http://www.gfd-dennou.org

可変性と可読性を考慮した大気大循環モデルの開発手法の考案と実装試験

森川 靖大
北海道大学 理学院 宇宙理学専攻
博士後期課程 3 年

目次

- はじめに
  - 大気大循環モデルとは
  - なぜ可変性と可読性か
  - 何が問題か
  - 可変性と可読性を考慮したこれまでの試み
  - 問題解決の方針
- dcmodelプログラミングガイドライン
- データ入出力ライブラリによるソースの簡素化と統一
- モジュール設計の工夫
- RDoc による解説文書の自動生成
- プログラムのテスト実行の自動化
- モジュール雛形作成ツールの整備
- モデルの実装の現状
- 今後の計画

大気大循環モデルとは

- 惑星大気に関して 3 次元の流体計算を行い、全球規模の大気の運動を予測、再現するプログラム
- 力学過程と物理過程で構成
  - 力学過程: 流体の流れの場 (静水圧近似) の計算
  - 物理過程: 放射、積雲など
- プログラミング言語は大抵 Fortran
  - スパコンで高速に動作させるため
  - 過去のプログラム資源を活用するため
- 実行速度向上のためのチューニングが施される
  - 計算時間が (大抵) 数時間〜数週間かかるため
- ソースコードはおおむね数万〜数十万行
- 世の中にはいろいろな大気大循環モデルがある
  - AFES、CCSR/NIES AGCM、PRISM、UCAR CAM …
  - 電脳倶楽部AGCM5、GFDL FMS、WRF、CCSM …
    - これらはフリーで手に入れられる

なぜ可変性と可読性か

- 様々な (現実的 and 仮想的) 惑星大気循環を手軽に計算できるようにしたい

http://earthobservatory.nasa.gov/Study/LivingEarth/

http://www.solarviews.com/raw/jup/jupiter.gif

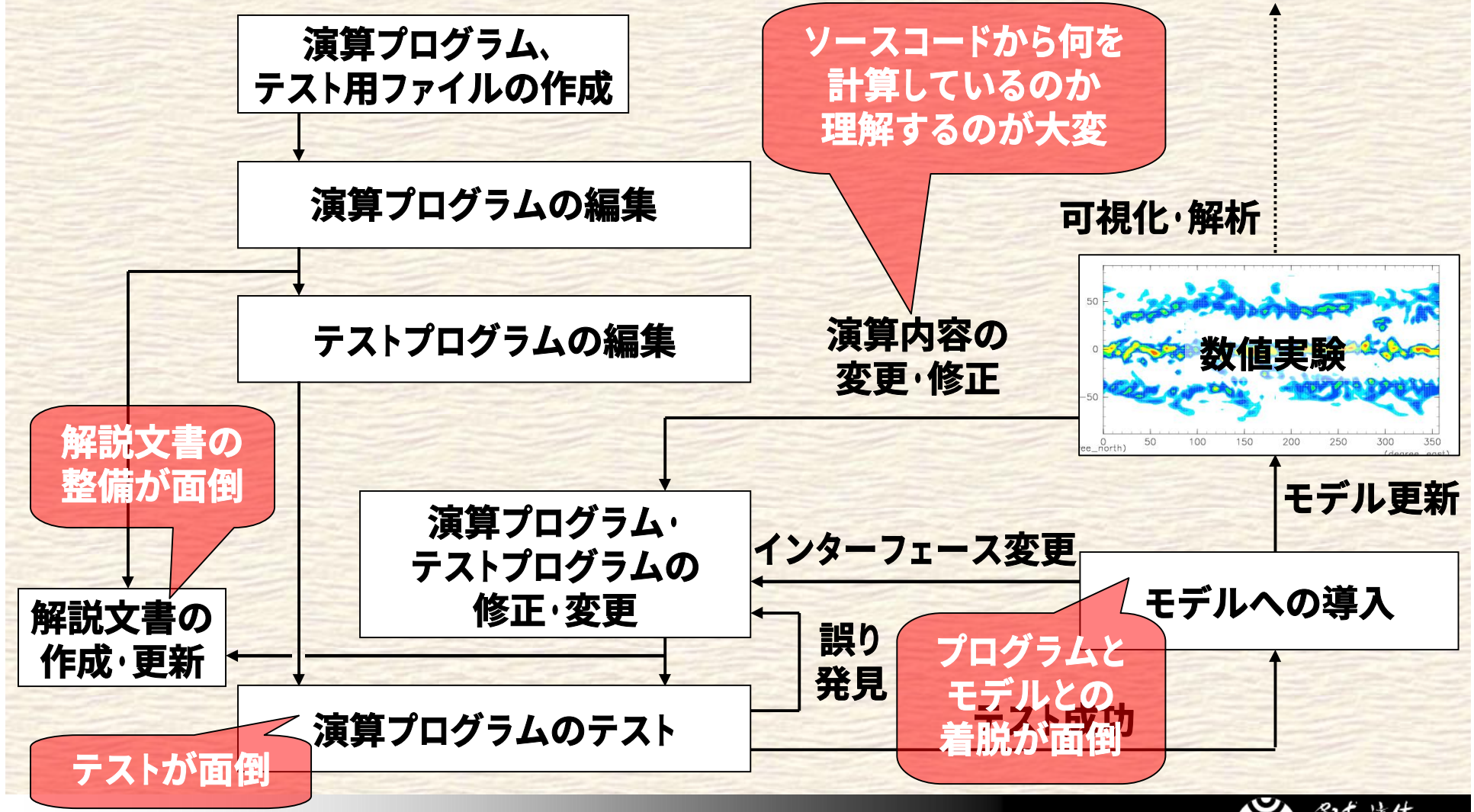
- 一般に仮想惑星大気計算のためには
  - 計算条件を手軽に変更
    - 大気組成、入射太陽放射量、重力加速度、大気圧、自転周期 etc.
  - 可変性・可読性に優れた大気大循環モデル
    - 何を計算しているか、ソースコードを読んで分かる
    - スキームの交換や分離が容易にできる

何が問題か

■ モデルの開発・整備と数値実験の作業の流れ

何が問題か

■ モデルの開発・整備と数値実験の作業の流れ



はじめに:
可変性と可読性を考慮したこれまでの試み (1)

- **AGCM5 (沼口, 1992; SWAMP Project, 1998)**
  - http://www.gfd-dennou.org/library/agcm5
  - FORTRAN 77
  - **変数命名規則・プログラム書法の工夫**
    - 「AGCM5 マニュアル第3部 コード解説」に (一部は) まとめられている
  - **例**

```
      CALL SMTV2S ( MMAX, IMAX, IDIM, JMAX, JDIM, KMAX,
     &              U, V, DIV, VOR, WORK,IT, T, IP, P, QUSDER, R, ML)
```

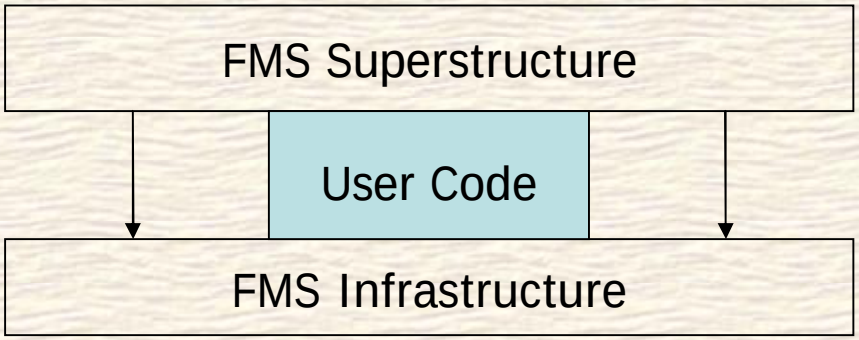
```
      CALL SMTV2S
     I      ( MMAX , IMAX , IDIM, JMAX, JDIM, KMAX,
     I        GTUA , GTVA ,
     O        WTDIV, WTVOR,
     W        WORK,
     I        IT, T, IP, P,QUSDER, R, ML)
```

物理量を表す変数の命名規則
・格子点か波数空間か ⇒ G or W
・どの時間ステップの値か⇒
B (t-Δt), A(t+Δt), T (**時間変化**)

引数の入出力属性の記述

はじめに: 可変性と可読性を考慮したこれまでの試み (2)

- **FMS (Flexible Modeling System; GFDL, 2005)**
  - http://www.gfdl.noaa.gov/fms/
  - Fortran 90/95
  - I/O, **並列化部分を** Infrastructure、**モデル (大気, 海洋等) 結合部分を** Superstructure **へ隠蔽**
    - The FMS Manual **にまとめられている**
      - http://www.gfdl.noaa.gov/~vb/FMSManual/FMSManual.html
- **研究を行う各ユーザは User Code 部分を編集**



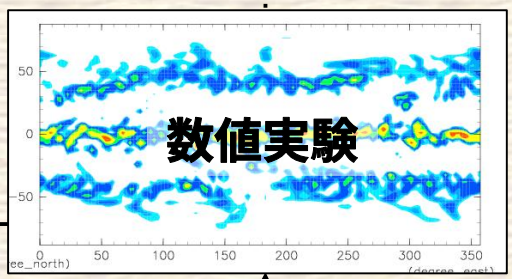
はじめに:
可変性と可読性を考慮したこれまでの試み (3)

- **AGCM5 の変数命名規則とプログラム書法**
  - FORTRAN 77 の限界
    - 変数の 6 文字ルール
    - 配列演算に関数使用不可
    - WORK 領域の指定
    - 配列サイズ要指定

ソースコードから何を計算しているのか理解するのが大変

演算内容の変更・修正

可視化・解析



モデル更新

インターフェース変更

モデルへの導入

プログラムとモデルとの着脱が面倒

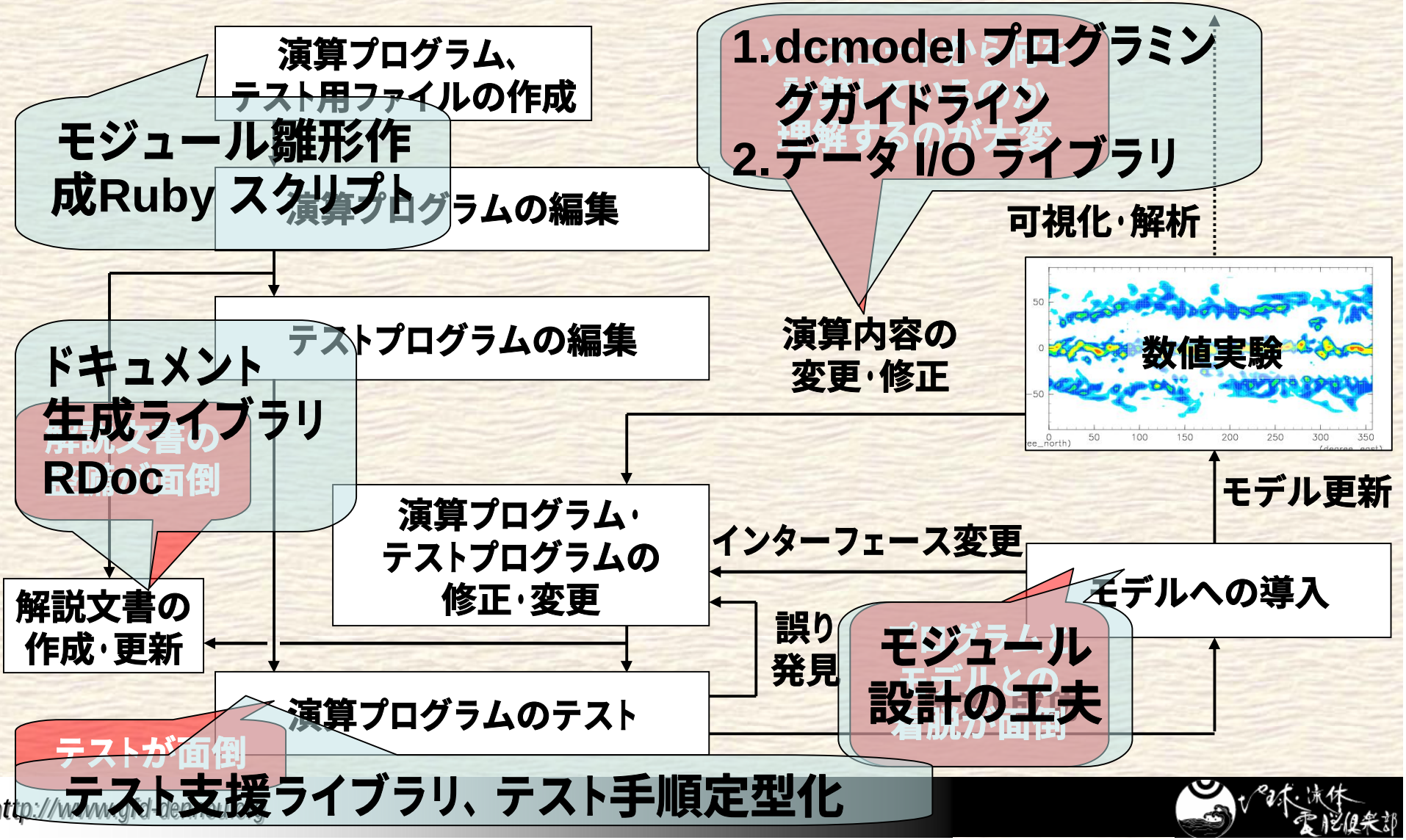
解説文書の整備が面倒

- **FMS モデリングシステム**
  - User code 内部でのプログラムの着脱についてより工夫したい
    - 例: 放射や積雲パラメタリゼーションなどのスキーム

テストが面倒

問題解決の方針

■ モデルの開発・整備と数値実験の作業の流れ



目次

- はじめに
- dcmodelプログラミングガイドライン
- データ入出力ライブラリによるソースの簡素化と統一
- モジュール設計の工夫
- RDoc による解説文書の自動生成
- プログラムのテスト実行の自動化
- モジュール雛形作成ツールの整備
- モデルの実装の現状
- 今後の計画

dcmodel プログラミングガイドライン

- http://www.gfd-dennou.org/library/dcmodel/
- 以下を目的に策定している (しつつある) Fortran 90/95 ソースコード書法のガイドライン
  - 読みやすいソースコードを作成すること
  - 異なる数値モデル間でのソースコードの流用･交換を容易にすること
- ソースコード中の演算実行部分からどのような計算を行っているか, ひいては元の支配方程式がどのようなものかを比較的簡単に想像できるようにする

例題

- **GCM で使用される渦度方程式 (多少簡略化)**

$$\frac{\partial \zeta(t)}{\partial t} = \frac{1}{a\cos\varphi}\left(\frac{\partial V_A(t)}{\partial \lambda} - \frac{\partial (v\cos\varphi) U_A(t)}{\partial \varphi}\right)$$

$$\zeta(t+\Delta t) = \zeta(t-\Delta t) + 2\Delta t \times \frac{\partial \zeta(t)}{\partial t}$$

λ ：経度
φ ：緯度
σ ：圧力/地表面圧力
t ：時刻
a ：惑星半径
f ：コリオリパラメータ

$\zeta\ (\lambda, \varphi, \sigma, t)$ ：渦度
$U_A\ (\lambda, \varphi, \sigma, t) = (\zeta + f)\ v \cos\varphi$
$V_A\ (\lambda, \varphi, \sigma, t) = -(\zeta + f)\ u \cos\varphi$
$u\quad (\lambda, \varphi, \sigma, t)$ ：東西風速
$v\quad (\lambda, \varphi, \sigma, t)$ ：南北風速

AGCM5 (F77) でのプログラム例


```
      CALL SMTV2S
     I      ( MMAX , IMAX , IDIM, JMAX, JDIM, KMAX,
     I        GTUA , GTVA ,
     O        WTDIV, WTVOR,
     W        WORK,
     I        IT, T, IP, P, QUSDER, R, ML)

      CALL SMTG2S
     I      ( MMAX , IMAX, IDIM, JMAX, JDIM, KMAX,
     I        GBVOR,
     O        WBVOR,
     W        WORK,
     I        IT, T, IP, P, QGS )

      DO 7100  K = 1 , KMAX
         DO 7100  NM = 1 , NMDIM
            WAVOR( NM,K ) =  WBVOR( NM,K ) + WTVOR( NM,K )*DELT2
7100 CONTINUE

      CALL SMTS2G
     I      ( MMAX, IMAX, IDIM, JMAX, JDIM, KMAX,
     I        WAVOR,
     O        GAVOR,
     W        WORK,
     I        IT, T, IP, P, QGS )
```

$$\frac{\partial \zeta(t)}{\partial t} = \frac{1}{a\cos\varphi}\left(\frac{\partial VA(t)}{\partial \lambda} - \frac{\partial (v\cos\varphi)UA(t)}{\partial \varphi}\right)$$

$$\zeta(t+\Delta t) = \zeta(t-\Delta t) + 2\Delta t \times \frac{\partial \zeta(t)}{\partial t}$$

- **工夫**
  - 変数命名規則
  - プログラム書法
- **可読性低下の原因**
  - サブルーチンコール
  - 配列サイズ情報
  - WORK 領域
  - 6 文字制限

階層的モデル群でのプログラム例

- 数式に近い形でソースコードを記述可能に

$$\frac{\partial \zeta(t)}{\partial t} = \frac{1}{a\cos\varphi}\left(\frac{\partial V_A(t)}{\partial \lambda} - \frac{\partial (v\cos\varphi) U_A(t)}{\partial \varphi}\right)$$

$$\zeta(t+\Delta t) = \zeta(t-\Delta t) + 2\Delta t \times \frac{\partial \zeta(t)}{\partial t}$$

```
wz_DVorDtN =                                                  &
       & wa_Div_xya_xya( xyz_VaN ,  - xyz_UaN )  &
       & / Rplanet
xyz_VorA    =                                                 &
       &  xya_wa( wa_xya( xyz_VorB )                          &
       &           + 2. * DelTime * wz_DVorDtN )
```

配列演算関数 (F90/95) の活用

- Fortran 90/95 では配列の演算を一行で記述可能
  - このような、配列を返す関数をユーザも定義可能

```
! FORTRAN 77 style
      REAL*8    A(10,10), B(10,10)
      INTEGER   I, J
  ...
      DO 1000 J=1,10
          DO 1000 I=1,10
              B(I,J) = EXP( A(I,J) )
 1000 CONTINUE
```

```
! Fortran 90/95 style
real(8) :: a(10,10), b(10,10)
...
b = exp(a)
```

変数・関数命名規則

- **変数**
  - (次元の接頭詞)_(物理的意味)(時間方向添字) のように書く
    - 経度 (1 次元) x_Lon
    - 温度 (3 次元, 格子点) 時刻 $t-\Delta t$ xyz_TempB
    - 渦度 (1 次元, スペクトル) 時刻 $t+\Delta t$ w_VorA
- **配列関数**
  - (出力型)_(機能)_(入力型) のように書く
    - スペクトル変換 w_Vor=w_xy(xy_Vor)
    - 逆変換 xy_Vor=xy_w(w_Vor)
    - 発散 w_Div=w_Div_xy_xy(xy_U, xy_V)
  - 引数の型を間違えることが少ない

目次

- はじめに
- dcmodelプログラミングガイドライン
- データ入出力ライブラリによるソースの簡素化と統一
- モジュール設計の工夫
- RDoc による解説文書の自動生成
- プログラムのテスト実行の自動化
- モジュール雛形作成ツールの整備
- モデルの実装の現状
- 今後の計画

データ入出力のプログラムを書く際の問題点

- ソースコードが煩雑になる
  - ファイルID や 変数ID の管理
- ソースコードの書き方が揃わない
  - 同じ形式のファイル出力に対して様々な書き方がある

gt4f90io ライブラリ

- **gtool4 規約に基づく Fortran90 netCDF I/O ライブラリ**
  - gtool4 netCDF **規約に基づくデータを入出力するためのライブラリ**
  - Fortran 90/95 **で書かれたプログラムで使用**
  - **データ入出力のための簡潔なインターフェース (サブルーチン群) を提供**
  - **その他、汎用なライブラリも同梱**
    - **文字列処理、デバッグ支援、CPU時間計測、メッセージ出力、コマンドライン引数の処理、テストプログラム作成支援** etc...

gt4_history

- gt4f90io ライブラリに含まれるデータI/O用モジュール
- 最低限 5 つのサブルーチンでデータの入出力が可能

- HistoryCreate(file, title, ...)
  - 初期設定
    - 出力ファイル名、タイトル、...、次元変数名、次元サイズ、...
- HistoryAddVariable(varname, dims, ...)
  - 変数定義
    - 変数名、依存次元名、...
- HistoryPut(varname, value, ...)
  - 変数出力
    - 変数名、出力値、...
- HistoryClose
  - 終了処理
- HistoryGet(file, varname, ...)
  - 変数入力
    - ファイル名、変数名、...

違うデータ型も
同サブルーチンで対応

gt4_history 使用例

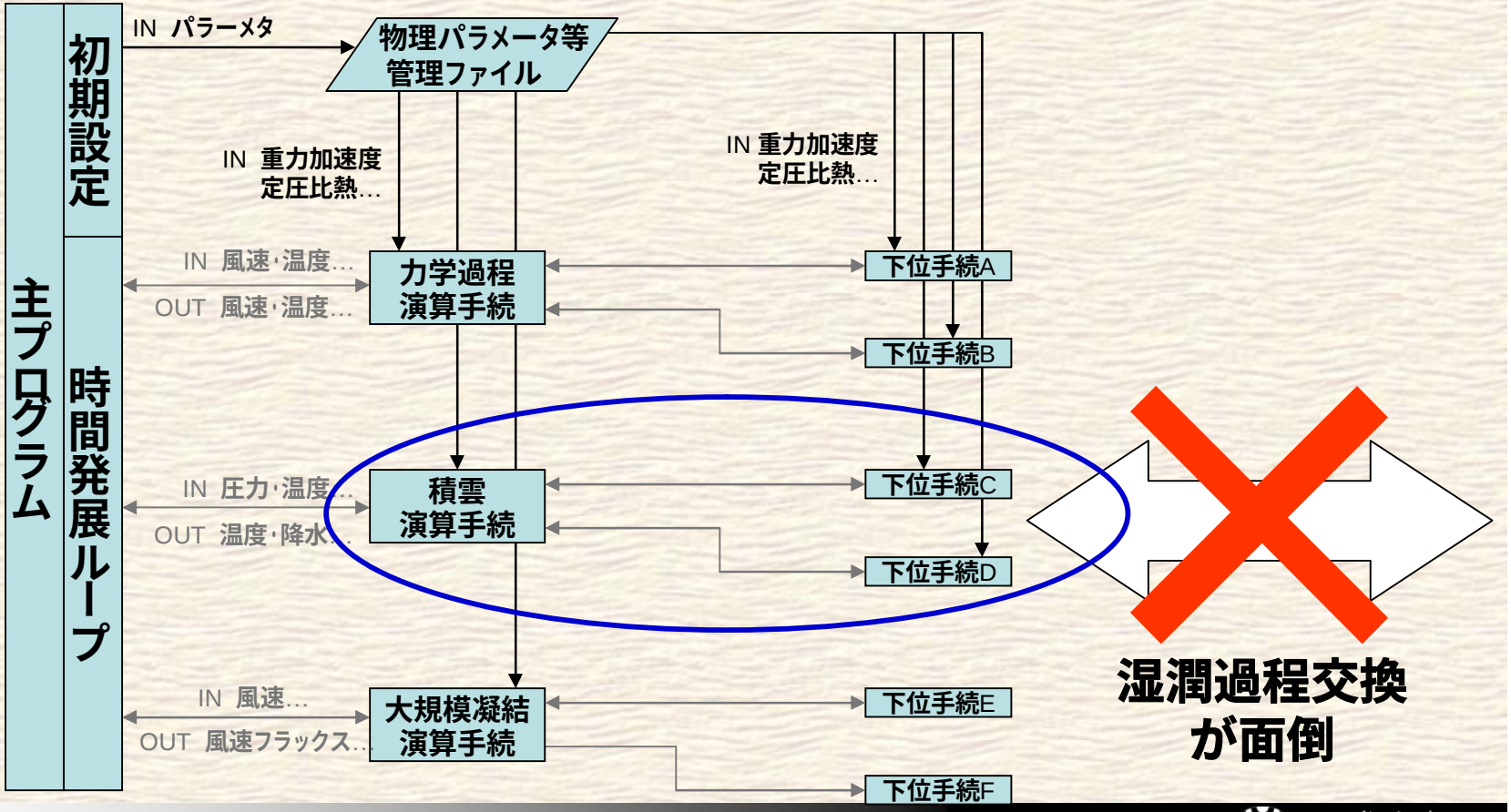

```
program sample
  use gt4_history                                          ! モジュールの使用を宣言
  [型宣言] ......
  call HistoryGet( file='init.nc' ... )                    ! 初期値入力

  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、..

  [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
```

目次

- はじめに
- dcmodelプログラミングガイドライン
- データ入出力ライブラリによるソースの簡素化と統一
- モジュール設計の工夫
  - AGCM5 (FORTRAN 77) のプログラム構造
  - モジュール設計の見直し
  - モジュール実装例
- RDoc による解説文書の自動生成
- プログラムのテスト実行の自動化
- モジュール雛形作成ツールの整備
- モデルの実装の現状
- 今後の計画

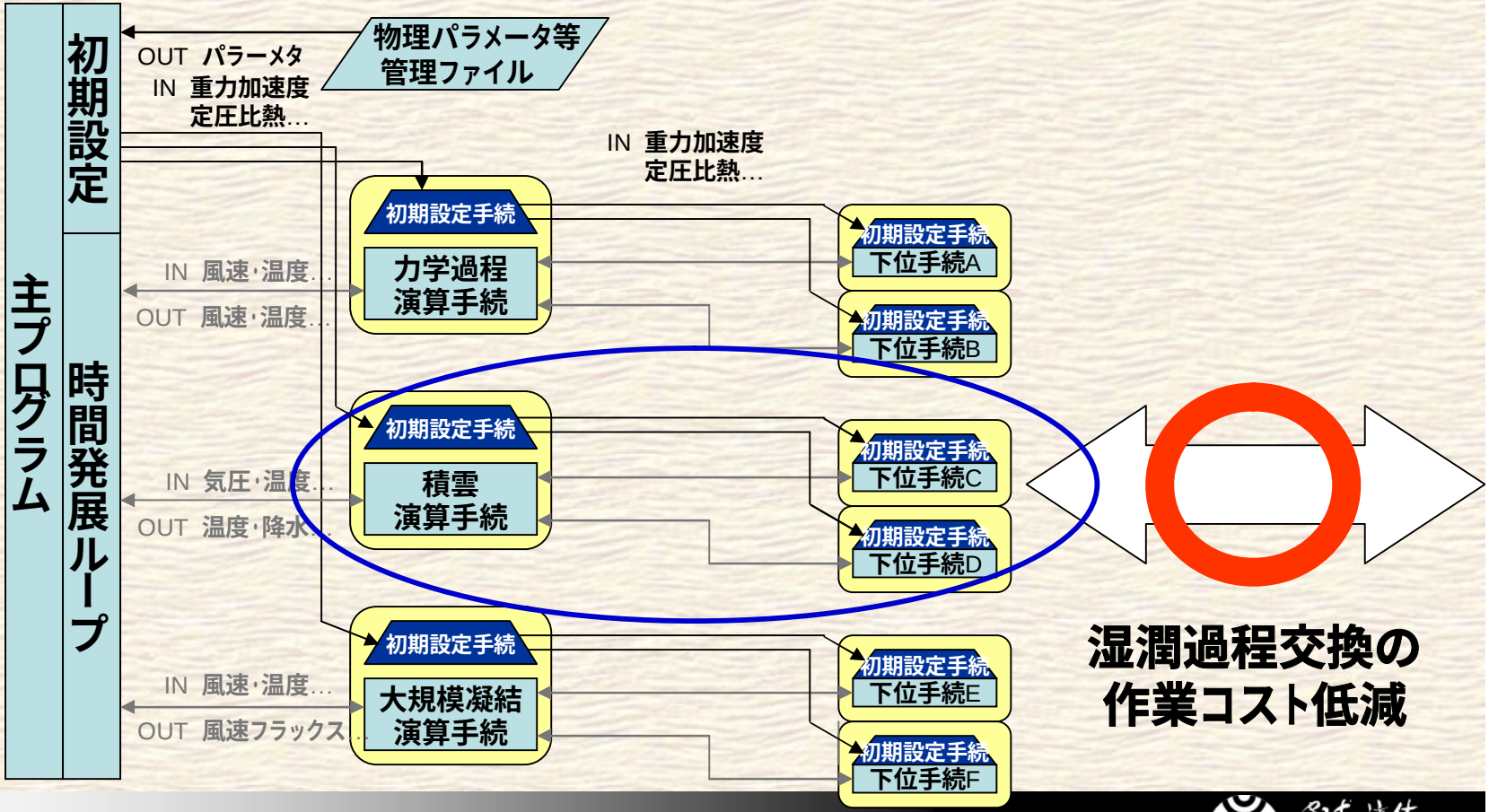
AGCM5 (FORTRAN 77) のプログラム構造

- 個々の演算で必要なパラメータを1つのファイルで集中管理する方法の問題点
  - モデルの一部を交換にはソースコード解読が必要



モジュール設計の見直し

- 今回の試み：個々の演算で必要なパラメータはモジュール毎に保持
  - 各モジュールで初期設定手続を用意し、その手続でパラメータを設定
  - スキーム交換に必要な情報が初期設定手続と演算手続の引数として集約



モジュール実装例

- 複数の積雲パラメタリゼーションの実装の場合
  - 異なるスキーム毎に異なるモジュール
    - 対流調節スキーム: phy_cumulus_adjust
    - Kuo スキーム: phy_cumulus_kuo
  - それぞれのスキームの使い方は同じになるよう実装
    - 総称手続きを用い、初期設定手続と演算手続の名称はそれぞれ Create および Cumulus
      - Create には重力加速度、気体定数、定圧比熱、 ...
      - Cumulus には気圧、温度、比熱、降水量、...
- スキーム交換に必要な情報が初期設定手続と演算手続の引数として集約
  - 初期設定: 重力加速度、気体定数、定圧比熱、 ...
  - 演算: 気圧、温度、比熱、降水量、...

目次

- はじめに
- dcmodelプログラミングガイドライン
- データ入出力ライブラリによるソースの簡素化と統一
- モジュール設計の工夫
- RDoc による解説文書の自動生成
  - 解説文書の重要性と問題点
  - RDoc とは
  - RDoc Fortran 90/95 強化版によるドキュメントの自動生成
- プログラムのテスト実行の自動化
- モジュール雛形作成ツールの整備
- モデルの実装の現状
- 今後の計画

解説文書の重要性と問題点

- ここで言う解説文書とは
  - サブルーチンなどの使い方を記した文書
    - 内容：名称、必要な引数、引数の型、概要、etc...
- 解説文書の利便性
  - ソースコードを読まずにプログラムの交換が可能に
  - 他人とプログラムの共有を行う際にも必須
- 解説文書の維持・更新のコスト高
  - モデルを構成するプログラムは頻繁に交換・変更されることを想定
  - セットとなる解説文書も同時に手動で維持・更新するのは面倒
    - ソースコードの書き換えだけで十分大変
    - ソースコードの編集と似て非なる作業を再度行うのは苦痛

RDoc とは

- オブジェクト指向スクリプト言語 Ruby で書かれた、ソースコードからドキュメントを自動生成するライブラリ (Ruby の標準ライブラリの1つ)
- Fortran 90/95 の解析も可能
  - ソースコード解析機構とマニュアル生成機構が分離しているため、他の言語で書かれたソースコードも解析可能
  - 標準で C および Fortran95 用の解析機構が付属
  - 当初は解析機能が不十分で実用に耐えなかった

RDoc Fortran 90/95 強化版による ドキュメントの自動生成

- **Fortran 90/95 の解析機能を強化**
  - サブルーチンや関数の引数、総称名称なども解析
  - MathML の利用により数式も表現可能

```
module phy_cumulus_adjust
  != 積雲パラメタリゼーション:
  !  対流調節スキーム
  !== Prodedures list
  ! Create        :: 初期設定
  ! Calculation   :: 演算
                :
contains
                :
 subroutine PhyCumulusAdjustCreate( &
    & phy_cum_ad, &
    & Grav, RAir, Cp, ... )
                :
 end subroutine PhyCumulusAdjustCreate
                :
end phy_cumulus_adjust
```

目次

- はじめに
- dcmodelプログラミングガイドライン
- データ入出力ライブラリによるソースの簡素化と統一
- モジュール設計の工夫
- RDoc による解説文書の自動生成
- プログラムのテスト実行の自動化
- モジュール雛形作成ツールの整備
- モデルの実装の現状
- 今後の計画

プログラムのテスト実行の半自動化(1)

- 個別の演算プログラムに対してのテスト
  - 初期設定手続に与えられたパラメタがモジュール内で正しく設定されているか「答えあわせ」する
  - 積雲や大規模凝結などに関する各スキームがそれぞれ予期された計算をおこなっているか「答えあわせ」する
- テストを行う上での問題点
  - テストの実行が面倒
    - 可視化や解析の作業を定常的に行うのは面倒
  - テストプログラム整備が面倒
    - 配列同士の比較など、コーディングにかかる手間が大きい

プログラムのテスト実行の半自動化(2)

- テストプログラムのコードを簡素化
  - dc_test モジュールによる多次元配列比較コード統一
    - 組込み型変数･配列 (1～7次元) に関して 2 つの引数を比較 (大小、等しいかどうかのチェック) するためのサブルーチンを用意
    - 値が異なる場合には両者の値と配列内での位置を出力して終了
- テスト実行手順の定型化
  - 各モジュールにテストプログラムを作成
    - モジュール (hoge.f90) に対して
      テストプログラム (hoge_test.f90) と
      テスト実行シェルスクリプト (hoge_test.sh) を作成
  - Makefile を整備し、make test コマンドでテストを実行

目次

- はじめに
- dcmodelプログラミングガイドライン
- データ入出力ライブラリによるソースの簡素化と統一
- モジュール設計の工夫
- RDoc による解説文書の自動生成
- プログラムのテスト実行の自動化
- モジュール雛形作成ツールの整備
- モデルの実装の現状
- 今後の計画

モジュール雛形作成ツールの整備

- Fortran ファイルやシェルスクリプトを自動生成する Ruby スクリプトを準備
- 積雲対流スキームの場合に生成されるファイルの例
  - モジュール
    - ファイル名: phy_cumulus_adjust.f90
    - 積雲対流スキームを実際に計算するための Fortran ファイル
  - テストプログラム
    - ファイル名: phy_cumulus_adjust_test.f90
    - 上記モジュールを読み込み、テストを実行する Fortran ファイル
  - テスト実行シェルスクリプト
    - ファイル名: phy_cumulus_adjust_test.sh
    - 上記テストプログラムの実行を行うシェルスクリプト

モジュール雛形作成ツールの使い方

- プログラムを実行するだけ
- モジュール名、引数キーワードなどに対して対話的に入力

使用例

```
$ make template

ruby dcmodel_f90sample_maker.rb  -E module
 Input Module name : phy_cumulus
 Title of module (for English) : Cumulus scheme
 Title of module (for Japanese) : 積雲スキーム
 Input basename [PhyCumulus]:
 Input arg_type [PHYCUM]:
 Input arg_keyword [phy_cum]:
 Input Your name [unknown]: Yasuhiro MORIKAWA
 Input Copyright [GFD Dennou Club]:

 Message: phy_cumulus.f90 is generated ... done.
 Message: phy_cumulus_test.f90 is generated ...
 Message: phy_cumulus_test00.nml is generated ...
 Message: phy_cumulus_test.sh is generated ... d
```

雛形生成 Ruby スクリプト

入力項目

モジュール
phy_cumulus.f90

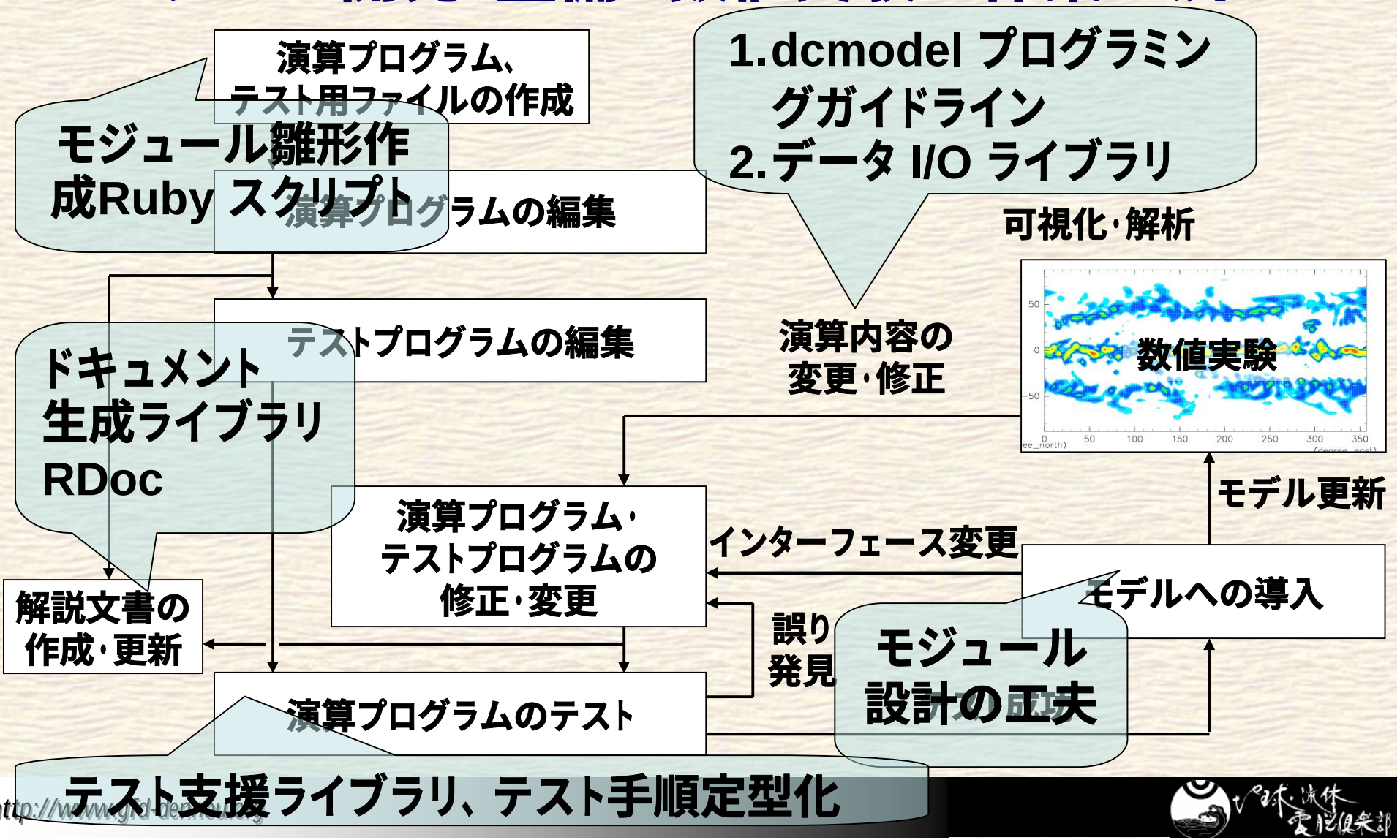
テストプログラム
phy_cumulus_test.f90

NAMELIST ファイル
phy_cumulus_test00.nml

テスト実行シェルスクリプト
phy_cumulus_test.sh

開発・整備の作業コスト低減

■ モデルの開発・整備と数値実験の作業の流れ



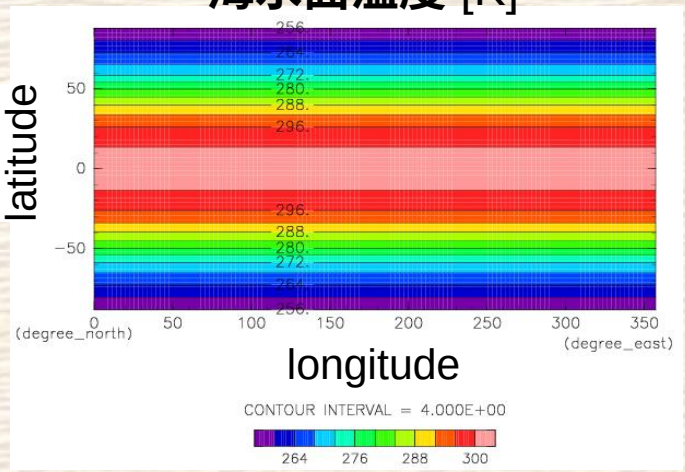
目次

- はじめに
- dcmodelプログラミングガイドライン
- データ入出力ライブラリによるソースの簡素化と統一
- モジュール設計の工夫
- RDoc による解説文書の自動生成
- プログラムのテスト実行の自動化
- モジュール雛形作成ツールの整備
- モデルの実装の現状
- 今後の計画

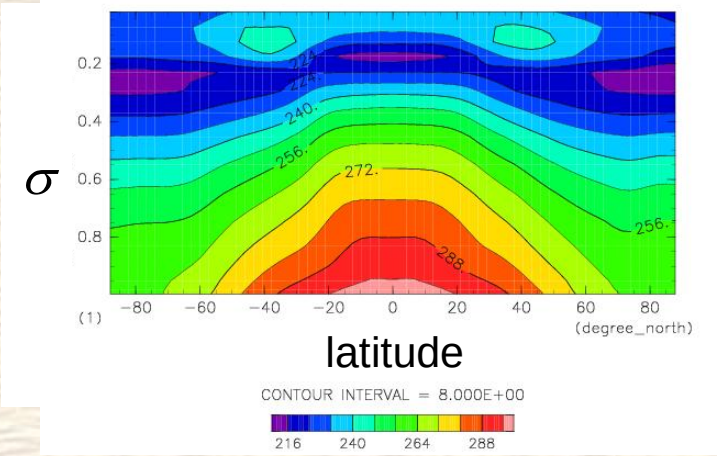
モデルの実装の現状

- **以下の物理過程をモジュール化して実装**
  - **湿潤対流調節**
  - **大規模凝結**
  - **放射過程** (4**色バンドモデル**)
  - **鉛直拡散** (Mellor Yamada, Level 2)
  - **地表面フラックス** (**バルク法**)
- **水惑星実験**
  - SST **分布** (Hosaka et al, 1998)

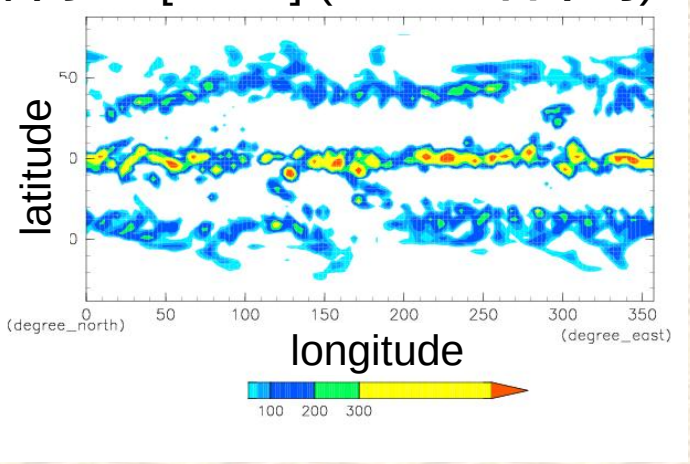
海水面温度 [K]



温度 [K] **東西平均** (90-100 **日平均**)



降水量 [W/m^2] (90-100 **日平均**)



目次

- はじめに
- dcmodelプログラミングガイドライン
- データ入出力ライブラリによるソースの簡素化と統一
- モジュール設計の工夫
- RDoc による解説文書の自動生成
- プログラムのテスト実行の自動化
- モジュール雛形作成ツールの整備
- モデルの実装の現状
- 今後の計画

今後の計画

- AGCM5 との結果および実行速度の比較
  - 物理過程がまともに実装されているか？
  - どこで遅くなっているか？
- プログラミングスタイル, モジュール設計についての速度の面からの見直し
- 木星を念頭においた湿潤惑星の計算
  - 杉山 (2007) と同様な計算設定
- 今よりも現実的な地球大気の計算
  - 海水面温度を現実的に
  - 地形がある
  - 陸面過程がある
  - 今よりは現実的な放射

参考文献

- Balaji, V.: The FMS Manual: A developer's guide to the GFDL Flexible Modeling System. http://www.gfdl.noaa.gov/~vb/FMSManual/FMSManual.html
- The flexible modeling system (FMS). http://www.gfdl.noaa.gov/~fms/, GFDL
- Hosaka, M., Ishiwatari, M., Takehiro, S., Nakajima, K., Hayashi, Y.-Y., 1998: Tropical precipitation patterns in the response to a local warm SST area placed at the equator of an aqua planet. J. Meteor. Soc. Japan, 76, 289--305.
- 森川 靖大, 小高正嗣, 石渡 正樹, 林 祥介, gtool4 開発グループ, 2006: gt490io ライブラリ, http://www.gfd-dennou.org/library/gtool4/, 地球流体電脳倶楽部.
- 森川靖大, 石渡正樹, 堀之内武, 小高正嗣, 林祥介, 2007: RDoc を用いた数値モデルのドキュメント生成. 天気, 54, 185--190.
- 沼口 敦, 1992: 博士論文.
- RDoc: http://www.ruby-doc.org/stdlib/libdoc/rdoc/rdoc/
- Ruby: http://www.ruby-lang.org/
- SWAMP Project, 1998: AGCM5. http://www.gfd-dennou.org/library/agcm5/. 地球流体電脳倶楽部
- 竹広 真一, 小高 正嗣, 石岡 圭一, 石渡 正樹, 林 祥介, 2006: 階層的地球流体スペクトルモデル集 SPMODEL. ながれマルチメディア 2006.
- 竹広真一, 石岡圭一, 森川靖大, 小高正嗣, 石渡正樹, 林祥介, SPMODEL 開発グループ, 2004: 階層的地球流体力学スペクトルモデル集 (SPMODEL), http://www.gfd-dennou.org/library/spmodel/, 地球流体電脳倶楽部.