',T_Right,' right')
print( 'At T = ',Tref,'K, standard deviation sqrt(< dT^2 >) = ', Tref_SD,'K' )
if( SCHEME_FLAG == 0 ):
    print('-- Forward Euler Scheme --')
else:
    print('== Predictor-Corrector Scheme ==')

# Plot temperature versus position
plt.plot(xx, T,'g*',xx, T0,'b--',xx[0],T[0],'sk',xx[-1],T[-1],'sk')
plt.xlabel('Position (m)'); plt.ylabel('Temperature (K)')
plt.title('Initial state (stars), steady state (dashed), boundary cells (squares)')
plt.show()


# In[ ]:


#* Initialize sampling plus misc. coefficients and arrays
Nsamp = 0                  # Count number of statistical samples
sumT  = np.zeros(Ncells)    # Running sum of T_i ; used to compute mean
sumT2 = np.zeros(Ncells)    # Running sum of (T_i)**2 ; used to compute variance
sumTT = np.zeros(Ncells)    # Running sum for correlation T_i * T_iCorr
sumSk = np.zeros(Ncells)    # Running sum for structure factor S(k)

iCorr = np.int_(Ncells/4)   # Grid point used for correlation

# Coefficients used in the main loop calculations
coeffDetFE = kappa * dt / dx**2
coeffStoFE = alpha * dt / dx
coeffZnoise = 1. / np.sqrt( dt * dV )

# Arrays used in the main loop calculations
Determ = np.zeros(Ncells); Stoch = np.zeros(Ncells)
Znoise = np.zeros(Ncells); Tface = np.zeros(Ncells)
PreT = np.copy(T)             # Used by Predictor-Corrector scheme
Spectrum = np.zeros(Ncells)   # Used to compute structure factor S(k)


# In[ ]:


#* Loop over the desired number of time steps.
NstepInner = 10                                       # Number of time steps (inner loop)
NstepOuter = np.int_(MegaSteps*1000000/NstepInner)    # Number of time steps (outer loop)
NskipOuter = NstepOuter/10        # Number of outer steps to skip before sampling begins
NdiagOuter = NstepOuter/20        # Number of outer steps between diagnostic outputs

for iOuter in range(NstepOuter):     # Outer loop

    # Print diagnostics
    if (iOuter % NdiagOuter) == 0 :
        print( 'Finished ',np.int_(100*iOuter/NstepOuter),' percent of the time steps')

    for iInner in range(NstepInner):       # Inner loop
```

```
        # Deterministic update for temperature
        for ii in range(cellLo,cellHi):
            Determ[ii] = coeffDetFE * ( T[ ii+1 ] + T[ ii-1 ] - 2*T[ ii ] )
        if( BC_FLAG == 0 ):
            Determ[0] = coeffDetFE * ( T[ 1 ] + T[ cellHi-1 ] - 2*T[ 0 ] )  # Periodic BC

        # Generate random noise Z
        Znoise = coeffZnoise * np.random.normal(size=Ncells)
        if( BC_FLAG == 0 ):
            Znoise[-1] = Znoise[0]   # Periodic BC

        # Tface[ i ] is average between T of cells i-1 and i; value on the left face of cell i
        for ii in range(cellLo,cellHi):
            Tface[ii] = 0.5 * (T[ ii-1 ] + T[ ii ])
        if( BC_FLAG == 0 ):
            Tface[cellHi] = Tface[0]   # Periodic BC
        else:
            Tface[cellHi] = 0.5 * (T[ cellHi-1 ] + T[ cellHi ])   # Dirichlet BC

        # Stochastic update for temperature
        for ii in range(cellLo,cellHi):
            Stoch[ii] = coeffStoFE * ( Tface[ii+1]*Znoise[ii+1] - Tface[ii]*Znoise[ii] )

        if( SCHEME_FLAG == 0 ):
            # Forward Euler scheme
            for ii in range(cellLo,cellHi):
                T[ii] += Determ[ii] + Stoch[ii]   # Total update for temperature
        else:
            # Predictor-Corrector scheme
            for ii in range(cellLo,cellHi):
                PreT[ii] = T[ii] + Determ[ii] + Stoch[ii]     # Predictor step
            if( BC_FLAG == 0 ):
                PreT[cellHi] = PreT[0]     # Periodic BC
            # Corrector step
            for ii in range(cellLo,cellHi):
                Determ[ii] = coeffDetFE * ( PreT[ ii+1 ] + PreT[ ii-1 ] - 2*PreT[ ii ] )
            if( BC_FLAG == 0 ):
                Determ[0] = coeffDetFE * ( PreT[ 1 ] + PreT[ cellHi-1 ] - 2*PreT[ 0 ] )  # Periodic BC
            for ii in range(cellLo,cellHi):
                Tface[ii] = 0.5 * (PreT[ ii-1 ] + PreT[ ii ])
            if( BC_FLAG == 0 ):
                Tface[cellHi] = Tface[0]   # Periodic BC
            else:
                Tface[cellHi] = 0.5 * (PreT[ cellHi-1 ] + PreT[ cellHi ])
            for ii in range(cellLo,cellHi):
                Stoch[ii] = coeffStoFE * ( Tface[ii+1]*Znoise[ii+1] - Tface[ii]*Znoise[ii] )
            for ii in range(cellLo,cellHi):
                T[ii] = 0.5*( T[ii] + PreT[ii] + Determ[ii] + Stoch[ii] )

        if( BC_FLAG == 0 ):
            T[cellHi] = T[0]     # Periodic BC

    # End of Inner loop

  # Take statistical sample
    if( iOuter > NskipOuter ):
        Nsamp += 1
        for ii in range(cellLo,cellHi):
            sumT[ii] += T[ii]            # Running sum for temperature average
            sumT2[ii] += T[ii]**2        # Running sum for temperature variance
            sumTT[ii] += T[ii]*T[iCorr]  # Running sum for temperature correlation

        if( NONEQ_FLAG == 0 ):
            # Take Fourier transform and record sampled spectrum in running sum
            Spectrum[cellLo:cellHi] = np.abs( np.fft.fft( T[cellLo:cellHi] ) )**2
            for ii in range(cellLo,cellHi):
                sumSk[ii] += Spectrum[ii]

# End of Outer loop


# In[ ]:


#* Calculate average, variance, and correlation
aveT = np.zeros(Ncells)     # Average < T_i >
varT = np.zeros(Ncells)     # Variance < (T_i - <T_i>)**2 >
corrT = np.zeros(Ncells)    # Correlation < T_i T_iCorr >

for ii in range(cellLo,cellHi):
    aveT[ii] = sumT[ii] / Nsamp
    varT[ii] = sumT2[ii]/Nsamp - aveT[ii]**2
for ii in range(cellLo,cellHi):
    corrT[ii] = sumTT[ii]/Nsamp - aveT[ii]*aveT[iCorr]


# In[ ]:
```

```
#* Plot temperature versus x
plt.plot(xx[cellLo:cellHi], T[cellLo:cellHi],'g*',
         xx[cellLo:cellHi],aveT[cellLo:cellHi],'ro',xx[cellLo:cellHi],T0[cellLo:cellHi],'b--')
plt.xlabel('Position (m)'); plt.ylabel('Temperature (K)')
plt.title('Final state (stars), Average (circles), Deterministic (line)')
plt.show()


# In[ ]:


#* Plot varicance of temperature versus x; compare with theory
varT_Th = np.zeros(Ncells)   # Theoretical value
varT_Th[cellLo:cellHi] = kB*aveT[cellLo:cellHi]**2 / (rho * c_V * dV)
# Conservation of energy correction for periodic boundary case
if( BC_FLAG == 0 ):
    varT_Th *= (1 - 1/(Ncells-1))

plt.plot(xx[cellLo:cellHi], varT[cellLo:cellHi],'ro',
         xx[cellLo:cellHi],varT_Th[cellLo:cellHi],'b--')
plt.xlabel('Position (m)'); plt.ylabel('Variance < dT**2 >')
plt.title('Average (circles), Equilibrium theory (dashed)')
plt.show()


# In[ ]:


#* Plot temperature cell correlation versus x
varTheory = kB*aveT[iCorr]**2 / (rho * c_V * dV)  # Temperature variance in cell iCorr
if( BC_FLAG == 0 ):
    corrT_Th = - varTheory * 1/(Ncells-1) * np.ones(Ncells)  # Conservation of energy correction
    corrT_Th[iCorr] = varTheory * (1 - 1/(Ncells-1))         # for periodic boundary case
else:
    corrT_Th = np.zeros(Ncells)                              # Kronecker delta correlation
    corrT_Th[iCorr] = varTheory                              # for Dirichlet boundary case

plt.plot(xx[cellLo:cellHi], corrT[cellLo:cellHi],'ro',
         xx[cellLo:cellHi],corrT_Th[cellLo:cellHi],'b--')
plt.xlabel('Position (m)'); plt.ylabel('Correlation < dT(x) dT(x*) >')
plt.title('Average (circles), Theory (dashed)')
plt.show()
print('Correlation for x* = ', xx[iCorr],' (m)')


# In[ ]:


#* Plot temperature cell correlation versus x WITHOUT the equilibrium delta function
corrTd = np.array(corrT)
corrTd[iCorr] = np.nan        # Delete data point at iCorr using NaN

# Calculate theoretical values for < dT(x) dT(x') > where x' = xx[iCorr]
if( NONEQ_FLAG == 0 ):
    corrTd_Th = np.array(corrT_Th)
    corrTd_Th[iCorr] = np.nan        # Delete data point at iCorr using NaN
else:
    corrTd_Th = np.zeros(Ncells)
    for ii in range(Ncells):
        # Non-equilibrium correlation; see J. Stat. Phys. 47 209 (1987)
        corrTd_Th[ii] = kB*(T_Right-T_Left)**2 / (rho * c_V * Area * Length**3)
        if( ii < iCorr ):
            corrTd_Th[ii] *= xx[ii]*(Length - xx[iCorr])
        else:
            corrTd_Th[ii] *= xx[iCorr]*(Length - xx[ii])

plt.plot(xx[cellLo:cellHi], corrTd[cellLo:cellHi],'ro',
         xx[cellLo:cellHi],corrTd_Th[cellLo:cellHi],'b--')
plt.xlabel('Position (m)'); plt.ylabel('< dT(x) dT(x*) > [delta-function removed]')
plt.title('Average (circles); Theory (dashed)')
plt.show()


# In[ ]:


#* Compute the measured structure factor and compare with linear theory
if( NONEQ_FLAG == 0 ):     # Only compute S(k) for equilibrium
    NN = len(range(cellLo,cellHi))
    Sk = np.zeros(NN)                           # Power spectrum S(k), AKA the structure factor
    Sk[0:NN] = sumSk[cellLo:cellHi] / Nsamp     # Average S(k)
    Nyq = np.int_(np.floor(NN/2))               # Nyquist frequency is kSpect[NNNy+1]
    ki = np.arange(1,Nyq)                       # Wavenumber index, skipping the zero index
    Sk_eq = varTheory*NN * np.ones(Nyq+1)       # Equilibrium structure factor, which is constant
```

```
# Plot structure factor, skipping the k=0 wavenumber index
plt.plot( ki, Sk[1:Nyq],'ro', ki, Sk_eq[1:Nyq],'b--',)
plt.xlabel('Wavenumber index'); plt.ylabel('Structure factor S(k)')
plt.title('Average (circles); Theory (dashed)')
plt.show()
```