PROJECT SUMMARY

Overview: Scientific discovery across the physical sciences is increasingly dependent on analysis of volumetric data. Whether the data come from a supercomputer simulation, direct measurement, or models, researchers wish to apply domain-specific ontologies to the data, extracting semantic meaning via visualization or an analysis algorithm. In practice, the ways in which scientists process volumetric data are analogous across domains, but cross-disciplinary knowledge transfer and tool development is blocked by barriers of terminology and unavailability of required analysis tasks. The `yt` project is an analysis and visualization toolkit that is currently primarily used for astrophysical simulations. The design focus of `yt` is reproducible, inquiry-driven discovery. It is used and developed by a vibrant community of practice with a wide range of participants. This grant proposes to expand the `yt` community beyond theoretical astrophysics via both technical and social means. Technical improvements to the `yt` codebase will procede along three avenues. The first is to develop a system that adapts the way `yt` presents data via a set of domain contexts that encode the ontology, jargon, and common analysis tasks for a given field of study. This will include creating a domain context system as well as a set of six pilot domain contexts developed in collaboration with domain practitioners. The second is to overhaul the `yt` field system, adding more versatility and enabling significant optimizations. The final improvement will be the development of a non-local analysis system, allowing generalized path traversal as well as domain convolutions. To ensure wide dissemination and use of these improved capabilities, we will design domain-specific documentation and training materials, and organize outreach and training events for early-career researchers.

Intellectual Merit: The proposed work will expand access to reproducible, inquiry-driven workflows for analysis and visualization of volumetric data across a broad cross-disciplinary community. By explicitly supporting data formats used in real research workflows, we lower the barrier to entry for working with volumetric datasets of all sizes. By alleviating the burden of writing custom implementations of common analysis algorithms, `yt` can enhance the ability of individual researchers to generate insights and discoveries. The direct output of this SI2-SSI will drive forward a myriad of research goals, from improving weather prediction, planning deep-ocean dives, to informing the design of nuclear reactors. Direct collaboration across scientific domains will enable the cross-pollination of analysis workflows. The proposing team brings broad experience with the development of scientific tools and includes insight from a wideranging cross-section of scientific domains to ensure critical milestones are reached in a realistic timeline.

Broader Impacts: This grant will substantially expand the capabilities and potential userbase of `yt` by expanding support to data formats and analysis techniques needed across the physical sciences, enabling the acceleration of inquiry and discovery that `yt` currently enables in the astrophysics community. By lowering the barrier to working with complex volumetric data, `yt` expands access to scientific discovery to new audiences, particularly students and early-career scientists. Since `yt` is developed as a community effort, expanding the community of `yt` users will also expand the community of `yt` developers, providing training in distributed scientific software development to a new generation of researchers. The development of Data Carpentry curricula for seismological and weather data will foster research capability and self-direction for researchers from those domains. In addition, we plan to begin targeted outreach to mentor new developers from underrepresented backgrounds, expanding access to training in real-world scientific software development using industry-standard practices. Finally, expanding access to new kinds of datasets from `yt` will lower barriers to producing visualizations suitable for STEM-based public outreach, building deeper capabilities for outreach and public understanding of science.

PROJECT DESCRIPTION

1 The yt Project

yt is an open-source platform for inquiry. By abstracting data layout, format, and organization, yt enables researchers to conduct scientifically-motivated analysis and visualization of data. **We will expand the reach of yt beyond its home domain of astrophysics, implementing necessary technical improvements to expand the research communities it can serve, and work to ensure its sustainability as a computational platform.**

Originally developed to analyze the results of astrophysical simulation data produced by a single simulation code, the yt project [38] is now a community-driven system for ingesting, processing, analyzing and visualizing a diverse selection of research data formats, varying in organizational scheme, file format, problem domain, and increasingly, fields of study. The community around yt has been supported at varying times by grants for development, workshops, but throughout has maintained a high-degree of "peer-production" [discussed in 39, 13].

yt has been successful in the domain of astrophysical simulations, both through easing the process of analyzing data in existing, known ways, as well as opening up entirely new types of analysis to a broad audience. It is not only used as a system for creating plots, but as an *engine* for asking and answering questions that are rooted in the physical meaning of data.

As a result of its history, primary userbase, and the snowballing effect of that userbase driving development, yt is still mostly used in the domain of computational astrophysics, with some preliminary uses in seismology, materials science, and nuclear engineering. The underlying data structures in yt are neutral to domains, as are the majority of the processing routines; however, supporting new domains requires expanding data ingestion capabilities as well as modifying how semantic information and data representations are presented to users from different domains.

1.1 yt In Practice

yt is written primarily in Python, is licensed under the BSD 3-clause license, and is guided through extensive community interactions and a codified governance structure. Since its inception in August of 2006, yt has been used in an estimated 300 papers (the method paper from 2011 has been cited 270 times as of writing), with contributions from over 100 distinct individuals in the source repository. yt can read and process data composed of discrete points (such as particles), uniform resolution meshes, adaptive meshes (in patches or octrees) and unstructured meshes (including higher-order elements). It is able to analyze and visualize generic coordinate systems, including built-in support for Cartesian, spherical, cylindrical, and geographic coordinates. MPI-based Parallelism is transparently accessible to end-users, and yt has facilities for 2D visualization as well as interactive and non-interactive 3D volume rendering. Fields can be defined by end users in the form of a python function and can compute fully-local or fixed stencil operations; all mathematical operations in yt are augmented with units that can be algebraically manipulated.

Here we show some basic usages of yt to demonstrate common workflows and illuminate the design philosophy behind the yt API.

Quick Calculations

yt can be used to quickly calculate statistics for a field in a subvolume of a dataset. This example uses an adaptive mesh refinement dataset containing data defined at multiple spatial resolutions. Individuals do not need to know how the data are stored or even that they are of varying resolution to extract meaningful inferences from the data.

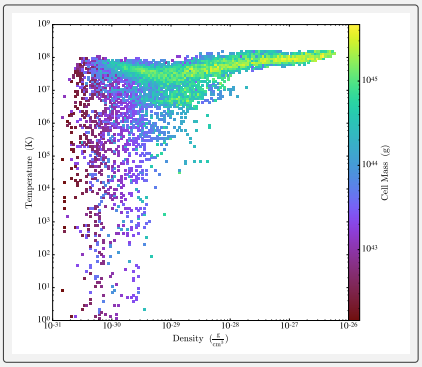
```
import yt
from yt.units import kiloparsec
ds = yt.load('IsolatedGalaxy/galaxy0030/galaxy0030')
sph = ds.sphere(center=ds.domain_center, radius=300*kiloparsec)
mean = sph.mean('temperature', weight='cell_mass')
mininum = sph.min('temperature')
maximum = sph.max('temperature')
std = sph.std('temperature', weight='cell_mass')
msg = "Minimum: {}\nMean: {}\nVariance: {}\nMaximum: {}\n"
print(msg.format(mininum, mean, std, maximum))
```

```
Minimum:  20.8445072174 K
Mean:  11212.3343006 K
Variance:  22968.9738919 K
Maximum:  24826104.0 K
```

Data Selection

Subselected volumes can be used as the source of data for further analysis. In the following example this capability is used to generate a phase diagram, binning the data as a function of non-spatial attributes, using only the gas contained within a sphere centered on a density peak.

```
import yt
from yt.units import megaparsec
ds = yt.load('Enzo_64/DD0043/data0043')
max_val, max_loc = ds.find_max('density')
sp = ds.sphere(max_loc, 10*megaparsec)
plot = yt.PhasePlot(sp, 'density', 'temperature',
                    'cell_mass', weight_field=None)
plot.save('my_phase.png')
```



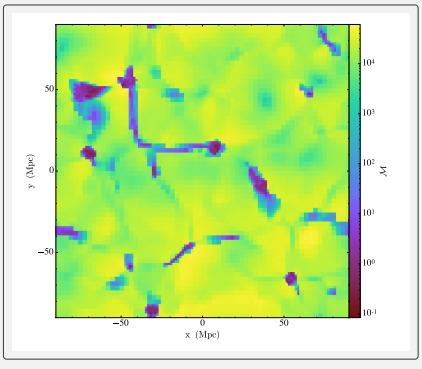
Visualizing User-Defined Fields

`yt` is able to generate visualizations using physically motivated values for the colorbar and axis scale in a few lines of code. A user of `yt` can avoid thinking about the formatting or structure of the data, instead using an API based on physical concepts to inquire into the properties of a dataset. New volume-filling fields are created through defining python functions.

```
import yt

@yt.derived_field(units='', display_name=r'\mathcal{M}')
def mach_number(field, data):
    return data['velocity_magnitude']/data['sound_speed']

ds = yt.load('Enzo_64/DD0043/data0043')
p = yt.SlicePlot(ds, 'z', 'mach_number')
p.save('mach_slice.png')
```



Volume Rendering

`yt` provides a physically-motivated API for generating 3D volume renderings (via software or GPU) that can target standard video formats, immersive devices such as headsets or Google cardboard, and planetarium domes. While this feature in `yt` is increasingly of interest to scientists to construct visualizations, it has been developed to enable interoperability with different rendering systems as well as to enable simple radiative transfer and synthetic observation construction.

1.2 Engineering Process

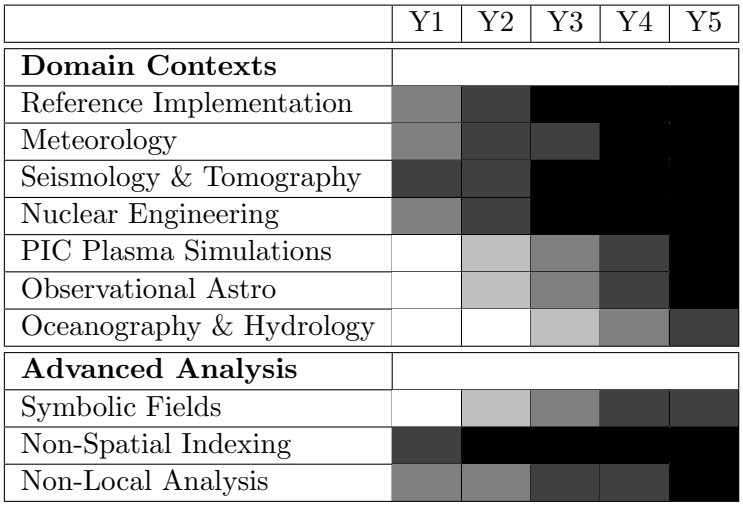


Figure 1: Implementation timeline, colored by phase of work: for conceptualization, for design, for implementation, and for support.

Major modifications to `yt`'s code base, such as the introduction of a new feature or breaking an existing API, are preceded by the creation of a `yt` Enhancement Proposal (YTEP). A YTEP is a document describing proposed changes, serving as a source of information and a starting point for discussion within the `yt` community; these are stored in a public, version-controlled repository, and discussed through public peer review. Once consensus is reached within the `yt` community about the design laid out in the YTEP, the authors proceed to implement the ideas presented in the document and issue a pull request for a public review of the proposed code changes. Where appropriate in this proposal, we cite existing YTEPs; these are accessible through the `yt` website. Our proposed development (§2) is designed to preserve community engagement while ensuring that deliverable timelines are met. All proposed work will undergo the full procedure for code submissions to `yt` to ensure complete community engagement.

2 Proposed Development

We propose to expand `yt` into a platform for analysis and visualization of data in a diverse set of scientific domains. Each development goal has been selected to **maximize impact** on existing or newly supported research communities, and **minimize barriers** for enhancing network effects of collaboration. Through community discussion, outreach to research groups, and pilot work we have identified four major development areas to drive adoption of `yt` in other domains.

Domain contexts: (§ 2.1) We will develop and deploy specialized "domain contexts" for `yt` that provide overlays of specialized analysis, terminology, and domain-specific models.

Advanced Fields: (§ 2.2) We will implement a new derived field engine, allowing for advanced symbolic computation of derived fields as well as performance optimizations.

Non-Spatial Indexing: (§ 2.3) At present, `yt` mandates that the dimensions of the data correspond directly to spatial (Cartesian, spherical, etc) dimensions. We will remove this restriction and extend it to support organization along non-spatial dimensions.

Non-Local Analysis: (§ 2.4) Building on existing functionality for Cartesian ray-casting through datasets, we will implement functions that provide higher-level access to traversal functions, enabling individuals to construct path-focused analysis.

In conjunction with this *technical* development, we will conduct *social* development. Focusing on directed outreach to individuals that are either current or potential users of `yt`, we will endeavor to engage communities through both new and existing channels; we detail these efforts in § 3.1.

The development timelines for each subcomponent of this proposal are given in Figure 1. Our timeline rubric includes several phases of work, beginning with conceptualization, transitioning to design (including pre-development and prototyping work), followed by the implementation and integration of the new feature into the `yt` codebase, and finally the "long-term support" phases. The support phase will include community outreach, improvement and iteration on algorithms and APIs, and integration with other proposed work. New code added as part of the proposal will be done so in a sustainable way, including new test cases and documentation, so that the new features will be maintainable after funding from the proposal is exhausted.

2.1 Domain-Specific Analysis Contexts

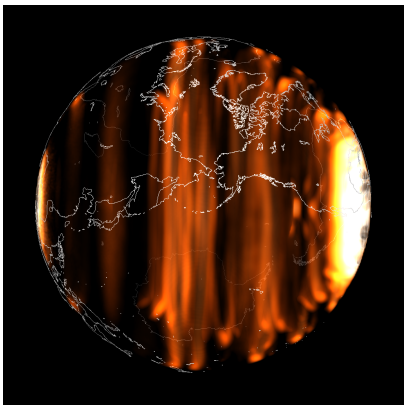


Figure 2: `yt`-generated volume rendering of a simulation of the 2004 Indian Ocean earthquake. *(Simulation courtesy Daniel Peter)*

The codebase and API of `yt` are structured to reduce intermingling between internal representations of data and the model those data present to researchers. This has enabled `yt` to take preliminary steps toward analyzing and visualizing non-astrophysical data, such as in the geosciences and engineering domains. Unfortunately, while the raw underlying data access, processing, and analysis methods are functional and useful in non-astrophysical domains, each of these different domains brings domain-specific jargon and units, analysis and processing tasks, relationships between derived and primary quantities, and in some cases fundamentally different outlooks on the relative merit of types and sources of data. Ideally, when researchers from different domains utilize `yt` for analyzing data, the interface, data representations and algorithms should be customized and customizable to the needs of their domain. This enables the underlying implementations to be directly transferred between domains while ensuring that the barrier to entry (particularly for early-stage researchers who may not have substantial computing background) remains low.

To address the lingering biases toward astrophysical data analysis in `yt`, while minimizing or circumventing any disruption to existing workflows, we will develop both a *framework* for implementing contexts for domain-specific analysis and *implementations* addressing the needs of several pathfinding communities of researchers. Below, we identify several scientific domains within which we will work to both develop domain analysis contexts, and identify representatives of those communities who have agreed to serve in an advisory capacity to this project (letters attached for each advisor). We emphasize that these are the *pilot* communities already identified; we will build them as both model and implementation of how to add new domain-specific contexts to `yt`.

2.1.1 Domain: Whole Earth Seismology and Tomography

At present, `yt` is able to volume render time-dependent seismic wavefields [12] from the output of SPECFEM simulations [18, 31], as illustrated in Figure 2. These renderings have been a key component of the "Seismodome" program in the Hayden Planetarium, American Museum of Natural History, New York. In one of the central elements of the program, the audience is immersed in the center of the planet and seismic waves travel around them, with sonified seismic data synchronized with the moving waves. The images, generated with yt as described in [12] with sound, provide a

dramatic experiential lesson in the physics of the Earth. While their value in public education is clear, there remains much potential for the value of wave field visualization in basic seismological research and teaching. High spatial-temporal resolution simulations are increasingly cheap [25]. The domain context for whole Earth analysis, building on existing seismology capabilities, will combine visualization, sonification, analysis and abstraction of natural and synthetic data, creating an end-to-end tool for physically-motivated analysis of seismological phenomena.

Global and regional tomographic models (imaging the structure of the interior of the planet) and their interpretation are one of the central activities of geophysics. Currently, interpretation of regional and global tomography is plagued by subjectivity in rendering. Regions of high or low seismic velocity anomalies generally are illuminated by iso-surfaces, where the physical interpretation of these volumes depends largely on the isosurface chosen. Building on existing volumetric analysis capabilities of `yt`, this domain context will reduce reliance on isosurfaces as a method of 3D visualization and explore the sensitivity of interpretation to choices in the rendering parameters, towards a general visual language for the rendering of tomography. These images will have profound impact in both basic research and public education contexts.

Coordinating Institution: Columbia University **Community Advisory:** *Ben Holtzman* (funded)

2.1.2 Domain: Weather Simulations

Numerical models serve a crucial role in weather and climate research and operational weather forecasting. While operational meteorologists typically utilize a standardized set of products derived from model output in their work, the same is not true for the research community. A popular weather research model, the Bryan Cloud Model [or CM1; 6] is an actively developed model designed for the study of atmospheric phenomena such as thunderstorms and tornadoes (as in Figure 3). CM1 has been cited in over 100 peer reviewed scientific publications [5] and is the primary tool for conducting research on severe thunderstorms and tornadoes by Co-PI Orf [27, 28, 29]. CM1 offers a handful of I/O options including netCDF [33] as well as a custom-built I/O system developed by Orf utilizing the HDF5 format [35] and currently being utilized on the NSF-sponsored Blue Waters supercomputer [4, 21].

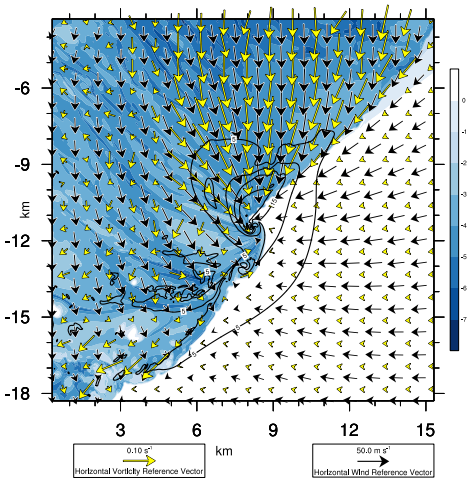


Figure 3: Visualization of a 2D slice through the CM1 model. As part of this proposal we will add support for creating these visualizations quickly and easily within `yt`.

In a typical use case, CM1 is run to completion, after which model output is plotted or visualized. As a part of the proposed work, we will develop both a domain context for weather data as a domain, as well as detailed support for the netCDF and HDF5 CM1 output formats. We will also support the concurrent generation of imagery in order to easily monitor key physical properties during simulations, including surface winds, cloud and precipitation, and vorticity. Developing a broad domain context, including specific units and distance metrics common within this community, will enable full analysis and visualization pipelines to be converted to utilize `yt`, as well as guide the development of features such as non-local analysis.

Coordinating Institution: CIMSS **Community Advisory:** *Leigh Orf* (funded)

2.1.3 Domain: Nuclear Engineering

Detailed numerical simulations of the combined effects of neutronics, heat transport, momentum transfer, and nuclear burn-up inform the development of new reactor designs [41]. Typically, these

simulations are performed on an unstructured or semistructured 3D mesh, where the primitive fields are evolved according to coupled nonlinear solvers. `yt` currently fully supports the EXODUS II file format used by the MOOSE code as well as the MOAB [11] data format used by the PyNE nuclear engineering toolkit [36]. In addition, `yt` supports generating visualizations of slices through these meshes using a high-order interpolation scheme, real-time 3D visualization of the mesh structure and field values, and geometric subselection.

As part of the proposed work, we will expand official support to more codes commonly used in reactor research. In addition, we will create a nuclear engineering domain context within the domain context system. This domain context will include a set of fields commonly used in nuclear engineering workflows, including fields that employ the non-local analysis system (§ 2.4), in particular for streamline analysis and generalized line queries. This work will also further strengthen `yt`'s ability to manage and analyze unstructured mesh datasets from other domains.
Coordinating Institution: NCSA **Community Advisory:** *Katy Huff*, Assistant Professor at the University of Illinois at Urbana-Champaign in the Department of Nuclear, Plasma, and Radiological Engineering *Alex Lindsay*, Postdoctoral Researcher at NCSA *Patrick Shriwise*, Graduate Student at the University of Wisconsin in the Department of Nuclear Engineering

2.1.4 Domain: Particle-In-Cell Plasma Simulations

In the plasma simulations community, scientists often conduct research using particle-in-cell methods for simulating complex interactions between ion and neutral fluids. We will develop a domain context that enables much simpler methods for describing in detail the particle deposition methods, as well as faster and more accurate visualization methods for particles from these simulations. This work will also be supported by current efforts to incorporate bitmap file indexing in `yt` and reduce the overhead of querying large datasets containing discrete points split across multiple files [19].
Coordinating Institution: NCSA **Community Advisory:** *Dr. Michael Bussman*, Junior Group Leader, Computational Radiation Physics and Laser Particle Acceleration groups at the Helmholtz Center for Research, Dresden-Rossendorf.

2.1.5 Domain: Observational Astronomy

In the observational astronomy community, researchers are increasingly making use of large 3D datasets generated by radio and millimeter telescopes as well as optical and infrared fiber-fed integral field unit spectrographs. In these datasets, the data can be mapped to a position-position-velocity (PPV) data cube, where each position-position coordinate corresponds to a position on the sky, and the velocity axis corresponds to the spectral dispersion direction. As a pixel on the sky corresponds to a complete spectrum, the data are large, unwieldy, and benefit from volumetric techniques such as those available in `yt`.

We will develop a domain context for observational astronomy that builds on the initial support for PPV data cubes present in `yt`; this will utilize non-spatial indexing, include enhancements to the units and terminology used in `yt` for these types of data, and interoperate with community tools such as AstroPy [1].
Coordinating Institution: NCSA **Community Advisory:** *Adam Ginsburg*, Postdoc and Jansky Fellow, National Radio Astronomy Observatory, Soccoro New Mexico; *Sarah Tuttle*, Assistant Professor, University of Washington.

2.1.6 Domain: Oceanography and Hydrology

In Figure 4, we show an example of a common workflow in hydrological analysis; this process requires multiple disconnected steps from multiple software packages. A standard bathymetric survey of a water reservoir is performed via sonar measurements in lines perpendicular to the water flow

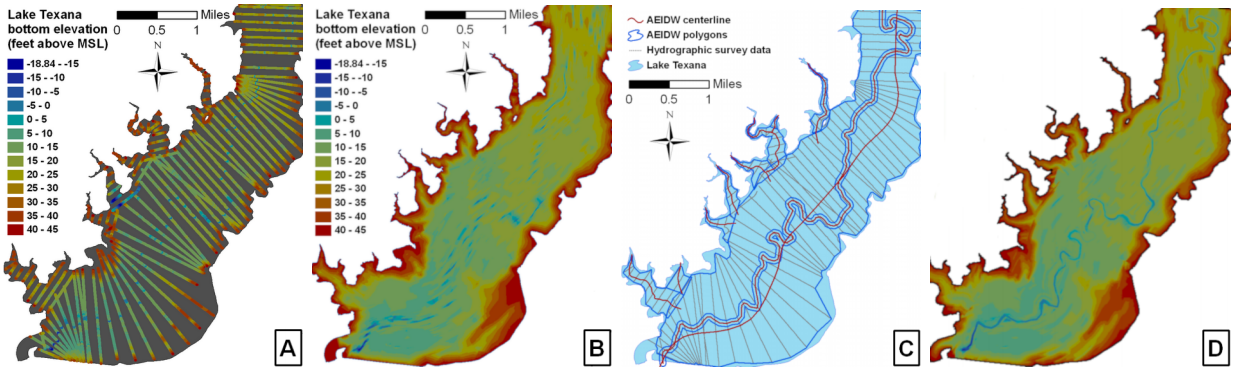


Figure 4: Illustration of several algorithms to produce bathymetry models from sonar data (see main text for details). (*Images courtesy of Dr. Dharhas Pothina*)

that are approximately 500ft apart; the resulting sparse, unstructured data (panel A) are then interpolated using Delaunay triangulation to create a full 3D model of a lake (panel B). However, this naïve approach, that is common in commercial hydrology software, often underestimates the true volume of the reservoir [20]. [22] suggests a better method, taking into the account the anisotropy of the river channel bed and performing Inverse Distance Weighting (IDW) interpolation within a circular search neighborhood (panel C). This technique can utilize highly-optimized and parallel methods already available in `yt`. Implementation of this method would allow domain scientists to efficiently create an accurate Digital Elevation Model (panel D), making `yt` highly competitive with the existing proprietary software solutions.

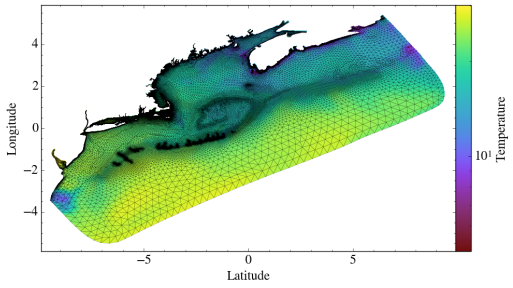


Figure 5: A visualization produced using `yt` of the forecasted water temperature off the eastern seaboard of the United States and Canada. In late July 2016. (*Image courtesy of Dr. Andrew Myers and Dr. Richard Signell*).

The oceanography and hydrology communities produce petabytes of publicly available data to inform forecasts of subsurface conditions. In Figure 5, we show a temperature forecast for the Northeastern ingested into `yt` directly from the Northeast and Coastal Ocean Forecast System's (NECOFS) THREDDS server using the OPeNDAP protocol. Providing native support for Climate and Forecast conventions and OPeNDAP protocol in `yt` will have a high impact on its adoption within the Oceanography community.

Coordinating Institution: NCSA **Community Advisory:** *Christopher Barker*, Oceanographer and modeler with the US National Oceanic and Atmospheric Administration's Emergency Response Division; *Richard Signell*, research oceanographer at the US Geological Survey in Woods Hole.

2.2 Advanced and Symbolic Fields

The data model used by `yt` is based around the idea of an abstract *data object* that defines an arbitrarily shaped subset of a volumetric *dataset*. The dataset itself is made up of individual *data elements*, which might be particles, rectilinear mesh zones, or unstructured mesh elements. Each data element is associated with a set of *on-disk fields* that `yt` can read in directly from an output file. Additionally, `yt` can construct *index fields* based on geometric properties of the dataset. Much of the power of `yt` comes from the ability to query data objects for the values of custom *derived*

fields that can be generated by composing on-disk fields, index fields, and other derived fields.

As a part of the proposed work, we will rework the `yt` field system so adding new fields can be done at a higher level of abstraction using a new field system based on the `SymPy` symbolic computation library [23]. Rather than defining fields in terms of python functions that operate on `NumPy ndarray` instances [40], fields will instead be defined using operations on `SymPy Symbol` objects so that operations in field definitions return a symbolic field object. This will allow us to transform a derived field definition into a depth-first traversal of an expression tree encoded in the symbolic object that corresponds to a given derived field. Once fields are defined in terms of `SymPy` objects, it will be feasible to implement algebraic optimizations like common subexpression elimination, as well as a system for caching on-disk fields that are used multiple times in a single field computation, avoiding unnecessary repeated I/O operations. Higher-level representation of field computations will enable experimentation with code generation, just-in-time compilation, as well as offloading to GPUs and other hardware accelerators.

The implementation of the new field system will proceed through the following steps: (1) Design a prototype field system based on `SymPy` objects, independent of `yt`; (2) Write a YTEP describing the design of the prototype field system, detailing any possible concerns for users and outlining a clear migration path for users to take advantage of the new field system; (3) Integrate the prototype into the `yt` codebase, following the community code submission and review process.

Co-I Goldbaum has extensive experience using `SymPy` and integrating it into `yt` in a performant, maintainable manner in the context of the `yt` unit system. `SymPy` is already a hard dependency of `yt` and expanding our usage within `yt` comes at no additional installation burden.

Community Advisory: *Anthony Scopatz*, Assistant Professor at University of South Carolina, Department of of Mechanical Engineering.

2.3 Non-spatial Indexing

Since `yt` was primarily developed for use in the domain of computational astrophysics, many of the core algorithms were implemented based on an assumption that the underlying data structures (particularly for purposes of selection and units) are defined in terms of spatial coordinates. This means that `yt` is currently unable to properly treat basic geographical data that is often stored as an array in which axes are latitude, longitude and, for instance, pressure. Current support for curvilinear coordinates and position-position-velocity data are based on fragile workarounds that do not address the core assumptions built into `yt`.

In order to alleviate the shortcomings of the current framework, we propose an implementation of an array object that would allow for inhomogenous units for each of its axes. Integration of such an array within `yt` mandates providing users with an ability to add a custom data coordinate system that would describe the order of the axes as they are stored within a data object, along with their respective units. That would enable users to conduct a meaningful analysis for a non-spatial data using the existing `yt`'s mechanism such as slices or projections. Visualization of non-spatial data will require a modification of the plotting routines, which in majority were designed for a spatial data. We will refactor the current plotting interfaces to a common base class and handle each case of a custom indexing through a specialized subclass, e.g. data indexed with *latitude* and *longitude* would be plotted using a subclass offering a map projection.

The implementation plan has been already submitted for review to the `yt` community in the form of YTEP-0027. We propose to continue this work in the following three distinct steps: (1) Creation of an array class that represents columnar vector data with inhomogeneous units; (2) Adapt the `yt` coordinate handler to make use of the new array class. At this point users should be able to load arbitrary non-spatial data into `yt`. (3) Extension of `yt`'s plotting routines. This should result in users being able to plot their data using an appropriate visual representation, e.g. map projection.

2.4 Non-Local Analysis

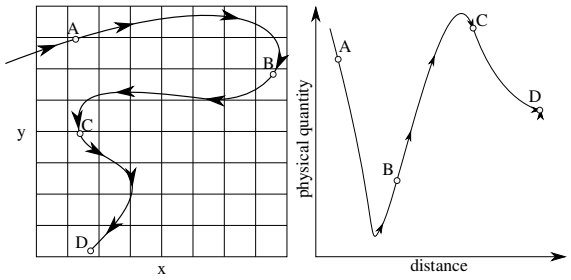


Figure 6: Diagram of "non-local" analysis, wherein rays can traverse a domain, accumulate and/or sample values, and update their direction or heading based on those values. Left panel is an example of a path through uniform resolution gridded data which corresponds to a sampling demonstrated in the right panel.

The `yt` field system (see § 2.2) is based on the notion that fields can be computed based either on fully-local field values, or the values of fields in nearby data elements (where the neighborhood size is specified in advance). This allows for calculation of fields like the kinetic energy of a fluid as well as stencil-based fields like the divergence of a vector field. Defining more complex fields, such as those that might relate to the accumulated values along streamlines emanating from each point in the system, is considerably more complex and not currently available; while the techniques (such as ray tracing, Monte Carlo diffusion approximations, and so forth) are computationally feasible, the user interface to these techniques is typically quite complex and inaccessible to most researchers.

We will design and implement a high-level interface for *non-local analysis*, starting with simple path-traversal accumulation and building to complex, full-domain convolution. Some initial design work for this task has been discussed in YTEP-0016. In addition, a simple prototype has been written but not integrated into the `yt` codebase.

`yt` includes support for generic ray tracing, either through the custom, built-in ray tracing system (supporting unstructured meshes and adaptive Cartesian meshes) or by utilizing the Embree framework. A planned collaboration will extend `yt`'s ability to utilize the SI2-funded `GraviT` ray tracing framework, as well. `yt` uses these engines for a variety of purposes, including volume rendering and stream lines. In contrast to the developed work here, the existing framework is complex to use and to extend; it is low-level, in contrast to the high-level definitions of derived fields that are found throughout `yt`, and it does not allow, for instance, defining paths that are updated based on criteria from accumulated or sampled values.

We will build and develop a generic system for describing high-level "path" operations that satisfies these requirements: (1) Paths can accumulate, sample or deposit values; (2) Path directions can be algorithmically updated; (3) Path operations are described at a high-level, for instance in Python; (4) Path operations can "convolve" the domain (for instance, to determine a "mean-free path" for each mesh element or zone).

Each of these will be an independent milestone, identified in our work plan, coupled with specific use cases from our identified domain contexts. The advisory board and the `yt` community will evaluate these for suitability and acceptance into the primary `yt` codebase.

3 Project Plan

3.1 Community Engagement

The community of users and developers (and the blurry line that distinguishes the two) that supports `yt` is often referred to as its best "feature." The `yt` community is active, each month dozens of pull requests are issued and more than one hundred messages are posted on the "user" and "developer" mailing lists. During the course of this work, we will continue to support the community, engaging with individuals through existing and new mechanisms to ensure that `yt` remains useful, easy, and performant.

Ensuring that development undertaken through this proposed work supports the *existing* com-

munity as well as fostering the development of *new* communities of developers and users is key to its success. PI Turk and Co-I Goldbaum have dedicated considerable resources to building, strengthening, and empowering the `yt` community. This takes the form of technical constructs and social engagement, focused on providing a friendly atmosphere that fosters collaboration, technical interchange, and recognition that priorities and contributions take many forms, including and beyond code. As described in YTEP-1776, a `yt` Steering Committee is in place; we have attached letters from those members of the Steering Committee not listed as PI, Co-I or SP on this proposal (Hilary Egan, Cameron Hummels, Sam Skillman, Britton Smith, and John ZuHone) indicating that they will engage with the development in this proposal, following community norms. PI Turk, Co-I Goldbaum and SP Kowalik are members of the `yt` Steering Committee.

The previous SI2 grant (SSE) that supports the development of `yt` explicitly identifies computational astrophysics as the target community; this grant, conversely, has identified several domains of physical science outside of astrophysics for targeted development and outreach (see §2.1). Introducing a new software tool to a community is challenging, and requires addressing both the *actual* needs of that community as well as the *perceived* needs of that community. Furthermore, interdisciplinary collaborations bring with them both opportunity and risk, as differences in nomenclature, modeling, shared understanding and choice of software tools can produce considerable impedance mismatches [30]. As interdisciplinary collaboration and building software tools that cross domains is a key component of this proposal, we have assembled an advisory board of "domain advisors," identified specific developments that support application of `yt` in other domains, and have prepared plans to conduct outreach to community members in other domains. This hybrid between top-down and bottom-up approaches to collaboration, focused on constructing pragmatically-useful tools and community support around those tools, will result in technology diffusion between disparate groups. The proposed work has been designed, structured and scheduled to enable building of "layers" of components that can act in superposition, in accordance with findings regarding technology transfer and collaboration [14]. In-person meetings focused on improving scientific productivity will further foster collaborations between team members and community members [9].

3.2 Curriculum Development

As part of the proposed work, we will hold at least one in-person and one online workshop each year of the project. These workshops will target both new and experienced users, will particularly solicit attendance by underrepresented and minoritized individuals, and will seek to broaden the participation in both the "user" and "developer" `yt` communities. The `yt` community has had success with workshops, including user-focused workshops in 2012 and 2016 (upcoming), and developer workshops in 2013 and 2014; as part of this proposal we will focus on "user" workshops. Additionally, all new features will be documented, through both API and narrative documentation styles. This will include tutorials, domain-specific examples, and interactive Jupyter notebooks deployed on the `yt` Hub (developed as a part of NSF SI2-SSE 1535651).

The University of Wisconsin has an existing an partnership with the Data Carpentry (DC) organization; DC trains researchers in the core data skills for efficient, shareable, and reproducible research practices and teaches openly available, high-quality, domain-tailored lessons. In collaboration with Dr. Tracy Teal, Executive Director of Data Carpentry, at the University of Wisconsin we will develop two full DC "modules." The first of these will be focused on analysis and visualization of weather data, and the second on seismology and tomography. In doing so, we will focus on developing not only the tools for analyzing and visualizing data, but the necessary curriculum to ensure their uptake amongst early-stage researchers. Developed curriculum for Data Carpentry will also be disseminated through the Midwest Big Data Hub (letter from MBDH Executive Director Melissa Cragin attached).

3.3 Release Process and Licensing

The development process for `yt` is conducted completely in the open, with code review happening in a public repository on Bitbucket. The release process for `yt` has been codified through YTEP-0008 to account for both bugfix releases and major development milestones. While numbered and versioned releases do occur with regularity, the development line is always available and in a constant state of release; most developers utilize the current development line for production work. The `yt` community employs continuous integration, ensuring that existing test cases do not regress as development continues. `yt` is licensed under the permissive, non-copyleft BSD 3-Clause licensing; all code produced as part of the proposed development will be similarly licensed.

Releasing a version of `yt` includes tagging a particular version, migrating open bugs to the next milestone, uploading the source distribution to the Python Packaging Index and distributing binary builds for Linux, OS X and Windows through multiple channels (`conda-forge`, Python packages, and Docker images). Major and minor versions of `yt` are available as a binary package through multiple channels, accounting for the vast majority of installation scenarios. Augmenting this strategy is the `yt` Hub, where runnable Jupyter notebooks are available, with the "stable" and "development" branches of `yt`. These have access to a variety of datasets as well as tutorial materials and provide the lowest barrier to entry to learning and utilizing `yt`.

3.4 Security, Reproducibility and Usability

The `yt` platform is typically utilized by individual researchers without adding any additional security implications beyond those present in running simulation or analysis code on either their local machine or an HPC resource such as one provided by XSEDE. We do not anticipate any security concerns that will need to be addressed; however, in the case that they are, we will coordinate with either upstream security teams (such as those for upstream packages like Jupyter, NumPy, etc) or with specialized teams such as the Cybersecurity division at NCSA.

Reproducibility of scientific results and the improvement of the usability of software toward those results are key to both the existing `yt` project and the approach of the proposed development. Reproducibility can be addressed through several avenues, both technical and social, but the largest barrier to a pervasive culture of reproducibility is social [26, 34]. The development process of `yt` has been designed to make the technical aspects of reproducibility easier through API-driven analysis and simplified sharing of code and data. Simultaneously, the overall culture of openness, sharing, and peer-support in the `yt` community has been effective in fostering an environment conducive to reproducibility. We will continue that through this proposed development.

The usability of `yt`, particularly when addressing the needs, preconceptions and vocabularies of new domains of study, is an important component to ensuring its overall uptake within those communities. In conjunction with our domain-specific advisory boards, we will conduct informal studies of usability in advance of API development as well as subsequent to that API development (through the educational outreach programs proposed here) to guide the overall usability of `yt`.

3.5 Use Cases

In collaboration with the domain context advisors and the `yt` community, we will work to build concrete use cases that apply the development to specific scientific problems. The product of building these use cases will be specific YTEPs that contain both the narrative description of those use cases as well as target scripts. While these scripts will be developed in advance of the API and implementation, development itself will be guided by the YTEP design process.

4 Outreach and Uptake

4.1 Sustainability

Sustainability, as a topic and as a process, is the subject of a considerable amount of discussion within the academic software community. For instance, there have been workshops such as the WSSSPE series [17], and the SI2 PI meetings [37], as well as others such as CSESSP and the UK Research Software Engineer conference. PI Turk has been involved in these conferences, including reviewing submissions, participating in the organizing committees, contributing to reports, attending, and in two cases provided keynote talks.

Sustainability of software is a well-recognized problem that is difficult to "solve." Typically, the concerns regarding "sustainability" take the approach of examining either responsiveness to maintenance requests (whether they be technical maintenance or community maintenance) or to developing new features (which may include adapting to new hardware or expanding to new communities.) One of the advantages of an open source, community-driven project is that oftentimes unaffiliated individuals will contribute to the codebase (or its affiliated resources, such as documentation) and that this then improves the overall project.

`yt` has benefited from volunteer contributions enormously in the past; while typically large infrastructural changes are conducted through funding (such as NSF SI2) an enormous amount of functionality that is directly end-user facing, such as IO methods and analysis modules, has been added by volunteers. We note, however, that an inherent risk in relying on this type of contribution is that it can preferentially select against underrepresented and minoritized groups, or non-traditional developers [10].

The `yt` approach to sustainability builds on the best practices identified in those software sustainability groups, emphasizing development of community leaders, ensuring that unfunded stakeholders remain invested rather than alienated in the presence of external funding, and attempting to foster a community of peers. We focus our efforts on sustainability on (1) ensuring that community members are enabled to conduct their research, thus ensuring that they have an indirect incentive to assist the community, (2) that they have interest in contributing back by receiving professional credit for doing so, and (3) that the technical barriers to doing so are not unnecessarily high.

These factors will combine to help, but not guarantee, that the project will remain sustainable past the lifetime of this award. The explicit broadening of impact of `yt` in this proposal will diversify the communities that may take part in sustaining it through community-maintenance and participation. In the NSF SI2-SSE grant that funded `yt` to develop astrophysical simulation tools, we took the approach that we would "prime the pump" with directed efforts at engaging the community. This has been successful so far, and we will continue to do so, by enabling individuals to participate in governing decisions, fostering career paths that benefit from participation in `yt`, and building connections with larger communities of researchers.

5 Broader Ecosystem

5.1 Alternative Software Packages

There are several mature environments for producing 3D visualizations of volumetric data. Many of the alternative offerings offer a desktop application where the primary interaction mode focuses on mouse-driven panning and zooming through 3D datasets, with additional tasks accessible via buttons or dropdown menus and dialogs. Below, we explore several alternative software packages, explaining how `yt` differs and provides a programmatic, inquiry-focused approach that is not emphasized in existing offerings; we view `yt` not as an alternative, but as complementary.

`VisIt` [8] is a visualization platform that can load data in many on-disk formats and indexing styles. The primary means of interacting with `VisIt` is via a windowed GUI, although a Python

scripting facility is also provided. Most scripts are initially generated by converting interactions generated from a GUI session into a Python script; typically this does not lead to exploration through a scripting interface. While `VisIt` is open source, development happens in a centralized repository, and there is no publicly documented contribution process. Additionally, its complexity adds barriers for new community members who want to submit contributions.

`ParaView` [2], like `VisIt`, is a visualization platform for volumetric mesh-based and particle-based data with a user interface built around interacting with a graphical application. At a low level, `ParaView` is powered by the `VTK` C++ library, with a Python scripting interface as well as low-level Python bindings to the C++ `VTK` library. The tight coupling between the Python bindings and `VTK` results in a non-idiomatic pythonic style that requires familiarity with the `VTK` API to be productive. `Mayavi` [32] is an alternative `VTK`-powered visualization package for volumetric data, with high-level bindings to `VTK` that hide most of the complexity of the C++ library; despite this, `Mayavi` is still focused primarily on visualization rather than analysis or scientific inquiry.

5.2 Interoperability

`yt` is built on top of the ecosystem of Scientific Python and PyData packages that are commonly used by researchers in a variety of physical sciences and data science roles. Array operations are managed by `NumPy` [40], plotting is driven by `matplotlib` [16], symbolic computation by `SymPy` [23], and IO can be managed through a variety of mechanisms, including `NetCDF4` and `HDF5` (through `netcdf4-python` and `h5py`). As a result of the interconnectedness of this ecosystem, `yt` is broadly interoperable with an enormous number of tools. `yt` is a NumFOCUS project and members of the `yt` development team (including PI Turk and Co-I Goldbaum) regularly contribute patches to other ecosystem projects as a result of development in `yt`.

In § 5.1 we have identified a number of packages that provide alternatives to some of `yt`'s functionality. Each of the main packages (of `ParaView`, `VisIt` and `Mayavi`) exposes a Python API; in fact, each of these packages has, in the past, been used in concert with `yt`, either with `yt` receiving data or supplying data, or exchanging visualizations between the two. For instance, [24] used `yt` for data processing and collection while `Mayavi` was used to generate 3D visualizations. We anticipate further interoperability may arise a result of community involvement and engagement.

5.3 New Technologies and Adaptation

As a project and a community, `yt` has a well-defined process for identifying, adapting to and adopting new technologies. At present, `yt` forms a core component of numerous astrophysics analysis pipelines. Many researchers, particularly those with less computational facility, are reluctant to rely on new technologies or migrate existing ones in an effort to avoid disruption and invasive changes.

The `yt` engineering process (described in § 1.2) was designed with the intention of balancing these concerns, as well as providing an opportunity for community members to provide feedback on changes. We will use this process to evaluate new "hard" dependencies for `yt`, although support for most new dependencies can be added via a "soft" dependency or through independently versioned and released `yt` extensions. As the primary purpose of the domain context development is to build friendly, domain-specific APIs, underlying functionality, the engine (`yt`) that drives that functionality can be adapted to new technology without disruptive user-facing changes.

For the most part, the ecosystem in which `yt` finds itself is a hybrid of the domain sciences communities and the scientific python ecosystem. Within the scientific python ecosystem, several new projects have been gaining momentum that require serious consideration, and which will be regularly evaluated with respect to their maturity, accessibility, and functionality. In particular, we have identified both `dask` and `numba` as being potentially applicable to `yt`.

The first of these, `dask`, is a mechanism for enabling transparent parallelism for operations on

numpy arrays. While `yt` is currently MPI-parallel, the data decomposition and parallelism available in `dask` may be relevant for future development, and we have begun to evaluate how invasive a change it would be to directly interoperate. The other emerging technology, `numba`, is a just-in-time (JIT) compiler for python code that targets several execution backends; support for `numba` is particularly interesting for the advanced field system (§ 2.2) and for non-local analysis (§ 2.4).

5.4 Metrics for Success

We have identified several metrics of the success and impact of this project.

1. **Scientific**: the number of citations to the extant `yt` method paper and any subsequent method paper, roughly broken down by domain of study. We will seek a year-over-year increase, particularly in non-astrophysics domains.
2. **Project**: we will seek an increase in the total number of contributors, particularly those contributors that are from underrepresented communities. We will also seek to see an increase in the number of participants on the mailing lists for `yt`.
3. **Development**: Each development effort described in this document is accompanied by a corresponding design document. We will seek to identify the overall utilization of those features, either by traffic on the mailing lists or through the scientific literature.

While these metrics are proxies for the overall impact and success of the project, we will also be exploring additional methods for determining whether we have met our goals of disseminating the software to users [15]; this includes the number of downloads (a poor metric), social media engagement, and the number of extension packages developed to build on `yt`'s functionality.

6 Results from Prior Support

Benjamin Holtzman (Columbia PI). NSF EAR-1147763: *Immersive Audio-visualization of Seismic Wave Fields in the Earth (EarthScope Education & Outreach)*, **Amount**: $94,708, **Period**: 07/01/2012-06/30/2014 (NFE to 2016). **Intellectual Merit**: This project lead to development of innovative methods of cinematic display (i.e. simultaneous, synchronized visual and auditory) of global and local seismic wave fields (using `yt`) and other aspects of earthquake physics, to teach the general public about seismology and earthquakes as a window into the workings of the planet. **Broader Impacts**: The visual and sonic representation of seismic data excite curiosity in the public, and also help researchers perceive complex patterns in their data. Products: The centerpiece is a planetarium show that has come to be called "SeismoDome" in the Hayden Planetarium at the American Museum of Natural History (collaborative on this project) NY, performed 4 times for over 1200 people, and a website with movies for teaching.

Nathan Goldbaum (NCSA Co-I): no NSF support in the past five years.

Leigh Orf (University of Wisconsin PI). NSF Grant 1550405: *INSPIRE: Quantitative Estimation of Space-Time Processes in Volumetric Data (QUEST)*, **Amount:** $999,589, **Period:** 1/1/2016 – 12/31/2019. **Intellectual Merit:** This cross-disciplinary grant seeks to identify signatures of tornado genesis and maintenance from multiple-Doppler radar data from the Doppler-On-Wheels network in conjunction with tornado simulations using the CM1 model. Dual-Doppler radar data and CM1 simulation data are being analyzed utilizing a novel analysis approach called entropy field decomposition, an approach that has been used to identify structural and functional modes of the human brain via functional Magnetic Resonance Imagery. **Broader Impacts:** This novel methodology has the potential to transform the way analysis is conducted across disciplines, and thus have significant impact in numerous fields that utilize digital imaging techniques.

Matthew Turk (NCSA PI). NSF 15-35651: *yt: Reusable Components for Simulating, Analyzing and Visualizing Astrophysical Systems*, **Amount**: $427,003.00, **Period**: 11/1/2014-9/30/2017, **Products**: `yt` source code **Intellectual Merit**: The method paper for `yt` has been cited over 260

times and the source code has been used to dramatically advance the speed and detail of research in astrophysical sciences. **Broader Impacts**: The developments from this grant have materially contributed to analysis results and visualizations used in numerous astrophysical papers. `yt` has additionally been used in the development of planetarium shows for education and public outreach.

7 Broader Impacts

The key deliverables of the proposal will lead to the development of an inquiry-based analysis and visualization platform for volumetric data suitable for use by a cross-disciplinary community of researchers. Students and early-career researchers will be able to make use of `yt` for data exploration, quick visualization, and analysis tasks, obviating the need to implement bespoke I/O implementations for common data formats and to reimplement common analysis algorithms. As we have found in astrophysics, going from a status quo of students needing to implement readers for their data, along with custom implementations of analysis algorithms, to a community solution like `yt` can substantially reduce the cognitive overhead and struggle to produce research results. Reducing barriers to entry and building a feeling of self-efficacy [3, 7] can lead to increased long-term retention of students, particularly among historically underrepresented or minoritized groups.

In addition to expanding the community of `yt` users, this proposal will also expand the community of `yt` developers. In large part, the community of users and developers of `yt` is an overlapping set, so these goals are mutually reinforcing [39]. Increasingly, software development is happening in the open on social coding sites like Github and Bitbucket. The `yt` community offers a way to train students and early-career scientists in the norms of social coding practices. In addition, we will identify mentors for targeted outreach in scientific software development, in particular women and members of minoritized or underrepresented groups, with the ultimate goal of expanding the diversity of the `yt` development community. Norms in the `yt` community, including a code of conduct and a formal process for resolving disputes, allows developers to interact in the community as a safe space free of judgement and and harassment, ideal for new developers. Given typical career outcomes for students in STEM fields, many of the early career researchers exposed to the `yt` developer community will end up working in industry. Since the `yt` community uses industry-standard development practices, including social, distributed development and distributed version control, continuous integration, and established practices, expectations, and workflows for new code contributions, we are also performing workforce development for the next generation of software developers and data scientists. Several former `yt` developers who left academia for industry cite participation in the `yt` community as singularly helpful for making their transition to industry.

The development of Data Carpentry curriculum and workshops will allow people beyond the `yt` user and developer community to learn about `yt` at workshops where there will be students working in fields that `yt` can operate. Many students and researchers have identified Data Carpentry (and its sibling organization, Software Carpentry) as a turning point in their careers, where they became comfortable with managing, interacting with and understanding research data and software. Distributing the `yt`-focused lessons through the Midwest Big Data Hub will engage with universities throughout the midwest as well as with private industry and other groups.

This project will enable outreach programs making use of data produced at NSF-funded research platforms like XSEDE, CIG, and IRIS. As `yt` is increasingly able to read research data formats, it will become much easier to extract postprocessed data for advanced visualizations and planetarium shows for the public as well as the scientific community. As demonstrated through the success of Seismodome, developing visually-thrilling, curiosity-provoking images and animations using tools *already used by scientists* can broaden STEM outreach and make outreach accessible to larger swaths of both the public and the scientific communities. By developing `yt` to reach more domains of science, we will build deeper capabilities for outreach and public understanding of science.

References

[1] Astropy Collaboration, T. P. Robitaille, E. J. Tollerud, P. Greenfield, M. Droettboom, E. Bray, T. Aldcroft, M. Davis, A. Ginsburg, A. M. Price-Whelan, W. E. Kerzendorf, A. Conley, N. Crighton, K. Barbary, D. Muna, H. Ferguson, F. Grollier, M. M. Parikh, P. H. Nair, H. M. Unther, C. Deil, J. Woillez, S. Conseil, R. Kramer, J. E. H. Turner, L. Singer, R. Fox, B. A. Weaver, V. Zabalza, Z. I. Edwards, K. Azalee Bostroem, D. J. Burke, A. R. Casey, S. M. Crawford, N. Dencheva, J. Ely, T. Jenness, K. Labrie, P. L. Lim, F. Pierfederici, A. Pontzen, A. Ptak, B. Refsdal, M. Servillat, and O. Streicher. Astropy: A community Python package for astronomy. *A&A*, 558:A33, October 2013.

[2] Utkarsh Ayachit. *The ParaView Guide: A Parallel Visualization Application*. Kitware, Inc., USA, 2015.

[3] Albert Bandura. *Self-efficacy*. Wiley Online Library, 1994.

[4] Brett Bode, Michelle Butler, Thom Dunning, Torsten Hoefler, William Kramer, William Gropp, and Wen-Mei Hwu. *The Blue Water Super-System for Super-Science*, pages 339–366. Chapman & Hall/CRC Computational Science. Chapman and Hall/CRC, Apr 2013.

[5] G Bryan. CM1 homepage. `http://www2.mmm.ucar.edu/people/bryan/cm1/`, 2008. Accessed: 2016-08-31.

[6] George H. Bryan and J. Michael Fritsch. A benchmark simulation for moist nonhydrostatic numerical models. *Monthly Weather Review*, 130(12):2917–2928, 2002.

[7] LaVar Charleston and Raul Leon. Constructing self-efficacy in stem graduate education. *Journal for Multicultural Education*, 10(2):152–166, 2016.

[8] Hank Childs, Eric Brugger, Brad Whitlock, Jeremy Meredith, Sean Ahern, David Pugmire, Kathleen Biagas, Mark Miller, Cyrus Harrison, Gunther H. Weber, Hari Krishnan, Thomas Fogal, Allen Sanderson, Christoph Garth, E. Wes Bethel, David Camp, Oliver Rübel, Marc Durant, Jean M. Favre, and Paul Navrátil. VisIt: An End-User Tool For Visualizing and Analyzing Very Large Data. In *High Performance Visualization–Enabling Extreme-Scale Scientific Insight*, pages 357–372. Oct 2012.

[9] K. Crowston, J. Howison, C. Masango, and U. Y. Eseryel. The role of face-to-face meetings in technology-supported self-organizing distributed teams. *IEEE Transactions on Professional Communication*, 50(3):185–203, Sept 2007.

[10] Ashe Dryden. The ethics of unpaid labor and the oss community. `https://www.ashedryden.com/blog/the-ethics-of-unpaid-labor-and-the-oss-community`, 2013.

[11] Derek Gaston, Chris Newman, Glen Hansen, and Damien Lebrun-Grandi. Moose: A parallel computational framework for coupled systems of nonlinear equations. *Nuclear Engineering and Design*, 239(10):1768 – 1778, 2009.

[12] Benjamin Holtzman, Jason Candler, Matthew Turk, and Daniel Peter. *Seismic Sound Lab: Sights, Sounds and Perception of the Earth as an Acoustic Space*, pages 161–174. Springer International Publishing, Cham, 2014.

[13] James Howison. Sustaining scientific infrastructures: transitioning from grants to peer production (work-in-progress). *iConference 2015 Proceedings*, 2015.

[14] James Howison and Kevin Crowston. Collaboration through open superposition: A theory of the open source way. *Mis Quarterly*, 38(1):29–50, 2014.

[15] James Howison, Ewa Deelman, Michael J. McLennan, Rafael Ferreira da Silva, and James D. Herbsleb. Understanding the scientific software ecosystem and its impact: Current and future measures. *Research Evaluation*, 2015.

[16] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

[17] Daniel S Katz, Sou-Cheng T Choi, Hilmar Lapp, Ketan Maheshwari, Frank Löffler, Matthew Turk, Marcus D Hanwell, Nancy Wilkins-Diehr, James Hetherington, James Howison, Shel Swenson, Gabrielle Allen, Anne Elster, Bruce Berriman, and Colin Venters. Summary of the first workshop on sustainable software for science: Practice and experiences (wssspe1). *arXiv preprint arXiv:1404.7414*, 2014.

[18] Dimitri Komatitsch and Jeroen Tromp. Introduction to the spectral element method for three-dimensional seismic wave propagation. *Geophysical journal international*, 139(3):806–822, 1999.

[19] Meagan Lang and Matthew J. Turk. Bitmap Indexing for Fast Analysis of Large Astrophysical Datasets. *Comput. Sci. Eng.*, submitted.

[20] Tyler McEwen, Dharhas Pothina, and Solomon Negusse. Improving efficiency and repeatability of lake volume estimates using python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 10th Python in Science Conference*, pages 103 – 108, 2011.

[21] Celso L Mendes, Brett Bode, Gregory H Bauer, Jeremy Enos, Cristina Beldica, and William T Kramer. Deploying a large petascale system: The blue waters experience. *Procedia Computer Science*, 29:198–209, 2014.

[22] Venkatesh M. Merwade, David R. Maidment, and John A. Goff. Anisotropic considerations while interpolating river channel bathymetry. *Journal of Hydrology*, 331(3-4):731–741, 12 2006.

[23] A Meurer, CP Smith, M Paprocki, O ertk, SB Kirpichev, M Rocklin, A Kumar, S Ivanov, JK Moore, S Singh, T Rathnayake, S Vig, BE Granger, RP Muller, F Bonazzi, H Gupta, S Vats, F Johansson, F Pedregosa, MJ Curry, AR Terrel, Rouka, A Saboo, I Fernando, S Kulal, R Cimrman, and A. Scopatz. Sympy: Symbolic computing in python. *PeerJ*, 2016.

[24] S. J. Mumford and R. Erdélyi. Photospheric logarithmic velocity spirals as MHD wave generation mechanisms. *MNRAS*, 449:1679–1685, May 2015.

[25] T. Nissen-Meyer, M. van Driel, S. C. Stähler, K. Hosseini, S. Hempel, L. Auer, and A. Fournier. AxiSEM: broadband 3-D seismic wavefields in axisymmetric media. *Solid Earth Discussions*, 6(1):265–319, 2014.

[26] B. A. Nosek, J. R. Spies, and M. Motyl. Scientific Utopia: II. Restructuring incentives and practices to promote truth over publishability. *ArXiv e-prints*, May 2012.

[27] Leigh Orf, Erica Kantor, and Eric Savory. Simulation of a downburst-producing thunderstorm using a very high-resolution three-dimensional cloud model. *Journal of Wind Engineering and Industrial Aerodynamics*, 104106:547 – 557, 2012. 13th International Conference on Wind Engineering.

[28] Leigh Orf, Robert Wilhelmson, Bruce Lee, Cathy Finley, and Adam Houston. Evolution of a long-track violent tornado within a simulated supercell. *Bulletin of the American Meteorological Society*, (2016), 2016.

[29] Leigh Orf, Robert Wilhelmson, and Louis Wicker. Visualization of a simulated long-track {EF5} tornado embedded within a supercell thunderstorm. *Parallel Computing*, 55:28 – 34, 2016. Visualization and Data Analytics for Scientific Discovery.

[30] Carole L. Palmer. Structures and strategies of interdisciplinary science. *Journal of the American Society for Information Science*, 50(3):242–253, 1999.

[31] Daniel Peter, Dimitri Komatitsch, Yang Luo, Roland Martin, Nicolas Le Goff, Emanuele Casarotti, Pieyre Le Loher, Federica Magnoni, Qinya Liu, Céline Blitz, Tarje Nissen-Meyer, Piero Basini, and Jeroen Tromp. Forward and adjoint simulations of seismic wave propagation on fully unstructured hexahedral meshes. *Geophysical Journal International*, 186(2):721–739, 2011.

[32] P. Ramachandran and G. Varoquaux. Mayavi: 3D Visualization of Scientific Data. *Computing in Science & Engineering*, 13(2):40–51, 2011.

[33] Russ Rew, Glenn Davis, Steve Emmerson, Cathy Cormack, John Caron, Robert Pincus, Ed Hartnett, Dennis Heimbigner, Lynton Appel, and Ward Fisher. Unidata netcdf, 1989.

[34] Victoria Stodden and Sheila Miguez. Best practices for computational science: Software infrastructure and environments for reproducible and extensible research. *Journal of Open Research Software*, 2(1):e21, jul 2014.

[35] The HDF Group. Hierarchical Data Format, version 5, 1997. http://www.hdfgroup.org/HDF5/.

[36] the PyNE Development Team. PyNE: The Nuclear Engineering Toolkit, 2014.

[37] Frank Timmes, Matthew Turk, Stan Ahalt, Shaowen Wang, Ray Idaszak, Richard Brower, Chris Lenhardt, and Karl Gustafson. 2016 software infrastructure for sustained innovation (si2) pi workshop. Technical report, USA, 2016.

[38] M. J. Turk, B. D. Smith, J. S. Oishi, S. Skory, S. W. Skillman, T. Abel, and M. L. Norman. yt: A Multi-code Analysis Toolkit for Astrophysical Simulation Data. *ApJS*, 192:9, January 2011.

[39] Matthew J Turk. Scaling a code in the human dimension. In *Proceedings of the Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery*, page 69. ACM, 2013.

[40] Stefan van der Walt, S. Chris Colbert, and Gael Varoquaux. The numpy array: A structure for efficient numerical computation. *Computing in Science and Engg.*, 13(2):22–30, March 2011.

[41] R.L. Williamson, J.D. Hales, S.R. Novascone, M.R. Tonks, D.R. Gaston, C.J. Permann, D. Andrs, and R.C. Martineau. Multidimensional multiphysics simulation of nuclear fuel behavior. *Journal of Nuclear Materials*, 423(13):149 – 163, 2012.