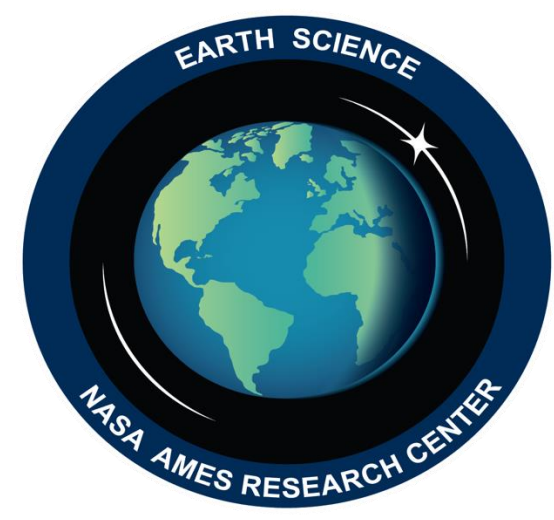


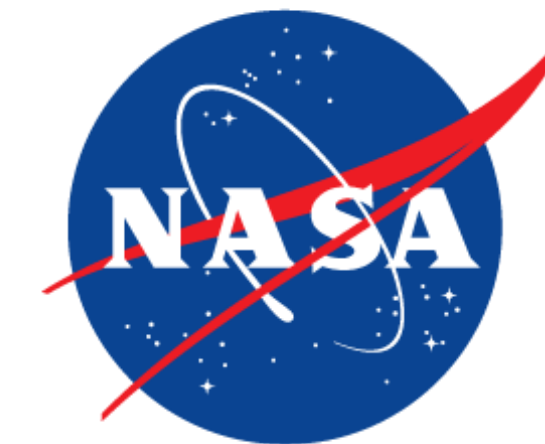


Gigatraj: An Atmospheric Trajectory Model

Leslie R. Lait¹, Paul A. Newman², Weiyan Jiang³, Thomas Clune², Thomas Grubb²

¹NASA Ames Research Center; ²NASA Goddard Space Flight Center; ³Science Systems and Applications Inc.

National Aeronautics and
Space Administration



1. Why yet another trajectory model?

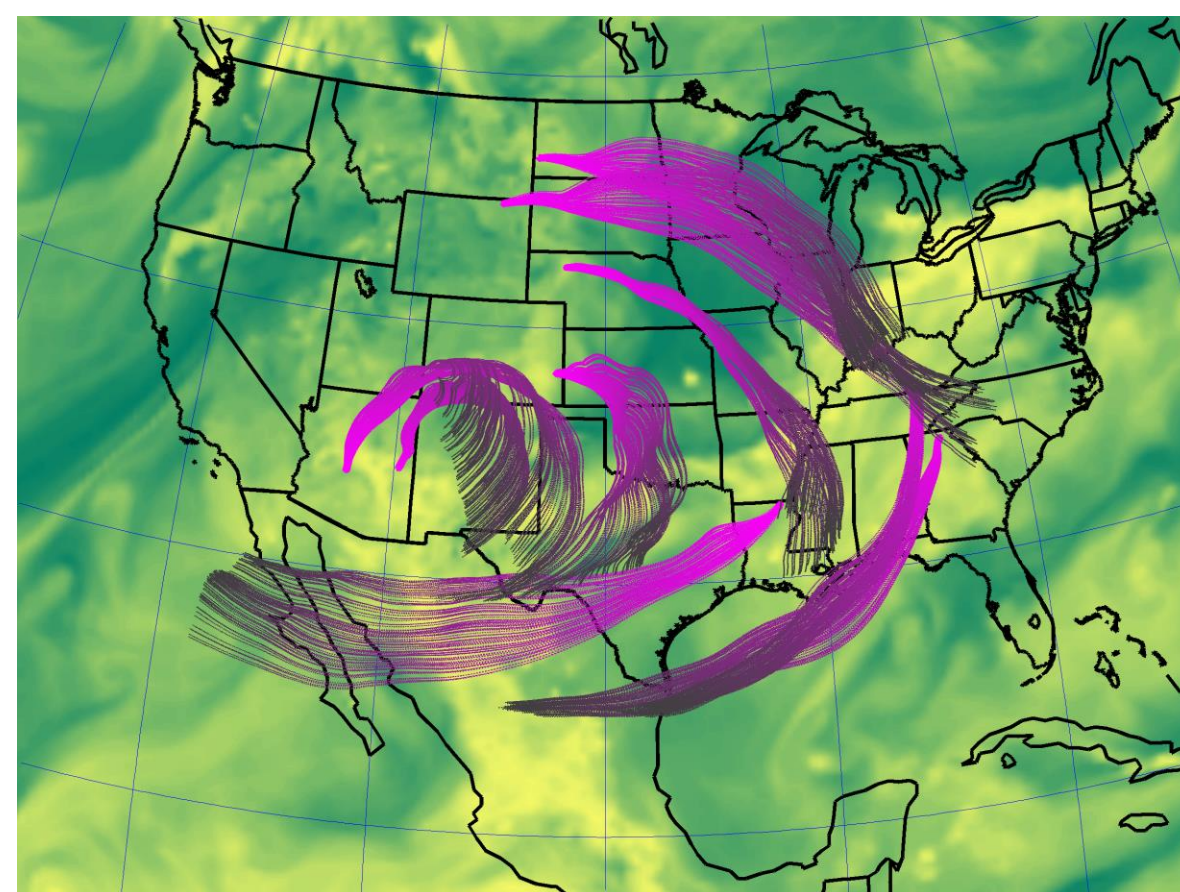
Trajectory models are useful for tracing air parcel histories (e.g., Schoeberl and Sparling, 1995), to aid examination of the physical and/or chemical processes parcels undergo.

Any such model involves making tradeoffs between performance and quality. For example, vertical interpolation using splines can yield better results than linear interpolation but is more computationally expensive.

We wish to put these tradeoffs into the users' hands, providing a choice of integrators, interpolators, and data sources at run time.

Also, since we work closely with the Goddard Modelling and Assimilation Office (GMAO), we wanted a trajectory model suitable for inclusion as a module in their Goddard Earth Observing System (GEOS), an earth systems model.

Figure 1: Five-day gigatraj trajectories initialized just above the tropopause over sites of strong convection on 2021-07-17T03 UTC, then traced forward quasi-isentropically using GEOS FP winds and heating rates. The green background is the GEOSI FP relative humidity field at 650 hPa, valid at 00Z on July 17.



2. Features of gigatraj:

- Software library toolkit for building atmospheric trajectory models
- Includes a sample standalone model that uses the toolkit
- Provides a choice of vertical coordinates:
 - Pressure (for kinematic trajectories)
 - Potential Temperature θ (for quasi-isentropic trajectories)
- Provides a choice of data sources:
 - Tilted Solid-Body Rotation winds (for testing)
 - MERRA-2
 - GEOS FP
- Extensible for new data sources.
- Uses 4th-order Runge Kutta integration by default
- Interpolates winds correctly even near the poles
- Has the ability to swap out interpolators, integrators
- Can cache meteorological gridded products, save/restore state
- Implemented as object-oriented classes written in C++
- Optionally uses OpenMPI for large multiprocessing runs, with performance gains above 1000 parcels (See Fig. 2).

Figure 2: Log-log plot of model run time versus the number of parcels, for varying numbers of processors used in the calculation (colored lines). In this test, each parcel-tracing processor reads its own meteorological data.

