Geosci. Model Dev., 5, 449–455, 2012
www.geosci-model-dev.net/5/449/2012/
doi:10.5194/gmd-5-449-2012

Geoscientific Model Development

"Gtool5": a Fortran90 library of input/output interfaces for self-descriptive multi-dimensional numerical data

M. Ishiwatari[1], E. Toyoda[2], Y. Morikawa[3], S. Takehiro[4], Y. Sasaki[5], S. Nishizawa[6], M. Odaka[1], N. Otobe[7], Y. O. Takahashi[6], K. Nakajima[8], T. Horinouchi[9], M. Shiotani[10], Y.-Y. Hayashi[6], and Gtool development group[11]

[1]Hokkaido University, Graduate school of Science, Sapporo, Japan
[2]Japan Meteorological Agency, Tokyo, Japan
[3]Parmy, Inc., Japan
[4]Kyoto University, Research Institute for Mathematical Sciences, Kyoto, Japan
[5]Kyoto University, Department of Mathematics, Kyoto, Japan
[6]Kobe University, Graduate school of Sciences, Kobe, Japan
[7]Fukuoka University, Faculty of Science, Fukuoka, Japan
[8]Kyushu University, Faculty of Science, Fukuoka, Japan
[9]Hokkaido University, Graduate school of Environmental Earth Science, Sapporo, Japan
[10]Kyoto University, Research Institute for Sustainable Humanosphere, Kyoto, Japan
[11]GFD Dennou Club, Japan

Correspondence to: S. Takehiro (takepiro@gfd-dennou.org)

Received: 4 November 2011 – Published in Geosci. Model Dev. Discuss.: 19 December 2011
Revised: 19 March 2012 – Accepted: 19 March 2012 – Published: 3 April 2012

Abstract. A Fortran90 input/output library, "gtool5", is developed for use with numerical simulation models in the fields of Earth and planetary sciences. The use of this library will simplify implementation of input/output operations into program code in a consolidated form independent of the size and complexity of the software and data. The library also enables simple specification of the metadata needed for post-processing and visualization of the data. These aspects improve the readability of simulation code, which facilitates the simultaneous performance of multiple numerical experiments with different software and efficiency in examining and comparing the numerical results. The library is expected to provide a common software platform to reinforce research on, for instance, the atmosphere and ocean, where a close combination of multiple simulation models with a wide variety of complexity of physics implementations from massive climate models to simple geophysical fluid dynamics models is required.

1 Introduction

General circulation models of the atmosphere and ocean (GCMs) are commonly utilized for research on the surface environment of the Earth and other planets. However, as described in the next section, research with GCMs requires not only high-end complex GCMs but also simplified models and/or process study models with a wide variety of complexity. In order to perform numerical experiments efficiently using a variety of simulation models with different complexities, numerical data should be self-descriptive and input/output of such data must be handled in a consolidated manner among the models. NetCDF is a data format commonly used in the Earth and planetary sciences and its manipulation library can handle metadata contained in netCDF data. However, netCDF operations are relatively low-level, so that many small steps are needed for data manipulation. This leads to a diversity of ways to implement input and output in models. These are considered to be obstacles for researchers who are developing and/or using numerical models in these fields. With these issues in mind, we have developed an input/output Fortran90 library, called "gtool5", which can be used commonly among numerical simulation

models, and whose interfaces are improved so as to minimize the operational steps required in handling numerical data. The gtool5 library is distributed on our web page: http://www.gfd-dennou.org/library/gtool/gtool5.htm.en.

In the following paper, Sect. 2 will explain, using research with atmosphere and ocean models as an example, technical issues of concern in simulation research in the Earth and planetary sciences that have led us to the development of gtool5. In Sect. 3, the data format of the gtool5 library will be presented, and the sophisticated input/output interfaces of the library are described. Section 4 will introduce examples of implementation of the gtool5 library interfaces to numerical models. We will also demonstrate some tools for post-processing/visualization of output data. Concluding remarks are made in Sect. 5.

2 Current issues in research with atmosphere/ocean models

2.1 Necessity of hierarchical models

GCMs are numerical simulation models that calculate fluid motion of the atmosphere, oceans, and ice sheets, and incorporate various kinds of elemental physical processes such as radiation, turbulence, cloud formation and precipitation, and ground surface processes. Computation of each elemental physical process in GCMs is not based on the first principles of physics, but instead is carried out by implementing a model system derived as a certain approximation of the elemental process. This method of implementation is often referred to as "parameterization". Validity of the numerical results of GCMs that contain many parameterization processes is not guaranteed a priori, and should be checked independently. Comparison with observational data is the most direct validation method. However, acquiring observational data is a significant undertaking in itself and observable physical properties are limited.

In order to compensate for the limitation of observations, process study models that more precisely describe elementary processes or simplified conceptual models that enable us to understand possible behaviors of the system concerned have been developed and utilized. Some of the parameterizations implemented in GCMs have been validated by comparing the numerical results with those of corresponding process study models. Examples of simplified conceptual models used to capture the rough behavior of GCMs are energy balance models where hydrodynamic calculations are neglected, and geophysical fluid dynamics models that do not calculate radiation processes. In recent years, however, it has become increasingly difficult to check the validity of GCMs using these procedures, since GCMs have become so complicated that it is not easy for a researcher to understand the program as a whole in order to recognize the effect of a certain elementary physical process, or to reduce the system in order to build up a conceptual model. This has been recognized as a "model-gap" problem (Held, 2005). In order to fill the gap between high-end complex models (GCMs) and process studies or simplified conceptual models, it is necessary to prepare a software environment that enables the preparation of an arbitrary set of models in a hierarchical fashion with various levels of complexity ranging from GCMs to simplified conceptual models or process study models. Subsequently, we need to compare the results of multiple simultaneous numerical experiments with these hierarchical models.

2.2 Requirements for hierarchical models

In order to build up various numerical models with different complexities and carry out numerical experiments efficiently, readability of the program code is an important factor. Readability is a critical issue not only for model maintainers but also for model users. The model users, who are not necessarily professionals in software technology, should be able to understand how the source code of the numerical models realize the original physical systems, i.e. the corresponding set of mathematical equations. This is the most peculiar requirement for the software utilized in the research of these fields.

In addition to the readability of the program code, the content and structure of the output data also affect the efficiency of the research process, since the typical size of the output data produced by these models tends to be quite large. In order to perform post-processing and visualization operations on these massive datasets, they must be self-descriptive; the data of the dependent variables and coordinate variables are needed. In addition, for management of many massive datasets, we need to record the parameters of the experimental conditions in each dataset. However, if the input/output interfaces for such self-descriptive datasets differ in the program code, readability of the code from one program to another is reduced; and we may not be able to manipulate multiple simulation models hierarchically.

In these circumstances, we need an input/output interface in a consolidated form that is independent of the model complexities. Moreover, the output interface for recording metadata should not be complicated; it should be adequately simplified so as not to reduce the readability of the program code. Most GCMs and related models in atmosphere and ocean sciences have been developed as Fortran programs. In order to satisfy these requirements, we have designed and implemented "gtool5", as described in the following section.

3 Gtool5: a Fortran90 library for data input/output interfaces

3.1 Gtool5 data format

Gtool5 handles time series of multi-dimensional (mainly from one to three dimensional) grid data. For these data,

we adopted a self-descriptive data format netCDF that can contain the supplementary information needed for post-processing and parameter values used in numerical experiments together with numerical data. NetCDF has been developed by Unidata, part of the University Corporation for Atmospheric Research (UCAR). NetCDF files can store arrays of arbitrary size and attach an arbitrary number of metadata, which are called "attributes", to the file and variables.

In order to use netCDF metadata in post-processing/visualization tools, output data must contain attributes describing the title, variable names, variable units, and so on. Naming rules of attribute names and the format of attribute values should be standardized, since the usage of attributes are not prescribed in the framework of netCDF itself. Therefore, we firstly designed conventions which provide naming rules and usages of attributes of netCDF data suitable for research with GCMs and associated hierarchical models, i. e. gtool4 NetCDF Conventions. Gtool4 netCDF conventions started to be developed and have been maintained since 2001, when CF conventions (Eaton et al., 2003), used in meteorology and oceanography widely nowadays, were not present. In 2006, since CF conventions (ver.1.0) became popular, we reviewed our conventions and concluded that: gtool4 NetCDF Conventions were compatible enough to CF conventions in pragmatic sense of the time; some features missing in CF conventions were useful or deserved future testing, and hence it is too early to decide on migration into CF conventions. The two conventions have little difference so that most data can circulate only with changing "Conventions". The most notable extension to CF conventions (and its predecessors) is meaning of "positive" attribute. We generalized it to specify system behavior instead of description of coordinates represented by the variable. The attribute can be attached to any coordinate variables to specify "default direction of rendering". The default value positive = "up" stands for bottom-to-top or left-to-right direction of coordinate, and "down" for the opposite. Other extensions are attributes for visualization hints such as default values of contour intervals and ranges.

3.2 Gtool5 data input/output interfaces

3.2.1 Design policy of gtool5 interfaces

The gtool5 library provides input/output interfaces that resolve the problem of the gap of granularity between the standard netCDF application programming interface (API) and model data object. The standard netCDF API corresponds to a basic data structure such as an array or string, while "a unit of dataset" is usually considered as a number of associations of arrays and text attributes for researchers using GCMs. Many operation steps are needed for the original netCDF API to be used for self-descriptive data output of several variables, since preparative operations for the output must be called for each variable in the program source code. However, for the variables used in the GCMs and associated hierarchical models, there are many common preparative operations, such as settings of coordinate variables. Therefore, by hiding such common preparative operations into the input/output library, we reduce the number of operation steps in the program code of the models.

The input/output interfaces are designed not to degrade the readability of the program source code as described in Sect. 2. In order to retain traceability of source code for calculation of fluid motion and elemental physical processes, the library tries to encapsulate details of data input/output inside the library. Model users do not have to concern themselves with computational science concepts such as file handles (unit numbers in Fortran), filenames, or division of grids in parallel processing once they are configured in the initialization subroutine. The library provides two kinds of interfaces suitable for small and massive models. The interfaces have been constructed to be as similar as possible to minimize the cost of learning.

3.2.2 Data input/output interface for small models

A sample interface is shown in Table 1. This interface is for small models in which data are output at regular time intervals and multiple variables are output into a single file. Major subroutines of this interface are HistoryCreate, HistoryAddVariable, HistoryPut, HistoryAddAttr, and HistoryClose. The functions and major arguments of these five subroutines are summarized in Table 1.

The subroutine HistoryCreate initializes an output file with configurations for names, sizes, and units of coordinate (independent) variables, the title of the dataset, and data source information (such as a data provider or a model name). The supplementary information supplied is designed to meet the minimum requirements for post-processing.

One of the characteristics of the gtool5 library is that all variables in the output are referenced by their names, specified in initializer routines such as HistoryCreate, unlike its lower layer netCDF library where variables are referred to by the numbers of handles returned from NF_OPEN. All netCDF handles for files, dimensions, and variables are intentionally hidden and managed in the library. Otherwise, the readability of the simulation code would be significantly compromised. By hiding handles used in the netCDF library, we can easily implement data input/output operations to model programs.

3.2.3 Program example

Figure 1 shows an example of a typical program source code of a numerical model using the gtool5 library. Lines 45 and 46 describe the model equation (finite difference form of the diffusion equation), and calculate grid values of the variable "temp" at each time step. Lines 37–39 in Fig. 1 configure the metadata: the arguments "longname" and "units"

Table 1. Data input/output interface for small models in which multiple variables are output into a single file. Only major subroutines and some optional arguments (represented by parentheses) are shown. The item with an italic font does not exist in the interface shown in Table 2.

Operation	Subroutine name	Major arguments
Initialization	`HistoryCreate`	*File name*, title, source (the method of production of the data), institution (specifies where the data was produced), names of dimensions, sizes of dimensions, descriptive names of dimensions, units of dimensions, [interval of output].
Variable definitions	`HistoryAddVariable`	Name of variable, dimensions on which variable depends, descriptive name of variable, units of variable.
Attribute settings	`HistoryAddAttr`	Name of variable, name of attribute, value of attribute.
Output of data	`HistoryPut`	Name of variable, names of dimensions, numerical data.
Finalization	`HistoryClose`	(No Argument)

Table 2. Data input/output interface for massive models, where output data are separately stored into multiple files so that each file contains an output variable. Only major subroutines are shown. The items with an italic font do not exist in the interface shown in Table 1.

Operation	Subroutine name	Major arguments
Initialization	`HistoryAutoCreate`	Title, source (the method of production of the data), institution (specifies where the data was produced), names of dimensions, sizes of dimensions, descriptive names of dimensions, units of dimensions.
Output of dimension variable	*HistoryAutoPutAxis*	Name of dimension, data
Variable definitions	`HistoryAutoAddVariable`	Name of variable, dimensions on which variable depends, descriptive name of variable, units of variable.
Attribute settings	`HistoryAutoAddAttr`	Name of variable, name of attribute, value of attribute.
Output of data	`HistoryAutoPut`	*Time*, name of variable, data.
Finalization	`HistoryAutoClose`	(No Argument)

of subroutines HistoryAddVariable specify the variable name and unit of the variable, respectively. These metadata are contained in the output data as attributes, and are used in the post-processing/visualization tools described in Sect. 4. In the program shown in Fig. 1, there are only six subroutine calls for data output, and an additional variable, such as a handle for netCDF API, is not required. In contrast, when the same operations are implemented with standard netCDF API, the number of subroutine calls for data output becomes two or three times that of Fig. 1, and several handle variables are required.

3.2.4 Data input/output interface for massive models

An alternative interface is shown in Table 2. This is intended for massive simulation code with larger size and/or number of output variables, where it is unrealistic to combine all output into a single file. With this interface, output data are stored separately in multiple files so that each file contains an output variable. Also, output data can be recorded at unequal time intervals.

Major subroutines of this interface are HistoryAutoCreate, HistoryAutoPutAxis, HistoryAutoAddVariable,

```
1    != Sample program for gtool_history/gtool5
2    !
3    ! Solving diffusion equation
4    ! \[
5    !     du/dt = \kappa d^2 u/dx^2
6    ! \]
7    ! for giving values of $u$ at $x=[0,1]$.
8    !
9    program diffusion
10
11     use gtool_history                              ! specification of necessary module
12
13     integer, parameter     :: nx=30                    ! grid number
14     integer, parameter     :: nt=200                   ! number of time steps
15     integer, parameter     :: ndisp=10                 ! output interval
16     real(8), parameter     :: dx=1.0/(nx-1)            ! grid spacing
17     real(8), parameter     :: dt=0.0005                ! time step
18     real(8), dimension(nx) :: x=(/(dx*(i-1),i=1,nx)/) ! coordinate variable
19     real(8), dimension(nx) :: temp                     ! temperature
20     real(8), parameter     :: kappa=1.0                ! diffusion coefficient of heat
21
22     tinit = 0.0                                        ! initial time
23
24     temp = exp(-((x-0.5)/0.1)**2)                      ! initial value
25
26     call HistoryCreate( &                          ! Initialization
27       & file='diffusion_1.nc', title='Diffusion equation', &
28       & source='Sample program of gtool_history/gtool5',   &
29       & institution='GFD_Dennou Club davis project',       &
30       & dims=(/'x','t'/), dimsizes=(/nx,0/),               &
31       & longnames=(/'X-coordinate','time        '/),       &
32       & units=(/'m','s'/),                                 &
33       & origin=real(tinit), interval=real(ndisp*dt) )
34
35     call HistoryPut('x',x)                         ! output of coordinate variable
36
37     call HistoryAddVariable( &                     ! definition of dependent variable
38       & varname='temp', dims=(/'x','t'/), &
39       & longname='temperature', units='K', xtype='double')
40
41     call HistoryPut('temp',temp)                   ! output of dependent variable
42
43     do it=1,nt
44
45       temp(2:nx-1) = temp(2:nx-1) &                 ! TIME INTEGRATION
46         & + kappa*(temp(3:nx)-2*temp(2:nx-1)+temp(1:nx-2))/dx**2*dt
47
48       if ( mod(it,ndisp) == 0 ) then
49         call HistoryPut('temp',temp)               ! output of dependent variable
50       endif
51     enddo
52
53     call HistoryClose
54     stop
55   end program diffusion
```

Fig. 1. An example of program source code of a numerical model using the gtool5 library. Numbers on the left-hand side indicate line numbers.

HistoryAutoPut, and HistoryAutoClose. As shown in Tables 1 and 2, the arguments of the two interfaces are almost identical, which contributes to the readability of program source code and reduces the maintenance cost of the library. The filenames of output data are automatically generated with this interface, whereas only a single filename is explicitly given in the simpler interface. Coordinate variables, called dimension or axis, are specified in the distinct subroutine, HistoryAutoPutAxis, while in the former interface, HistoryPut can deal with both data (dependent) and coordinate (independent) variables. This asymmetry is chosen to simplify the implementation with parallel processing in mind.

3.2.5 Data input with gtool5 library

The gtool5 library also hides netCDF handles when reading data, and users can specify a variable by its name. For a case of input of a variable specified by "U" contained in netCDF file inputfile.nc, the subroutine HistoryGet should be called as follows:

```
real(8) :: xyz_U(10,20,30)
...
call HistoryGet('inputfile.nc', 'U', xyz_U).
```

HistoryGet does not read metadata. The subroutine HistoryGetAttr is used for input of metadata, if necessary. By

specifying optional arguments, a portion of variable data can be extracted as follows:

```
real(8) :: y_V(20)
...
call HistoryGet('inputfile.nc', 'V', y_V,
range='x=180.0,y=-10.0:10.0,t=3.5').
```

The above example shows a case where a part of a variable specified by "V" is extracted in the range of $x = 180, -10 \leq y \leq 10, t = 3.5$ and is stored in an numerical array. The extracted range is specified with an optional argument in the form of "range = ⋯".

4 Software related to the gtool5 library

Based on the gtool5 library, we are now developing a series of hierarchical numerical models for research on the surface environment of the Earth and other planets (GFD Dennou Club dcmodel project). Our aims are to validate the results obtained by high-end models such as GCMs and to understand the circulation structures of the atmosphere and ocean by performing numerical experiments by using associated hierarchical models. Our high-end models are a three-dimensional general circulation model of the atmosphere, "DCPAM", and a three-dimensional non-hydrostatic convection model, "deepconv" . At the same time, we have been developing a series of simplified models illustrating standard problems of geophysical fluid dynamics, "SPMODEL" (Takehiro et al., 2006). The input/output parts of these models with different complexity are arranged in a consolidated form without reducing readability, thanks to the interface routines of gtool5.

We have been constructing a series of software based on the script language "ruby" for post-processing/visualization of data produced by the gtool5 library (GFD Dennou Club Davis project). As for the main end-user tools, there is a class library for analysis and visualization of multi-dimensional gridded physical quantities, "GPhys", and web-based data and knowledge server software, "Gfdnavi" (Horinouchi et al., 2010; Nishizawa et al., 2010) (Fig. 2). With Gfdnavi, we are aiming to construct a system for archiving and sharing of documentation, image data, and supplementary information for visualization. Using these tools, we can analyze output data of models of different sizes and complexities.

5 Conclusions

In this paper, we present a Fortran90 input/output library, "gtool5", for hierarchical models ranging from high-end models to simplified models. The input/output interfaces of the library can be implemented in model programs in a consolidated form independent of the complexity of the models. The interfaces are sufficiently simplified in spite of the concurrent output of supplementary information, which contributes to the readability of the program code.

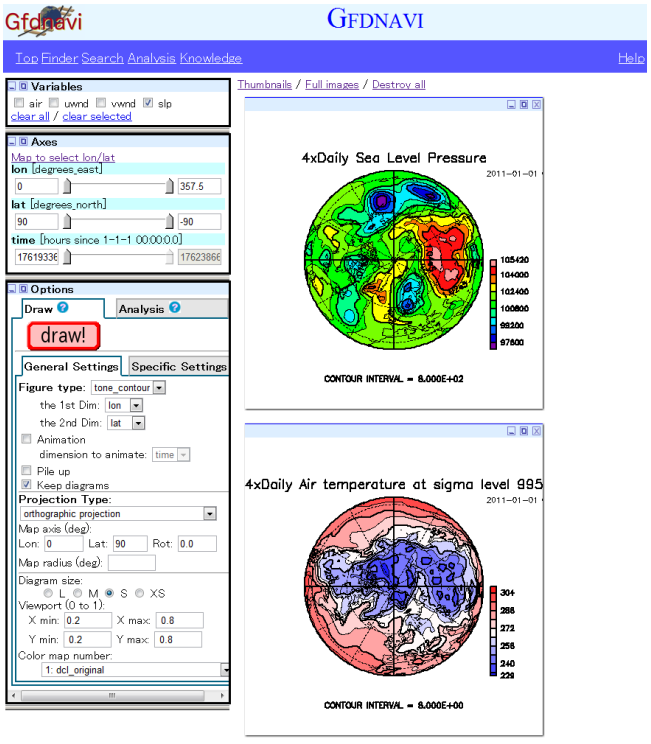

Fig. 2. A screenshot of web-based data and knowledge server software, Gfdnavi.

Moreover, use of gtool5 gives us possibility of adding to its capabilities in the future without changing existing interfaces used in current models. Relinking with a new version of the library would automatically provide the new functionality. For example, the present version of the gtool5 library can read compressed files by linking with netCDF-4 library, whereas it cannot output compressed data. However, it is expected to realize compression of model output by using newly implemented netCDF-4 functions, such as "nf90_def_var_deflate" and "nf90_def_var_chunking", instead of the old functions of netCDF-3 without changing the programs of present models. Another example is the parallel support. MPI-1 is originally implemented in the present version of gtool5 library so that, in parallel computations, you can realize separate input/output from each processor or bulk output by collecting data into a single processor even when you use netCDF-3 library without parallel functionality. We are now improving the gtool5 library to use parallel API of netCDF-4. Since the API of gtool5 would not be changed, present models can achieve better parallel functionality only by relinking the new version of the libraries.

Performance of data input/output using the present version of gtool5 is not so good. It takes about 50 % longer computational time for output of data with 8 GB size ($100\,000 \times 1000$ two-dimensional array with double precision) using gtool5 API than that with raw netCDF API. Moreover, data input takes about twice the time. Nevertheless, the computational

cost of data IO is sufficiently small in actual numerical computations of our models with gtool5 API, and benefits of the library, such as simplification of programming and improvement of readability, are provided satisfactory. However, there is potential possibility that data IO with gtool5 library may become a bottleneck in model calculations on a massive scale. We are now starting to improve performance of data IO of gtool5 library.

The gtool5 library proposed in this paper is not novel from the viewpoint of information technology; the data format mainly owes its design to the netCDF library used in the lower level of gtool5 library. The main achievement of the gtool5 library is to provide convenient interfaces for researchers working on numerical models for the Earth and planetary sciences. As is emphasized in Sect. 2, a particular requirement for the software utilized in the research of these fields is that the source code must be readable even by users. Hence, one of the issues for numerical research in these fields is the growing difficulty in tracing model source code because of the enlargement of the system. Expanded source code causes an "information explosion" for model developers and users. Construction of the gtool5 library and development of hierarchical models using gtool5 represent one approach to overcoming such difficulties.

Acknowledgements. Some of the authors were supported by a Grant-in-Aid for Scientific Research on Priority Areas "IT Infrastructures for Info-plosion" (subject number A01-14).

Edited by: P. Jöckel

References

Eaton, B., Gregory, J., Drach, B., Taylor, K., and Hankin, S.: NetCDF Climate and Forecast (CF) Metadata Conventions, Version 1.0., available at: http://cf-pcmdi.llnl.gov/documents/cf-conventions/1.0/cf-conventions.html (last access: 2 April 2012), 2003.

GFD Dennou Club Davis project: available at: http://www.gfd-dennou.org/library/davis/, last access: 2 April 2012.

GFD Dennou Club dcmodel project, available at: http://www.gfd-dennou.org/library/dcmodel/, last access: 2 April 2012.

Gfdnavi: available at: http://www.gfd-dennou.org/library/davis/gfdnavi/, last access: 2 April 2012.

GPhys: multi-purpose class to handle Gridded Physical quantities, available at: http://www.gfd-dennou.org/library/ruby/products/gphys/, last access: 2 April 2012.

gtool4 NetCDF Conventions: available at: http://www.gfd-dennou.org/library/gtool/gt4ncconv.htm.en, last access: 2 April 2012.

Held, I. M.: The gap between simulation and uderstandign in climate modeling, B. Am. Meteorol. Soc., 86, 1609–1614, 2005.

Horinouchi, T., Nishizawa, S., Watanabe, C., Tomobayashi, A., Otsuka, S., Koshiro, T., Hayashi, Y.-Y., and GFD Dennou Club: Gfdnavi, Web-based Data and Knowledge Server Software for Geophysical Fluid Sciences, Part I: rationales, stand-alone features, and supporting knowledge documentation linked to data, in: Database Systems for Advanced Apllications, Springer-Verlag, LNCS, 6193, 93–104, 2010.

Morikawa, Y., Takahashi, O. Y., Sasaki, Y., Nishizawa, S., Otobe, N., Ishiwatari, M., Odaka, M., Takehiro, S., Horinouchi, T., Hayashi, Y.-Y., Toyoda, E., and Gtool Devlopment Group: The gtool5 library, available at: http://www.gfd-dennou.org/library/gtool/ (last access: 2 April 2012), 2010.

NetCDF: available at: http://www.unidata.ucar.edu/software/netcdf/, last access: 2 April 2012.

Nishizawa, S., Horinouchi, T., Watanabe, C., Isamoto, Y., Tomobayashi, A., Otsuka, S., and GFD Dennou Club: Gfdnavi, web-based data and knowledge server software for geophysical fluid sciences, Part II: RESTful web services and object-oriented programming interface, in: Database Systems for Advanced Apllications, Springer-Verlag, LNCS, 6193, 105–116, 2010.

Ruby: available at: http://www.ruby-lang.org/en/, last access: 2 April 2012.

Sugiyama, K., Odaka, M., Yamashita, T., Nakajima, K., Hayashi, Y.-Y., and deepconv Development Group: Non-hydrostatic model deepconv, available at: http://www.gfd-dennou.org/library/deepconv/ (last access: 2 April 2012), 2010.

Takahashi, Y. O., Ishiwatari, M., Noda, S., Odaka, M., Horinouchi, T., Morikawa, Y., Hayashi, Y.-Y., and DCPAM Devlopment Group: DCPAM: planetary atmosphere model, available at: http://www.gfd-dennou.org/library/dcpam/ (last access: 2 April 2012), 2010.

Takehiro, S., Odaka, M., Ishioka, K., Ishiwatari, M., Hayashi, Y.-Y., and SPMODEL Development Group: SPMODEL: A series of hierarchical spectral models for geophyiscal fluid dynamics, Nagare Multimedia, available at: http://www2.nagare.or.jp/mm/2006/spmodel/index.htm (last access: 2 April 2012), 2006.