The Journal of Open Source Software

Kamodo: A functional api for space weather models and data

Asher Pembroke[1], Darren DeZeeuw[2, 3], Lutz Rastaetter[2], Rebecca Ringuette[4, 2], Oliver Gerland[5], Dhruv Patel[5], and Michael Contreras[5]

1 Asher Pembroke, DBA **2** Community Coordinated Modeling Center, NASA GSFC **3** University of Michigan **4** ADNET Systems Inc. **5** Ensemble Government Services

DOI: 10.21105/joss.03963

Software

- Review
- Repository
- Archive

Editor: Pending Editor

Submitted: 29 November 2021
Published: 30 November 2021

Summary

Kamodo is a functional programing interface for scientific models and data. In Kamodo, all scientific resources are registered as symbolic fields which are mapped to model and data interpolators or algebraic expressions. Kamodo performs function composition and employs a unit conversion system that mimics hand-written notation: units are declared in bracket notation and conversion factors are automatically inserted into user expressions. Kamodo includes a LaTeX interface, automated plots, and a browser-based dashboard interface suitable for interactive data exploration. Kamodo's json API provides context-dependent queries and allows compositions of models and data hosted in separate docker containers. Kamodo is built primarily on sympy (Meurer et al., 2017) and plotly (Plotly Technologies Inc., 2015). While Kamodo was designed to solve the cross-disciplinary challenges of the space weather community, it is general enough to be applied in other fields of study.

Statement of need

Space weather models and data employ a wide variety of specialized formats, data structures, and interfaces tailored for the needs of domain experts. However, this specialization is also an impediment to cross-disciplinary research. For example, data-model comparisons often require knowledge of multiple data structures and observational data formats. Even when mature APIs are available, proficiency in programing languages such as python is necessary before progress may be made. This further complicates the transition from research to operations in space weather forecasting and mitigation, where many disparate data sources and models must be presented together in a clear and actionable manner. Such complexity represents a high barrier to entry when introducing the field of space weather to newcomers at space weather workshops, where much of the student's time is spent installing and learning how to use prerequisite software. Several attempts have been made to unify all existing space weather resources around common standards, but have met with limited success.

Kamodo all but eliminates the barrier to entry for space weather resources by exposing all scientifically relevant parameters in a functional manner. Kamodo is an ideal tool in the scientist's workflow, because many problems in space weather analysis, such as field line tracing, coordinate transformation, and interpolation, may be posed in terms of function compositions. Kamodo builds on existing standards and APIs and does not require programing expertise on the part of end user. Kamodo is expressive enough to meet the needs of most scientists, educators, and space weather forecasters, and Kamodo containers enable a rapidly growing ecosystem of interoperable space weather resources.

Usage

Kamodo Base Class

Kamodo's base class manages the registration of functionalized resources. As an example, here is how one might register the non-differentiable Weierstrass function (Weierstrass, 1872).

```
from kamodo import Kamodo, kamodofy
import numpy as np

@kamodofy(
    equation=r"\sum_{n=0}^{500} (1/2)^n cos(3^n \pi x)",
    citation='Weierstrass, K. (1872). Uber continuirliche functionen eines reellen
    )
def weierstrass(x = np.linspace(-2, 2, 1000)):
    '''
    Weierstrass function (continuous and non-differentiable)

    https://en.wikipedia.org/wiki/Weierstrass_function
    '''
    nmax = 500
    n = np.arange(nmax)

    xx, nn = np.meshgrid(x, n)
    ww = (.5)**nn * np.cos(3**nn*np.pi*xx)
    return ww.sum(axis=0)

k = Kamodo(W=weierstrass)
```

When run in a jupyter notebook, the latex representation of the above function is shown:

$$W(x) = \sum_{n=0}^{500} (1/2)^n cos(3^n \pi x) \tag{1}$$

This function can be queried at any point within its domain:

```
k.W(0.25)

# array([0.47140452])
```

Kamodo's plotting routines can automatically visualize this function at multiple zoom levels:

```
k.plot('W')
```

The result of the above command is shown in Figure 1. This exemplifies Kamodo's ability to work with highly resolved datasets through function inspection.

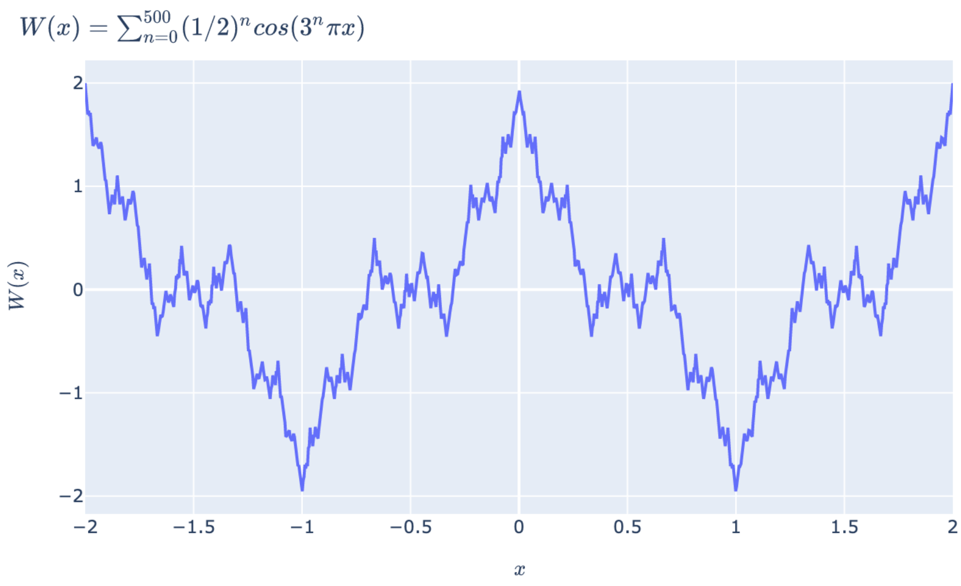

Figure 1: Auto-generated plot of the Weirstrass function.

Kamodo Subclasses

The Kamodo base class may be subclassed when third-packages are required. For example, the pysatKamodo subclass preregisters interpolating functions for Pysat (Stoneback et al., 2019) Instruments:

```
from pysat_kamodo.nasa import Pysat_Kamodo

kcnofs = Pysat_Kamodo('2009, 1, 1', # Pysat_Kamodo allows string dates
         platform = 'cnofs', # pysat keyword
         name='vefi', # pysat keyword
         tag='dc_b',# pysat keyword
         )
kcnofs['B'] = '(B_north**2+B_up**2+B_west**2)**.5' # a derived variable
```

Here is how the kcnofs instance appears in a jupyter notebook:

$$\mathrm{B}_{\mathrm{north}}(t)[nT] = \lambda(t) \tag{2}$$

$$\mathrm{B}_{\mathrm{up}}(t)[nT] = \lambda(t) \tag{3}$$

$$\mathrm{B}_{\mathrm{west}}(t)[nT] = \lambda(t) \tag{4}$$

$$\mathrm{B}_{\mathrm{flag}}(t) = \lambda(t) \tag{5}$$

$$\mathrm{B}_{\mathrm{IGRFnorth}}(t)[nT] = \lambda(t) \tag{6}$$

$$\mathrm{B}_{\mathrm{IGRFup}}(t)[nT] = \lambda(t) \tag{7}$$

$$\mathrm{B_{IGRFwest}}\,(t)[nT] = \lambda(t) \tag{8}$$

$$\mathrm{latitude}\,(t)[degrees] = \lambda(t) \tag{9}$$

$$\mathrm{longitude}\,(t)[degrees] = \lambda(t) \tag{10}$$

$$\mathrm{altitude}\,(t)[km] = \lambda(t) \tag{11}$$

$$\mathrm{dB_{zon}}\,(t)[nT] = \lambda(t) \tag{12}$$

$$\mathrm{dB_{mer}}\,(t)[nT] = \lambda(t) \tag{13}$$

$$\mathrm{dB_{par}}\,(t)[nT] = \lambda(t) \tag{14}$$

$$B(t)[nT] = \sqrt{\mathrm{B_{north}}^2\,(t) + \mathrm{B_{up}}^2\,(t) + \mathrm{B_{west}}^2\,(t)} \tag{15}$$

Units are explicitly shown on the left hand side, while the right hand side of these expressions represent interpolating functions ready for evaluation:

```
kcnofs.B(pd.DatetimeIndex(['2009-01-01 00:00:03','2009-01-01 00:00:05']))
```

```
2009-01-01 00:00:03    19023.052734
2009-01-01 00:00:05    19012.949219
dtype: float32
```

Here, the function `B(t)` returns the result of a variable derived from preregisterd variables as a pandas series object. However, kamodo itself does not require functions to utilize a specific data type, provided that the datatype supports algebraic operations.

Kamodo can auto-generate plots using function inspection:

```
kcnofs.plot('B_up')
```

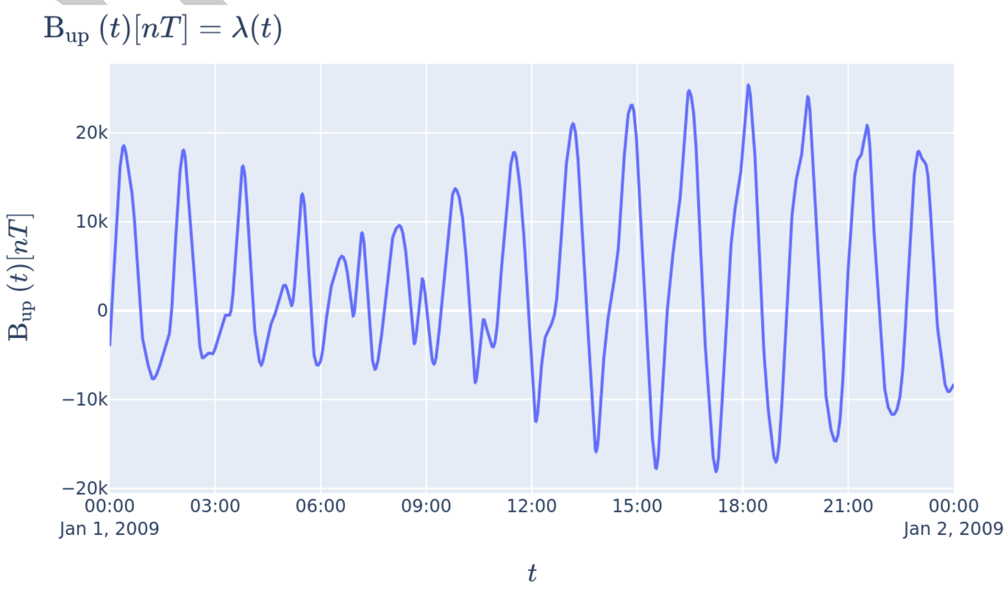

Figure 2: Auto-generated plot of CNOFs Vefi instrument.

The result of the above command is shown in Figure 2. To accomplish this, Kamodo analyzes the structure of inputs and outputs of B_up and selects an appropriate plot type from the Kamodo plotting module.

Citation information for the above plot may be generated from the meta property of the registered function:

```
kcnofs.B_up.meta['citation']
```

which returns references for the C/NOFS platform (Beaujardière, 2004) and VEFI instrument (Pfaff et al., 2010).

Related Projects

Kamodo is designed for compatibility with python-in-heliophysics (Ware et al., 2019) packages, such as PlasmaPy (PlasmaPy Community et al., 2020) and PySat (Stoneback et al., 2018), (Stoneback et al., 2019). This is accomplished through Kamodo subclasses, which are responsible for registering each scientifically relevant variable with an interpolating function. Metadata describing the function's units and other supporting documentation (citation, latex formatting, etc) may be provisioned by way of the @kamodofy decorator.

The PysatKamodo (Asher Pembroke, 2021) interface is made available in a separate git repository. Readers for various space weather models and data sources are under development by the Community Coordinated Modling Center and are hosted in their official NASA repository (D. Pembroke A, 2021).

Kamodo's unit system is built on SymPy (Meurer et al., 2017) and shares many of the unit conversion capabilities of Astropy (Astropy Collaboration, 2013) with two key differences: Kamodo uses an explicit unit conversion system, where units are declared during function registration and appropriate conversion factors are automatically inserted on the right-hand-side of final expressions, which permits back-of-the-envelope validation. Second, units are treated as function metadata, so the types returned by functions need only support algebraic manipulation (Numpy (Harris et al., 2020), Pandas (team, 2020), etc). Output from kamodo-registered functions may still be cast into other unit systems that require a type, such as Astropy (Astropy Collaboration, 2013) and Pint (Aaron Coleman, 2021).

Kamodo can utilize some of the capabilities of raw data APIs such as HAPI, and a HAPI kamodo subclass is maintained in the ccmc readers repository (D. Pembroke A, 2021). However, Kamodo also provides an API for purely functional data access, which allows users to specify positions or times for which interpolated values should be returned. To that end, a prototype for functional REST api (Fielding, 2000) is available (P. Pembroke A, 2021) and an RPC api (Nelson, 2020) for direct access from other programming languages is under development.

Kamodo container services may be built on other containerized offerings. Containerization allows dependency conflicts to be avoided through isolated install environments. Kamodo extends the capabilities of space weather resource containers by allowing them to be composed together via the KamodoClient, which acts as a proxy for the containerized resource running the KamodoAPI.

Acknowledgements

Development of Kamodo was initiated by the Community Coordinated Modeling Center, with funding provided by Catholic University of America under the NSF Division of Atmospheric and Geospace Sciences, Grant No 1503389. Continued support for Kamodo is provided by Ensemble Government Services, LTD. via NASA Small Business Innovation Research (SBIR) Phase I/II, grant No 80NSSC20C0290, 80NSSC21C0585, resp. Additional support is provided by NASA's Heliophysics Data and Model Consortium.

The authors are thankful for the advice and support of Nicholas Gross, Katherine Garcia-Sage for, and Richard Mullinex.

References

Aaron Coleman, et al. (2021). Pint. In *GitHub repository*. https://github.com/hgrecco/pint; GitHub.

Asher Pembroke, et al. (2021). PysatKamodo. In *Github repository*. https://github.com/pysat/pysatKamodo; GitHub.

Astropy Collaboration. (2013). Astropy: A community Python package for astronomy. *Astronomy and Astrophysics*, *558*. https://doi.org/10.1051/0004-6361/201322068

Beaujardière, O. (2004). C/NOFS: A mission to forecast scintillations. *Journal of Atmospheric and Solar-Terrestrial Physics*, *66*, 1573–1591. https://doi.org/10.1016/j.jastp.2004.07.030

Fielding, R. T. (2000). REST: Architectural styles and the design of network-based software architectures. *Doctoral Dissertation, University of California*.

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J. K., Singh, S., Rathnayake, T., Vig, S., Granger, B. E., Muller, R. P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., … Scopatz, A. (2017). SymPy: Symbolic computing in python. *PeerJ Computer Science*, *3*, e103. https://doi.org/10.7717/peerj-cs.103

Nelson, B. J. (2020). *Remote procedure call, 1981*. PARC CSL-81-9, Xerox Palo Alto Research Center, Palo Alto, CA.

Pembroke, D., A. (2021). nasaKamodo. In *Github repository*. https://github.com/nasa/Kamodo; GitHub.

Pembroke, P., A. (2021). Kamodo-core. In *Github repository*. https://github.com/EnsembleGovServices/kamodo-core; GitHub.

Pfaff, R., Rowland, D., Freudenreich, H., Bromund, K., Le, G., Acuña, M., Klenzing, J., Liebrecht, C., Martin, S., Burke, W. J., Maynard, N. C., Hunton, D. E., Roddy, P. A., Ballenthin, J. O., & Wilson, G. R. (2010). Observations of DC electric fields in the low-latitude ionosphere and their variations with local time, longitude, and plasma density during extreme solar minimum. *Journal of Geophysical Research: Space Physics*, *115*(A12). https://doi.org/https://doi.org/10.1029/2010JA016023

PlasmaPy Community, Everson, E., Stańczak, D., Murphy, N. A., Kozlowski, P. M., Malhotra, R., Langendorf, S. J., Leonard, A. J., Stansby, D., Haggerty, C. C., Mumford, S. J., Beckers, J. P., Bedmutha, M. S., Bergeron, J., Bessi, L., Bryant, K., Carroll, S., Chambers, S., Chattopadhyay, A., … Skinner, C. (2020). *PlasmaPy* (Version 0.5.0) [Computer software]. Zenodo. https://doi.org/10.5281/zenodo.4313063

Plotly Technologies Inc. (2015). *Collaborative data science*. Plotly Technologies Inc. https://plot.ly

Stoneback, R. A., Burrell, A. G., Klenzing, J., & Depew, M. D. (2018). PYSAT: Python Satellite Data Analysis Toolkit. *Journal of Geophysical Research: Space Physics*, *123*(6), 5271–5283. https://doi.org/10.1029/2018JA025297

Stoneback, R. A., Klenzing, J. H., Burrell, A. G., Spence, C., Depew, M., Hargrave, N., Bose, V. von, Luis, S., & Iyer, G. (2019). *Python satellite data analysis toolkit (pysat) vX.y.z*. https://doi.org/10.5281/zenodo.1199703

team, T. pandas development. (2020). *Pandas-dev/pandas: pandas* (Version 1.3.4) [Computer software]. Zenodo. https://doi.org/10.5281/zenodo.3509134

Ware, A., Barnum, J., Candey, R., Cecconi, B., Christe, S., Faden, J., Grimes, E., Harris, B., Harter, B., Kilcommons, L., Loh, A., McGuire, R., Mumford, S., Narock, A., Nguyen, Q. N., Niehof, J., Maldonado, A., Murphy, N., Panneton, R., … Woodraska, D. (2019). *Python in heliophysics community meeting*. Zenodo. https://doi.org/10.5281/zenodo.2537188

Weierstrass, K. (1872). Uber continuirliche functionen eines reellen arguments, die fur keinen worth des letzteren einen bestimmten differentailqutienten besitzen, akademievortrag. *Math. Werke*, 71–74.