(25.10.23) Computational Physics B

class 1

- Richtinger meshkov instability
  - shock gives the pressure gradient, and can interact with the density gradient to generate vorticity.
  - similar to RT instability, but shock driven.
- How to map astrophysical phenomenon to numerical simulation?
  - "Computers do not know your units."
  - We choose the units to ensure the dimensionless numbers "invariant", i.e. these numbers should correspond to the real physical system whatever units we choose.
    - The specific values of physical quantities dose not matter, only their ratios matter.
  - What are the characteristic lengthscales/timescales?
    - timescale: dynamic timescale $t_{\rm dyn} \sim \frac{L_{\rm box}}{v}$, cloud crushing timescale $t_{\rm cc} \sim \frac{R_{\rm cloud}}{v}$, sound crossing timescale $t_{\rm sc} \sim \frac{L_{\rm box}}{c_s}$.
      - for example, $\frac{t_{\rm sc}}{t_{\rm dyn}} = \frac{L_{\rm box}/c_s}{L_{\rm box}/v} = \frac{v}{c_s} = \mathcal{M}$, is th mach number, which is important in physical systems.
      - and this gives requirements on our initial settings, e.g. $t_{\rm max} > t_{\rm grow}$.
    - determine gas ρ, T, P, E, v.
      - these quantities are not independent, using equation of state to determine all these quantities.
- Parallel computing
  - Grndahl law?
    - speedup $S_p = \frac{1}{(1-f)+\frac{f}{p}}$, where f is parallelized function, $1-f$ is the serial function, p is the number of processors.
    - efficiency $E_p = \frac{S_p}{p}$, shows unit CPU's contribution to speedup. E_p drops quickly with increasing p.
      - $1-f>0$,
      - the cost of communication between different processors ,
      - more ghost zones increase the computation cost.
- HOMEWORK: WRITE A CODE TO DO PARALLELIZATION USING MPI.

class 2

I cannot understand anything about MPI or any other things, and I give up it.

- How to solve ODE?
  - a system of 1st order ODEs: $\frac{d\vec{y}}{dt} = \vec{f}(t, \vec{y})$
    - explicit method
      - $y_k^{n+1} = y_k^n + \Delta t \cdot f_k(t^n, \vec{y}^n)$
      - 1st order Euler method
      - quantity at $n+1$ time step is determined by quantity at n time step.
      - most popular explicit method for ODEs: 4th order Runge-Kutta method (RK4)
        - $y_k^{n+1} = y_k^n + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4)$, where
          - $k_1 = f_k(t^n, \vec{y}^n)$
          - $k_2 = f_k(t^n + \frac{\Delta t}{2}, \vec{y}^n + \frac{\Delta t}{2}\vec{k}_1)$
          - $k_3 = f_k(t^n + \frac{\Delta t}{2}, \vec{y}^n + \frac{\Delta t}{2}\vec{k}_2)$
          - $k_4 = f_k(t^n + \Delta t, \vec{y}^n + \Delta t\vec{k}_3)$
        - note that we still use the value at n time step to calculate the value at $n+1$ time step.
        - RK4 is a multistage method, means we need to calculate several intermediate steps to get slopes and finally use a weighted slope to get the final value.
    - implicit method
      - $y_k^{n+1} = y_k^n + \Delta t \cdot f_k(t^{n+1}, \vec{y}^{n+1})$
      - the quantity at $n+1$ time step is determined by the quantity at $n+1$ time step itself.
      - this seems possible to solve "stiff" problems? -- the function is sensitive to small changes, so small changes in step size can lead to large changes in the result.