A Pythonic interface to a particle-resolved Monte Carlo aerosol simulation framework

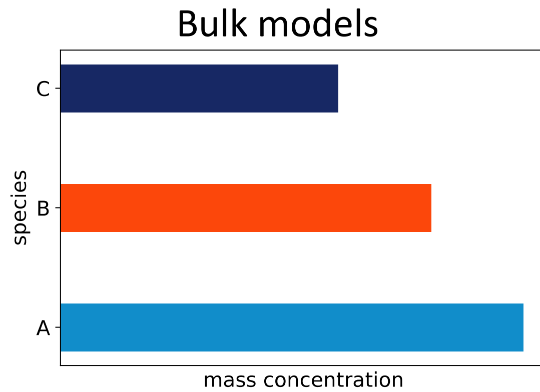
Zach D'Aquino[1], Sylwester Arabas[3], Jeffrey Curtis[1], Nicole Riemer[1], Matthew West[2]

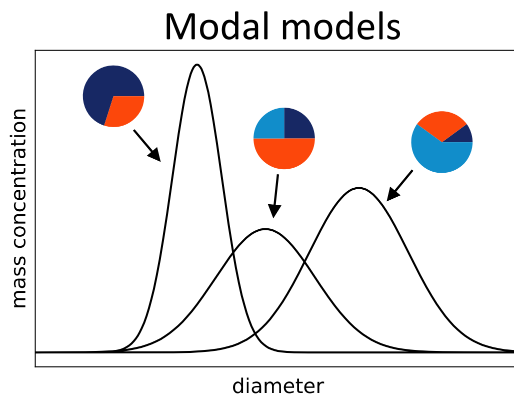
International Aerosol Modeling Algorithms Conference 2023

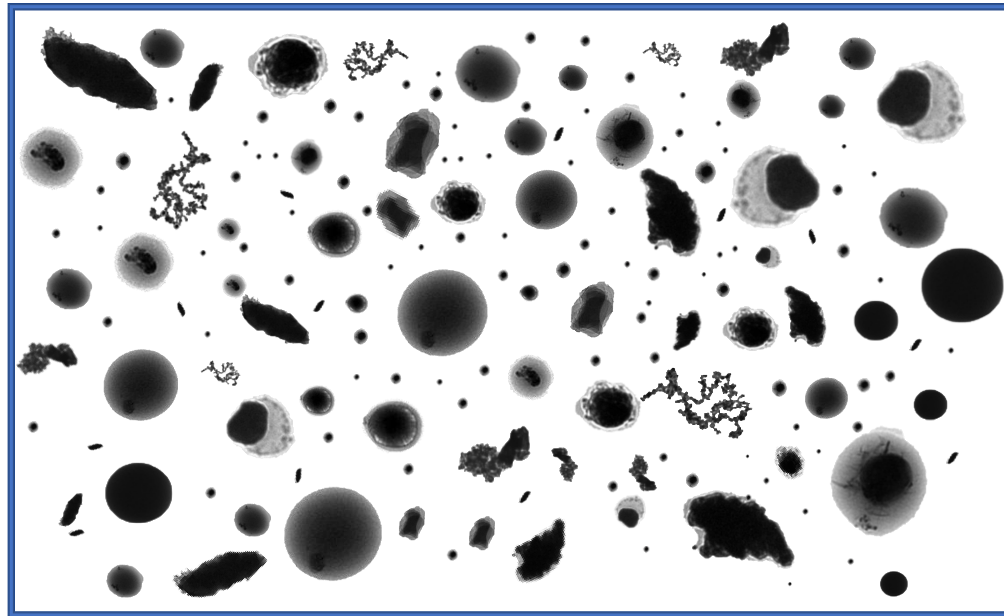
December 7th, 2023

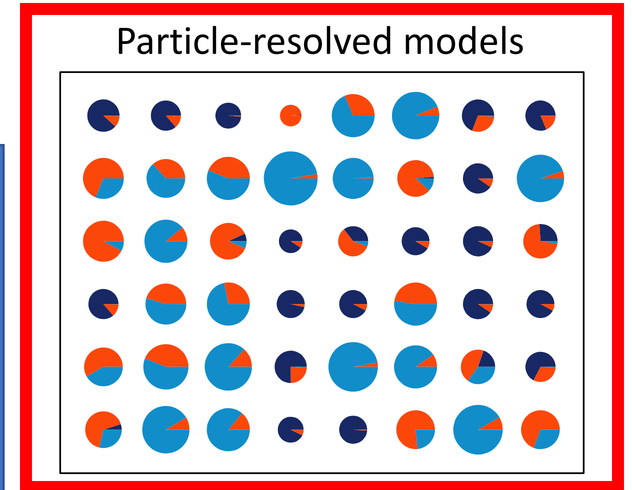
[1]Atmospheric Sciences, [2]Mechanical Science and Engineering, University of Illinois Urbana-Champaign
[3]Physics and Applied Computer Science, AGH University of Kraków

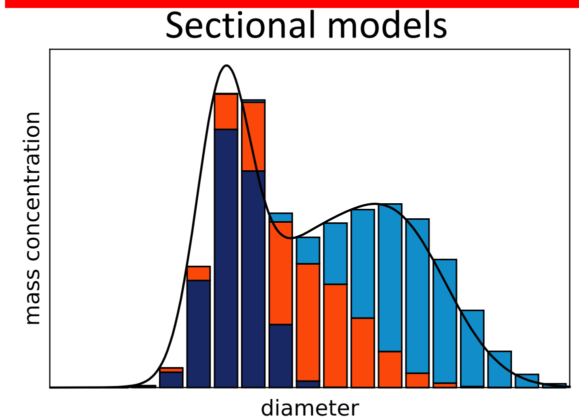
Particle-resolved modeling



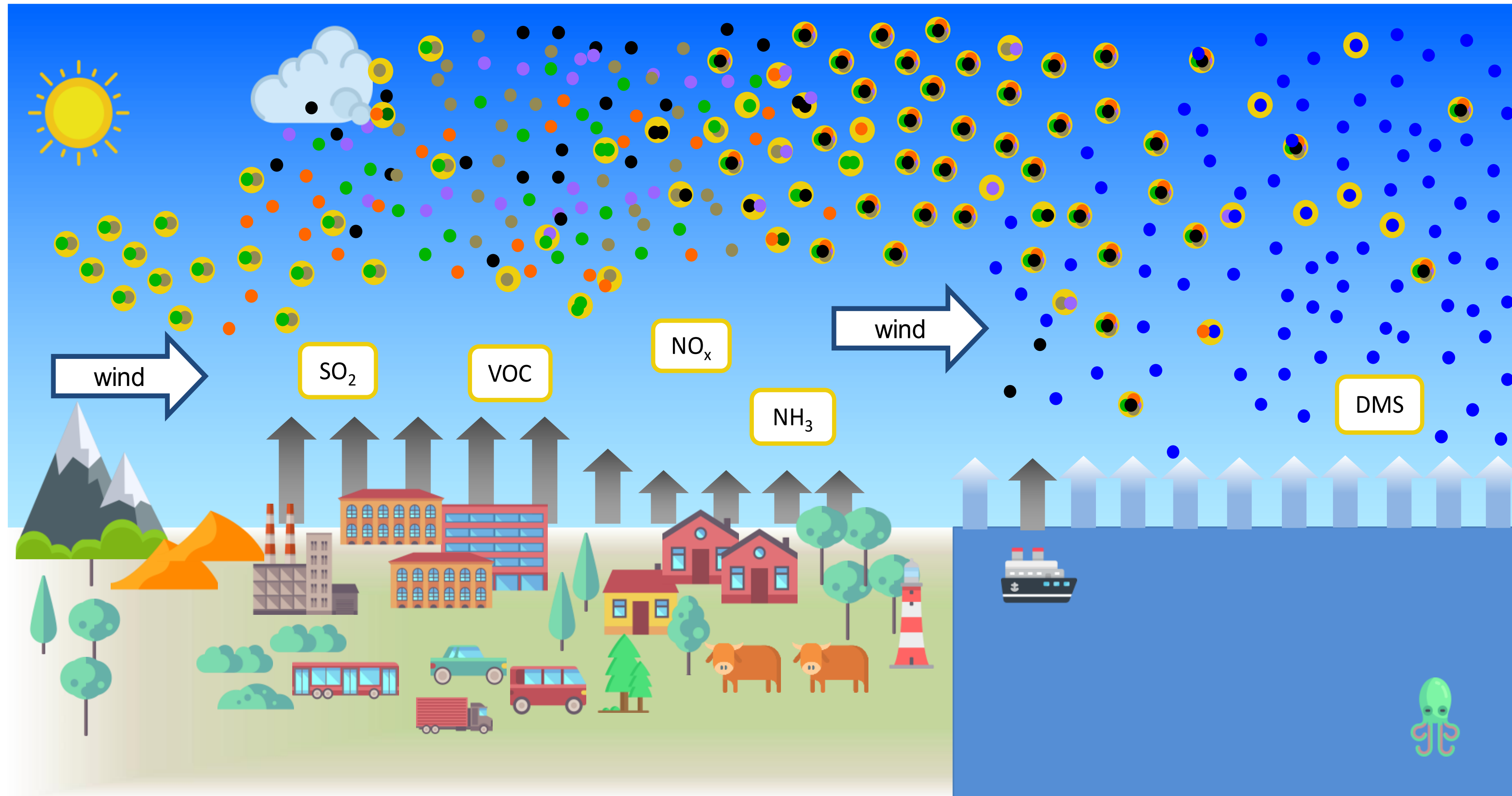








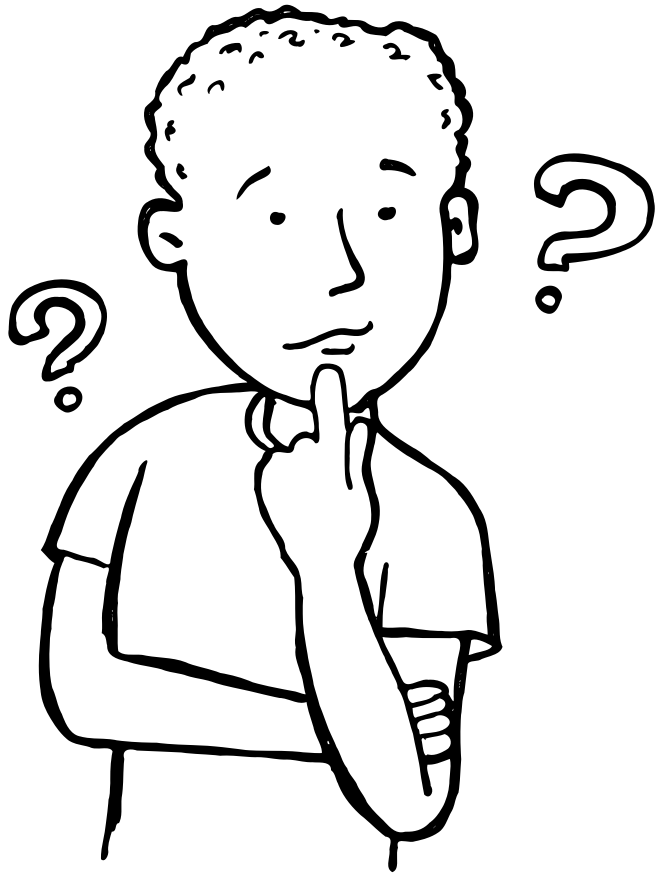
PartMC

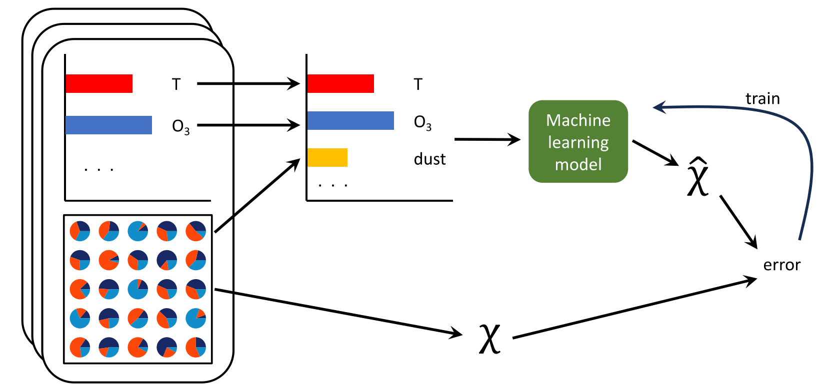


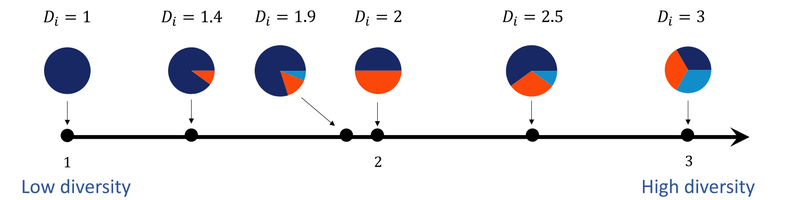
- Emission from primary sources
- Brownian coagulation
- Nucleation scavenging
- Dry deposition

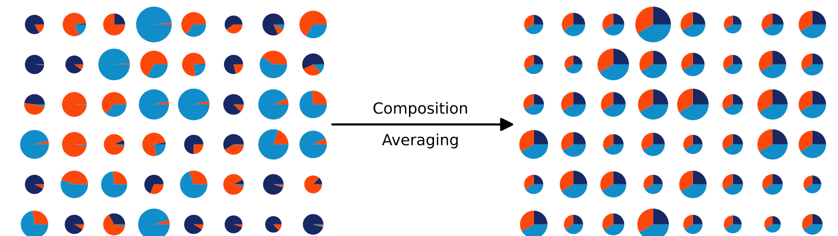
Riemer, Ault, West, Craig, Curtis, *Rev. Geophys.*, 2019

Why do we need PyPartMC?

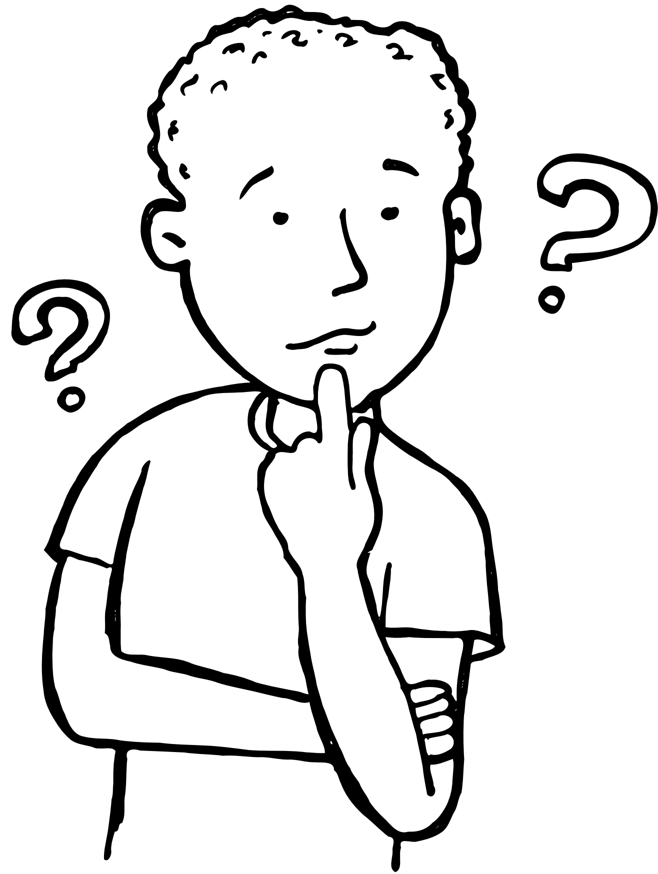


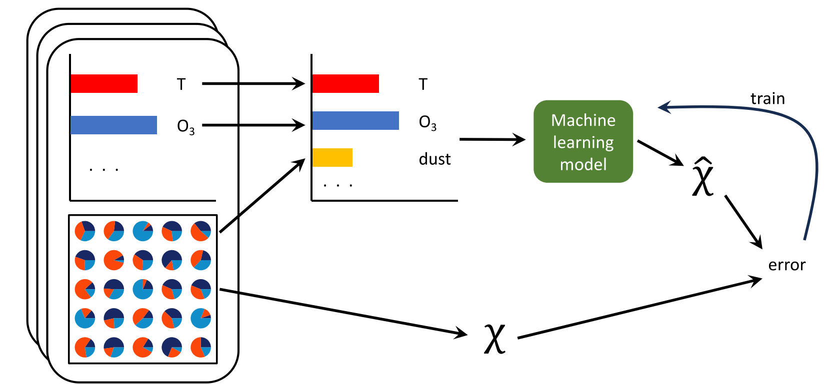


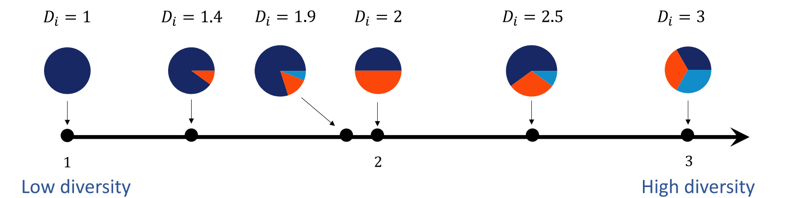


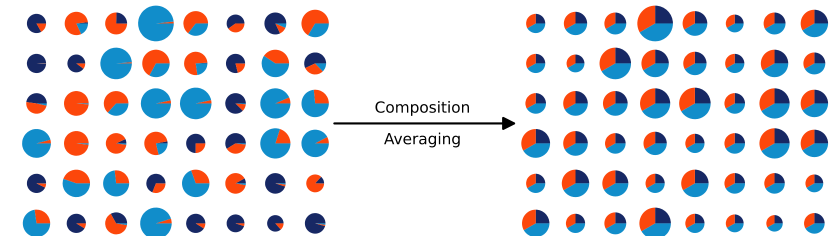


Why do we need PyPartMC?

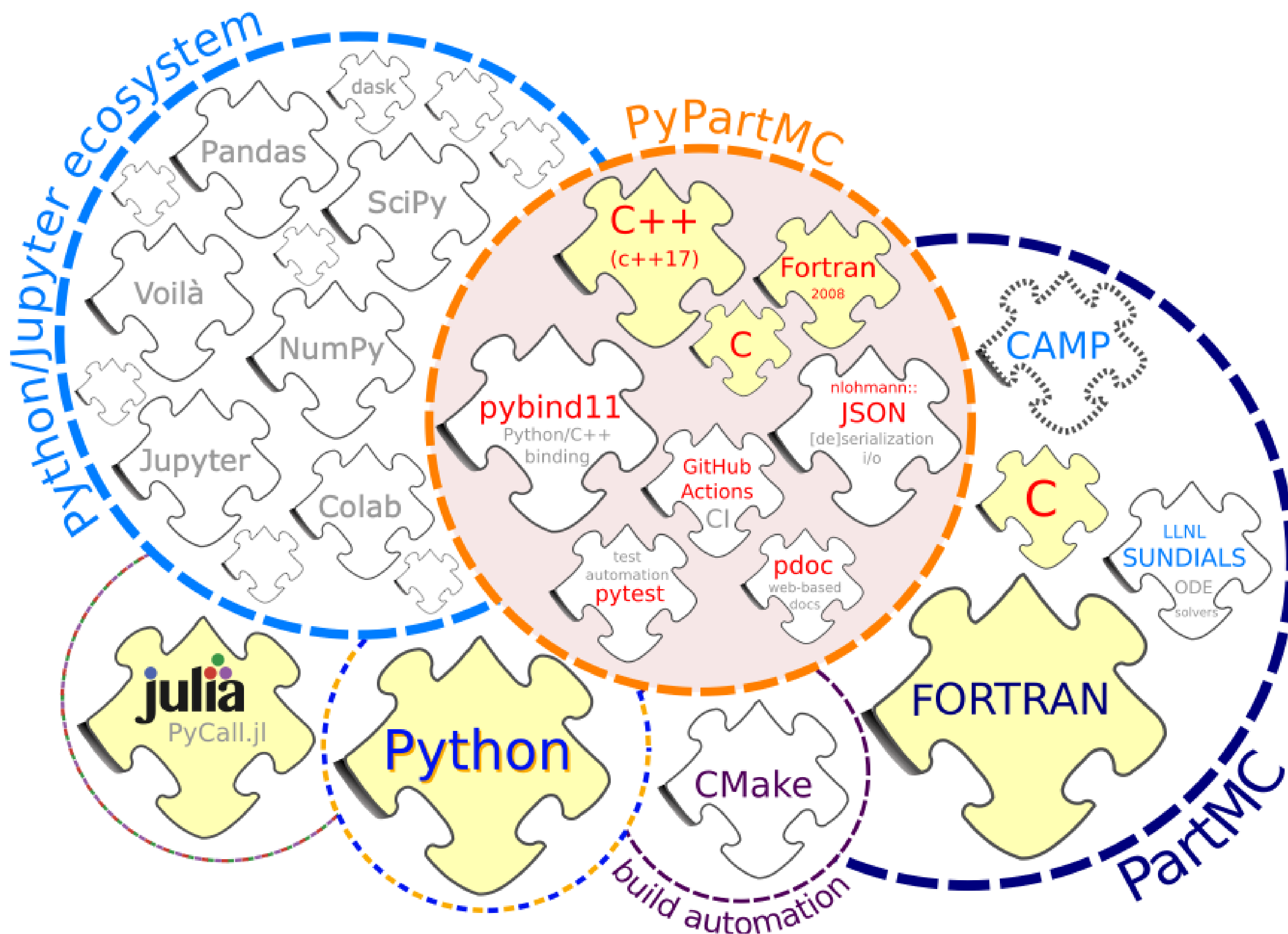








Key technologies



Key technologies

Index

Functions

- condense_equilib_particle
- condense_equilib_particles
- diam2rad
- histogram_1d
- histogram_2d
- input_state
- loss_rate
- loss_rate_dry_dep
- output_state
- pow2_above
- rad2diam
- rand_init
- rand_normal
- run_part
- run_part_timeblock
- run_part_timestep
- sphere_rad2vol
- sphere_vol2rad

Classes

- **AeroData**
  - densities

Package **PyPartMC**

▶ EXPAND SOURCE CODE

Functions

```
def condense_equilib_particle(arg0: _PyPartMC.EnvState, arg1: _PyPartMC.AeroData, arg2: _PyPartMC.AeroParticle)
```

Determine the water equilibrium state of a single particle.

```
def condense_equilib_particles(arg0: _PyPartMC.EnvState, arg1: _PyPartMC.AeroData, arg2: _PyPartMC.AeroState)
```

Call condense_equilib_particle() on each particle in the aerosol to ensure that every particle has its water content in equilibrium.

```
def diam2rad(arg0: float) -> float
```

Convert diameter (m) to radius (m).

```
def histogram_1d(arg0: _PyPartMC.BinGrid, arg1: List[float], arg2: List[float]) -> List[float]
```

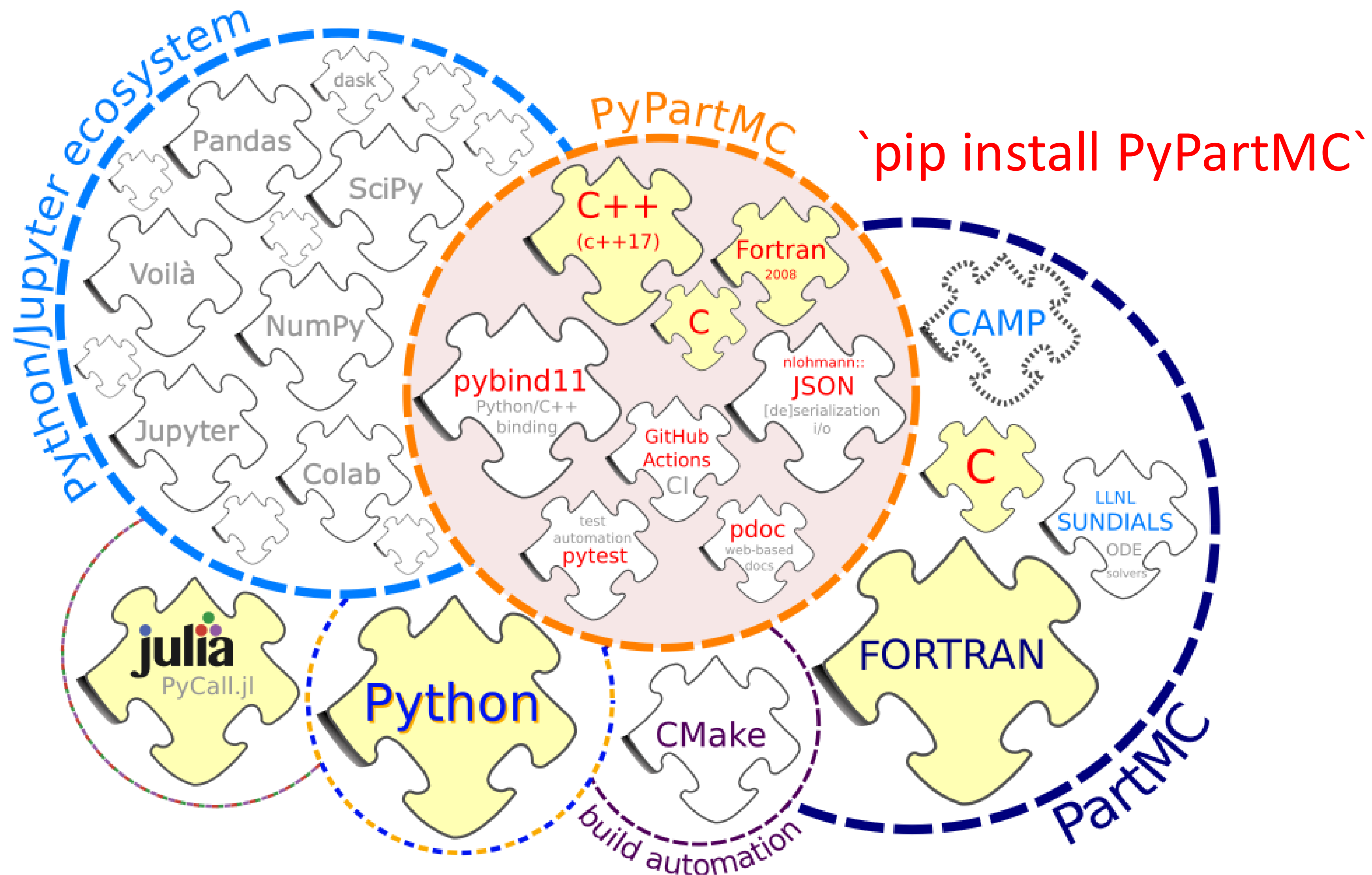
Return a 1D histogram with of the given weighted data, scaled by the bin sizes.

```
def histogram_2d(arg0: _PyPartMC.BinGrid, arg1: List[float], arg2: _PyPartMC.BinGrid, arg3: List[float],
        arg4: List[float]) -> List[List[float]]
```

Return a 2D histogram with of the given weighted data, scaled by the bin sizes.

build automation

Key technologies

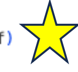


`pip install PyPartMC`

```
module pmc_aero_particle

  type aero_particle_t
     !> Constituent species volumes [length # of species] (m^3).
     eal(kind=dp), allocatable :: vol(:)
  end type aero_particle_t

contains

  !> Total volume of the particle (m^3).
  elemental real(kind=dp) function aero_particle_volume(aero_particle)

     !> Particle.
     type(aero_particle_t), intent(in) :: aero_particle

     aero_particle_volume = sum(aero_particle%vol)

  end function aero_particle_volume

end module
```

- PyPartMC operates with unmodified PartMC
- Data remains in underlying PartMC Fortran types
- Modularity of PartMC allows wrapping structure to work
- C++ wrappers allow pybind11 to automate API generation

```
module PyPartMC_aero_particle
  use iso_c_binding
  use pmc_aero_particle
  implicit none

  contains

  subroutine f_aero_particle_volume(ptr_c, vol) bind(C)
    type(aero_particle_t), pointer :: ptr_f => null()
    type(c_ptr), intent(in) :: ptr_c
    real(c_double), intent(out) :: vol

    call c_f_pointer(ptr_c, ptr_f)

    vol = aero_particle_volume(ptr_f)
  end subroutine

end module
```



```
extern "C" void f_aero_particle_volume(const void *ptr, double *vol) noexcept;

namespace py = pybind11;
struct AeroParticle {
    static auto volume(const AeroParticle &self) {
        double vol;
        f_aero_particle_volume(
            self.ptr.f_arg(),
            &vol
        );
        return vol;
    }
}
```

```
PYBIND11_MODULE(_PyPartMC, m) {
    pybind11::class_<AeroParticle>(m, "AeroParticle",
        R"pbdoc(
            Single aerosol particle data structure.

            The \c vol array stores the total volumes of the different
            species that make up the particle.
        )pbdoc"
    )
        .def_property_readonly("volume", AeroParticle::volume,
            "Total volume of the particle (m^3).")
}
```

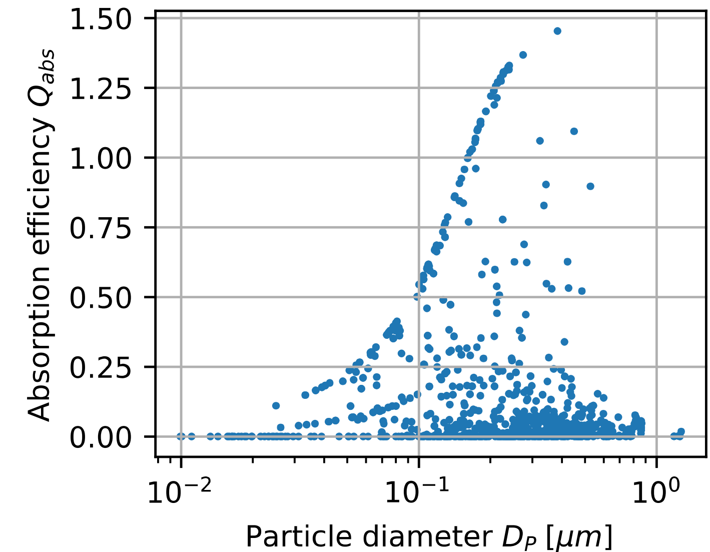
```
import PyPartMC as ppmc
from PyPartMC import si
import gc

class TestAeroParticle:

    @staticmethod
    def test_volume():
        aero_data_arg = (
            {"H2O": [1000 * si.kg / si.m**3,
             1, 18e-3 * si.kg / si.mol, 0]}
        )
        aero_data = ppmc.AeroData(aero_data_arg)
        volumes = [1e-6]
        sut = ppmc.AeroParticle(aero_data, volumes)
        aero_data = None
        gc.collect()

        vol = sut.volume

        assert vol == sum(volumes)
```

Example using external package

User-defined function employs the PyMieScatt package to compute optical properties at each timestep

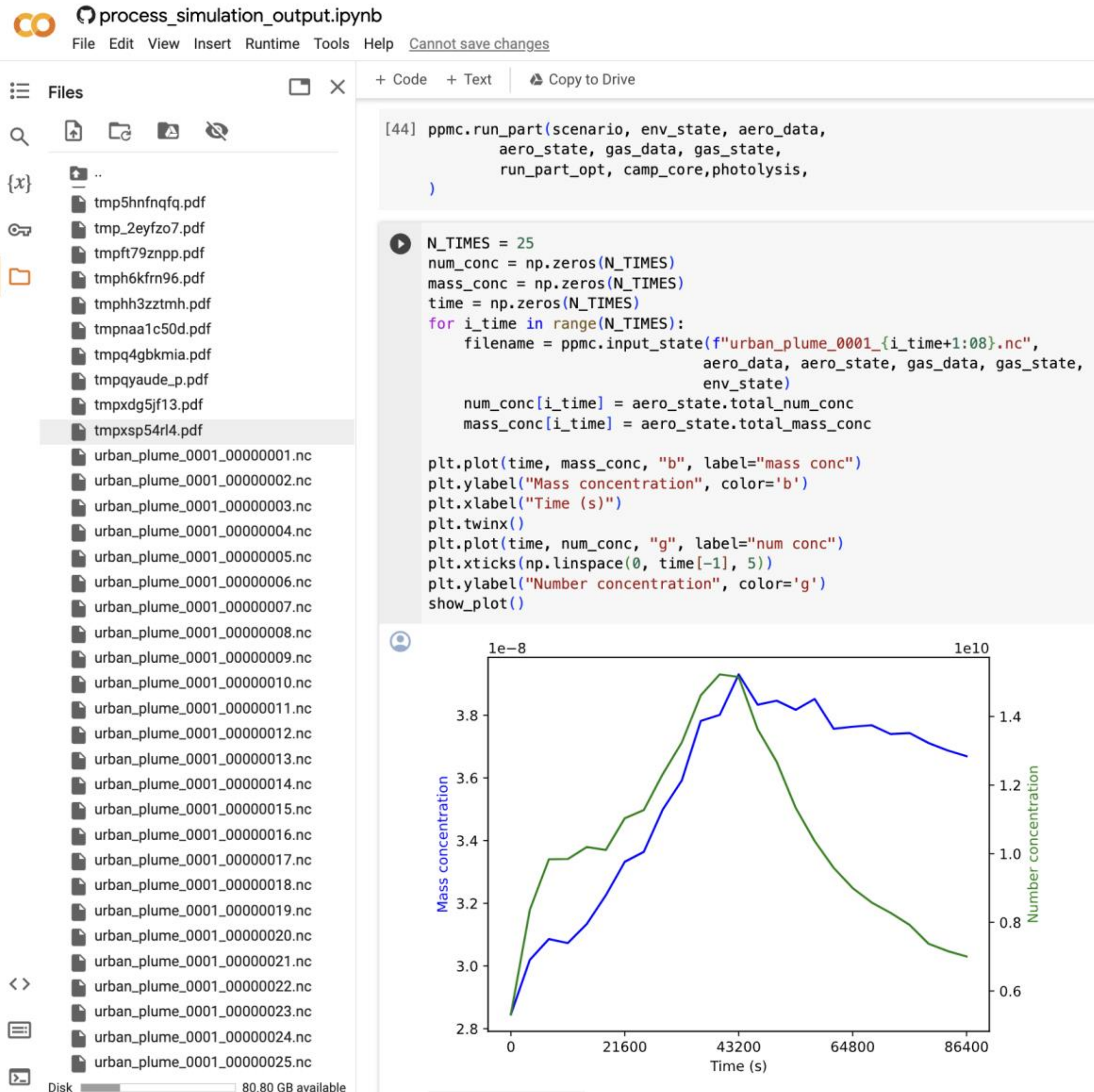
```python
def aero_state_compute_optical(aero_state_optical):
    wl = 550.0 # unit: nm
    refr_shell = 1.52+0*1j
    refr_core = 1.82+0.74*1j
    diams_core = np.array(aero_state_optical.diameters(include=["BC"])) * 1e9 # unit: nm
    diams_total = np.array(aero_state_optical.diameters()) * 1e9 # unit: nm

    qsca_part = np.zeros(len(aero_state_optical))
    qabs_part = np.zeros(len(aero_state_optical))
    for i_part in range(len(aero_state_optical)):
        val = PyMieScatt.MieQCoreShell(refr_core,
                                       refr_shell,
                                       wl,
                                       diams_core[i_part],
                                       diams_total[i_part],
                                       asDict=True
                                       )
        qsca_part[i_part] = val['Qsca']
        qabs_part[i_part] = val['Qabs']

    cross_section = np.pi * (diams_total / 2 / 1e9)**2

    num_concs = aero_state_optical.num_concs
    B_abs = np.sum(qabs_part * cross_section * num_concs)
    B_sca = np.sum(qsca_part * cross_section * num_concs)

    return (diams_total, qsca_part, qabs_part, B_sca, B_abs)
```

```python
for i_block in range(1,N_BLOCKS+1):
    i_next = int(N_STEPS_PER_BLOCK * i_block)
    ppmc.run_part_timeblock(
        scenario,
        env_state,
        aero_data,
        aero_state,
        gas_data,
        gas_state,
        run_part_opt,
        camp_core,
        photolysis,
        i_time,
        i_next,
        0,
    )
    time[i_block] = env_state.elapsed_time
    diameters, qsca, qabs, Bsca[i_block], Babs[i_block] = aero_state_compute_optical(aero_state)
    i_time = i_next + 1
```



https://pymiescatt.readthedocs.io/en/latest/

Time stepping in Fortran



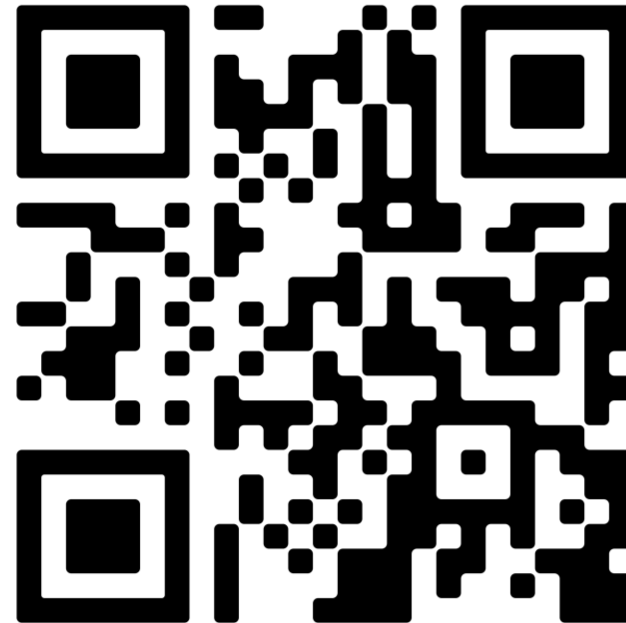
Time stepping in Fortran

Processes data from outputted netCDF files and use matplotlib to visualize

- Entire simulation runs in Google Colab
- Choice in time stepping control
- Data generation and processing in a single notebook

More than just a wrapper!

GitHub



Documentation