Parallelization and Vectorization of nuDust

Ezra Brooker,
Sarah Stangl

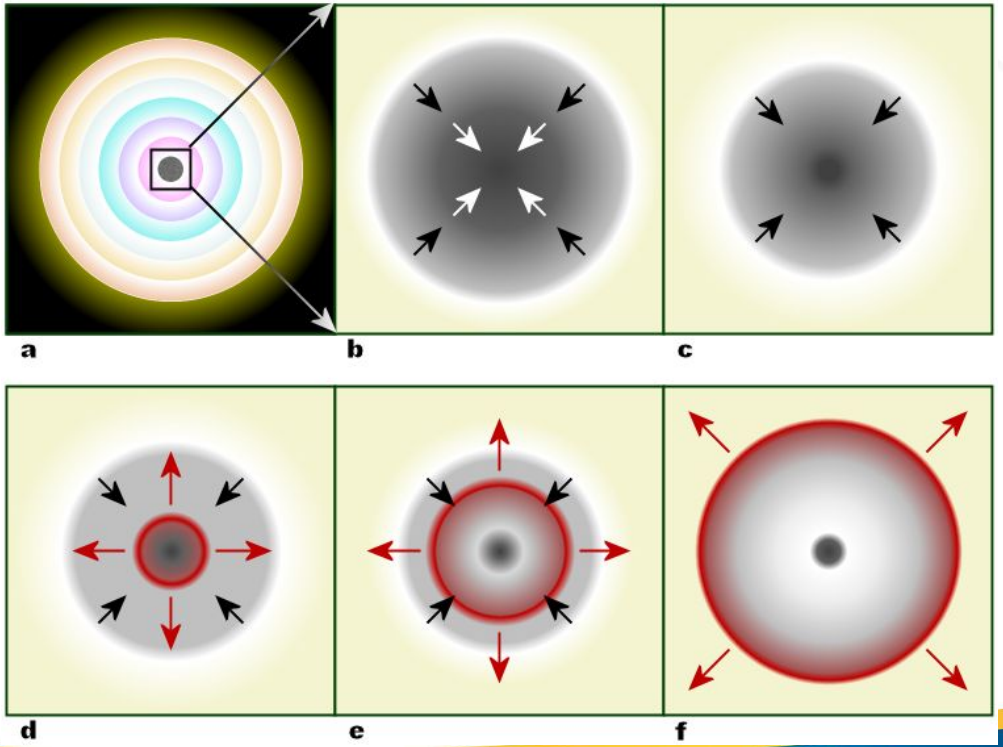
Mentors: Chris Fryer, Chris Mauney

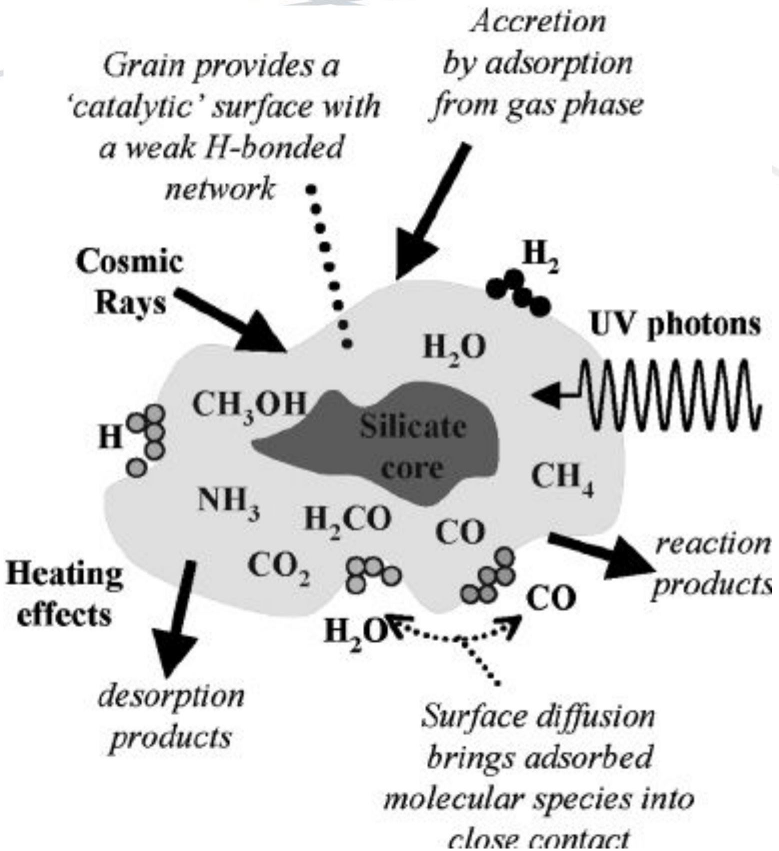
LA-UR-20-26163

What is Dust?

- Affects Observations
  - re-emits light in longer wavelengths
- Seed for more complicated molecules
  - Needed for water to form
- Enriches ISM, proto-galaxies/stars
- Multi-Messenger signal
- Sources of Dust
  - AGB Atmospheres
  - Supernova Outflows
  - Formation in Cold ISM

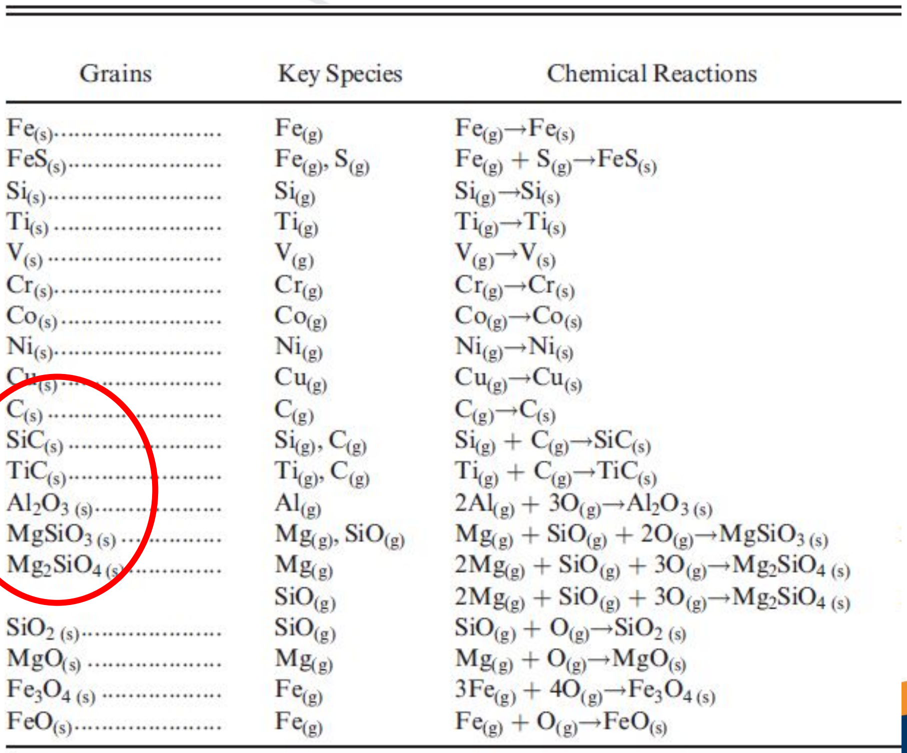
CCSN




Grain provides a 'catalytic' surface with a weak H-bonded network
Accretion by adsorption from gas phase
Cosmic Rays
H_2
UV photons
H_2O
CH_3OH
Silicate core
H
CH_4
NH_3
H_2CO
CO
CO_2
Heating effects
reaction products
CO
H_2O
desorption products
Surface diffusion brings adsorbed molecular species into close contact

Formation of Dust - Key Species

- Nucleation rate
  - governed by key species
    - the reaction rate is much larger than the decay rate
    - species with the least collisional frequency, controls nucleation and growth



Grains	Key Species	Chemical Reactions
$Fe_{(s)}$	$Fe_{(g)}$	$Fe_{(g)} \rightarrow Fe_{(s)}$
$FeS_{(s)}$	$Fe_{(g)}$, $S_{(g)}$	$Fe_{(g)} + S_{(g)} \rightarrow FeS_{(s)}$
$Si_{(s)}$	$Si_{(g)}$	$Si_{(g)} \rightarrow Si_{(s)}$
$Ti_{(s)}$	$Ti_{(g)}$	$Ti_{(g)} \rightarrow Ti_{(s)}$
$V_{(s)}$	$V_{(g)}$	$V_{(g)} \rightarrow V_{(s)}$
$Cr_{(s)}$	$Cr_{(g)}$	$Cr_{(g)} \rightarrow Cr_{(s)}$
$Co_{(s)}$	$Co_{(g)}$	$Co_{(g)} \rightarrow Co_{(s)}$
$Ni_{(s)}$	$Ni_{(g)}$	$Ni_{(g)} \rightarrow Ni_{(s)}$
$Cu_{(s)}$	$Cu_{(g)}$	$Cu_{(g)} \rightarrow Cu_{(s)}$
$C_{(s)}$	$C_{(g)}$	$C_{(g)} \rightarrow C_{(s)}$
$SiC_{(s)}$	$Si_{(g)}$, $C_{(g)}$	$Si_{(g)} + C_{(g)} \rightarrow SiC_{(s)}$
$TiC_{(s)}$	$Ti_{(g)}$, $C_{(g)}$	$Ti_{(g)} + C_{(g)} \rightarrow TiC_{(s)}$
$Al_2O_{3\,(s)}$	$Al_{(g)}$	$2Al_{(g)} + 3O_{(g)} \rightarrow Al_2O_{3\,(s)}$
$MgSiO_{3\,(s)}$	$Mg_{(g)}$, $SiO_{(g)}$	$Mg_{(g)} + SiO_{(g)} + 2O_{(g)} \rightarrow MgSiO_{3\,(s)}$
$Mg_2SiO_{4\,(s)}$	$Mg_{(g)}$	$2Mg_{(g)} + SiO_{(g)} + 3O_{(g)} \rightarrow Mg_2SiO_{4\,(s)}$
	$SiO_{(g)}$	$2Mg_{(g)} + SiO_{(g)} + 3O_{(g)} \rightarrow Mg_2SiO_{4\,(s)}$
$SiO_{2\,(s)}$	$SiO_{(g)}$	$SiO_{(g)} + O_{(g)} \rightarrow SiO_{2\,(s)}$
$MgO_{(s)}$	$Mg_{(g)}$	$Mg_{(g)} + O_{(g)} \rightarrow MgO_{(s)}$
$Fe_3O_{4\,(s)}$	$Fe_{(g)}$	$3Fe_{(g)} + 4O_{(g)} \rightarrow Fe_3O_{4\,(s)}$
$FeO_{(s)}$	$Fe_{(g)}$	$Fe_{(g)} + O_{(g)} \rightarrow FeO_{(s)}$

Nozawa et al. 2003

Dust Growth via grain nucleation

- Growth (key species)
  - material collides and sticks to the grain
  - once the key species is used up, reaction stops
  - abundance of key species is determined by a system of coupled nonlinear ODEs
- Moment Equations
  - number density, radius, surface area, key species depletion

$$\frac{dr_j}{dt} = \alpha_{sj}\Omega_j \left(\frac{kT}{2\pi m_{1j}}\right)^{1/2} c_{1j}(t) = \frac{1}{3} a_{0j} \tau_{\mathrm{coll},j}^{-1}(t)$$

$$\frac{dK_j^{(0)}}{dt} = \frac{J_j(t)}{\tilde{c}_{1j}(t)} \frac{4\pi}{3\Omega_j}$$

$$\frac{dK_j^{(i)}}{dt} = \frac{J_j(t)}{\tilde{c}_{1j}(t)} \frac{4\pi}{3\Omega_j} r_{c,j}^{i} + iK_j^{(i-1)} \frac{dr_j}{dt}$$

(for $i = 1$–3)

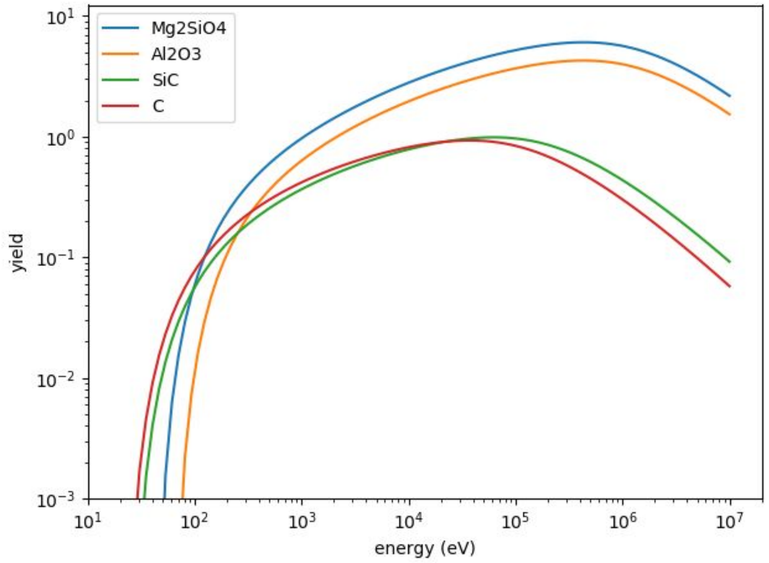
Sputtering Yield

- The amount of sputtered atoms per ion.
  - Depends on the surface binding energy, and the energy of the incoming particle.

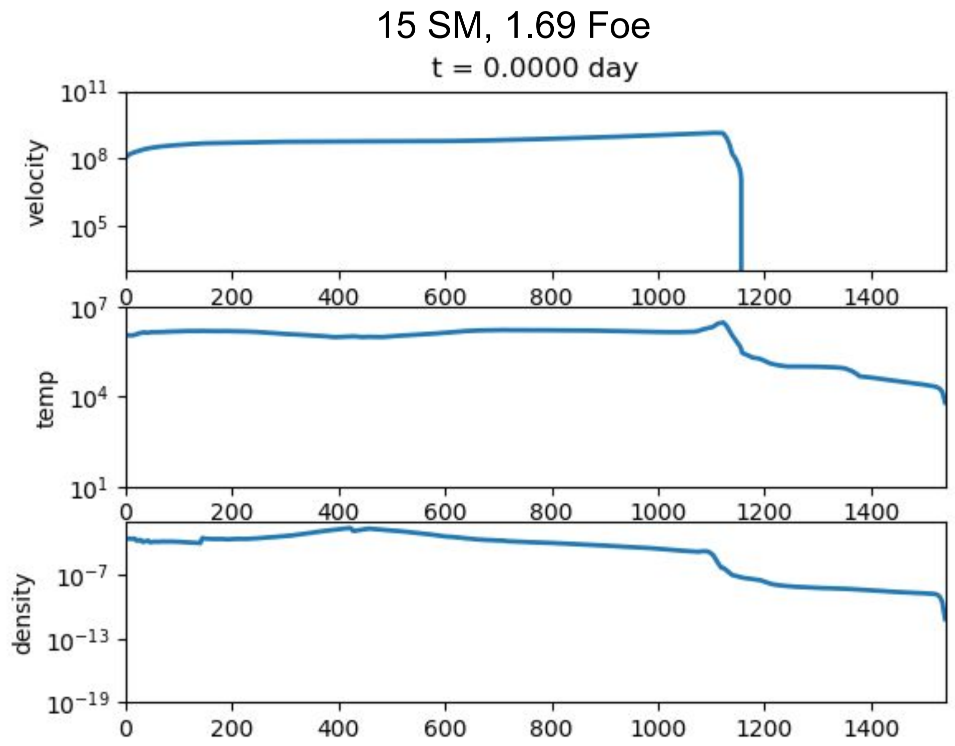
$$Y_i(E) \approx \frac{S_i(E)}{U_0}[1-(\frac{E_{th}}{E})^{2/3}](1-\frac{E_{th}}{E})^2$$

$$\frac{1}{n_H}\frac{da}{dt} \approx -\sum A_i(\frac{8kT}{\pi m_i}) \int \epsilon_i e^{-\epsilon_i} Y_i(\epsilon_i) d\epsilon_i$$

$$\frac{1}{n_H}\frac{da}{dt} \approx -v_d \sum A_i Y_i(E=1/2 m_i v_d^2)$$



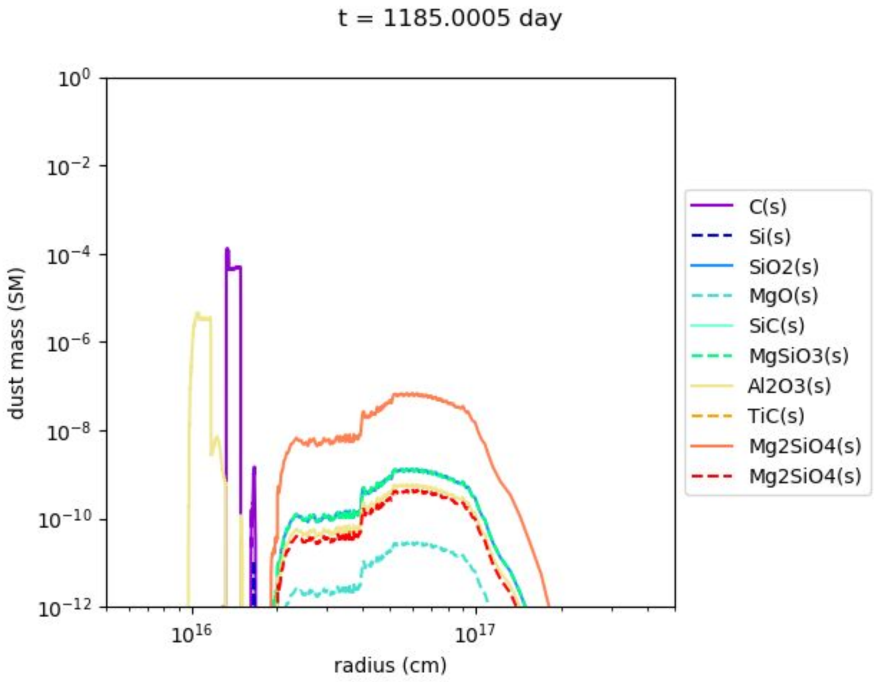
Hydro Results



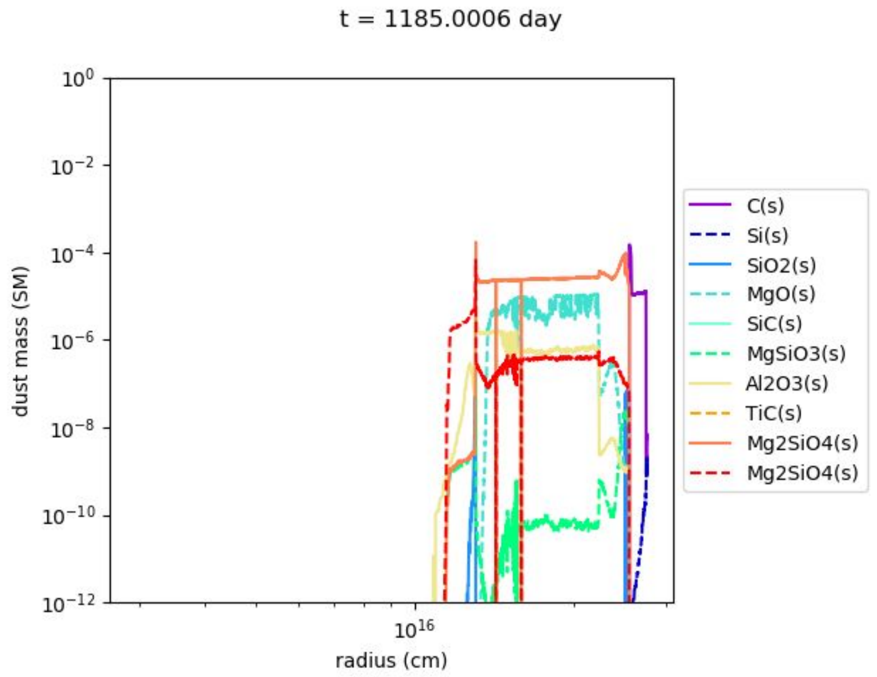
55 simulation models with ~1000-1800 cells = ~100,000 cell calculations

Dust Formation

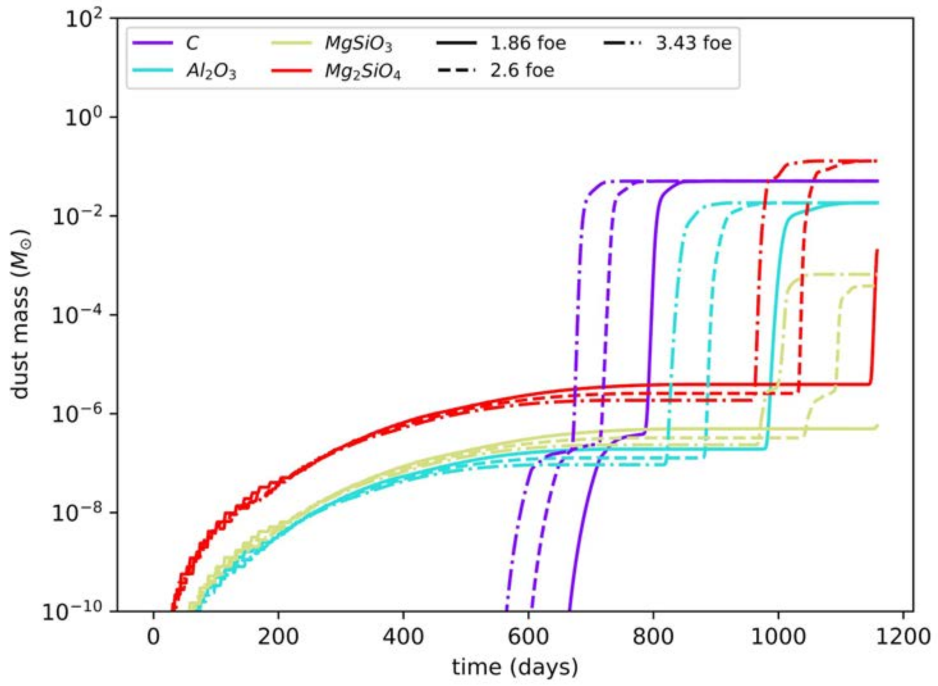
15 SM, 2.47 Foe

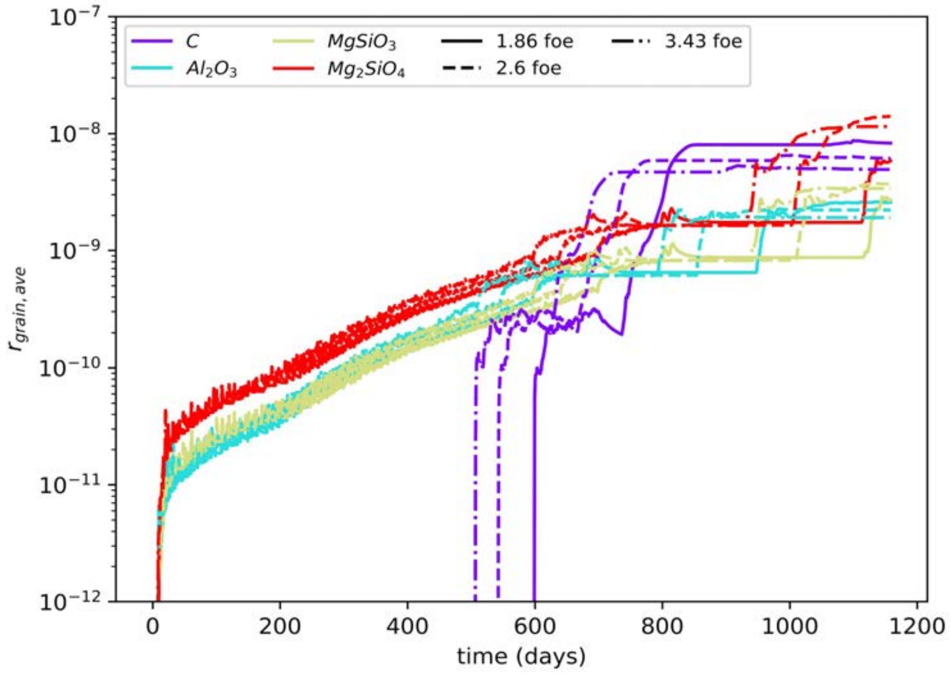


20 SM, 2.85 Foe



Dust Nucleation, Average Grain Radius

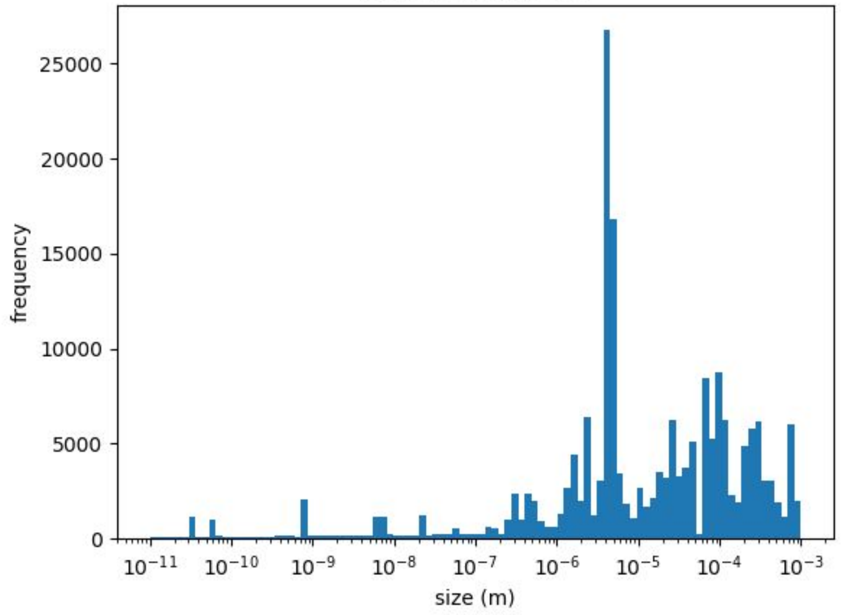




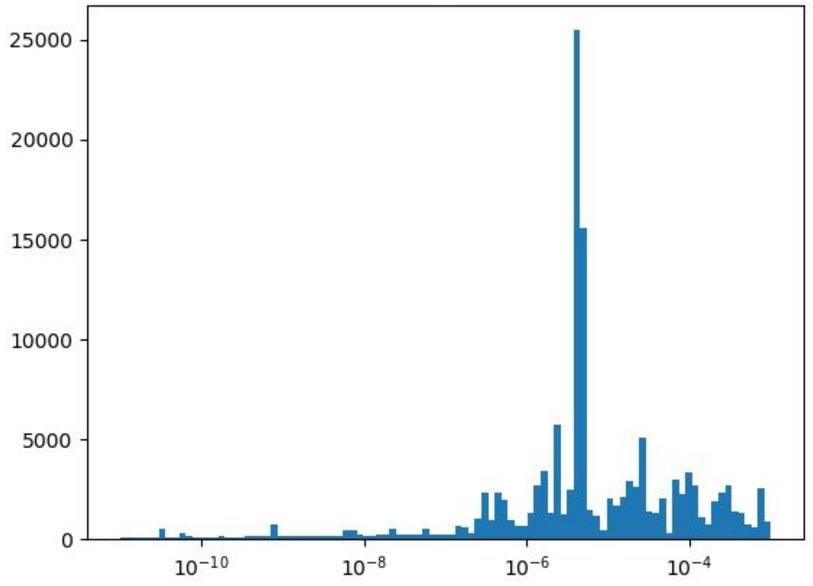
Dust Destruction

15 SM, 2.63 Foe, C grains

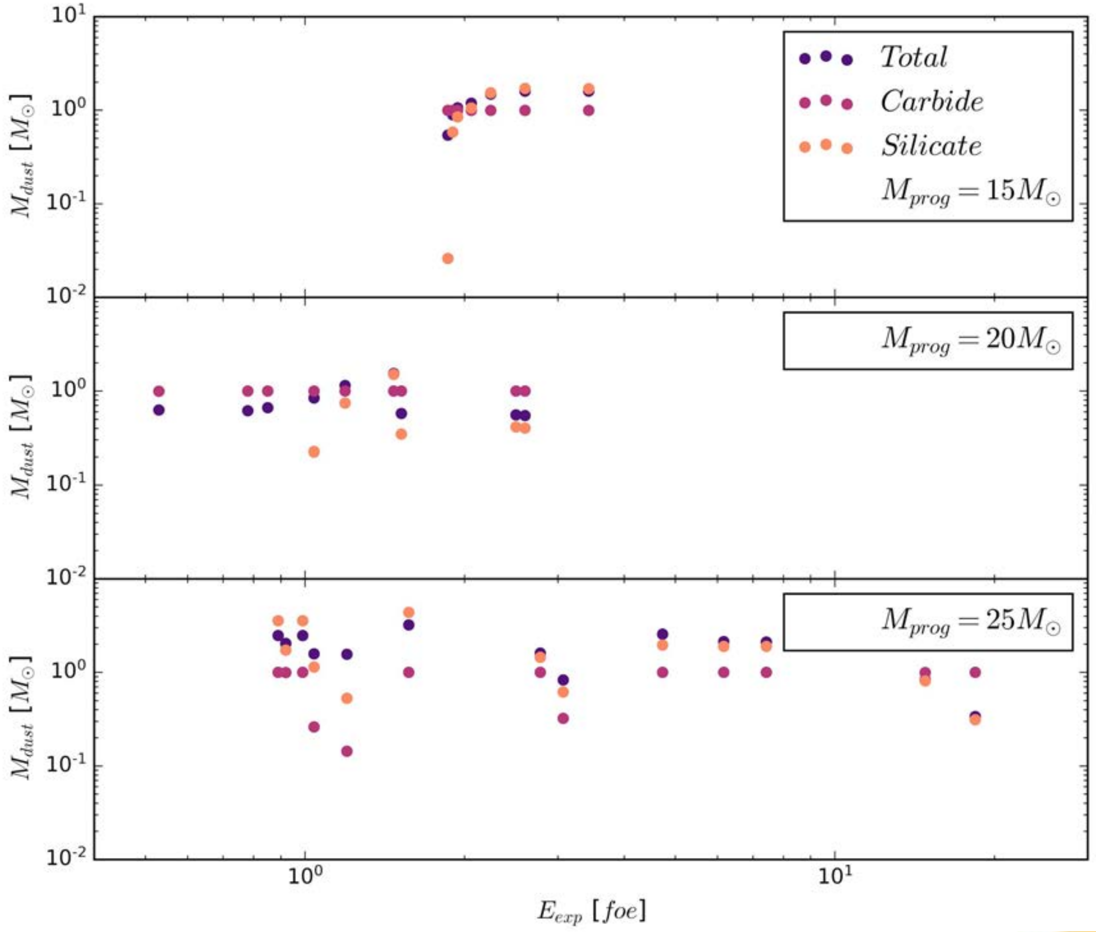
No shock



Shock Destruction

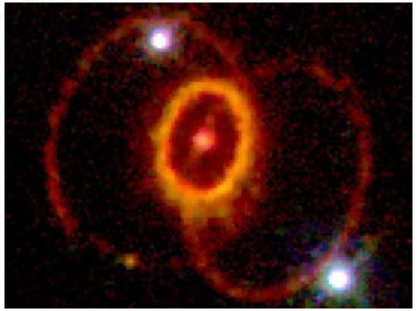


Dust Mass



10^1
10^0
10^{-1}
10^{-2}
$M_{dust}\ [M_\odot]$
● ● ● Total
● ● ● Carbide
● ● ● Silicate
$M_{prog} = 15M_\odot$
$M_{prog} = 20M_\odot$
$M_{prog} = 25M_\odot$
10^0
10^1
$E_{exp}\ [foe]$

Scaling up and model complexity



CCSNe are 3-D, dust production is as well

Extend physics model (gas chemistry, shock destruction, radioactive decay, etc)

Scaling up and increasing complexity requires more efficient code

Current Codes

- 1-D hydrodynamical code using initial data from Fryer et. al. 2018, to model outflow, shocks, and cooling of the ejecta
- Python code, *nuDust*
  - open-source **nu**cleating **dust** code, available at https://github.com/lanl/sndust
  - takes composition/hydrodynamics data and evaluates the kinetics of nucleation named

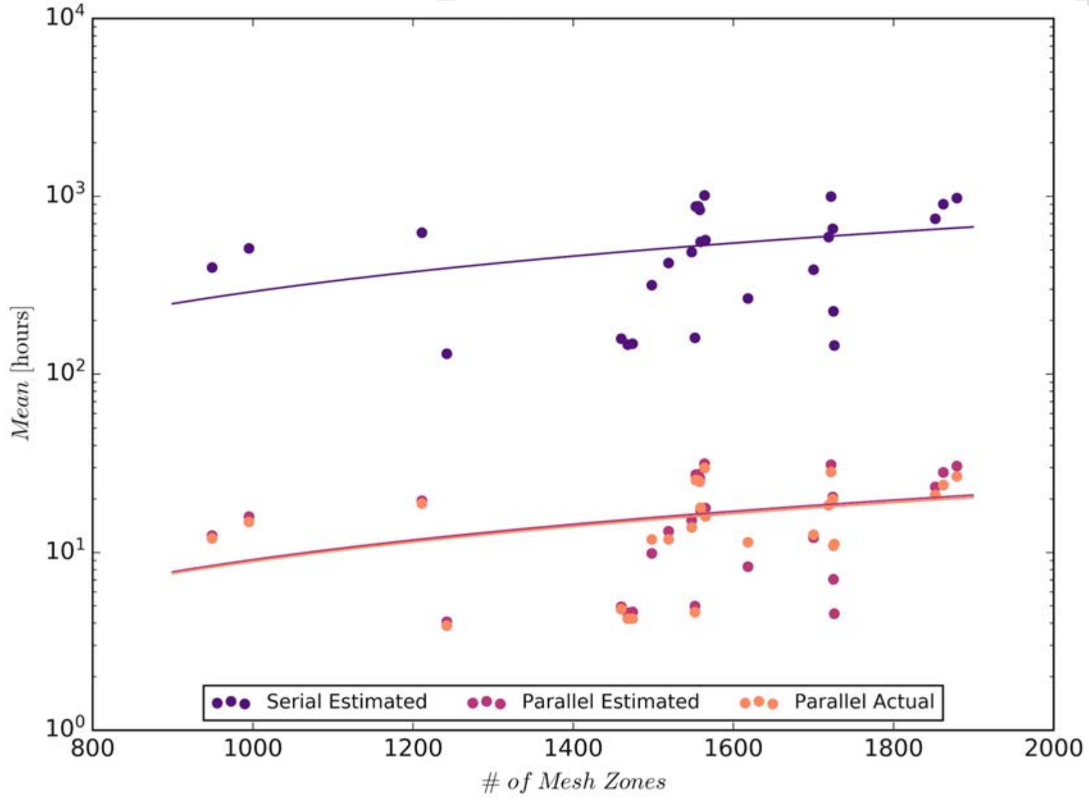
nuDust (Version 1)

- OOP Python 2/3 compatible code
- *numpy* vectorization used when possible
- Utilizes the *LSODA* ODE integrator in *scipy*
- Parallelized using the *multiprocessing* package
  - Useful for single node, multi-core jobs
  - A single dust model can be relegated to one node
- Reads mesh cell data from several master tables
  - Density/temperature trajectory, gas composition

nuDust v1 performance

- Required computation time model dependent
  - Parallelization still vastly superior to serial execution of a single model due to lagrangian structure
- Depending on input data
  - Single cell can take an average 20 minutes in a model
  - Longer computation time for stiffer ODE systems
  - Average model times:
    - Serial: 522.81 hours (~22 days)
    - Parallel: 16.34 hours (<1 day)

Model Total Computation Times



Serial total time estimated from mean time taken for a single cell from parallel runs

Parallel total time estimated similarly to serial total time, but workload simple-distributed over 32-cores of a single compute node

Actual parallel run times for 32-core single node jobs per model

Parallel estimate assumes nothing about the load balancing for parallel work distribution

nuDust (Version 2 developmental)

- *mpi4py* package replaces *multiprocessing* for easier, more stable and *MPI*-like HPC use
- *numba* package used to more efficiently vectorize computations and parallelize
- Data on-loaded/off-loaded as a "particle" per mesh cell
  - Reduces code startup time, reduces lookup tables

nuDust

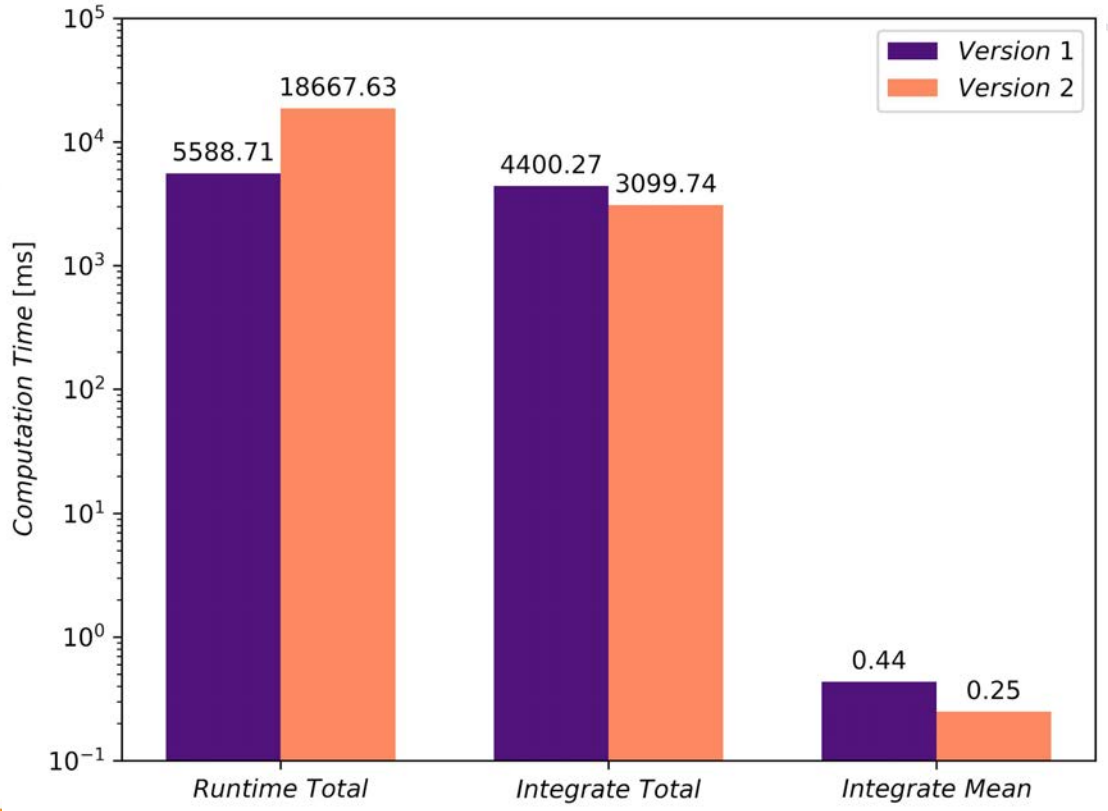
```python
@jit((numba_dust_calc[:], numba_dust_type[:], double[:], double[:], double[:]))
def dust_moments(calc_t, dust_t, y, cbar, dydt):
    for i in prange(calc_t.size):
        if dust_t[i].active == 0: continue
        if calc_t[i].ncrit < 2.0: continue

        gidx = dust_t[i].prod_idx[0]
        dydt[gidx] = calc_t[i].Js / calc_t[i].cbar

        for j in range(1, N_MOMENTS):
            jdbl = np.float64(j)
            dydt[gidx + j] = dydt[gidx] * np.power(calc_t[i].ncrit, jdbl / 3.) \
                                + (jdbl / dust_t[i].a0) * calc_t[i].dadt * y[gidx + j - 1]

        dydt[dust_t[i].react_idx[:dust_t[i].nr]] -= \
            calc_t[i].cbar * dydt[gidx + 3] * calc_t[i].r_nu[:dust_t[i].nr]
```

nuDust v2 performance



Single zone

~1.75 speedup per integrator call

~1.4 speedup overall for integration

Need to optimize startup costs

Future

- Science
  - Run hydrodynamics models in 2- & 3-D
  - Observational predictions
    - light-curves, spectra
- Development
  - ODE solve (matrix inversion) into Numba, GPU

Questions?

Thanks for listening!