修士論文

惑星大気大循環モデル DCPAM の開発：データ I/O ライブラリの改良および力学コアの設計と実装実験

Development of "DCPAM", a general circulation model for planetary atmospheres: improvement of data I/O library, and design and implementation tests of dynamical core

森川 靖大

Yasuhiro Morikawa

北海道大学大学院理学研究科 地球惑星科学専攻
地球流体力学研究室

Division of Earth and Planetary Sciences,
Graduate School of Science, Hokkaido University,
Geophysical Fluid Dynamics Laboratory

2005/01/28

要旨

プログラム構造の可変性とソースコードの可読性の向上を模索するべく, GCM (General Circulation Model, 大気大循環モデル) を新たに設計し, そのプログラム実装と試験計算を行った. (以後, このモデルを DCPAM, Dennou-Club Planetary Atmosphere Model と呼ぶことにする). 同時に, netCDF による自己記述的多次元格子点データの入出力が可能となるように I/O ライブラリの開発整備をおこなった. 可読性の向上によりプログラムの改良や変更のコストの削減が期待され, 可変性の向上により新たなプログラムの追加や既に組み込まれているプログラムの分離を容易にすることが期待できる. また計算結果を自己記述的多次元格子点データに出力することで, 解析のコストを削減すること目指した.

DCPAM の設計は, AGCM5 (SWAMP Project, 1998) を参照しつつ Fortran90 の機能を積極的に活用することを念頭において行った. 可変性向上のために, モジュール・構造型・総称手続きなどを用いたオブジェクト指向的設計によってモデルの内部構造の階層化と I/O ライブラリの分離を試みた. 可読性を向上するために, SPMODEL ライブラリ (Takehiro et al., 2004) において用いられていた関数および変数命名規則を拡張した命名規則を考案した. さらに, 気象庁標準コーディングルール (Muroi, et al., 2002) と The FMS Manual (Balaji, 2002) における Fortran90 コーディングルールを参考にし, 変数の宣言方法のルールやモジュールの初期化・終了処理の統一方法のルールなどを取り入れた. プログラムの解説ドキュメントの生成を容易に行う試みとして, Fortran90 ソースコードに RD 形式 (Ruby Documentation Project, 2005) のコメント文を埋め込み, ソースコードから HTML 形式や TEX 形式を自動生成できるようにした. この仕組みにより, プログラムとその解説ドキュメントとの乖離を防ぎ, ドキュメント管理コストを削減することができる.

DCPAM における I/O ライブラリの分離とモジュール化のために, gtool4 ツール・ライブラリ (Toyoda et al., 2002) を改良し, gt4f90io として整備した. gtool4 では入出力データの形式として COARDS 上位互換の gtool4 netCDF 規約 (Toyoda et al., 2000) を定めており, これにより, データに関する情報の取得や解析のコストの削減が期待できる. これを改良し, Fortran90 のためのデータ I/O に特化した gt4f90io は DCPAM だけでなく, 様々な数値モデルに用いることが可能である.

DCPAM の力学部分の動作試験として, Held and Suarez (1994) の GCM 力学コアのベンチマークテストを行った. 1200 日の積分を行った結果, 傾圧不安定やハドレー循環は表現できることが確認された. ただし, 亜熱帯ジェットの位置に関しては Held and Suarez (1994) の結果とは差異が見られた. この原因は波の活動度の違いやモデルの空間解像度の違いであると考えられる.

目 次

図 目 次

1　はじめに

GCM (General Circulation Model; 大気大循環モデル) とは, 全球規模の大気循環のシミュレーションを行うための数値モデルであり, 天気予報などの気象予測や地球の温暖化等の気候変動予測などに用いられている. GCM は大気の流れを駆動する 3 次元の力学 (「力学過程」と呼ばれる) や, より細かいスケールでの乱流拡散や積雲対流 (「物理過程」と呼ばれる) など様々な要素を含むため, 複雑なモデルとなる. また, 予報精度を高めるため, 個々の GCM に特有なチューニングが必要となる. こういった事情から, 大抵の GCM は解読の非常に難しいモデルとなり, チューニングを行った本人以外のユーザが GCM を自分好みに改変して利用するということは大変困難である. 結果として大半のユーザはその中身でどのようなコーディングがなされているのか知らないまま GCM を動かすことが多い. またそのような理由によって, GCM を成す個々の力学過程や物理過程を分離して再構成するなどといったことも容易ではない. 例えば, GCM でコーディングされている元の 3 次元の方程式系は, 数式でならば簡単に 2 次元や 1 次元に書き換えることが可能であり, その変形を追うことで次元の異なる力学に対する考察を深めることができた. しかし, GCM のコードは数式に比べてあまりに複雑化しすぎ, もはや 2 次元や 1 次元のモデルとはまったくの別物として隔離されてしまっている. 従って, 現状の GCM を用いて得られた結果を力学的に解析するため, GCM の系を順次簡略化して数値実験を行い, その結果を比較するなどしようとするのは非常に困難である. また物理過程は, 予報する対象に応じて経験則を盛り込んだチューニングが行われるため, 気軽に交換することが難しくなっている. 結果的に, GCM は予報のための道具として発達できた代償として, 数式のような物理を理解するための道具としては非常に使いづらいものとなってしまった.

モデルを作るにあたり, 力学過程の簡略化や物理過程の交換を容易にするために必要なことは, ソースコードの可読性の向上と, 階層化プログラム構造による可変性の向上であろうと考えられる. ソースコードの可読性の向上は FORTRAN77 で作成されたこれまでの GCM においても, 変数の命名規則の制定やフォーマットの統一など様々な試みがなされてきた. それは, ソースが読みやすいことがそのまま, メンテナンスコストを下げ, 改良や変更を容易にすることにつながっていたからであった. しかし, FORTRAN77 には変数名が 6 文字までという厳しい制限があったため, 変数の命名規則の制定はその有用性をかなり減じられてしまった. 一方, 可変性はどうであったろうか. ソースコードの可読性の向上がプログラム内のコード一行一行の単位での話だとすると, 階層化プログラム構造による可変性の向上はプログラム単位での構想である. できる限り個々のサブルーチンや関数を独立に動作できるようにし, 交換や分離を容易にしておくことで, 力学過程の簡略化や物理過程の交換を容易にしようというものである. その際, 1 つのサブルーチンや関数

にどれだけの役割を与えるかということも重要な問題である. この階層化プログラミング構造も, これまでの GCM で考慮されなかったわけではなく, FORTRAN77 という言語において, それなりに個々の役割に応じてプログラムの分離が行われた. しかしやはり, GCM のような巨大プログラムになってくると, その独立性を維持するのは難しく, 一部のサブルーチンや関数を分離や交換するのはそれなりの労力を要してしまっていた. 原因の一つとして, FORTRAN77 の COMMON 宣言文や INCLUDE 文が使いづらいということもあった.

そのような中, Fortran の現行規格が FORTRAN77 から Fortran90, そして Fortran95 へ移行し, それらの新規格で記述された言語を処理できるコンパイラも増えてきた. だが, 現在多くの GCM は未だ FORTRAN77 (または一部 Fortran90 を含むものの, 実際にはその機能をあまり活用しない FORTRAN77 的なコーディング) で記述されており, Fortran90 とその新機能を GCM に応用し, 力学過程の簡略化や物理過程の交換を容易にすることが可能かどうか, その有効性を実証する試みが広く深く行われているとは言いがたい状況である. そこで本研究では, Fortran90 とその新機能を駆使することで, 力学過程の簡略化や物理過程の交換が容易に可能であるような GCM を新たに設計, 実装する. 地球大気だけでなく, 他の惑星大気の表現にも使えることを目標にして, この GCM を DCPAM (Dennou-Club Planetary Atmosphere Model) と名付ける.

可読性を向上させることが可能な Fortran90 の新機能として, 変数の文字数の制限が緩和された事や, 配列を簡潔に表記できるようになったことなどが挙げられる. DCPAM での狙いは, これまでの GCM で用いられてきた可読性向上の試みを継承しつつ, それらを Fortran90 によって改良していくことである. 可変性を向上させることが可能な機能としては, モジュール機能が挙げられる. この機能により, プログラムの階層化をより明確に行えるようになった. public や private 文によってそのモジュール内の公開要素を明示的に指定できること, use 文と only 句によってアクセスする変数を限定できることなどもその利便性を強化した. また構造データ型を用いることで, 1 つの変数にデータとそのデータに関する情報を格納することも可能となった. DCPAM では, これらの Fortran90 の新機能によって GCM をどこまで階層化できるか実証しようと試みる.

第一歩として, AGCM5 (SWAMP Project, 1998) のバージョン 5.3 を参照し, このモデルと同様の方程式系および離散化を用いた GCM を Fortran90 で作成する. このモデルを参照した理由は主に, このモデルが大学等での研究・教育用に利用できる形に整理がなされており, 詳細なマニュアルが付属していることであるが, 他にも上記で述べたような FORTRAN77 にして可読性や可変性を向上させるような試みが多くなされたモデルで, Fortran90 でモデルを記述する際にも参考となる部分が多々あるからである.

なお, モデルによる計算と, その結果の理解との間に大きな隔たりが存在することは Held (2004) によっても指摘されており, GFDL [†1] では Fortran90 の機能を活かした力学的考察のための階層的モデル FMS (Flexible Modeling System; GFDL, 2005) の開発が行われている. このモデルは柔軟にモデルの組み換えが可能な階層的プログラム構造を目指すと共に, 並列化, 高速化も重視している. また, モデルそのものをまるごと提供するのではなく, データ I/O や並列化, 機種の違いを吸収する下層のプログラム (Infrastructure と呼ばれる) と, 大気と海洋のモデルを連結するための上層のプログラム (Superstructure と呼ばれる) を提供している. FMS を使って数値モデルを開発するユーザは, 自分の書いたプログラム (User code と呼ばれる) をその間に挟むことにより, 機種の違いやデータ I/O , および並列化を Infrastructure に委ねることができ, また大気と海洋など, 他のモデルとの連結に関しては Superstructure に委ねることが可能となる.

FMS と DCPAM とでは, 階層的プログラム構造による可変性の向上という点に関してはかなり近い発想に基づいている. しかし, DCPAM ではモデルのコードそのものの可読性も重視している. 言わば, FMS での User code 自体をどのように記述するのかという部分を重視しているのである. FMS では大気モデルや海洋モデルの結合に関するインターフェースは提供するが, User code をどのように記述するかということについては提示していない. また, 階層的プログラム構造による可変性の向上に関しても, FMS は大気や海洋のモデルをそれぞれ 1 つのプログラム単位と考えているのに対し, DCPAM では力学過程内の一つ一つの演算を 1 つのプログラム単位として見ているという点では大きく異なっている. よって, FMS は, 系の力学過程の簡素化などを効率的に行うことなどは, 少なくとも現状では想定していないように思われる. よって, DCPAM では, FMS とはまた異なった, 可読性や可変性を向上させるための方法を, 実際にコードを作成しながら提案することを目指す.

モデルの内部構造のみならず, モデルが扱うデータの形式も重要である. まず, データが汎用的に利用できることが重要である. 例えば, 上述したように系の簡素化などを行ってデータを出力し, それを複雑なモデルと比較するような場合, データ形式は 3 次元に固定されるものではなく, 次元の違いを柔軟に吸収するものの方が良い. しかし, 利用する状況が限定される場合には逆に汎用性を狭めた方が便利な場合もある. 例えば, 常に 3 次元のデータのみを扱うのであれば, データ形式は 3 次元に固定されているものを使った方が便利であろう. つまり, 適度な汎用性を持つデータ形式であることが重要である. 次に, そこにあるデータが, 誰がどこでいつどのようにして作成した何のデータなのか, というデータに関する情報 (このような付加情報のことをメタデータと呼ぶ) を簡単に知るための手段が用意され

[†1] Geophysical Fluid Dynamics Laboratory (`http://www.gfdl.noaa.gov`). NOAA (National Oceanic and Atmospheric Administration. `http://www.noaa.gov`) のミッションの 1 つ.

ているかという点が重要である. 計算機やネットワークの急速な進歩により大量のデータがやり取りされる中, 研究者個々人が扱うデータの数もそれに比例して増えている. その結果, データの処理だけでも大変な労力が必要になっており, メタデータの取得といった作業にはできる限り労力を裂きたくないというのが現状である. その労力をできるだけ減らすために試みられている方法の 1 つにメタデータをデータそのものに付加するというものがある (このように, メタデータをデータ自身に含むものを自己記述的データ形式と呼ぶ). この自己記述的データ形式を用い, 必要なメタデータを付加しておくことで, メタデータを得るための労力を格段に減らすことが可能になる. 3 つ目に重要なのはそのデータの可搬性である. データが機種依存するものである場合, 例えばスーパーコンピューターで計算した結果を手元のパソコンで解析・可視化しようとする際に何らかの変換処理が必要となってしまう. 上述したように昨今は個々人が扱うデータの数も増えており, そのような労力はできるだけ低減したい. そのためには, データは機種に依存しない可搬性の高いものが望ましい.

先にあげた AGCM5 では, これら 3 つのうち, 汎用性と自己記述性に関しては考慮され, データ形式として GTOOL3 形式を策定し, そのデータ形式に対応した解析・可視化ツールおよび数値モデルの I/O ライブラリとして GTOOL3 (Numaguti, et al. 1989) を開発し, 利用していた. GTOOL3 形式は, FORTRAN の書式無しシーケンシャルデータのヘッダに付加情報を格納するという方法でそれなりの自己記述性を実現した. (ただし座標軸情報に関しては別のファイルが必要であった). このデータ形式は基本的に 3 次元格子点データを想定していたが, 1, 2 次元のデータも含むことができ, それなりに汎用性が高かったと言える. (ただし, 3 次元よりも多い次元のデータは扱えなかった).

しかし, DCPAM では GTOOL3 形式よりも, さらに汎用的, 自己記述的で, 可搬性に関しても考慮されたデータ形式を必要としている. 1 つ目に, DCPAM はその力学過程の簡素化や複雑化などを容易にし, より柔軟に利用できる GCM を目指している. そのような目的で DCPAM を利用することを考えると, データの次元は 3 次元以上の多次元データであることが望ましい. そのようなデータであれば, 例えば, パラメタ実験の際に, 4 次元目としてパラメタを指定することなどが可能になる. 2 つ目に, 数多くのデータのやり取りを想定すると, データは完全に自己記述的であって欲しい. GTOOL3 形式のデータももそれなりに自己記述性は高かった. しかし, 座標軸に関するデータに関しては別に保持, 管理が必要であった. これは, 扱うデータの数が増えることでより不便性を増してしまった. 3 つ目に, データの可搬性を向上させたいという希望がある. GTOOL3 形式のデータは書式無しデータであったため, 機種依存しており, 可搬性に関しては確保できなかった. DCPAM では機種を気にせずにデータを持ち運べるようなデータ形式が望ましい.

これら希望を満たすデータ形式として DCPAM が採用したのが gtool4 netCDF 規約 (Toyoda et al., 2000) である. I/O ライブラリとしては, 元々 gtool4 netCDF 規約に基づくデータの解析・可視化ツールおよび I/O ライブラリとして開発されていた gtool4 ツール・ライブラリ (Toyoda et al., 2002) が利用できる. ただし, gtool4 ツール・ライブラリ は, DCPAM が I/O ライブラリとして望む機能には十分でないため, gtool4 ツール・ライブラリの I/O ライブラリ部分を取り出し, その部分の I/O ライブラリとしての機能を充実させ, gt4f90io として再パッケージする.

本論文の構成は以下のとおりである. 2 節では, gtool4 netCDF 規約に関する詳細と, gt4f90io について紹介する. 3 節では, 力学過程を含めたモデルのデザイン, および可読性や可変性を向上させる試みを紹介する. 4 節では, GCM の力学過程の性能試験のためのテスト計算として, Held and Suarez (1994) の力学コアベンチマークテストを行う. 5 節では, 結論を述べる.

2　データ I/O の改良

DCPAM の入出力データの形式として gtool4 netCDF 規約 (Toyoda et al., 2000) を採用する. なぜなら, gtool4 netCDF 規約に基づくデータ (以下, gtool4 データと呼ぶ) は, DCPAM で望んでいる多次元自己記述的データであり, 機種依存しないために可搬性に優れているためである. これにより, データに関する情報の取得や解析のコストの削減が期待できる.

gtool4 データをモデルから入出力するための I/O ライブラリとして gtool4 ツール・ライブラリ (Toyoda et al., 2002) が利用できる. しかし, I/O ライブラリとしての機能が不十分なため, このソフトウェアから I/O 部分だけを抜き出し, DCPAM にとって必要な機能を付加して gt4f90io として再パッケージした. gt4f90io では最低限 4 行のサブルーチン call により, データ入出力を実現しており, これがソースコードの可読性の大幅な向上をもたらした.

本節では, 2-1 で gtool4 netCDF 規約 の特徴を解説する. 2-2 では gt4f90io について紹介する.

2-1　gtool4 netCDF 規約

多次元格子点データのための自己記述的表現方法として gtool4 netCDF 規約 (Toyoda et al., 2000) という規約がある.

gtool4 netCDF 規約はそのデータ形式として netCDF (Rew et al, 1997) を採用している. このことによって, 1) 次元の数に制約の無い多次元格子点データであること, 2) 座標情報も含むメタデータをデータ本体に付加できるため, 完全に自己記述的データ形式であること, 3) 機種依存しないため, 異なる機種の計算機から得られた計算結果も同様に扱うことができることが実現された.

gtool4 netCDF 規約は netCDF データに格納するメタデータに関しても規定している. 主に, 地球流体現象の研究で必要とされると想定されるメタデータを格納するよう規定されている. さらに, 可視化情報もメタデータに含むことにより, 数値データの効率的な可視化も考慮している.

この規約は COARDS 規約 (Cooperative Ocean/Atmosphere Research Data Service, 1995) や CSM 規約 (National Center for Atmospheric Research, 1997) との上位互換として利用できるよう策定されたので, COARDS 規約, CSM 規約のデー

タは gtool4 データを解釈するソフトウェアで利用可能であり, 逆に gtool4 データは COARDS 規約や CSM 規約として利用できる.

2-2　データ I/O ライブラリ gt4f90io

gtool4 ツール・ライブラリ (Toyoda et al., 2002) は gtool4 データの解析・可視化ツール, I/O ライブラリ, ならびに Fortran90 で記述された数値モデルに用いられることを想定した文字処理, デバッグ追跡補助, 日付処理, 単位処理などの汎用プログラム群を提供するソフトウェアであった.

DCPAM では gtool4 データを入出力するために, gtool4 データのための I/O ライブラリが必要である. しかし, gtool4 ツール・ライブラリの I/O ライブラリには, ドキュメントが不足していたり, 入力用サブルーチンが未整備だったことなどから, DCPAM でそのまま利用するには多少機能が不足していた. よって, DCPAM での要望に応える I/O ライブラリとして, gtool4 ツール・ライブラリ の I/O 部分を抜き出し, I/O ライブラリとして特化したソフトウェアとして gt4f90io (gtool4 規約に基づく Fortran90 netCDF I/O ライブラリ, Fortran90 netCDF I/O library with gtool4 conventions) を整備した. なおこの際, DCPAM にとっても有用であったことから, 文字処理やデバッグ追跡補助, 日付処理や単位処理などの汎用プログラム群も一緒に gt4f90io に移植し, ドキュメントの整備などを行った.

2-2-1　階層的内部構造

gt4f90io の内部構造は gtool4 ツール・ライブラリを引き継ぎ, Fortran90 の「モジュール」機能を活用することで, その役割に応じて明確に階層化されている. (図 1 参照). I/O 部分としては大別して 3 階層を成し, netCDF データとアクセスする下層ライブラリ (`an_generic` モジュール), 多次元数値データを扱う本体となる中層ライブラリ (`gtdata_generic` モジュール), そしてユーザインターフェースとなる上層ライブラリ (`gt4_history` モジュール) から構成される. そして, 汎用プログラム用モジュール群が用意されている.

上記のように内部構造をその役割に応じて明確に階層化することにより, 大幅な情報隠蔽が可能となった. また, Fortran90 ではあるがオブジェクト指向的なコーディング書法を試みており, クラスを構造データ型に, メソッドをサブルーチンに, 多態性を総称手続きに対応させた実装を行っている.

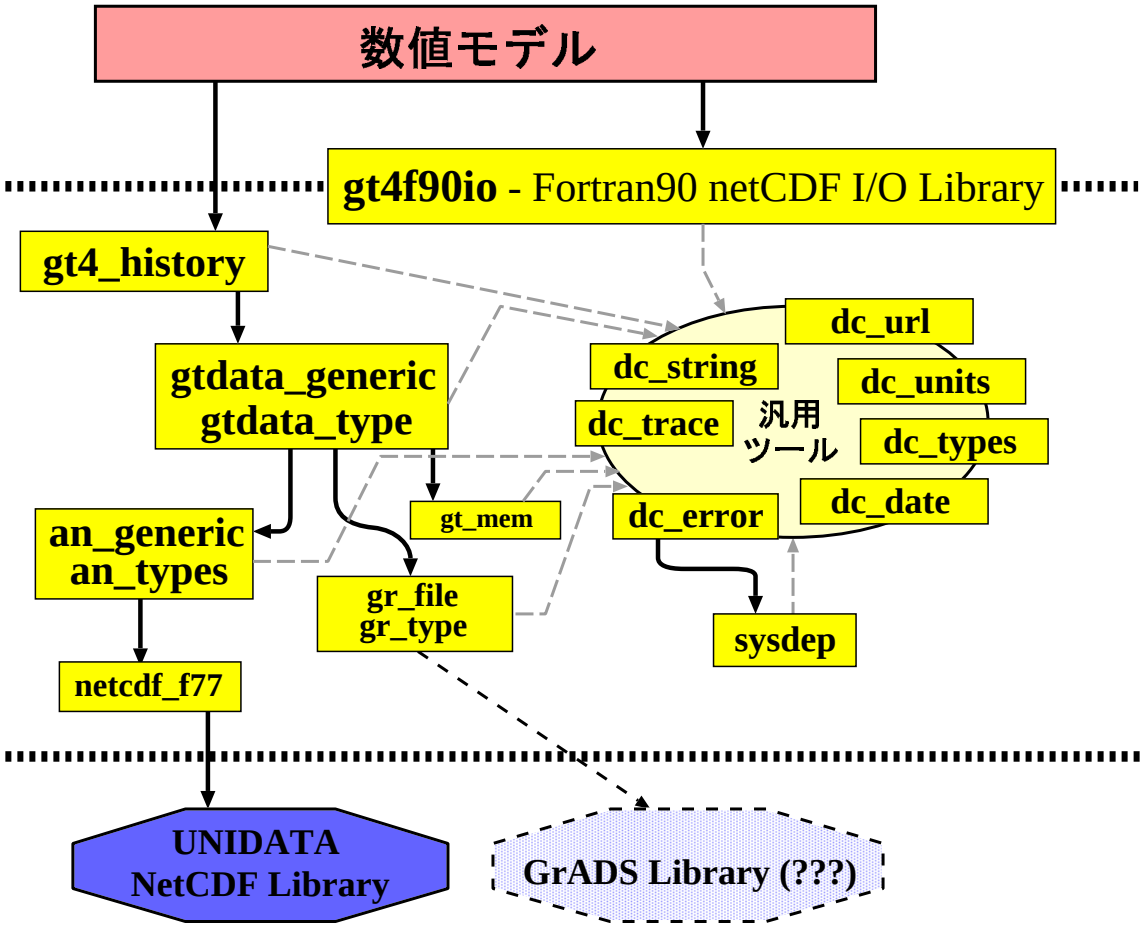


図 1: gt4f90io の内部構造概観図. 矢印はその依存関係を表している.

2-2-2 以降では, 各モジュールの概要を説明する.

2-2-2　an_generic

an_generic (**a**bstract **n**etCDF) モジュールは netCDF ファイルの作成, netCDF 変数の作成や削除, データ本体の入出力, 属性情報の問い合わせや付加や削除を行う. このモジュールの特徴は, 数値データも属性も「変数」に帰属するものとして扱うところにある. これは, netCDF というデータ形式が, 「変数」に数値データや属性を持たせるようにしているためである. その結果, an_generic モジュール使えば, 全ての動作を「変数」に対して行えばよくなるため, netCDF ファイルの操作が非常に容易になる. このモジュールは gtool4 ツール・ライブラリからその

まま引き継いだものである

2-2-3　gtdata_generic

gtdata_generic は元々のデータ形式に依らない, 多次元数値データの操作を行うモジュールである. 巨大な多次元データの全てを一度にメモリに格納することはできないため, 多次元データの一部の切り出し, および切り出した範囲の指定, 移動を行うための Slice, Slice_Next を用意し, 操作する部分のデータだけをメモリに格納するようにしている. このモジュールも gtool4 ツール・ライブラリから引き継いだものである.

2-2-4　gt4_history

gt4_history モジュールは, 数値モデルを開発するユーザが gtool4 データを入出力する際のインターフェースとなるサブルーチン群を提供する. an_generic によって netCDF データへの入出力が簡素化され, gtdata_generic によって多次元データの操作が行われるので, ユーザは gt4_history モジュールで用意される簡潔なサブルーチンを呼ぶだけで, 簡単に gtool4 データを読み書きできるようになる. 最低限必要となるのは以下の 5 つのサブルーチンのみである. 実際の使用例は図 2 のようになる. なお, このモジュールの基本部分は gtool4 ツール・ライブラリで開発されたもので, データ入力用のサブルーチンの開発やドキュメントの整備は gt4f90io として行った.

- HistoryCreate

  出力ファイル名, データのタイトル, 座標軸情報等の設定を行う.

- HistoryAddVariable

  変数情報を設定する.

- HistoryPut

  数値データの出力を行う. Fortran90 の総称手続きを用いることで, 0 次元の配列から多次元の配列まで同じサブルーチンを利用することが可能である.

- HistoryClose

  HistoryCreate によって作成したファイルを閉じる.

- HistoryGet

  gtool4 データを入力する. このサブルーチンも, Fortran90 の総称手続きを用いることで, 0 次元の配列から多次元の配列まで同じサブルーチンを利用することが可能である.

```
program sample
  use gt4_history                                        ! モジュールの使用を宣言

  [型宣言] ......

  call HistoryCreate( &                                  ! ヒストリー作成
       file='sample.nc', title='gt4_history',    &       !   ・ファイル名、タイトル
       ..., dims=(/'x','t'/), dimsizes=(/30,0/), &       !   ・次元変数、次元サイズ
       .......)
  call HistoryAddVariable( &                             ! 変数定義
       varname='temp', dims=('x','t'), .... )            !   ・変数名、依存次元、..

  [初期値入力]

  call HistoryGet(file='init.nc', varname='temp', array=temp)


  [時間積分ループ]
         :
        call HistoryPut(varname= 'temp', value=temp) ! 変数の出力
         :
  [時間積分ループ 終わり]

  call HistoryClose                                      ! 終了の処理
  stop
end program sample
```

図 2: gt4f90io の gt4_history モジュールを用いたサンプルソースコード.

2-2-5 dc_string

dc_string モジュールは文字処理のためのサブルーチンや関数群を提供する. 具体的には, Fortran では用意されていない大文字小文字の変換やデータの整形, 文字型データとその他のデータ型の変換, 文字列の置換などのためのサブルーチンや関数が用意されている. 主な機能は gtool4 ツール・ライブラリから引き継いだもので, gt4f90io ではさらにいくつかの機能を付加し, さらにドキュメントを作成した.

2-2-6　dc_message

dc_message モジュールはメッセージ出力のためのサブルーチンを提供する. メッセージの種類に応じて, 単なるメッセージ, 警告, エラー (2-2-8 参照) を選択できる. また, どのサブルーチンでメッセージを出力したのかという情報も出力するようになっている. これは gt4f90io において新たに作成したモジュールである.

2-2-7　dc_trace

dc_trace モジュールは, 数値モデルのデバッグ作業の補助となるサブルーチンを提供する. 各サブルーチンの開始, 終了部分でそれぞれ BeginSub EndSub というサブルーチンを呼ぶようにコーディングを行っていくことで, 図 3 のようなメッセージを出力できる (デフォルトでははは標準エラー出力に出力される). プログラムの階層構造を表示するようになっているため, 見やすく, バグの追跡を容易にする. 元々 gtool4 ツール・ライブラリで用意されていたものを, gt4f90io では出力のメッセージフォーマットの改良や多次元データのログへの出力を可能にし, 加えてドキュメントの作成を行った.

2-2-8　dc_error

dc_error はエラー処理のためのモジュールである. エラーとは, 個々のプログラムが期待される動作をすることができないといった事態である. エラーの取り扱いを明確に規定することで, gt4f90io を利用するユーザは, エラーの原因と対処法を把握できるようになる. このモジュールも gtool4 ツール・ライブラリから引き継いだもので, gt4f90io では機能の拡張とドキュメントの整備を行った.

dc_error モジュールでは gt4f90io 内の全てのエラーを一元管理している. そして個々のエラーに対し, エラーコード, コードに対応する文字列 (ニーモニックのようなもの), およびその内容を端的に示すエラーメッセージを用意している. gt4f90io の開発者は, あるプログラムにおいてエラーを発生させる場合, dc_error モジュール内から対応するエラーを探し, そのエラーコードに対応する文字列をハードコードする. gt4f90io を用いて数値モデルを作成するユーザは, エラーが生じた場合, そのエラーメッセージを元に原因を追求することが可能である.

```
     :
#call HistoryPut0
#| call HistoryPutEx : time
#| | call TimeGoAhead : varname=time head=1.
#| | | call lookup_dimension
#| | | | call gtvarinquire : var.mapid=1
#| | | | | call anvarinqurie : var.id=1
#| | | | | end anvarinqurie : ok
#| | | | |-name=time
#| | | | end gtvarinquire
#| | | end lookup_dimension : ord=1
#| | | call gtvarslice : var%mapid=1 dimord=1
#| | | |-[gt_variable 1: ndims=1, map.size=1]
#| | | |-[dim1 dimno=1 ofs=0 step=1 all=0 start=1 count=0 stride=1 url=]
#| | | |-[vartable 1: class=netcdf cid=1 ref=1]
#| | | |-[AN_VARIABLE(file=3, var=1, dim=1)]
#| | | |-map(dimord): originally start=1 count=0 stride=1
#| | | |-start=1 (1 specified)
#| | | |-count=1 (1 specified)
#| | | end gtvarslice
#| | end TimeGoAhead
#| |-anfiledefinemode
#| end HistoryPutEx
#end HistoryPut0
  :
```

図 3: dc_trace によるログの例.

3　モデルの設計

プログラム構造の可変性とソースコードの可読性の高い新しい GCM を設計, 開発する. 実際には, AGCM5 (SWAMP Project, 1998) のバージョン 5.3 を参照し, それらから方程式系や離散化などは引き継ぎつつ, Fortran90 によってより可読性や可変性に優れた新たな GCM を作成する. AGCM5 は球面プリミティブ方程式系に基づく FORTRAN77 で記述された大気大循環モデルで, 物理過程として若干の水過程と放射過程を含む. その特徴を以下に挙げる.

- 物理定数や積分時間や積分間隔, 拡散や放射に関するパラメタや, 出力ファイル, 出力変数など, 様々な値を NAMELIST で設定することが可能である. 従って, それらの値がハードコードされるような GCM に比べ, 数値実験を容易に行うことができる. (解像度に関しては再コンパイルが必要である).
- 変数の命名規則を定めているため, 可読性が高い. ただし, FORTRAN77 の制限により 6 文字までの変数名しか使えない.
- 型宣言やサブルーチンの呼び出しや DO ループなどで, フォーマットを決めたコーディングをしているため, 可読性が高い.
- 数理モデル, 離散化, コード内部の概要, および内部の各種サブルーチンの詳細な解説ドキュメントが揃っている他, 「ごくらく AGCM5」などのチュートリアルも用意されている.

DCPAM では, Fortran90 を用い, AGCM5 を以下のように改良したモデルを作成する.

- NAMELIST によって様々な値を設定できるという設計は引き継ぎ, さらに Fortran90 のポインタや動的割付配列を用いることで解像度なども設定できるようにする.
- Fortran90 では変数名やサブルーチン名の文字の制限が緩和されたので, より分かりやすい命名規則を用い, より可読性を向上させる.
- サブルーチンの呼び出しなどのフォーマットの統一する, ということに関しては AGCM5 の精神を引き継ぎ, DCPAM でもフォーマットの統一を行う. 配列同士の演算など, Fortran90 でより見やすく書ける部分では, さらに可読性の高いフォーマットにする.
- データ形式に gtool4 netCDF データ を用いることで, データの参照効率をより向上させる. また, データ I/O に gt4f90io を用い, I/O 部分の可読性も向上させる.

- SPMODEL ライブラリ (Takehiro et al., 2004) を用いることで, 特にスペクトル法の演算に関する部分の可読性を向上させる.
- 各種モジュールやサブルーチン, 関数に関してドキュメントを作成する.
- Fortran90 のモジュール・構造型・総称手続きを積極的に用い, オブジェクト指向的な設計を目指す. そのような設計により, 個々の演算ルーチンをオブジェクト化し, 物理過程の交換や力学過程の簡素化を容易にする.

なお, Fortran90 のモジュールデザインや, コードのフォーマットに関しては FMS (GFDL, 2005) の The FMS Manual (Balaji, 2002) も参考にした.

本節では, 3-1 で DCPAM の数理モデルの解説を, 3-2 でそのプログラムの構造を解説する. 3-3〜3-6 では可読性, 可変性を向上させるための試みを紹介する. 3-7 では, モデルを解説するドキュメントの作成・管理に関する新たな試みを紹介する.

3-1　数理モデルの設計

数理モデルは, AGCM5 と同様な方程式系, および離散化を用いる. 物理過程は, 放射, 乱流拡散, 積雲対流, 大規模凝結過程, 地表面フラックスといったものを考えている.

3-1-1　支配方程式系

座標系は, 水平方向には緯度 φ, 経度 λ を, 鉛直方向には $\sigma = \dfrac{p}{p_s}$ をとる (p は圧力で, p_s は地表面圧力である).

支配方程式系は以下のように表される.

連続の式

$$\frac{\partial \pi}{\partial t} + \boldsymbol{v}_H \cdot \nabla_\sigma \pi = -\nabla_\sigma \cdot \boldsymbol{v}_H - \frac{\partial \dot{\sigma}}{\partial \sigma} \tag{1}$$

静水圧の式

$$\frac{\partial \Phi}{\partial \sigma} = -\frac{RT_v}{\sigma} \tag{2}$$

運動方程式

$$\frac{\partial \zeta}{\partial t} = \frac{1}{a(1-\mu^2)}\frac{\partial VA}{\partial \lambda} - \frac{1}{a}\frac{\partial UA}{\partial \mu} + \mathcal{D}(\zeta) \tag{3}$$

$$\frac{\partial D}{\partial t} = \frac{1}{a(1-\mu^2)}\frac{\partial UA}{\partial \lambda} + \frac{1}{a}\frac{\partial VA}{\partial \mu} - \nabla_\sigma^2(\Phi + R\overline{T}\pi + KE) + \mathcal{D}(D) \tag{4}$$

熱力学の式

$$\begin{aligned}\frac{\partial T}{\partial t} = & -\frac{1}{a(1-\mu^2)}\frac{\partial UT'}{\partial \lambda} - \frac{1}{a}\frac{\partial VT'}{\partial \mu} + T'D \\ & -\dot{\sigma}\frac{\partial T}{\partial \sigma} + \kappa T\left(\frac{\partial \pi}{\partial t} + \boldsymbol{v}_H \cdot \nabla_\sigma \pi + \frac{\dot{\sigma}}{\sigma}\right) \\ & + \frac{Q}{C_p} + \mathcal{D}(T) + \mathcal{D}'(\boldsymbol{v})\end{aligned} \tag{5}$$

水蒸気の式

$$\begin{aligned}\frac{\partial q}{\partial t} = & -\frac{1}{a(1-\mu^2)}\frac{\partial Uq}{\partial \lambda} - \frac{1}{a}\frac{\partial Vq}{\partial \mu} + qD \\ & -\dot{\sigma}\frac{\partial q}{\partial \sigma} + S_q + \mathcal{D}(q)\end{aligned} \tag{6}$$

ここで, 各記号の意味は以下のとおりである.

- a : 惑星半径 [m]
- z : 高度 [m]
- t : 時刻 [s]
- u : 経度方向の風速 [m s^{-1}]
- v : 緯度方向の風速 [m s^{-1}]
- T : 絶対温度 [K]
- q : 比湿 [kg kg^{-1}]
- R : 大気の気体定数 [J K^{-1} kg^{-1}]
- C_p : 大気の定圧比熱 [J K^{-1} kg^{-1}]
- g : 重力加速度 [m s^{-2}]
- ε : 水と大気の分子量比 [1]
- Q : 放射, 凝結, 小規模運動過程等による加熱・温度変化 [J kg^{-1} s^{-1}]
- S_q : 凝結, 小規模運動過程等による水蒸気ソース項 [s^{-1}]
- $\mathcal{D}'(\boldsymbol{v})$: 摩擦熱 [K s^{-1}]

またその他の記号は以下のように定義される.

$$\begin{aligned}
\boldsymbol{v}_H &\equiv (u, v) && (7)\\
\theta &\equiv T\,(p/p_s)^{-\kappa} && (8)\\
\kappa &\equiv R/C_p && (9)\\
\Phi &\equiv gz && (10)\\
\pi &\equiv \ln p_s && (11)\\
\dot{\sigma} &\equiv \frac{d\sigma}{dt} && (12)\\
\mu &\equiv \sin\varphi && (13)\\
\varepsilon_v &\equiv 1/\varepsilon - 1 && (14)\\
T_v &\equiv T(1+\varepsilon_v q) && (15)\\
U &\equiv u\cos\varphi && (16)\\
V &\equiv v\cos\varphi && (17)\\
\zeta &\equiv \frac{1}{a(1-\mu^2)}\frac{\partial V}{\partial\lambda} - \frac{1}{a}\frac{\partial U}{\partial\mu} && (18)\\
D &\equiv \frac{1}{a(1-\mu^2)}\frac{\partial U}{\partial\lambda} + \frac{1}{a}\frac{\partial V}{\partial\mu} && (19)\\
UA &\equiv (\zeta+f)V - \dot{\sigma}\frac{\partial U}{\partial\sigma} - \frac{RT'}{a}\frac{\partial\pi}{\partial\lambda} + \mathcal{F}_\lambda\cos\varphi && (20)\\
VA &\equiv -(\zeta+f)U - \dot{\sigma}\frac{\partial V}{\partial\sigma} - \frac{RT'}{a}(1-\mu^2)\frac{\partial\pi}{\partial\mu} + \mathcal{F}_\varphi\cos\varphi && (21)\\
KE &\equiv \frac{U^2+V^2}{2(1-\mu^2)} && (22)\\
T'(\lambda,\varphi,\sigma) &\equiv T(\lambda,\varphi,\sigma) - \overline{T}(\sigma) && (23)\\
\boldsymbol{v}_H\cdot\nabla_\sigma &\equiv \frac{u}{a\cos\varphi}\left(\frac{\partial}{\partial\lambda}\right)_\sigma + \frac{v}{a}\left(\frac{\partial}{\partial\varphi}\right)_\sigma\\
&= \frac{U}{a(1-\mu^2)}\left(\frac{\partial}{\partial\lambda}\right)_\sigma + \frac{V}{a}\left(\frac{\partial}{\partial\mu}\right)_\sigma && (24)\\
\nabla_\sigma^2 &\equiv \frac{1}{a^2(1-\mu^2)}\frac{\partial^2}{\partial\lambda^2} + \frac{1}{a^2}\frac{\partial}{\partial\mu}\left[(1-\mu^2)\frac{\partial}{\partial\mu}\right]. && (25)
\end{aligned}$$

$\mathcal{F}_\lambda, \mathcal{F}_\varphi$ は小規模運動過程による力である. $\mathcal{D}(\zeta), \mathcal{D}(D), \mathcal{D}(T), \mathcal{D}(q)$ は水平拡散項であり, ラプラシアンの冪で表現される超粘性を用いる. 具体的には以下のように表される.

$$\mathcal{D}(\zeta) = -K_{HD}\left[(-1)^{N_D/2}\nabla^{N_D} - \left(\frac{2}{a^2}\right)^{N_D/2}\right]\zeta, \qquad (26)$$

$$\mathcal{D}(D) = -K_{HD}\left[(-1)^{N_D/2}\nabla^{N_D} - \left(\frac{2}{a^2}\right)^{N_D/2}\right]D, \tag{27}$$

$$\mathcal{D}(T) = -(-1)^{N_D/2}K_{HD}\nabla^{N_D}T, \tag{28}$$

$$\mathcal{D}(q) = -(-1)^{N_D/2}K_{HD}\nabla^{N_D}q. \tag{29}$$

この水平拡散項は計算の安定化のための意味合いが強い．小さなスケールに選択的な水平拡散を表すため, N_D としては, 4∼16 を用いる．

境界条件　鉛直流に関する境界条件は

$$\dot{\sigma} = 0 \quad at \quad \sigma = 0,\ 1. \tag{30}$$

である．よって (1) から, 地表気圧の時間変化式と σ 系での鉛直速度 $\dot{\sigma}$ を求める診断式

$$\frac{\partial \pi}{\partial t} = -\int_0^1 \boldsymbol{v}_H \cdot \nabla_\sigma \pi d\sigma - \int_0^1 D d\sigma, \tag{31}$$

$$\dot{\sigma} = -\sigma\frac{\partial \pi}{\partial t} - \int_0^\sigma D d\sigma - \int_0^\sigma \boldsymbol{v}_H \cdot \nabla_\sigma \pi d\sigma, \tag{32}$$

が導かれる．

3-1-2　離散化

水平離散化にはスペクトル法を用いる．展開関数は球面調和関数である．波数切断は三角形切断を用いている．鉛直離散化には Arakawa and Suarez(1983) のスキームを用いる．時間積分は, 重力波に関係する項は台形型 implicit スキームを, その他の項には leapfrog スキームを適用する semi-implicit スキームを用いて行う．leapfrog スキームを用いることによって生じる計算モードの増幅を抑えるため, Asselin(1972) の時間フィルターを 1 ステップ毎に適用する．

3-2　モジュール構造の概観

DCPAM では, プログラムを個々の役割に応じてモジュール化するようにしている．図 4 ではモジュール構造の概観を示している．

以下に, 各モジュールの役割を簡単に記す．

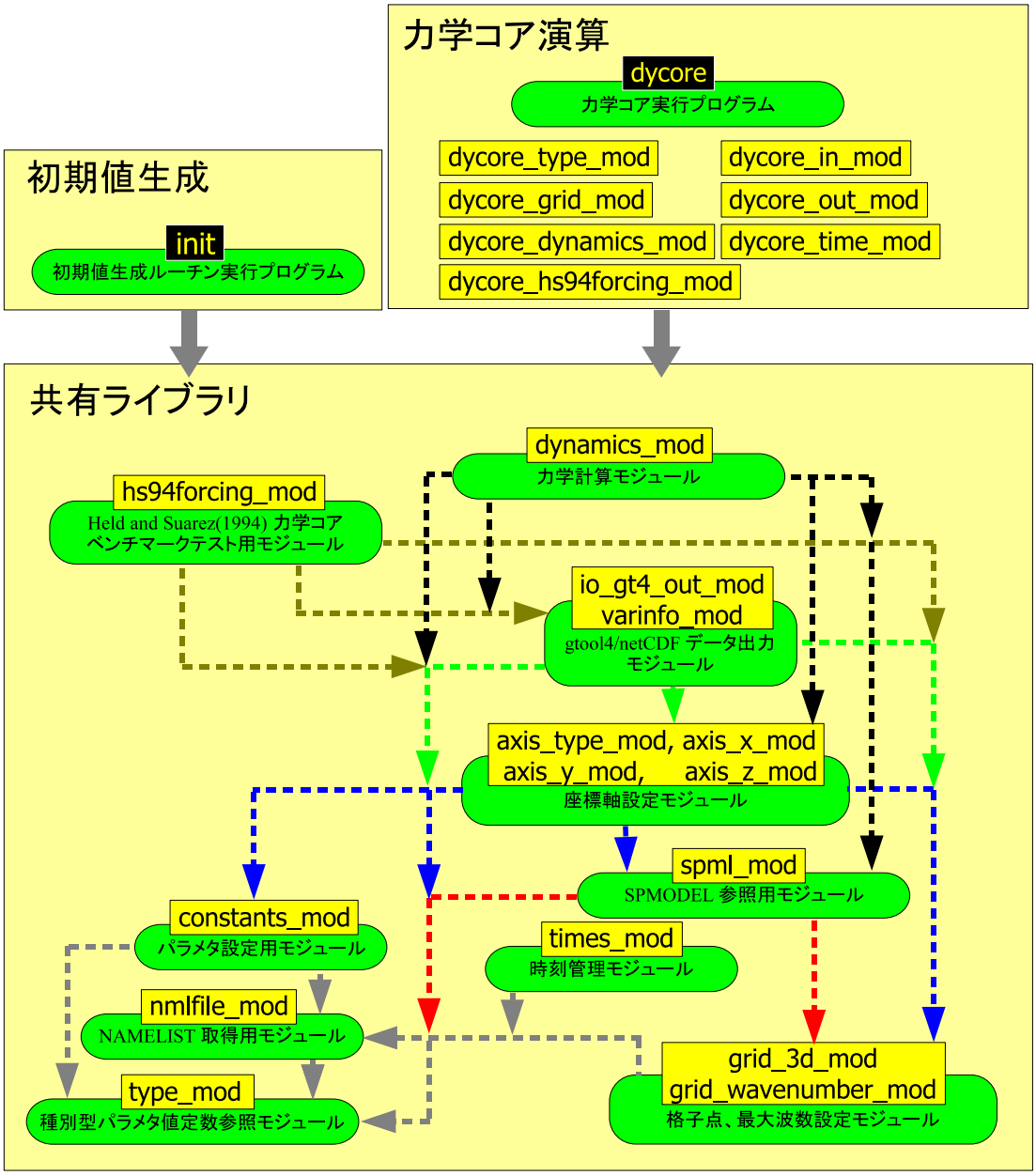


図 4: DCPAM のモジュール構造の概観図. 矢印は, その依存関係を表す. 交差している線のうち, 交差点に矢印があるものは, その矢印の元が矢印の先に依存することを表している. 交差点に矢印が無い場合, その交差している線同士に依存の関係は無い.

- `type_mod`

  種別型パラメタ値を提供するモジュールである. この他のモジュールおよびプログラムは原則的に全てこのモジュールを参照するようにする. そのようにすることで, 種別型パラメタ値を変更したい場合, このモジュール内の値を変更するだけで全てのモジュールにその変更が適用される.

- `nmlfile_mod`

  NAMELIST を取得するための支援モジュールである. ファイルから NAMELIST を取得する場合に便利なサブルーチンを用意している. NAMELIST を取得するモジュールおよびプログラムは, 原則的に全てこのモジュールを呼ぶ.

- `constants_mod`

  物理定数を設定するためのモジュールである. デフォルトでは地球上の標準状態の値を持っているが, NAMELIST により変更することも可能である.

- `time_mod`

  時刻や「計算ステップ」に関して設定するためのモジュールである.

- `grid_3d_mod`, `grid_wavenumber_mod`

  緯度, 経度, 高度に関する格子点の数, およびスペクトル法を用いる際の球面調和関数の最大波数の設定を行う.

- `spml_mod`

  SPMODEL ライブラリ (3-3 参照) のためのラッパーモジュールである. 他のモジュールから SPMODEL のサブルーチンおよび関数を呼ぶ場合には, 原則的にこのモジュールを介して呼び出す.

- `axis_type_mod`, `axis_x_mod`, `axis_y_mod`, `axis_z_mod`

  座標軸を設定するためのモジュールである.

- `io_gt4_out_mod`, `varinfo_mod`

  gtool4 netCDF 規約に基づくデータを出力するためのモジュールである. データの出力のために, gt4f90io (2-2 参照) を用いている.

- `dynamics_mod`

  力学部分の計算を行うモジュールである. スペクトル法による演算は SPMODEL ライブラリ (3-3 参照) を用いて行う.

  力学コアの簡素化を容易にするためには, このモジュールは個々の演算を行うモジュールから成り立ち, 簡素化の際には必要でない演算部分を容易に分

離できる必要がある. しかし現状では, まだモジュールの分離を行えておらず, 今後の課題である.

- `hs94forcing_mod`

  Held and Suarez(1994) の乾燥大気 GCM ベンチマーク用の加熱と散逸を計算するモジュールである.

- `init`

  初期値を生成用の実行プログラムである.

- `dycore`, `dycore_type_mod`, `dycore_grid_mod`, `dycore_in_mod`, `dycore_out_mod`, `dycore_time_mod`, `dycore_dynamics_mod`, `dycore_hs94forcing_mod`

  力学コア演算用の実行プログラムと, それに直接付随するモジュール群である.

3-2-1　モジュールの初期化処理の統一

上記のモジュール群では, その初期化処理を統一の手順で行っている. 初期化処理を以下に規定する動作に統一することで, 個々のモジュールは依存する下位のモジュールの初期化ルーチンを別個に呼び出しても良くなる. 従って, 上位のプログラムではそれらの依存の構造の詳細を知ることなく, 個々のモジュールの初期化処理を行うことが可能である. このアイディアは The FMS Manual (Balaji, 2002) の FMS coding conventions を参考にしている.

- モジュール `module_name_mod` の初期化ルーチンとして `module_name_init` を用意する.
- 初期化のフラグとして, モジュールの global 変数として `module_name_initialized` を用意する. 初期値は `.false.` とする.
- 初期化ルーチン `module_name_init` は以下の動作を行う.
  - モジュール内で用いる変数のうち, save 属性を持つもので allocate が必要なものは, 全て初期化ルーチンで allocate を行う.
  - モジュールで読み込む NAMELIST は全て初期化ルーチンで読む.
  - `module_name_init` が呼ばれた際に `module_name_initialized` が `.false.` の場合, 上記の動作を行うと同時に `module_name_initialized` を `.true.` にする. もしも, `module_name_init` が `.true.` であったならば, 何も動作せずに初期化ルーチンを終了する.

- モジュールが直接依存するその他のモジュールの初期化ルーチンを呼ぶ.
- モジュールのバージョンまたは最終更新日時などを出力する.

3-2-2　モジュールの終了処理の統一

上記のモジュール群では, その終了処理の手順も統一する. これらにより, 初期化ルーチンでの動作をクリアし, (できる限り) 初期化ルーチンが呼ばれる前の状態に戻す. このアイディアもまた The FMS Manual (Balaji, 2002) の FMS coding conventions を参考にしている.

- モジュール `module_name_mod` の終了ルーチンとして `module_name_end` を用意する.
- 終了ルーチン `module_name_end` は以下の動作を行う.
  - 初期化ルーチンで allocate されたものを全て deallocate する.
  - 初期化ルーチンで設定された初期値を全てデフォルトの値に戻す.
  - `module_name_end` が呼ばれた際に `module_name_initialized` が `.true.` の場合, 上記の動作を行うと同時に `module_name_initialized` を `.false.` にする. もしも, `module_name_init` が `.false.` であったならば, 何も動作せずに終了ルーチンを終了する.
  - モジュールが直接依存するその他のモジュールの終了ルーチンを呼ぶ.

3-3　SPMODEL によるスペクトル法の演算

スペクトル法の演算に SPMODEL ライブラリ (Takehiro et al., 2004) を用いることで, 可読性の向上を図る. SPMODEL ライブラリは Fortran90 で記述されたスペクトル法による数値計算のための, 配列を返す関数を提供している. SPMODEL で提供される関数群の特徴は, その関数名が以下のように統一されていることである.

(出力データ型)_(作用)_(入力データの型)

各データの型を表す記号は以下のとおりである.

- s : sin 展開スペクトルデータ
- c : cos 展開スペクトルデータ
- e : フーリエ展開スペクトルデータ
- w : 球面調和関数展開スペクトルデータ
- t : チェビシェフ多項式展開スペクトルデータ
- x : 格子点データ [x 座標 または 緯度座標]
- y : 格子点データ [y 座標 または 経度座標]
- z : 格子点データ [高度 (z や圧力や σ) 座標または 動径座標]
- g : 一般的な格子点データ
- a : 全ての種類

この関数に合わせて, 各変数を

(データ型)_(名前)

という形で定義すると, 以下のようにプログラムを書くことが出来る. 具体例は以下のようになる.

- xy 次元のデータからスペクトルデータへの変換関数.

```
w_xy(xy_data)
```

- スペクトルデータにラプラシアンを作用させる関数.

```
w_Lapla_w(w_data)
```

- 温度のスペクトルデータ w_Temp を格子点データ xy_Temp に変換する式.

```
xy_Temp = xy_w(w_Temp)
```

- 緯度経度方向の風速 xy_VelLon, xy_VelLat の格子点データから発散の格子点データ xy_Div を求める式.

```
xy_Div = &
    &  xy_w(                                              &
    &   w_Div_xy_xy( xy_VelLon , xy_VelLat ) / Rplanet &
    & )
```

これらの特徴を活かすことで, 可読性の高い, 従来よりも数式に近いコードを簡単に書くことが可能となる. 実際に, dynamics_mod (3-2 参照) では渦度の式 (3) を以下のように記述している.

```
wz_Vor_t = &
     & wa_Div_xya_xya( xyz_VA, - xyz_UA ) / R0 &
     &   + wz_DiffVorDiv * wa_xya(xyz_Vor)
```

なお, SPMODEL ライブラリはその演算の際に ISPACK (Ishioka, K., 2002) を用いている. ISPACK はスペクトル変換, 時間積分等の流体方程式の数値計算に必要となる基本的なサブルーチンをまとめた FORTRAN77 で記述されたライブラリである.

3-4　コーディングルール

DCPAM では Fortran90 ソースコードの可読性, 可変性, 移植性を損なわず, またメンテナンスコストを抑えるため, 原則的に気象庁標準コーディングルール (Muroi, et al., 2002) に則ったコーディングを行う. ただし, このコーディングルールは汎用性が高いために変数名やプログラム名, ドキュメントの生成の方法など関しては規定していない. 以降の節では, 階層的モデル群の構築にとって有益であろうと考えられ, 実際に DCPAM において採用しているコーディングルールを紹介する.

なお, 気象庁コーディングルールでは, 「最新の JIS 規格は Fortran 95 であるが、Fortran 90 しか使えない環境が依然多い状況であると思われるので、Fortran 95 の新機能は当面使わないことを推奨する。」とあるが, ポインタを使う上で, Fortran95 規格から使用可能な `null=>()` は非常に有用であることと, Fortran95 の新機能を利用可能な環境が増えてきたことから, 少なくとも `null=>()` に関しては DCPAM で積極的に用いていく. [†2].

3-5　変数命名規則

変数名のフォーマットを揃えることで, モデルを単純化・複雑化したり, 他のモデルを組み込んだりする際の作業の効率化だけでなく, モデルをより物理的に理解しやすいものにすることが期待できる. 以下では, DCPAM における変数の命名規則を記す. この命名規則は, 元々 SPMODEL (3-3 参照) で用いられていた命名規

[†2] `null=>()` とは宣言文においてポインタ変数の初期値として設定できるもので, ポインタの初期状態を「空状態」にすることができる. これを用いない場合, ポインタの初期状態は「不定」となるため, 安全のために必ず, プログラム内部において `nullify` 関数が必要となってしまう. または `allocate` を用いても良いが, 多重に `allocate` するとメモリにゴミを溜める事になるため推奨しない.

則を拡張したものである. これらの命名規則は, 将来的には SPMODEL や 2 次元雲解像モデル deepconv/arare (Sugiyama et al., 2004) とも統一することを考慮している.

- 基本命名規則

  原則的に変数は x_yyyy_z のように表す. x は空間次元に関する情報で, 2 次元, 3 次元の場合には xx, xxx のように数を増やす. yyyy はその変数の物理的意味を表す. z は時間に関して示す.

  空間次元に依存しない変数であったり, 時間に依存しないものに関しては, x_ や _z が無いこともあり得る.

- 空間次元に関する添え字

  変数の接頭詞 x_ には空間次元に関する情報が与えられる. 1 文字が 1 つの次元を表すようにしているため, 接頭詞の数と文字種だけで, その量がどの次元に依存した何次元の量なのかが一目でわかる. 以下に, 個々の文字の意味を記す.

  - s : sin 展開スペクトルデータ
  - c : cos 展開スペクトルデータ
  - e : フーリエ展開スペクトルデータ
  - w : 球面調和関数展開スペクトルデータ
  - t : チェビシェフ多項式展開スペクトルデータ
  - x : 格子点データ [x 座標 または 緯度座標]
  - y : 格子点データ [y 座標 または 経度座標]
  - z : 格子点データ [高度 (z や圧力や σ) 座標 または 動径座標]
  - p : 格子点データ [x の交互格子点上の座標]
  - q : 格子点データ [y の交互格子点上の座標]
  - r : 格子点データ [z の交互格子点上の座標]
  - a : 格子点データ [任意の座標]

- 時間に関する添え字

  変数の接尾詞 _z には時間に関する情報を与える.

  - aa : 時刻 $t + 2\Delta t$
  - a : 時刻 $t + \Delta t$
  - n : 時刻 t
  - b : 時刻 $t - \Delta t$
  - bb : 時刻 $t - 2\Delta t$

  a は **a**fter, n は **n**ow, b は **b**efore のそれぞれ頭文字をとったものである. 例えば, 時刻が $t + n\Delta t$ の場合には n 個分だけ a を, $t - n\Delta t$ の場合には n 個分だけ b をつける.

- 物理量に関して

  yyyy の部分にはその物理意味に関する情報を与える. 方針として, なるべく数式などの記号をそのまま用いるのではなく, その物理的意味 (呼び方) をできるだけ反映させた変数名にすべきと考えている. 当然, 全ての変数に対して規定をすることは難しいので, ここでは代表的に良く用いられるものに関して規定してある.

  - 座標に関する量

    - X : x 座標
    - Lon : 経度座標
    - Y : y 座標
    - Lat : 緯度座標
    - Z : z 座標
    - Rad : 動径座標
    - Press : 気圧座標
    - Sigma : σ 座標

    - DelX : x 座標格子間隔
    - DelLon : 経度座標格子間隔
    - DelY : y 座標格子間隔
    - DelLat : 緯度座標格子間隔
    - DelZ : z 座標格子間隔
    - DelRad : 動径座標格子間隔
    - DelPress : 気圧座標格子間隔
    - DelSigma : σ 座標格子間隔

  - 時間に関する量

    - Time : 時間
    - DelTime : 時間間隔

  - 予報変数に関する量

    - VelX : x 方向への風速
    - VelLon : 経度方向への風速
    - VelY : y 方向への風速
    - VelLat : 緯度方向への風速
    - VelZ : z 方向への風速
    - VelRad : 動径方向への風速
    - VelPress : 気圧方向への風速
    - VelSigma : σ 方向への風速

    - Vor : 渦度
    - HDiv : 発散

    - Temp : 温度
    - VirTemp : 仮温度
    - PotTemp : 温位

    - Press : 気圧

  - Height　：高度

- 具体例

  上記の規則を適用した例をいくつか示す.

  - x_Lon　　：経度方向の座標データ
  - r_DelSigma　：σ の半整数レベル格子間隔
  - xyz_Temp_b　：温度の 3 次元格子点データ (時刻 $t - \Delta t$)
  - wz_Vor_a　　：渦度の水平スペクトルデータ
    + 鉛直格子点データ (時刻 $t + \Delta t$)

3-6　ファイル, モジュール, プログラムの命名法

以下では, ファイル名とモジュール名, プログラム名に関する命名法を紹介する. ファイル名とその内部のプログラムとの名前に関連性を持たせることでメンテナンスコストを抑えることが目的である. 以下では, 原則的に一つのファイルに一つのモジュール (およびそのモジュールに含まれるサブルーチンや関数) を含むこととする. なお, 項目の 1 〜 3 番目は The FMS Manual (Balaji, 2002) の FMS coding conventions を模倣している.

- モジュール名には接尾語として _mod をつける. このようにすることで, 名称のみでそれがモジュールであることが識別できる.

- ファイル名は, モジュール名から接尾語の _mod を除いたものに .f90 をつけたものにする. (例えば, モジュール名が module_name_mod の場合, ファイル名は module_name.f90 とする.)

- モジュール module_name_mod 内のサブルーチン名称には, 接頭詞として module_name_ をつける. これにより, 外部から呼び出した際にそのサブルーチンがどのモジュールから呼ばれたものかが分かりやすくなる. なお, 現在のところ関数と変数に関しては規定していない. 関数や変数は代入式において用いられることが考えられるため, 長い名前にしてしまうと可読性が損なわれる可能性がある. よって, 呼び出し元のモジュールが一目で分かるというメリットと, 可読性の低下との兼ね合いを考慮すべきである.

- モジュール module_name_mod で NAMELIST を読み込む場合, その NAMELIST 変数は module_name_nml とする. なお, モジュールにおいて複数種の NAMELIST を読み込む必要がある場合, module_name_xxxx_nml という名称を使っても良い. (xxxx には任意の文字列を代入する). ただし, module_name_xxxx_mod というモジュールが存在しないことが条件である.

3-7　ドキュメント

モジュールには, その動作, 使用方法, エラー処理等に関して述べたドキュメントを添えておくべきである. ただし, ドキュメントの生成は手間がかかるため, できるだけその作成のためのコストを抑えることが望ましい. そのための 1 つの方法として Fortran90 のソースコードにドキュメントもそのまま埋め込み, 最終的にその部分だけを抜き出すことでドキュメントを生成するということが考えられる. (一例として FMS (GFDL, 2005) のドキュメント生成システムが挙げられる). しかし, その方法の欠点として, Fortran90 のソースコードの可読性が低下するという問題がある. 大抵ドキュメントとして想定される HTML, XML, TEX などは, 文字の修飾や体裁を整えるための「タグ」や「コマンド」といった記号が数多く必要となり, Fortran90 のソースコードとしては非常に見づらくなる可能性が高いのである.

ドキュメント生成のコストを抑えるもう 1 つの方法として, Fortran90 ソースコードに記述されているサブルーチンの引数やその引数のデータ型など, そのモジュールを使おうとするユーザに必要な情報に関してはコードをそのままドキュメントに変換するという方法がある. そのようにしておくと, プログラムが仕様変更された際, ドキュメントも自動的に更新することができる. これは単純にメンテナンスコストを下げるということだけでなく, コードとドキュメントとの乖離を防ぐという重要な役割を果たす. しかしこの方法にも上記と同様な欠点として, ドキュメントに反映するべき場所を指定するフラグが Fortran90 のソースコードを見づらくしてしまう可能性が高いのである.

以下では, Fortran90 内に Ruby Document (Ruby Documentation Project, 2005) フォーマットのドキュメントを埋め込むことで, Fortran90 としての可読性をできるだけ損なうことなく, ソースコードからドキュメントを生成する試みを紹介する.

3-7-1　Ruby Document の特徴

Ruby Document (以下, 略して RD と呼ぶ) とは, オブジェクト指向スクリプト言語 Ruby (Matsumoto, 2005) において, それらで作成されたプログラムのドキュメントを書くために考え出された汎用のドキュメントフォーマットである.

このフォーマットは以下のような特徴をもつ.

- テキスト形式に近い書式である.

HTML や TEX や roff などで必要となる文字の修飾のための記号が, RD では非常に少なくて済むようになっている. そのため, Fortran90 のソースコードに埋め込む「Fortran90 のコードでないもの」の数を極力抑えることが可能である.

- 多種の形式への変換が可能である.

  RD によって記述されたドキュメントは HTML や TEX, roff 形式へ機械的に変換することが可能である. これにより, Web での公開や紙面への印刷など, 場合に応じて適した形式にすることができる. また, HTML への変換が可能なことで, カスケードスタイルシートによる修飾も可能になる.

なお, RD では TEX のような数式を記述することは不可能である. そのため, 数式を含むドキュメントに関しては別途 TEX で記述することにし, ここではモジュール, およびその内部のサブルーチンの引用仕様を中心に記述したドキュメントの作成を前提とする.

3-7-2　Fortran90 ソースコードサンプル

Fortran90 ソースの冒頭部分には, そのモジュールに関する全体的な情報を記載する.

- `Module`

  `Module` の後ろにはモジュール名と, 可能ならばその要約を記述する.

  `Developers` にモジュールの作成, 改訂に関わった開発者の名前を記載する. `Version` にはバージョン番号または最終更新日時, もしくはその両方を記載する. `Tag Name` には CVS Tag を記載する. `Change History` には CVSWeb のモジュールに関する URL を記載する[†3].

  - `Overview`
    モジュールの概要を記載する.
  - `Error Handling`
    モジュールにおけるエラー処理の詳細を記述する.
  - `Known Bugs`
    モジュールにおける既知のバグ情報を記述する.

[†3] `Tag Name` と `Change History` に関しては, バージョン管理システム CVS (Berliner, B., 1994) を利用して開発していることを前提にしている. もしもそのようなバージョン管理システムを利用していない場合には必要のない項目である.

- Notes
  モジュールにおける備考を記述する.
- References
  モジュールにおいて参照した情報を記述する.
- Future Plans
  将来的に計画される仕様変更などの情報を記述する.

図 5 がその例である. !=begin および !=end は RD の開始と終了を示すフラグであり, これらの間のみが RD のドキュメントとして識別される. =, == は HTML で言えば <h1>...</h1> および <h2> ... </h2>, TEX で言えば \section{...} および \subsection{...} に相当する記号である.

Fortran90 のモジュールの宣言文からサブルーチンの開始の直前までの部分では, モジュール全体として依存するモジュールや公開要素に関する情報を記載する. 図 6 がその例である.

- Dependency
  モジュール全体として依存するモジュール, および依存する公開要素の一覧
- Public Interface
  公開要素として提供するサブルーチン名, 関数名, 変数名の一覧を記載する.
- Public Data
  公開要素として提供される変数の詳細情報を記載する.

Fortran90 のサブルーチンの宣言以降の部分には, サブルーチンの引用仕様等に関する情報を記載する. 図 7 がその例である.

- Procedure Interface
  サブルーチンに関する引用仕様を記載する. 個々のサブルーチンに関して記載する内容は以下に記す.
  - Subroutine または Function
    Subroutine の後ろにはサブルーチン名を, Function の後ろには関数名を記載し, 可能ならばその要約を記述する.
    その下にサブルーチン (または関数) の動作の詳細を記述する.

```
!-------------------------------------------------------------------
!     Copyright (C) Morikawa Yasuhiro, 2005. All rights reserved.
!-------------------------------------------------------------------
                                                                !=begin
!= Module module_name_mod : Sample module of RD in F90 source code.
!
!   * Developers: Morikawa Yasuhiro
!   * Version: $Id: module_name.f90,v 1.4 2005/01/09 morikawa Exp $
!   * Tag Name: $Name: Initial$
!   * Change History: ((<URL:https://www.gfd-dennou.org/dcpam/cvsweb>))
!
!== Overview
!module_name_mod の概要
!
!== Error Handling
!module_name_mod におけるエラー処理の詳細。
!
!== Known Bugs
!既知のバグに関する情報。
!
!== Notes
!備考。
!
!== References
!参考文献。
!
!== Future Plans
!将来的に計画される仕様変更等の情報。
                                                                !=end
```

図 5: RD を埋め込んだ Fortran90 の冒頭部分

- ▷ Dependency
  サブルーチン (または関数) として依存するモジュール, および依存する公開要素の一覧.
- ▷ NAMELIST
  サブルーチンが初期化ルーチンで, NAMELIST を読み込むものである場合に, 読み込む NAMELIST 変数について記載する.
- ▷ Input
  授受特性が in の引数に関して記載する.
- ▷ Output
  授受特性が out の引数に関して記載する.
- ▷ In/Out

```
module module_name_mod
                                                                    !=begin
  !== Dependency
  !
  use type_mod,   only: STRING, REKIND, DBKIND, INTKIND
                                                                    !=end
  implicit none
                                                                    !=begin
  !== Public Interface
  !
  private
  public :: module_name_init, module_name_end    ! subroutines
  public :: module_name_func                     ! functions
  public :: data1, data2                         ! variables

  !== Public Data
  !
  real(DBKIND), save  ::  & ! follow data is default values.
       &
       & data1     = 3.141592653589793 , & ! 円周率
       & data2     = 6.371d6               ! 球の半径
                                                                    !=end
```

図 6: RD を埋め込んだ Fortran90 の公開要素宣言部分

授受特性が inout の引数, またはポインタ引数に関して記載する.

3-7-3　F90 ソースコードからの HTML の生成

上記のように記述した Fortran90 のソースコードに, まず以下のように行頭からコメントアウトの文字である “!” を取り除くような処理を施す. (“[” と “]” の間には, タブ文字と空白が入っている).

```
$ sed 's/^[  ]*!//' module_name.f90 > module_name.rd
```

そして RDtool ((Ruby Documentation Project, 2005) 参照) の rd2 というコマンドにより HTML 化される.

```
$ rd2 module_name.rd > module_name.htm
```

これらの結果にカスケードスタイルシートによる修飾を加えることで, 図 8〜図 11 に示されるような HTML ドキュメントとなる. (本来は 1 つのページだが, 紙面の問題から分割している).

図 6 と図 9, 図 7 と図 10 〜図 11 をそれぞれ比較すると, 公開要素や引数などの部分のソースコードがそのまま HTML として出力されていることが分かる. このようにしておくことで, モジュールやサブルーチンの仕様に変更が生じた場合でも, 変更がそのままドキュメントに反映され, ソースとドキュメントの内容が食い違うことを防ぐことができる.

```
contains
                                                                    !=begin
  !== Procedure Interface
  !
  !=== Subroutine module_name_init : モジュールの初期化
  !
  !NAMELIST を入力し、グローバル変数を allocate する。
  !
  subroutine module_name_init(inchar, outint, inoutdata, inoutdb)
  !
  !==== Dependency
  !
    use type_mod   , only : REKIND, DBKIND, INTKIND, TOKEN, STRING
    use nmlfile_mod, only : nmlfile_init, nmlfile_open, nmlfile_close
    use dc_trace   , only : BeginSub, EndSub, DbgMessage
    use dc_message , only : MessageNotify
                                                                    !=end
    implicit none
                                                                    !=begin
    !==== NAMELIST
    !
    character(TOKEN)  :: name        = '' ! 名前
    integer(INTKIND)  :: length      = 0  ! 長さ

    namelist /module_name_nml/ &
         & name         , &  ! 名前
         & length       , &  ! 長さ
         & Loop              ! ループの初期値

    !==== Input
    character(*)    , intent(in) :: inchar  ! Input Character

    !==== Output
    integer(INTKIND), intent(out) :: outint ! Output Integer

    !==== In/Out
    real(REKIND),     intent(inout)    :: inoutdata  ! In/Out Data
    real(DBKIND),     pointer,optional :: inoutdb    ! In/Out Data
                                                                    !=end
  continue
                            :
                            :
  end subroutine module_name_init

                            :
end module sample_mod
```

図 7: RD を埋め込んだ Fortran90 のサブルーチン引用仕様部分

Module module_name_mod : Sample module of RD in F90 source code.

- Developers: Morikawa Yasuhiro
- Version: $Id: module_name.f90,v 1.4 2005/01/09 morikawa Exp $
- Tag Name: $Name: Initial$
- Change History: <URL:https://www.gfd-dennou.org/dcpam/cvsweb>

Overview

module_name_mod の概要

Error Handling

module_name_mod におけるエラー処理の詳細。

Known Bugs

既知のバグに関する情報。

Notes

備考。

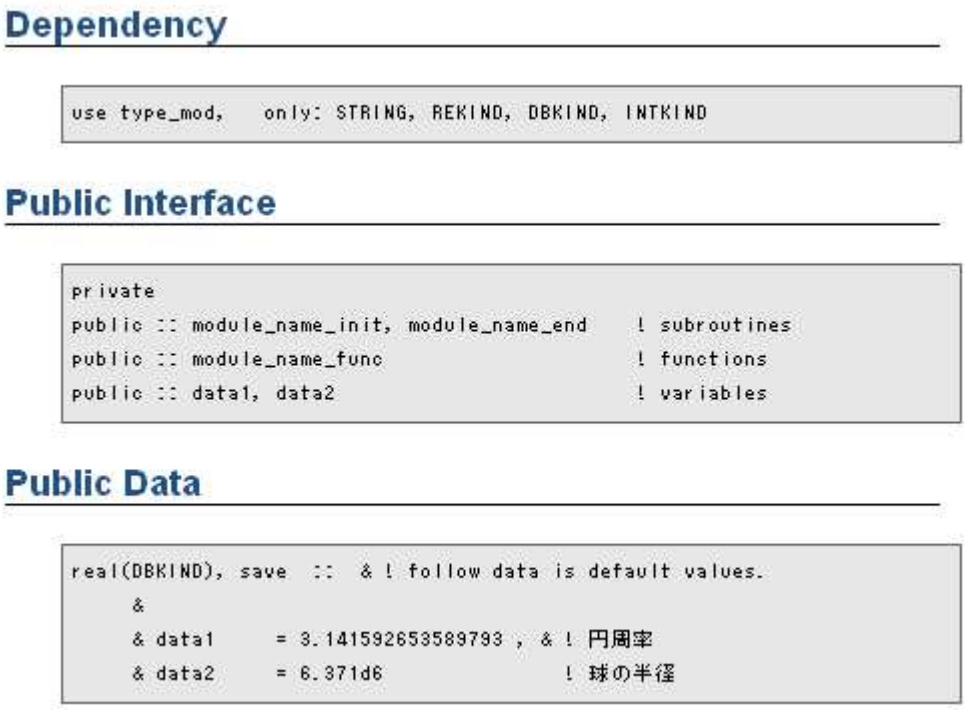
References

参考文献。

Future Plans

将来的に計画される仕様変更等の情報。

図 8: RD を埋め込んだ Fortran90 の冒頭部分の HTML への変換結果



Dependency

```
use type_mod,   only: STRING, REKIND, DBKIND, INTKIND
```

Public Interface

```
private
public :: module_name_init, module_name_end    ! subroutines
public :: module_name_func                     ! functions
public :: data1, data2                         ! variables
```

Public Data

```
real(DBKIND), save  ::  & ! follow data is default values.
     &
     & data1     = 3.141592653589793 , & ! 円周率
     & data2     = 6.371d6               ! 球の半径
```

図 9: RD を埋め込んだ Fortran90 の公開要素宣言部分の HTML への変換結果

Procedure Interface

Subroutine module_name_init : モジュールの初期化

NAMELIST を入力し、グローバル変数を allocate する。

```
subroutine module_name_init(inchar, outint, inoutdata, inoutdb)
```

Dependency

```
use type_mod   , only : REKIND, DBKIND, INTKIND, TOKEN, STRING
use nmlfile_mod, only : nmlfile_init, nmlfile_open, nmlfile_close
use dc_trace   , only : BeginSub, EndSub, DbgMessage
use dc_message , only : MessageNotify
```

NAMELIST

```
character(TOKEN)  :: name        = '' ! 名前
integer(INTKIND)  :: length      = 0  ! 長さ

namelist /module_name_nml/ &
     & name         , &  ! 名前
     & length       , &  ! 長さ
     & Loop              ! ループの初期値
```

図 10: RD を埋め込んだ Fortran90 のサブルーチン引用仕様部分の HTML への変換結果 1

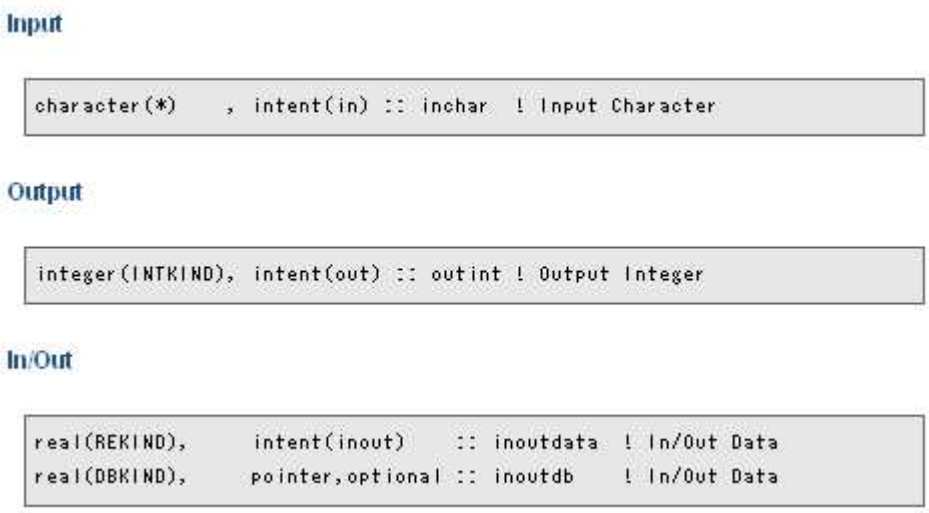


図 11: RD を埋め込んだ Fortran90 のサブルーチン引用仕様部分の HTML への変換結果 2

4　テスト計算

DCPAM の実装実験として, Held and Suarez (1994) の力学コアベンチマークテストを行った. このテストは, 物理過程に依存しない大気大循環モデルの力学過程を評価するためのベンチマーク計算である. この実験では, 温度場にニュートン冷却を与え, $\sigma < 0.7$ の速度場にレイリー摩擦を与える. ニュートン冷却で使用する平衡温度を図 12a に示す. ニュートン冷却の時定数 k_T とレイリー摩擦の時定数 k_v は以下のように与えられる.

$$k_T = k_a + (k_s - k_a) \max\left(0, \frac{\sigma - \sigma_b}{1 - \sigma_b}\right) \cos^4 \phi, \tag{33}$$

$$k_v = k_f \max\left(0, \frac{\sigma - \sigma_b}{1 - \sigma_b}\right). \tag{34}$$

ただし,

$$\sigma_b = 0.7, \tag{35}$$

$$k_f = 1 \text{ day}^{-1}, \tag{36}$$

$$k_a = \frac{1}{40} \text{ day}^{-1}, \tag{37}$$

$$k_s = \frac{1}{4} \text{ day}^{-1}. \tag{38}$$

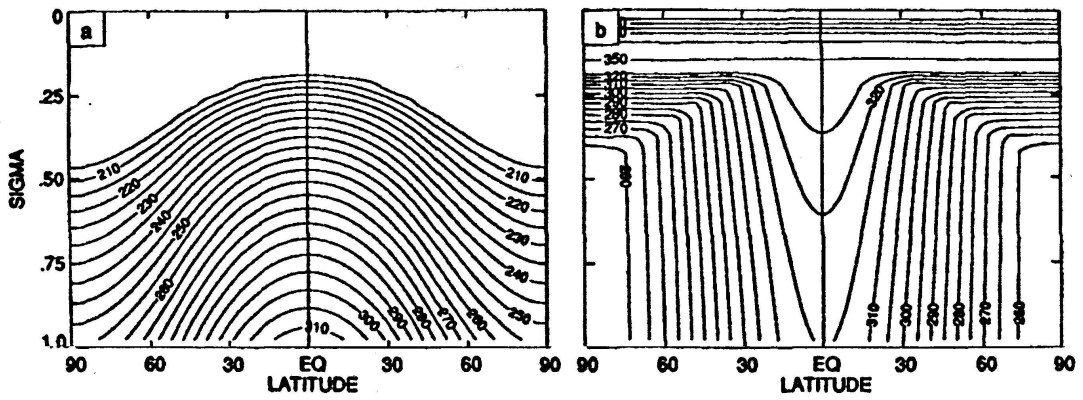


図 12: 与えられる放射平衡温度 (a) と温位 (b) 分布. (Held and Suarez(1994) の Fig. 1 より).

図 13 は, Held and Suarez(1994) において示された結果である. 以下のモデルで 1200 日積分し, その始めの 200 日を捨てて残りの 1000 日を平均した東西風速の帯

状平均である. モデルは T63 のスペクトルモデルで (T63 の “T” はスペクトル法の展開関数である球面調和関数の波数切断に三角形切断を用いていることを示し, “63” は最大波数を示す), 鉛直座標として σ 座標系, 時間積分として重力波項には implicit スキームを, その他の項には leapfrog スキームを適用する semi-implicit スキームを用いている. σ では等間隔に 20 の鉛直層がとられ, モデルの上端は気圧がゼロである. 非重力波項以外の時間積分には leapfrog スキームが用いられるので, 計算モードを制御するために Robert(1966) のタイムフィルターが使われる. 渦度, 発散, 温度の水平混合には 4 乗の Laplacian 型が用いられ, 系の最大波数の e-folding time は常に 0.1 日となるような強さに設定される.

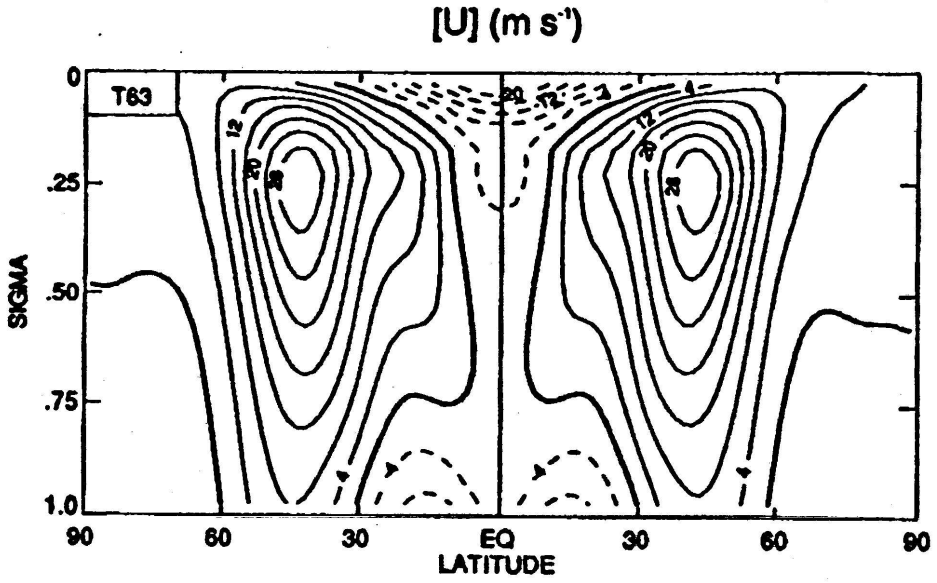


図 13: Held and Suarez(1994) の T63 のスペクトルモデルから得られた, 帯状平均した東西風. (Held and Suarez(1994) の Fig. 2 より).

DCPAM でも同様な実験を行った. まずはモデル設定の詳細について述べる. DCPAM の方程式系および離散化は原則的に 3-1 で述べたとおりだが, 今回のモデルでは物理過程を含んでおらず, また時間積分には explicit な leapflog スキームを用いている. 鉛直層とモデル上端の設定は Held and Suarez(1994) と同様である. (26)〜(29) で用いられる N_D は 8 とし, 最大波数に対する e-folding time を 0.1 日とする. これらも Held and Suarez(1994) と同様の設定である. 水平解像度 T21 で 1200 日積分させる. 初期値は等温無風で, ある 1 点に 1 K の擾乱を与える. 初期の温度場としては 272 K と 300 K の 2 種類を用いた.

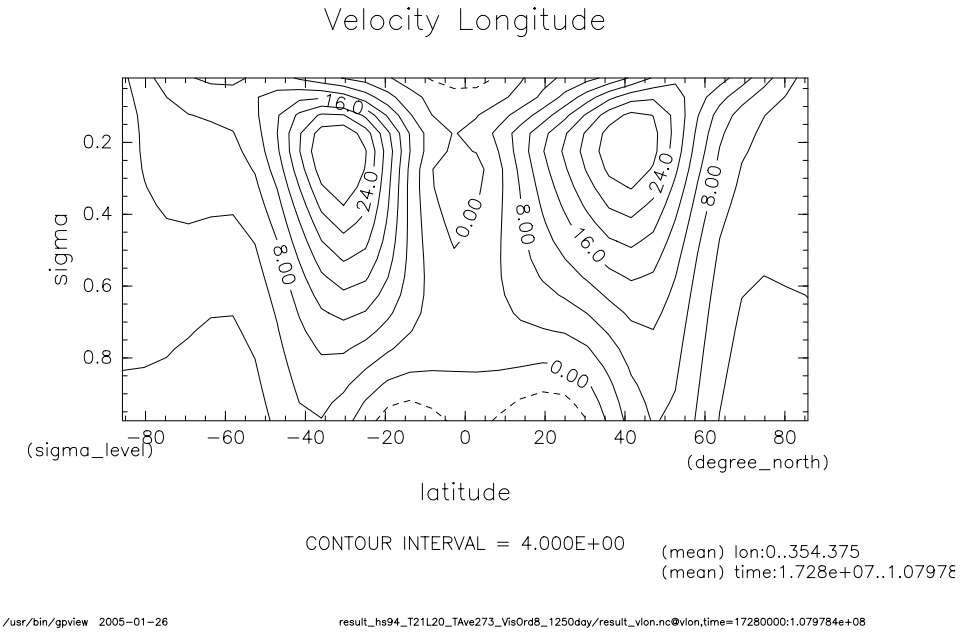
図 14 は, 最後の 1000 日分を平均した帯状平均の東西風である. 上の図の初期の温度場は 273 K で, 下の図は 300 K である. 図 13 と比較すると, 緯度 ±40 度, $\sigma = 0.25$ 付近で東向きの風が卓越してるなど全体の傾向では似ている. しかし, Held and Suarez (1994) では赤道付近で高度に依らず東風が吹くのに対し, DCPAM の結果では限られた高度でしか東風は吹かない. また, 図 13 のような綺麗な南北対称性が無いことも差異として挙げられる. なお, 上下の図を比べると, 初期値に応じて対称性に差があることが分かる. この差異の原因として, 2 つの実験における傾圧不安定帯から射出されるロスビー波の活動度の相違というものが 1 つの可能性として考 えられる.

西風が最も発達する $\sigma = 0.2$ における東西風速の時間変化を図 15 に示す. 初期温度が 273 K の方は南北対称性が崩れているのに対し, 300 K のものは比較的対称性を維持しているのが分かる. 従って, T21 のスペクトルモデルで, 1200 日積分する際には多少なりとも初期値が結果に影響するしていることが分かる.

南北風速の帯状平均を図 16 に, $\dot{\sigma}$ の帯状平均を図 17 に示す. ハドレー循環が起こっている様子が見て取ることができる. 地表面気圧の様子を見ると, 傾圧不安定が発生することも見ることができる. 温度の帯状平均を図 18 に示す. 温度は基本的に, ニュートン冷却によって Held and Suarez (1994) の平衡温度 (図 12 の上図参照) に近づくが, 子午面循環の影響を受け, 少し歪んだ分布となる. なお, 1200 日積分した際の全質量の誤差の割合は 10^{-4} % 以下であった.

波の活動度の違いが Held and Suarez (1994) と DCPAM のモデルの結果の差異をもたらした可能性があるので水平解像度 T42 でも計算を行った. 150 日積分させ, 最後の 50 日分を平均した帯状平均の東西風を図 19 の上図として示した. 下図は T21 の計算の 100 日から 150 日まで平均した帯状平均東西風である. 初期値には両方とも, 等温 300 K, 静止で, 1 点に 1 K の擾乱を与えた. 100 日〜150 日の段階では, T42 の結果は T21 に比べ非常に 図 13 と似ていることが分かる. 他の全ての条件が同じである. しかし, T42 とした場合に, Held and Suarez (1994) の結果に近づいたのが, 波の活動度の相違によるものであるかどうかについては今後さらに検討していく必要がある.

これらの結果から, ハドレー循環や亜熱帯ジェットといった, Held and Suarez (1994) のテスト計算において起こることが期待される現象は定性的には表現できた. しかし, 定量的な差が, 解像度によるものなのか, それともモデルのバグなのかは残念ながらまだ調査が必要である. 今後は, 解析やモデルの設定を変えての再計算によって原因を探る.



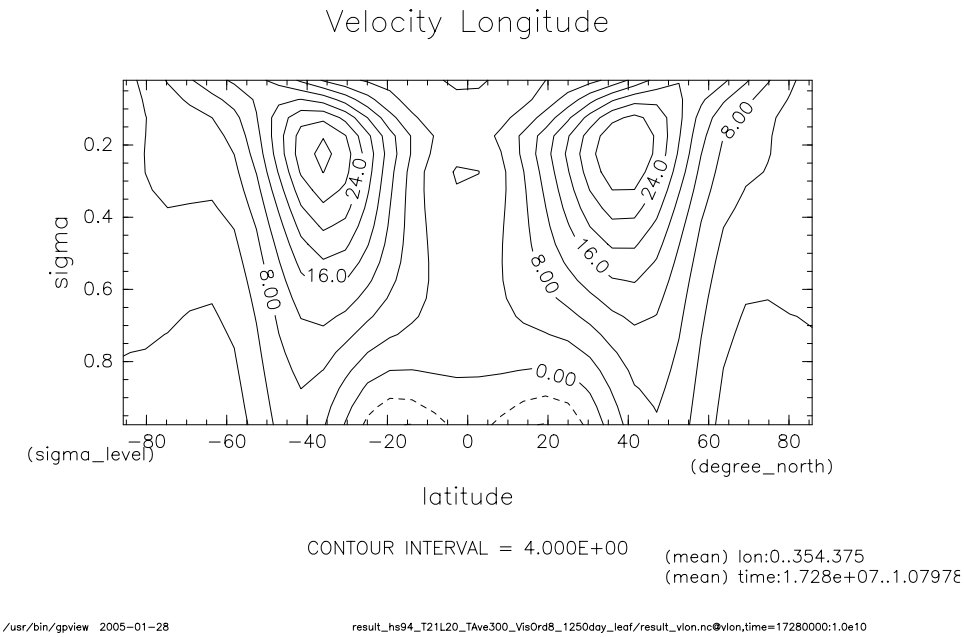
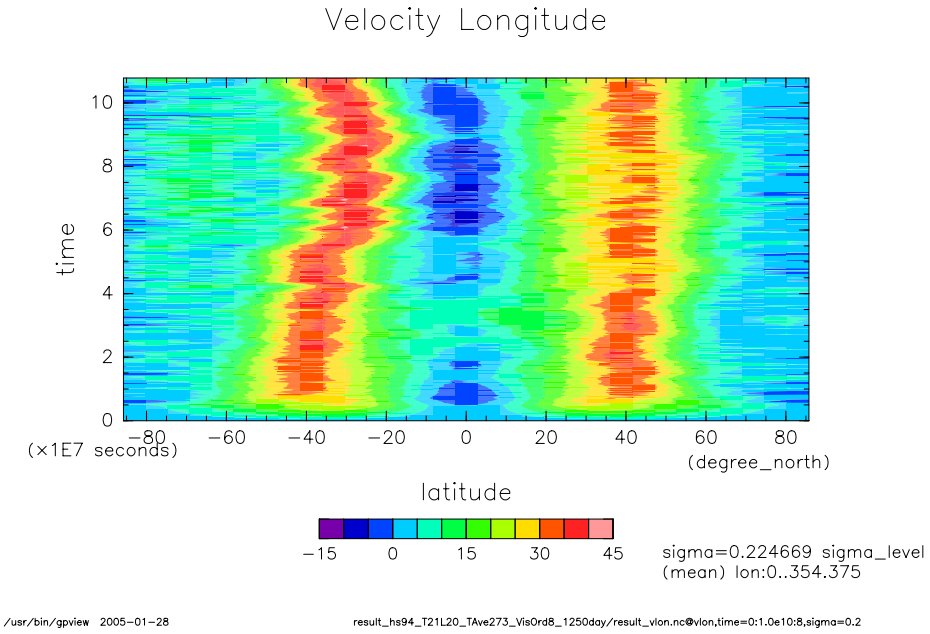


図 14: T21 スペクトルモデルで得られた帯状平均東西風 (単位は m/s). 200 日目から 1200 日目までの平均値. 初期値は等温静止で 1 点に 1 K の擾乱を与えた. 上の温度の初期値は 273 K, 下は 300 K.



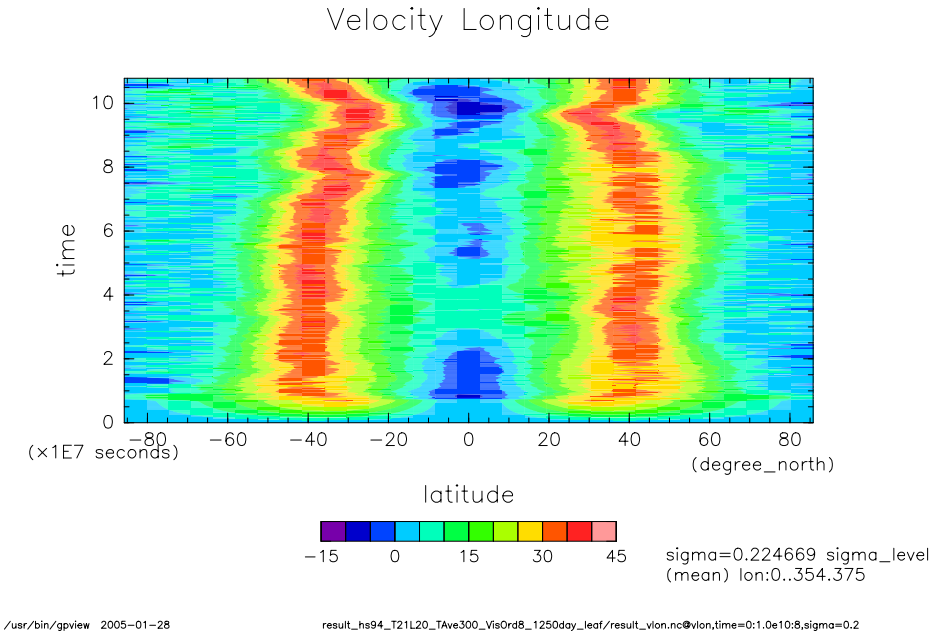


図 15: T21 スペクトルモデルで得られた帯状平均東西風 (単位は m/s). 最も西風が強くなる $\sigma = 0.2$ における時間変化を示す. 初期値は等温静止で 1 点に 1 K の擾乱を与えた. 上の温度の初期値は 273 K, 下は 300 K.

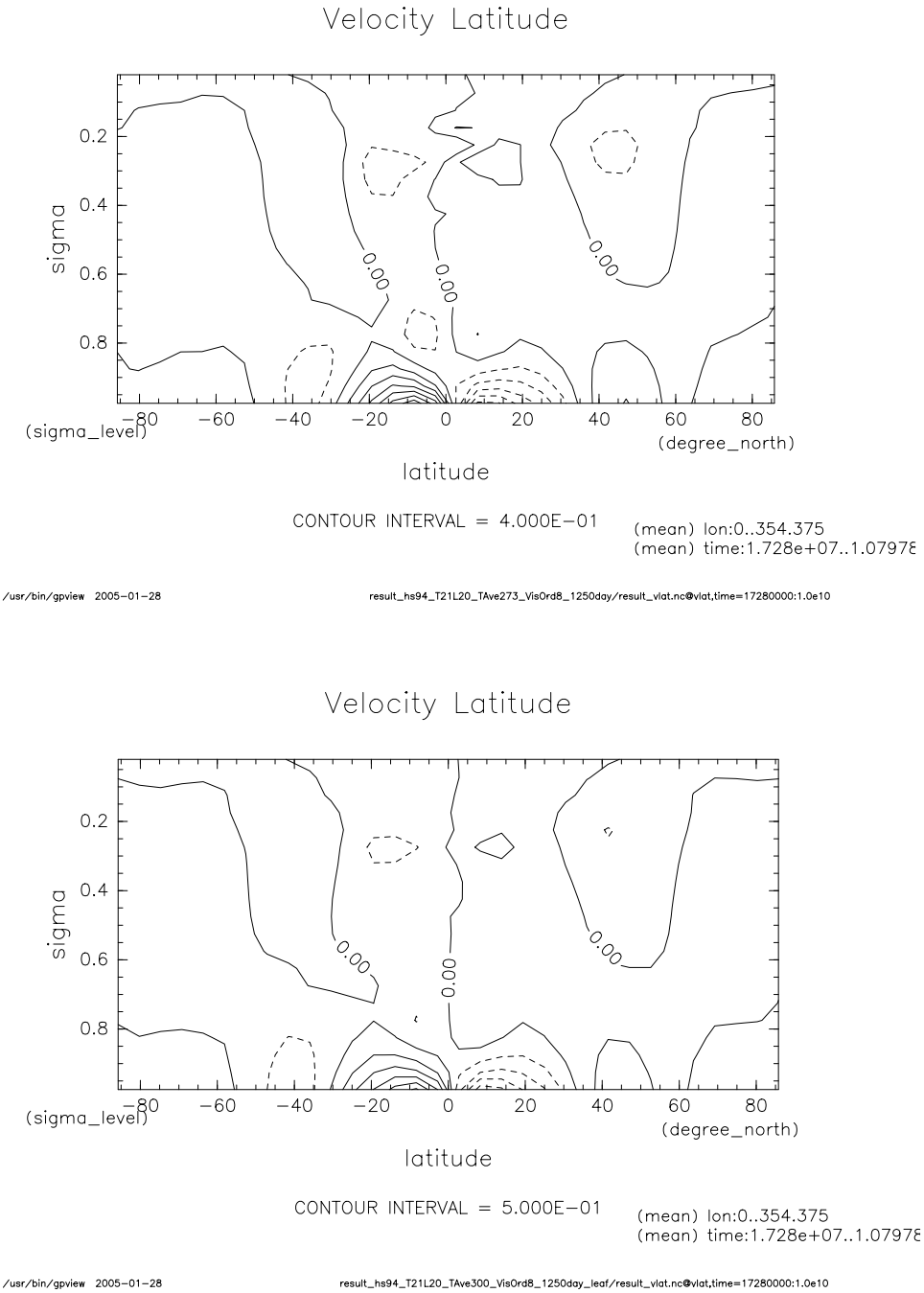
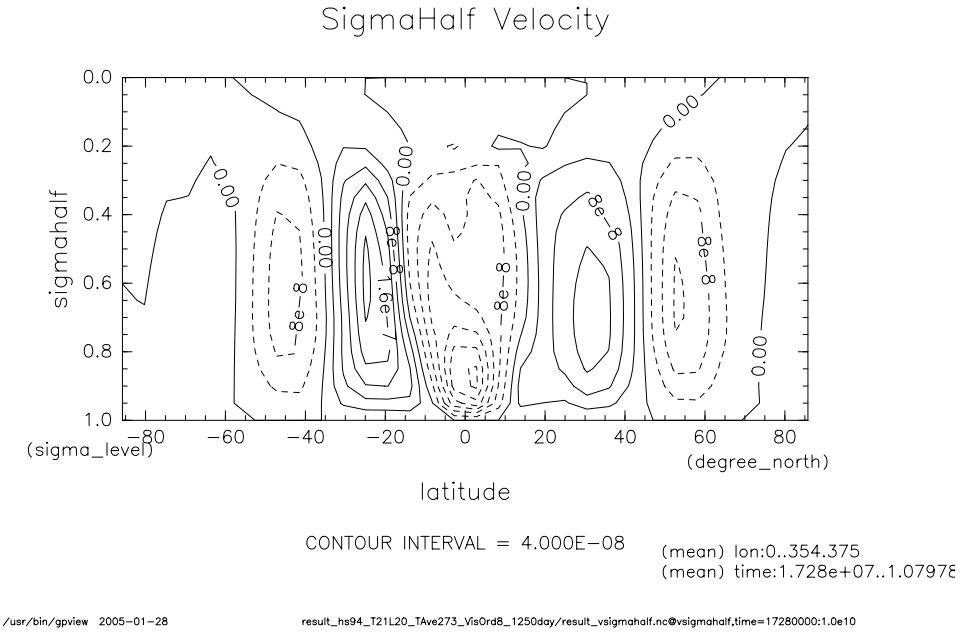


図 16: T21 スペクトルモデルで得られた帯状平均南北風 (単位は m/s). 200 日目から 1200 日目までの平均値. 初期値は等温静止で 1 点に 1 K の擾乱を与えた. 上の温度の初期値は 273 K, 下は 300 K.



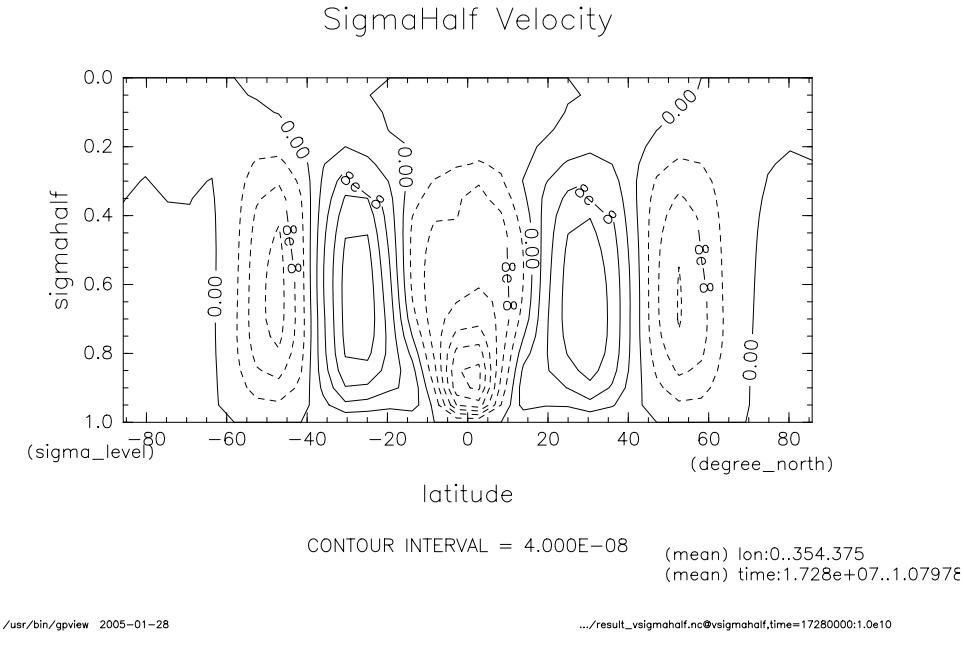
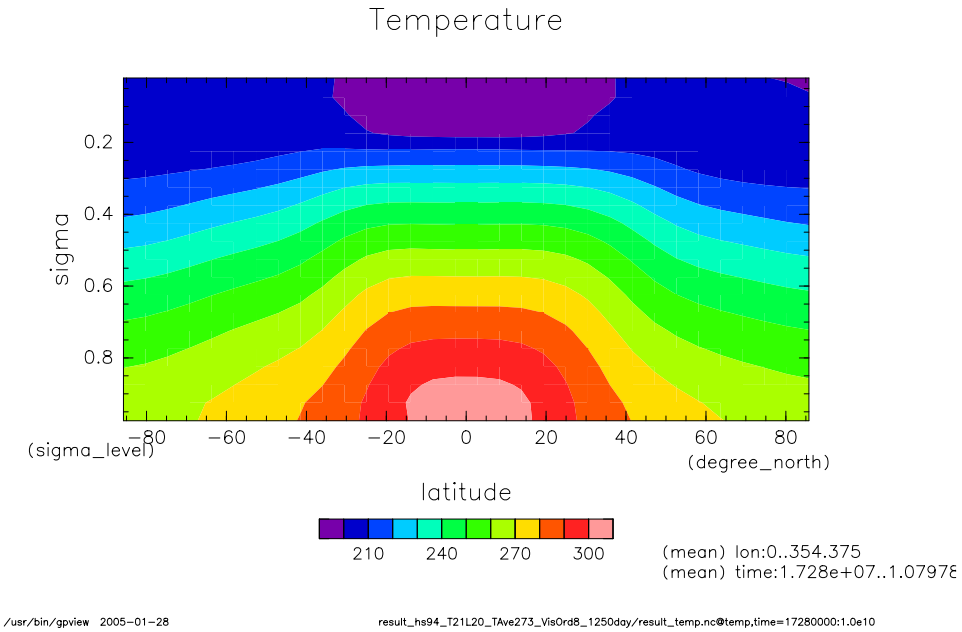


図 17: T21 スペクトルモデルで得られた帯状平均鉛直風 (単位は m/s). $\sigma = 1$ が地上で $\sigma = 0$ が大気上端なので, $\dot{\sigma} > 0$ が下降流, $\dot{\sigma} < 0$ が上昇流である. 200 日目から 1200 日目までの平均値である. 初期値は等温静止で 1 点に 1 K の擾乱を与えた. 上の温度の初期値は 273 K, 下は 300 K.



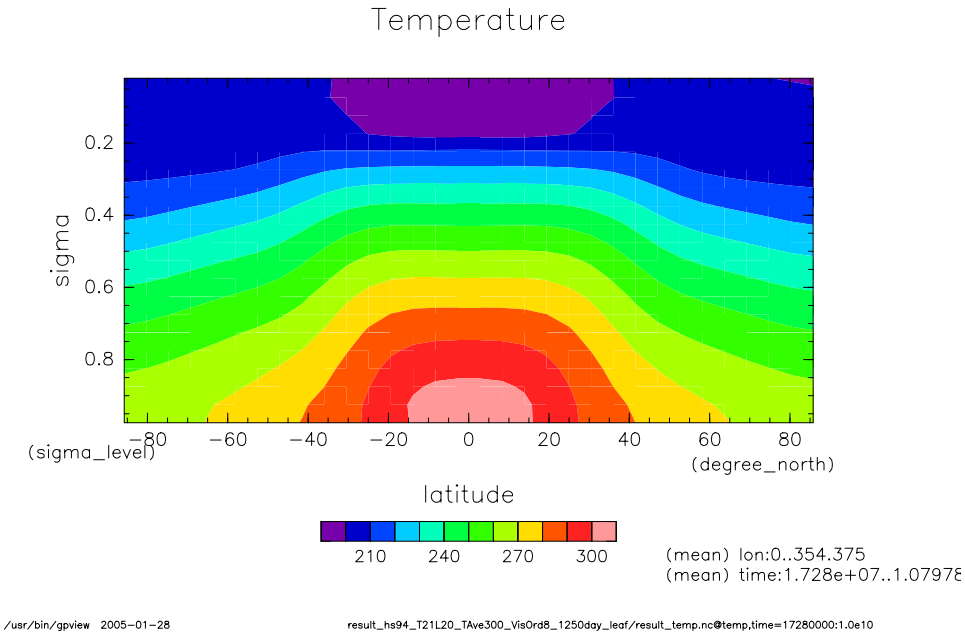
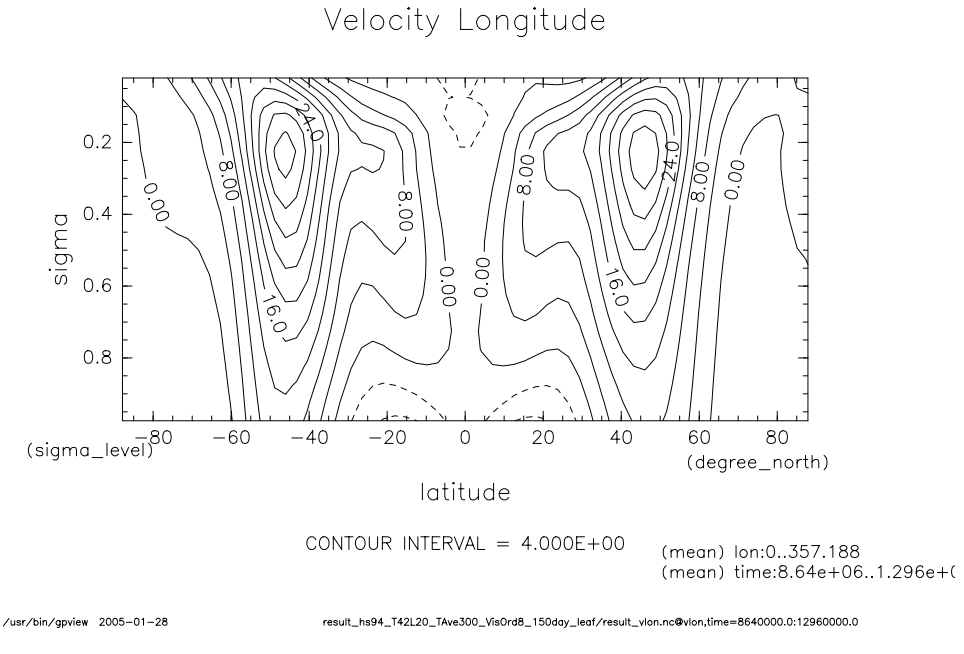


図 18: T21 スペクトルモデルで得られた帯状平均温度 (単位は K). 200 日目から 1200 日目までの平均値である. 初期値は等温静止で 1 点に 1 K の擾乱を与えた. 上の温度の初期値は 273 K, 下は 300 K.



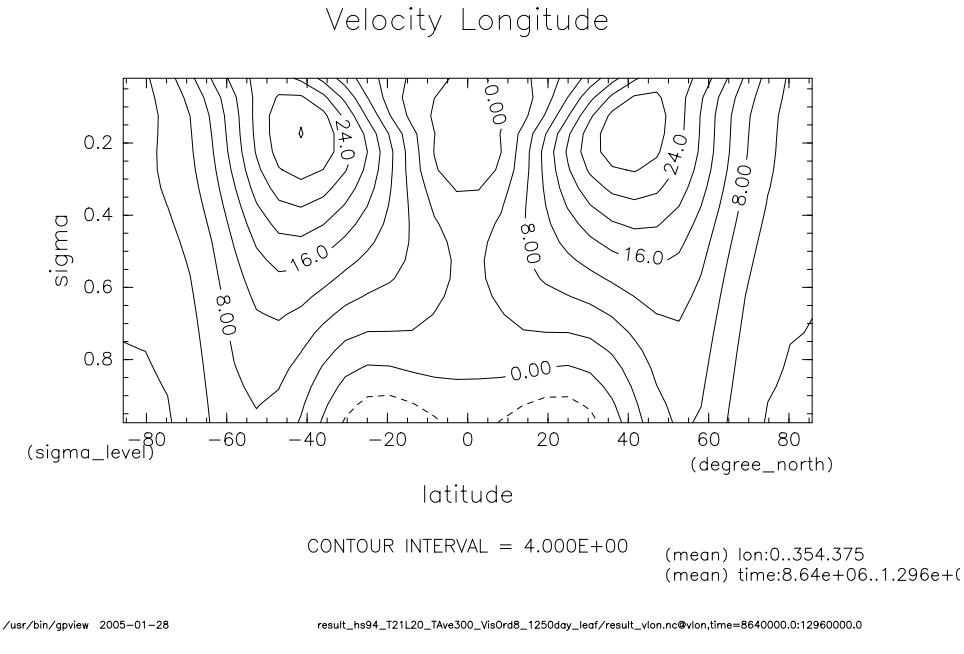


図 19: 上が T42, 下が T21 で得られた帯状平均東西風 (単位は m/s). 100 日目から 150 日目までの平均値である. 初期値は等温静止で 1 点に 1 K の擾乱を与えた. 温度の初期値は共に 300K である.

5　まとめ

プログラム構造の可変性とソースコードの可読性の向上を模索するべく, GCM を新たに設計し, これを DCPAM と名付け, そのプログラム実装と試験計算を行った.

I/O ライブラリとして gt4f90io を整備した. gtool4 netCDF 規約に基づくデータを用いることでそのデータ処理の効率を上げ, 解析のコストを削減することができることが期待される. gt4f90io は Fortran90 の新機能を生かすことで情報隠蔽などを可能としている. そのお陰で I/O のための簡潔なインターフェースを持ったサブルーチンを提供できる. 結果的に, DCPAM の I/O 部分のコードの可読性を高めることができた.

DCPAM のソースコードの可読性を向上するための様々な試みを行った. 1 つ目が変数の命名規則の策定である. Fortran90 になって変数の文字制限がなくなったことを利用し, SPMODEL の命名規則を拡張して変数名自体に依存する次元, 物理量の意味, どの時間のデータであるかという情報をこめることに成功した. 2 つ目は, スペクトル演算に SPMODEL ライブラリを用いたことである. 先の変数の命名規則と合わせることで, スペクトル計算部分のコードが非常に読みやすくなった. 3 つ目は, 気象庁標準コーディングルールの採用である. このコーディングルールに沿ったフォーマットでコードを記述することで, 紛らわしい表記を排除でき, 配列表記など, 読みやすい書式で統一できる. 4 つ目は, The FMS Manual のコーディングルールを模倣し, ファイル名とプログラム名との間に関連性をもたせたことである. このことによって, 複数のファイルをまたいでコーディングを行う際のコストが低減できた.

Fortran90 プログラムのドキュメント生成を容易にする方法を提示した. Fortran90 ソースコードに, RD 書式でコメントを埋め込むことで, Fortran90 ソースファイルからドキュメントを生成できるようになった. さらに, サブルーチンの引用仕様など, ソースからそのままドキュメントにすることで, ドキュメントとソースコードとの乖離を防ぐことが可能になった. また, RD 書式は HTML や TEX や roff など様々な書式に変換可能であるため, 場合に応じたドキュメント形式を作成可能である.

Fortran90 のモジュール機能で DCPAM の内部を階層化することで可変性の向上を試みた. モジュールの初期化・終了処理を統一することでモジュールの交換や分離といった可変性を向上させた. ただし, 力学過程の簡素化を容易にするために必要となる力学コアの階層化は課題として残っている.

DCPAM の力学コアの動作テストとして, Held and Suarez (1994) の力学コアベンチマークテストを行った. 亜熱帯ジェットやハドレー循環など, 期待される現象はある程度表現できたが, Held and Suarez (1994) で示される結果とは差異も多く, その原因を探る必要がある. まずは, 波の活動度の解析や, 設定を変えての再実験をおこなう予定である.

今後の開発計画としては, まず, 上記のような数値モデルのベンチマークテストを続け, GCM としての正当性の検証を行う. 次に, 物理過程の作成を行う. 例えば, 積雲対流過程に関しては湿潤対流調節スキーム, Kuo スキーム, Emanuel スキームなどをそれぞれ 1 つのモジュールとして実装し, モジュールの交換による容易な物理過程の交換が可能かどうかを調べてみる. そして, 鉛直差分を現在の L-grid (Arakawa and Suarez, 1983) に加えて CP-grid (Arakawa and Konor, 1996) の場合についても実装を行い, その鉛直差分の交換がどこまで容易にできるのかを検討する.

謝辞

本論文の作成にあたり, 多くの方々から多大なる御協力と暖かい御声援を頂いたことに深く感謝いたします.

指導教官である北海道大学大学院理学研究科地球惑星科学専攻 林祥介 教授には興味深いテーマを与えて頂きました. また学部時代から合わせて 3 年間, 私の力不足ゆえ遅々としてなかなか研究が進まぬ中, 温かく見守り研究を後押しして下さいました. 同大学地球環境科学研究科 石渡正樹 助手ならびに同大学理学研究科地球惑星科学専攻 小高正嗣 助手には日々叱咤激励していただき, 研究の方針から GCM 開発の詳細に至るまで事細かにアドバイスを頂きました. また本論文の構成や内容に関しても的確なコメントを数多くいただきました. 同専攻 高橋芳幸 博士には, GCM の力学コア開発の際, ソースコードのチェックから結果の解釈に至るまで丁寧に指導して頂くなど大変お世話になり, デバッグに真夜中まで付き合っていただくことも度々ありました. 気象庁予報部数値予報課 豊田英司 博士には gtool4 ツール・ライブラリを通して, Fortran90 に関する様々な知識・技術を学ばせていただきました. 京都大学生存圏研究所 堀之内武 助手には, gt4f90io 開発の際など, 数多くの有用なコメントを頂きました. SPMODEL の開発者である京都大学数理解析研究所 竹広真一 助教授, ISPACK の開発者である京都大学大学院理学研究科地球惑星科学専攻 石岡圭一 助教授, ならびに GFD-DENNOU Library の開発者の皆様には, それぞれソフトウェアの素晴らしい機能を通じて研究を支援していただきました. AGCM5 および GTOOL3 の開発者である故 沼口敦 氏からは, そのモデル・ツールから貴重なアイディアを数多く頂きました.

研究室の先輩である杉山耕一朗さんには Fortran90 によるプログラミングに関して相談に乗っていただいた他, RD によるドキュメントの自動生成に関して, 実際に試して頂き, コメントやアドバイスを頂きました. 同じく研究室の先輩である山田由貴子さん, ならびに同期の塚原大輔さんには解析や可視化のノウハウを数多く丁寧に教えていただきました. また, 研究室や同専攻 惑星物理学研究室の皆様には, ゼミや日頃の会話や議論を通じて様々なことを考えさせて頂きました. また日々の研究や論文の作成にあたっては暖かい声援を送って頂きました. この場を借りて深くお礼申し上げます.

本論文における解析・可視化ツールには電脳 Ruby ツールを用いました. これらの大変に便利なツール群を開発・整備している皆様に感謝いたします. また文書整形には pLaTeX を, スタイルファイルとして地球流体電脳倶楽部で開発されたマクロ定義スタイルファイル集 Dennou Style 6 [†4] を利用させて頂きました.

[†4] http://www.gfd-dennou.org/arch/cc-env/TeXmacro/dennou/SIGEN.htm

参考文献

地球流体電脳倶楽部, 1995: 格子点データ解析ツール GTOOL3 利用の手引 ver. 1.01b http://www.gfd-dennou.org/arch/gtool/doc/gtool.pdf

Arakawa, A., Suarez, M. J., 1983: Vertical differencing of the primitive equations in sigma coordinates. *Mon. Wea. Rev.*, **111**, 34–35.

Arakawa, A., and Konor, M., J., 1996: Vertical differencing of the primitive equations based on the Charney-Phillips grid in hybrid $\sigma - p$ vertical coodinates. *Mon. Wea. Rev.*, **124**, 511–528.

Asselin, R. A., 1972: Frequency filter for time integrations. *Mon. Wea. Rev.*, **100**, 487–490.

Balaji, V., 2002: The FMS Manual: A developer's guide to the GFDL Flexible Modeling System.
http://www.gfdl.noaa.gov/~vb/FMSManual/FMSManual.html

Berliner, B., 1994: CVS: Concurrent Versions System.
https://www.cvshome.org.

Cooperative Ocean/Atmosphere Research Data Service, 1995: Conventions for the st andardization of NetCDF files.
http://ferret.wrc.noaa.gov/noaa_coop/coop_cdf_profile.html

National Center for Atmospheric Research, 1997: NCAR-CSM NetCDF Conventions http://www.cgd.ucar.edu/cms/eaton/netcdf/NCAR-CSM.html

GFDL, 2005: FMS: The flexible modeling system.
http://www.gfdl.noaa.gov/~fms/.

GFD-DENNOU Club, 2004: GFD-DENNOU Library 5.3,
http://www.gfd-dennou.org/arch/dcl.

Held, I. M., 2004: The Gap Between Simulation and Understanding in Climate Modeling. http://www.gfdl.gov/~ih/papers/21.pdf.

Held, I. M., and Suarez, M. J., 1994: A proposal for the intercomparison of the dynamical cores of atmospheric general circuation models. *Bull. Am. Meteor. Soc.*, **75**, 1825–1830.

GFD-Dennou Club, 2003: Dennou Ruby Project.
http://www.gfd-dennou.org/arch/ruby

Ishioka, K., 2002: ispack-0.61, http://www.gfd-dennou.org/arch/ispack/, GFD Dennou Club.

Matsumoto, Y., 2001: The Object-Oriented Scripting Language Ruby. http://www.ruby-lang.org.

Morikawa, Y., Odaka, M., Ishiwatari, M., Hayashi, Y.-Y., Gtool4 Devlopment Group, 2004: gt4f90io Fortran90 netCDF I/O library with gtool4 conventions, http://www.gfd-dennou.org/arch/gtool4/, GFD Dennou Club.

Muroi, T., Toyoda, E., Yoshimura, Y., Hosaka, M., Sugi, M., 2002: 気象庁標準コーディングルール. 天気, **49**, 91–95 (in Japanese), http://www.mri-jma.go.jp/Project/mrinpd/coderule.html.

Robert, A. J., 1966: The integration of a low order spectal form of the primitive meteorological equations. *J. Meteor. Soc. Japan*, **44**, 237–245.

Ruby Documentation Project, 2005: http://www.rubyist.net/~rubikitch/RDP.cgi?cmd=view;name=top.

Rew, R., Davis,G., Emmerson, S., and Davies, H., 1997: netCDF User's Guide For Fortran, version 3. http://www.unidata.ucar.edu/packages/netcdf/guidef

Sugiyama, K., Kitamori, T., Odaka, M., and Nakajima K., and Hayashi, Y-Y., 2004: deepconv/arare, http://www.gfd-dennou.org/arch/oboro, GFD-Dennou Club

SWAMP Project, 1998: AGCM5, http://www.gfd-dennou.org/arch/agcm5, GFD-Dennou Club

Takehiro, S., Ishioka, K., Morikawa, Y., Odaka, M., Ishiwatari, M., Hayashi, Y.-Y., SPMODEL Development Group, 2004: Hierarchical Spectral Models for GFD (SPMODEL), http://www.gfd-dennou.org/arch/spmodel/, GFD Dennou Club.

Toyoda, E., Ishiwatari, M., Horinouchi, T., Akahori, K., Numaguti, A., Hayashi, Y.-Y., GFD Dennou Club Davis Project, 2000: gtool4 netCDF convention, http://www.gfd-dennou.org/arch/gtool4/, GFD Dennou Club.

Toyoda, E., Ishiwatari, M., Takehiro, S., Hayashi, Y.-Y., gtool4 Devlopment Group, 2002: gtool4 Fortran90 Tools/Library, http://www.gfd-dennou.org/arch/gtool4/, GFD Dennou Club.