Fortran 90を用いた
お手軽 grid model 構築ツール
GMSマニュアル

平成 23 年 8 月 30 日

目 次

第 I 部
Tutorial Document

1 はじめに

お手軽グリッドモデル構築ツール (GMS) は, 中野満寿男さんが開発を始めたもので, 現在は福岡大学に所属している安部実希が引き継いでいる. 我々は, このライブラリが多くの人に使われ, 地球流体力学の発展に寄与することを願う.

近年, PC の低価格化, 高速化が進み, 数値モデルを用いた研究をお手軽に行えるようなった. しかしながら, 数値モデルの構築は依然として難しい. また, 数値実験によってシミュレートされた現象の物理的解釈のためには, より簡単なモデルによる実験と比較しなければならないこともあり, そのために必要な数値モデルの改変をスムーズに行うためには, モデルのコードが高い可読性を備えている必要がある.

可読性の高い数値モデルを構築するためにはどのようにすればよいだろうか. 高い可読性のために解決すべき問題点として, まず配列添え字が挙げられる. 気象モデルでは変数配列に 3 つの添え字がついていることが多い. この 3 つの添え字は, 3 次元空間でどの格子の値であるのかを表している. この配列添え字は, たとえば DO ループを回す際に同じようなことを何度も記述する必要があり, 可動性を低くする一因となっていると考えられる. 次に考えられる問題点として, グリッド情報の管理が挙げられる. 気象モデルでは, 物理量を互い違いに配置する, いわゆる staggered grid がよく用いられる. よって一般に, 同じ添え字を持つ個々の変数は 3 次元空間で別々の位置に配置されることになる. 数値モデルをコーディングする際には, 常にそのことを考える必要があり, しばしばバグのすみかになっている. ここで, 変数の添え字が 3 次元空間でどの位置の値を表しているのかという情報を「グリッド情報」と呼ぶことにする.

上記で挙げた問題点を解決すべく Fortran90 を用いて GMS(Grid Modeling System) の開発を進めている. その主な特徴は次の通りである. まず, 変数の値を記憶する配列と, グリッド情報からなるデータ構造を GMS 変数として構造体を使って定義することにより, 変数自体に自分がどの位置に配置されているのかを含めることが可能となっている. 次に, GMS 変数を返り値とする空間差分演算子, 空間平均演算子を定義することにより, 差分や平均の演算結果は半格子分ずれるという情報を返り値に含めることが可能となっている. 以上より, GMS を使ってコードを書くことで, 可読性の高いモデルをお手軽に構築することができる.

「おてがる GMS」では 1 次元浅水波モデルを例にして, GMS の使い方を解説している.

2 インストール

以下のどちらの場合でもまず dclf90, gtool5, netcdf-dev をインストールする必要がある. dclf90 と gtool5 については, PPA の共有リポジトリにアップしている. また, 後のために gave もインストールする.

- 共有リポジトリ
  deb http://ppa.launchpad.net/gfd-dennou/ppa/ubuntu CODENAME main
  deb-src http://ppa.launchpad.net/gfd-dennou/ppa/ubuntu CODENAME main

CODENAME は lucid や maverick などの Ubuntu でのリリース名とする. 詳しくは 電脳 Ruby プロジェクトホームページの ubuntu ページに載せている.

2.1 gms パッケージがある場合

最低でも Ubuntu 環境でのパッケージは作成する予定である. 作成した際には、以下のコマンドでインストールできるようにする. 作成したものは上記の PPA リポジトリにアップする予定である.

```
$ sudo apt-get install gms
```

2.2 gms パッケージがない場合

サイトにある gms の tar ボール (gms_latest.tar.gz) をダウンロードする必要がある.

1) ダウンロードした tar ボールを展開し, できたディレクトリに入る.

```
$ tar zxvf gms_latest.tar.gz
$ cd gms/
```

2) libgms.a をインストール
コンパイラが gfortran の場合を例に挙げて説明する.

```
$ cd gms/source
```

必要ならば、source ディレクトリ内にある mklib-gfr の中を修正する.
・FC : Fortran コンパイラを指定. ex) FC=gfortran
・OPT : 指定した Fortran コンパイラのオプション指定. ex) OPT=”-wX -c”
・35 行目 - 36 行目 : mod ファイルとライブラリの移動先を指定する.
ex) mv *.mod /usr/local/gms-gfortran/include/
ex) mv libgms.a /usr/local/gms-gfortran/lib/
注) mod ファイルとライブラリの移動先をこの時点で作成しておく.

```
$ sudo ./mklib-gfr
```

3) gmsfrt をインストール
gms を用いたモデルのコンパイルは, gms/bin/gmsfrt を使う.

```
$ cd gms/bin/gmsfrt
```

必要ならば、gmsfrt の編集する.
・fc : Fortran コンパイラを指定. ex) fc=”gfortran”
・linker : fc と同じものを指定. ex) linker=”gfortran”
・mods : mod ファイルの指定. gms を使用するには, gtool5, dcl-f90, gms の mod ファイルが必要.
ex) mods=”-I/usr/lib/gtool5/include -I/usr/local/gms-gfortran/include -I/usr/local/dcl-f90/include”
・fflags : fc のオプション.
・libs : ライブラリファイルの指定. gms を使用するには, gtool5, dcl-f90, gms の ライブラリファイルが必要.
ex) libs=”-L/usr/lib/gtool5/lib -lgtool5 -L/usr/local/gms-gfortran/lib -lgms02 -lgms
-L/usr/local/dcl-f90/lib -lf90dcl”
・ldflags, ldlibs : ここでは, netcdf のフラグとライブラリパスを記入.
ex) ldflags=”-L/usr/netcdf/lib -L/usr/lib ”
ex) ldlibs=”-lnetcdf -lnetcdff -lgfortran”
gmsfrt のインストール. インストール先が /usr/local/gms-gfortran/bin の場合.

```
$ sudo cp gmsfrt /usr/local/gms-gfortran/bin/
```

注) 上記のライブラリパスやインクルードパスは上記の PPA リポジトリからインストールした場合の例なので, 各自環境に合わせて変える必要がある.

4) gmsfrt のパスを通す
~/.bashrc に, 以下を追記する.
PATH=$PATH:/usr/local/gms-gfortran/bin
export=PATH

5) ちゃんとインストールできたか確認

```
$ cd gms/sample/
$ gmsfrt shallow1d.f90
$ ./a.out
$ gave test.nc
```

ここで図 (例. 図 1) が正しく表示されることを確認できれば、GMS を正しくインストールできたことになる.

3 おてがる GMS

GMS の基本的な使い方と概念について, 例題を用いて具体的に説明を行う. ここでは, UNIX システムに GMS をマニュアル通りにインストールしていることを前提としている. そうでない環境で GMS を学び始めようという方は, まず, 基本操作のどこが違うか身近な先生達に直接尋ねる必要がある. ここで紹介する FORTRAN プログラム自体は, どんな環境でも同じように動くようになっている.

3.1 まずは 1 次元浅水波モデル

GMS を用いると, 可読性の高いコードで数値モデルを作ることができる. 最初の例題として, 数値モデルの中でも比較的簡単な 1 次元浅水波モデルをつくってみる.

3.1.1 gms を用いて作った 1 次元浅水波モデル

FORTRAN プログラムは, 次のページに載せている gms_shallow1d である. 時間積分は Leap-Frog 法を用いている.
プログラム自体は sample ディレクトリの中にある.

```
1:  program gms_shallow1d
2:  !—描画には gtool5 を用いる.
3:      use dcl
4:      use gtool_history
5:  !— gms ライブラリ
6:  !— これを書かなきゃgms が使えない.
7:      use gms      8:      implicit none
9:  !—モデルに使うパラメタ
10:         integer, parameter :: nx = 100
11:         integer, parameter :: nloop = 1000
12:         real, parameter :: fdump = 1000.0
13:         real(8), parameter :: delta_x = 2000.0D0
14:         real(8),parameter :: xmax = delta_x * nx, xmin = 0.0D0
15:         real(8), parameter :: depth = 1.0D0, grav=9.8D0, dt = 100.0D0
16:         real(8), parameter :: pi = 3.14159265358979D0
17:         integer :: i
18:         real :: time=0.0
19:         external initfunc
20:         real(8)::initfunc

21: !—糊代の数と糊代を含んだ配列の最小・最大
22:         integer, parameter :: margin = 1
23:         integer, parameter :: lbx = -margin, ubx = nx+margin

24: !—GMS 変数 宣言
25:         type(var_x) :: u, u_a, u_b
26:         type(var_x) :: h, h_a, h_b

27: !—GMS に必要な情報を与える
28:         call set_grid_num_x(nx) !格子点数
29:         call set_margin_x(margin) !糊代の長さ
30:         call set_real_min_x(xmin) !x 軸の最小座標値
31:         call gms_set_interval_x(delta_x) !dx
32:         call dump_gms_modelparm

33: !—work 領域の大きさ
34:         call allocate_work_area_x(20)

35: !—GMS 変数にグリッド情報を与える
36:         call def_var(u, on_none_none_grid)
37:         call def_var(u_a, on_none_none_grid)
38:         call def_var(u_b, on_none_none_grid)
39:         call def_var(h, off_none_none_grid)
```

```
40:         call def_var(h_a, off_none_none_grid)
41:         call def_var(h_b, off_none_none_grid)

42:  !—作成する NetCDF ファイルの設定
43:          call gt4init(h, "test.nc", fdump, "h", "displacement", "m")

44:  !—変数 h の初期状態設定
45:         call input_func_var(initfunc, h_b)
46:         u_b = 0.0D0

47:  !—境界条件設定
48:         call cyclic_boundary_x(u_b)
49:         call cyclic_boundary_x(h_b)

50:  !—Euler 法による時間積分 (1 ステップ分)
51:         h = h_b - depth * d_x(u_b) * dt
52:         u = u_b - grav * d_x(h_b) * dt

53:         call cyclic_boundary_x(u)
54:         call cyclic_boundary_x(h)

55:  !—Leap Frog 法による時間積分
56:       do i = 0, nloop

57:          time = real(dt) * i

58:            h_a = h_b - depth * d_x(u) * 2.0D0 * dt
59:              u_a = u_b - grav * d_x(h) * 2.0D0 * dt

60:            call cyclic_boundary_x(u_a)
61:              call cyclic_boundary_x(h_a)

62:  !—タイムフィルター
63:              u = u + 0.1D0 * (u_a - 2.0D0 * u + u_b)
64:              h = h + 0.1D0 * (h_a - 2.0D0 * h + h_b)

65:  !—NetCDF に書き出し
66:            if ( mod(time, fdump) == 0.0 ) then
67:              call gt4_timeout("t", time)
68:              call gt4_varout("h", h)
69:            end if

70:  !—次のステップ
```

```
71:              u_b = u
72:                h_b = h
73:                h = h_a
74:                u = u_a

75:       end do

76:   !—NetCDF への書き出し終了
77:       call gt4end

78: contain

79:    subroutine gt4init(var, fname, ndump, vname, lname, vunit)
80:       type(var_x), intent(in) :: var
81:       character(*), intent(in):: fname, vname, lname, vunit
82:       real , intent(in)::ndump

83:       call HistoryCreate(”test.nc”, ”gms_test”, ”gms_shallow1d”, ”masuo”, &
84:                                & (/’x’, ’t’/), (/size_x(var),0/), &
85:                                   (/”x-coordinate”,”time ”/), (/”m”,”s”/), 0.0, ndump)

86:       call HistoryPut(’x’, pos_x(var))

87:       call HistoryAddVariable( &！変数定義
88:                      varname=vname, dims=(/’x’,’t’/), &
89:                      longname=lname, units=vunit, xtype=’double’)

90:    end subroutine gt4init

91:    subroutine gt4_varout(vname, var)
92:       character(*), intent(in) :: vname
93:       type(var_x), intent(in) ::var

94:       call HistoryPut(vname, get(var) )

95:    end subroutine gt4_varout

96:    subroutine gt4_timeout(vname, time)
97:       character(*), intent(in) :: vname
98:       real, intent(in) ::time

99:       call HistoryPut(vname, time )

100:    end subroutine gt4_timeout
```

```
101:    subroutine gt4end
  102:        call HistoryClose
  103:    end subroutine gt4end

104:    end program gms_shallow1d

105:   !—初期条件
  106:          real(8) function initfunc(x)
  107:             real(8), intent(in) :: x
  108:             real(8), parameter ::pi = 3.14159265358979D0
  109:             initfunc = sin( 2.0D0 * pi * x / 2.0D5 )
  110:             write(*,*) x, initfunc
```

UNIX システムに GMS をマニュアル通りにインストールしている場合には,

```
$ gmsfrt -o shallow gms_shallow1d.f90
```

によって shallow という実行ファイルが作られる. そして,

```
$ ./shallow
```

とすると, 「test.nc」という NetCDF ファイルが作成される. ここで出力には gave を使うことにする.

```
$ gave test.nc
```

として, Contour & Tone で描画すると frame1 が出力される.

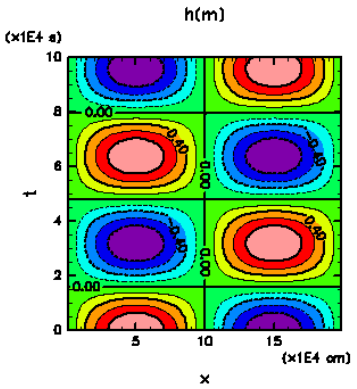

図 1: frame1

3.1.2　gms を用いて作った 1 次元浅水波モデルの説明

プログラム gms_shallow1d の最初から 48 行目までは, モデルの初期設定を行っている. それに次いで, NetCDF の設定, 浅水波方程式の計算, という順でコードが書かれている.

まず, 2 行目から 7 行目までは, 利用するモジュールを指定している. dcl と gtool_history は描画のときに使われるモジュールである. 次の gms が GMS ライブラリである.

次の 9 行目から 20 行目までは, モデルで使うパラメタである. これについてはここでは詳しく説明しない.

そして 21 行目から 41 行目までの部分で, GMS の初期設定を行っている.

- 21 行目 - 23 行目
  margin : [糊代の数]
  lbx, ubx : [配列の最小値と最大値]
  糊代の数と糊代を含んだ配列の最小・最大.

- 24 行目 - 26 行目：GMS 変数 宣言
  type(var_<dim>) :: value
  　GMS 変数の型宣言. <dim>：次元 (ex. x や xz), value：GMS 型変数として扱う変数

  下の図 2（2 次元 X-Y 平面）を例に考えると, 変数A・B・Cは次の様に書ける.
  　$type(var\_xy)$ :: A, B, C

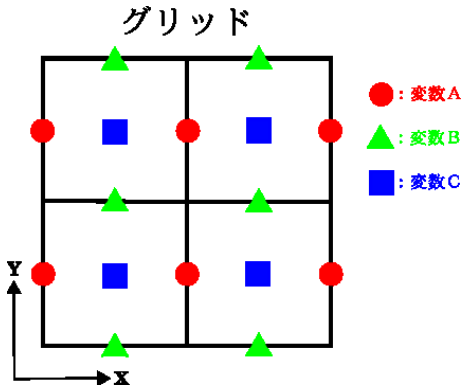

図 2: frame2

- 27 行目 - 32 行目：GMS に必要な情報を与える
  set_grid_num_<dir>(<dir>_number)
  　格子点の数を設定. <dir>：方向 (ex. x や y), <dir>_number：格子点数
  set_margin_<dir>(margin)
  　糊代の数を設定. <dir>：方向 (ex. x や y)
  set_real_min_<dir>(minimum)
  　最小座標の設定. <dir>：方向 (ex. x や y), minimum：最小座標
  gms_set_interval_<dir>(delta_<dim>)
  　グリッド幅の設定. <dir>：方向 (ex. x や y), delta_<dim>：グリッド幅
  dump_gms_modelparm
  　設定したことを表示

- 33 行目 - 34 行目：work 領域の大きさを設定
  allocate_work_area_<dim>(work_number)
  使用するメモリ領域の数を指定.

- 35 行目 - 41 行目：GMS 変数にグリッド情報を与える
  def_var(value, grid1_grid2_grid3_grid)
  GMS 変数にグリッド情報を与える. value：GMS 型変数, grid1, grid2, grid3：on か off を入力.
  on：座標軸上に変数を配置するとき.
  off：座標軸上に変数を配置しないとき.
  ・図 2（2 次元 X-Y 平面）を例に考えると, 変数A・B・Cは次の様に書ける.
  $call\ def\_var(\mathrm{A}, on\_off\_none\_grid)$
  $call\ def\_var(\mathrm{B}, off\_on\_none\_grid)$
  $call\ def\_var(\mathrm{C}, off\_off\_none\_grid)$

- 42 行目 - 43 行目：作成する NetCDF ファイルの設定
  ・お絵描きをする gms 型変数 が 1 つの場合
  call gt4init(var1, ”fname”, fdump, ”vname1”, ”lname1”, ”vunit1”)
  var1：gms 型変数, ”fname”：NetCDF ファイル名. 拡張子は nc, fdump：出力間隔,
  ”vname1”：変数の名前. この名前がグラフに出力される, ”lname1”：変数の長い名前,
  ”vunit1”：変数の単位

44 行目以降は, モデルの計算である.

- 44 行目 - 46 行目：初期値の設定
  call input_func_var(initfunc, var)
  gms 型変数 var に, 内部関数で計算した initfunc を代入している. モデルでは
  79 から 85 行目である.

- 47 行目 - 64 行目：境界条件の設定や, 計算
  ・call cyclic_boundary_<dir>(var)
  gms 型変数 var の境界を周期的条件で設定する.
  ・式中の d_x
  d_x(var)：gms 型変数 var の x 方向の 1 次精度後方差分をする.

- 65 行目 - 69 行目：NetCDF に書き出し
  ・call gt4_timeout(”t”, time)
  グラフに出力する時間の設定. ” ” の部分が出力される変数名, time の部分が出力される変数.

  ・call gt4_varout(”h”,h)
  グラフに出力する gms 型変数. ” ” の部分が出力される変数名, h の部分が出力される変数.

70 行目から 75 行目は省略.

76 行目 - 77 行目：NetCDF への書き出しを終了.

79 行目 - 103 行目：NetCDF への書き出しの設定を行っている. この部分の書き方は gtool5 を参考にしている.

(この時↑の注意)　参考 URL : gtool5 のサブルーチンの説明
HistoryCreate の設定を行う時, dimsize では size 関数を用いる.
また, HistoryPut の設定の時は, array 部分は position 関数を用いる.
ex) call HistoryCreate(fname,”title”,”source”,”institution”,dims,(/size_x(var),0/), ⋯
ex) call HistoryPut('x', pos_x(var))

以上で説明を終わる. 各関数, サブルーチンの詳細は, Reference に載せいている.

第 II 部
Reference

1 概要

2 MATH 数学処理パッケージ

中野 ver.

2.1 bar

- 機能
  grid 間での平均を取る.
- 書式
  output = bar_<dir>(var)
  <dir> : 方向. x or y or z
  ex) output = bar_x(var)
- 引数
  var : gms 型変数
- 備考
  なし
- 関連項目

2.2 cmean

- 機能
  一方向一列分の平均を取る. mean とは違って, 変数の位置を考慮に入れていない. 変数が軸上にない場合のみを定義している.
- 書式
  output = cmean_<dir>(var)
  <dir> : 方向. x or y or z
  ex) output = cmean_x(var)
- 引数
  var : gms 型変数
- 備考
  なし
- 関連項目

2.3 derivative

- 機能
  差分を計算する.
- 書式
  output = d_<num+dir>(var)
  <num+dir> について
  num : 差分の取り方の指定.
  なし → 1 つ後ろの値を, 2 → 1 つ前と 1 つ後ろの値を, 4 → 2 つ前と 2 つ後ろの値を使う
  dir : 方向. x or y or z
  ex) output = d_2x(var)
- 引数
  var : gms 型変数
- 備考
  なし
- 関連項目

2.4 divide

- 機能
  四則演算のひとつ. gms 型変数同士, または gms 型変数と実数の割り算を定義している.
- 書式
  gms 型変数 / gms 型変数
  gms 型変数 / 実数 (順序逆も可)
  ex) output = var1/var2 (計算では何も考えずに左記の様に書いてよい)
- 引数
  なし
- 備考
  なし
- 関連項目

2.5 function

- 機能
  gms 関数に対して組み込み関数が使えるように定義している.
- 書式

1. 引数が 1 つのもの :
   output = abs(var)
   他 : acos, aint, anint, asin, atan, cosh, exp, log, log10, sin, sinh, sqrt, tan, tanh

2. 引数が 2 つのもの：
output = atan2(var1, var2)
他：max, min, mod, modulo, sign

機能は Fortran90 の組み込み関数と同じである.

- 引数
var, var1, var2 : gms 型変数
- 備考
なし
- 関連項目

2.6 mean

- 機能
一方向一列分の平均を取る. cmean とは違って, 変数の位置を考慮に入れて定義してある.
- 書式
output = cmean_<dir>(var)
<dir> : 方向. x or y or z
ex) output = cmean_x(var)
- 引数
var : gms 型変数
- 備考
なし
- 関連項目

2.7 minus

- 機能
四則演算のひとつ. gms 型変数同士, または gms 型変数と実数の引き算を定義している. また, ただ gms 型変数にマイナス (-) をつけることができる.
- 書式
gms 型変数 - gms 型変数
gms 型変数 - 実数 (順序逆も可)
- gms 型変数
ex) output = var1 - var2 (計算では何も考えずに左記の様に書いてよい)
- 引数
なし
- 備考
なし
- 関連項目

2.8 plus

- 機能
  四則演算のひとつ. gms 型変数同士, または gms 型変数と実数の足し算を定義している. また, ただ gms 型変数にプラス (+) をつけることができる.

- 書式
  gms 型変数 + gms 型変数
  gms 型変数 + 実数 (順序逆も可)
  + gms 型変数
  ex) output = var1 + var2　(計算では何も考えずに左記の様に書いてよい)

- 引数
  なし

- 備考
  なし

- 関連項目

2.9 power

- 機能
  gms 型変数のべき乗, または実数を gms 型変数でべき乗する時の定義をしている.

- 書式
  gms 型変数 ** gms 型変数
  gms 型変数 ** 実数
  実数 ** gms 型変数
  ex) var1 ** var2　($= var1^{var2}$)

- 引数
  なし

- 備考
  なし

- 関連項目

2.10 times

- 機能
  四則演算のひとつ. gms 型変数同士, または gms 型変数と実数のかけ算を定義している.

- 書式
  gms 型変数 * gms 型変数
  gms 型変数 * 実数 (順序逆も可)
  ex) output = var1 * var2

- 引数
  なし
- 備考
  なし
- 関連項目

2.11 upwind1

- 機能
  1 次精度風上差分で計算する関数.
- 書式
  output = uw1_<dir>(var1,var2)
  <dir> について
  dir : 方向. x or y or z
  ex) output = uw1_x(var1,var2)
- 引数
  var1,var2 : gms 型変数
- 備考
  保存形.
  1 つ目の引数 (var1) で流れの向きを判断する様にしている。
  例）$\frac{\partial uT}{\partial x}$ に対して使用すると、(u、T：1 次元変数)
  output = uw1_x(u,T)
  output の位置は、2 つ目の引数 (var2) に合わせてある.
- 関連項目

2.12 upwind3

- 機能
  3 次精度風上差分で計算する関数.
- 書式
  output = uw3_<dir>(var1,var2)
  <dir> について
  dir : 方向. x or y or z
  ex) output = uw3_x(var1,var2)
- 引数
  var1,var2 : gms 型変数
- 備考
  保存形.
  1 つ目の引数 (var1) で流れの向きを判断する様にしている。
  例）$\frac{\partial uT}{\partial x}$ に対して使用すると、(u、T：1 次元変数)

output = uw3_x(u,T)
output の位置は、2 つ目の引数 (var2) に合わせてある.

- 関連項目

3 GRPH 図形処理パッケージ

安部 ver.

3.1 draw_graph

- 機能
  とにかくデータを NetCDF ファイルにはき出すためのサブルーチン. gms 型変数 5 つ分までサポートしている.

- 書式
  描画する変数の数が 1 つの場合.
  call gt4init<1>_<dim>(var1, ”fname”, ndump, ”vname1”, ”lname1”, ”vunit1”, dt)
  call gt4_varout<1>_<dim>(”vname1”, var1)
  <1>：描画する変数の数を記入. 1 から 5 まで.
  <dim>：変数の次元. x や xz など
  ex) call gt4init1_x(var1, ”test.nc”, ndump, ”vname1”, ”lname1”, ”vunit1”, dt)
  ex) call gt4_varout1_x(”vname1”, var1)

- 引数
  var1：描画したい gms 型変数.
  fname：ファイル名. 拡張子を .nc にする.
  ndump：出力間隔.
  vname1：変数名. これがグラフに出力される.
  lname1：正式な変数名.
  vunit1：単位. グラフに出力される.
  dt：時間ステップ幅

- 備考
  fname, vname1, lname1, nunit1 は ” ” に挟んで書く.
  自分で定義して描画したい場合は, gtool5 を自分で勉強する必要がある.
  (この時↑の注意)　参考 URL：gtool5 のサブルーチンの説明
  HistoryCreate の設定を行う時, dimsize では size 関数を用う.
  また, HistoryPut の設定の時は, array 部分は position 関数を用いる.
  ex) call HistoryCreate(fname,”title”,”source”,”institution”,dims,(/size_x(var),0/), ⋯
  ex) call HistoryPut(’x’, pos_x(var))

  2.2 節で gms のインストール方法を記載したが、その方法ではこの draw-graph はインストールされない.
  Fortran Compiler が gfortran ではなく、gmsfrt でなくてはならない. このモジュールを使用する方法は、要望があれば書く予定である.

- 関連項目

– 関連ルーチン (position, size)

4　その他のパッケージ

中野 ver.

4.1　assign

- 機能
  gms 型変数を含んだ計算において, 左辺と右辺をイコールでつなげられるように定義している.
  また, 左辺と右辺でグリッドの位置が等しいかを評価している.
- 書式
  gms 型変数 = gms 型変数
  ex) var1 = var2　(計算では何も考えずに左記の様に書いてよい)
- 引数
  なし
- 備考
  なし
- 関連項目

4.2　assign2

- 機能
  gms 型変数に実数を代入することができるように定義している.
- 書式
  gms 型変数 = 実数
  ex) var1 = real_num　(計算では何も考えずに左記の様に書いてよい)
- 引数
  なし
- 備考
  なし
- 関連項目

4.3　asym

- 機能
  境界条件に関するサブルーチン. 境界を挟んで反対称的な条件にする. ex) 粘着壁条件
- 書式
  call asym_boundary_<dir>(var)
  　<dir> : 方向. x or y or z
  ex) call asym_boundary_x(var)

- 引数
  var：gms 型変数
- 備考
  なし
- 関連項目

4.4　cyclic

- 機能
  境界を周期的条件に定義する.
- 書式
  call cyclic_boundary_<dir>(var)
  <dir>：方向. x or y or z
  ex) call cyclic_boundary_x(var)
- 引数
  var：gms 型変数
- 備考
  なし
- 関連項目

4.5　datatype

- 機能
  gms 型変数の構造体を定義している.
- 書式
  type(var_<dim>) :: var
  <dim>：変数の次元. x や xz など
  ex) type(var_x) :: var
- 引数
  var：gms 型変数
- 備考
  なし
- 関連項目
  なし

4.6　def_var

- 機能
  変数に grid 情報を与える.

- 書式
  call def_var_<dim>(var, gridinfo)
  　<dim>：変数の次元. x や xz など
  ex) call def_var_x(var, on_none_none_grid)

- 引数
  var：gms 型変数
  gridinfo：grid 情報. 記述の仕方は ”grid_info” を参照.

- 備考
  grid 情報の記述方法は関連ルーチンのものを使う.

- 関連項目
  – 関連ルーチン (grid_info)

4.7　get

- 機能
  gms 型変数を配列型変数に変換する.

- 書式
  output = get_<dim>(var)
  　output：得られた配列型変数
  　<dim>：変数の次元. x や xz など
  ex) output = get_x(var)

- 引数
  var：gms 型変数

- 備考
  物理空間のみ変換される. output にあらかじめ配列を与えておく必要がある.

- 関連項目

4.8　input_func_var

- 機能
  関数によって定義した値を gms 型変数に値を代入する.

- 書式
  call input_func_var_<dim>(func, var)
  　<dim>：変数の次元. x や xz など
  ex) call input_func_var_x(func, var)

- 引数
  func：関数によって定義した値. これは内・外関数などで自分で定義する必要がある.
  var：値を放り込まれる gms 型変数.

- 備考
  なし

- 関連項目

4.9　input_var

- 機能
  配列型変数を gms 型変数に変換する.
- 書式
  call input_var_<dim>(input, var)
  <dim>：変数の次元. x や xz など
  ex) call input_var_x(input, var)
- 引数
  input：配列型変数　var：gms 型変数
- 備考
  物理空間のみ変換される. input と var の配列個数が一致してないといけない (このことは get も同様).
- 関連項目

4.10　position

- 機能
  gms 変数の座標位置を配列型変数, または gms 型変数として求める.
- 書式
  output1 = pos_<dir>(var)
  output2 = pos_var_<dir>(var)
  output1：配列型変数
  output2：gms 型変数
  <dir>：方向. x or y or z
  ex) output1 = pos_x(var)
  ex) output2 = pos_var_x(var)
- 引数
  var：gms 型変数
- 備考
  物理空間のみ求められる.
- 関連項目

4.11　size

- 機能
  gms 型変数のサイズを求める.

- 書式
  output = size_<dir>(var)
  <dir> : 方向. x or y or z
  ex) output = size_x(var)
- 引数
  var : gms 型変数
- 備考
  物理空間のみ求められる.
- 関連項目

4.12　sym

- 機能
  境界条件に関するサブルーチン. 境界を挟んで対称的な条件にする. ex) すべり壁条件, 断熱条件
- 書式
  call sym_boundary_<dir>(var)
  <dir> : 方向. x or y or z
  ex) call sym_boundary_x(var)
- 引数
  var : gms 型変数
- 備考
  なし
- 関連項目

4.13　mem_manager

- 機能
  1. 使用する gms 型変数が必要となるメモリ (work area) を確保する.
  2. work area 上で計算結果に対応する値が入っているアドレスに印を付ける. 一番最初はユーザーがこれを行わないといけない. これは, def_var で変数に grid 情報を与える時に自動的になされる.
  3. work area 上のあいてるアドレスを問い合わせる. 計算結果をメモリに書き出すモジュールで自動的に問い合わせるようになっている.
  4. work area につけられている計算結果の印を新しい場所に付け替える
  5. ゴミ集め. やってるのは Address counter を 1 にリセットするだけである.

  *) 上記の 1. 以外はユーザーが定義する必要はない.
- 書式
  work area の確保
  call allocate_work_area_<dir>(num)
  <dir> : 方向. x or y or z
  ex) call allocate_work_area_x(10)

- 引数
  num：使用する gms 型変数の数
- 備考
  num は使用する数より少し多めに取っておくとよい.
- 関連項目

4.14　model_info

- 機能
  モデルの情報を設定するモジュール. 格子点数, 糊代の数, グリッドの大きさ (dx など), 物理空間での座標最小値, そして以上の設定をプログラム実行時に端末に表示する.
- 書式
  call set_grid_num_<dir>(num)
  call set_margin_<dir>(margin)
  call gms_set_interval_<dir>(grid_size)
  call set_real_min_<dir>(min)
  call dump_gms_modelparm
  <dir>：方向. x or y or z　　ex) call set_grid_num_x(num)
  ex) call set_margin_x(margin)
  ex) call gms_set_interval_x(dx)
  ex) call set_real_min_x(min)
  ex) call dump_gms_modelparm
- 引数
  num：格子点数
  margin：糊代の数
  grid_size：グリッドの大きさ (dx など)
  min：物理空間での座標最小値
  以上はすべてユーザー定義.
- 備考
  なし
- 関連項目

ここから安部 ver.

4.15　get_all

- 機能
  gms 型変数を配列型変数に変換する.

- 書式
  output = get_all_<dim>(var)
  output：得られた配列型変数
  <dim>：変数の次元. x や xz など
  ex) output = get_all_x(var)

- 引数
  var：gms 型変数

- 備考
  get と違うところ：糊代も含めたデータを扱う.

- 関連項目
  - 関連ルーチン (get)

4.16　grid_info

- 機能
  変数に与えるグリッド情報群. 変数の位置がグリッド境界上なら「on」, 半グリッドずれたところなら「off」と表現する. 考えない次元に対しては「none」と表現する.

- 書式
  def_var を用いて使用例を示す.
  call def_var_<dim>(var, grid1_grid2_grid3_grid)
  <dim>：変数の次元. x や xz など
  ex) gms 型変数 var が 2 次元変数で, 位置については x 軸に対してはグリッドの境界上に,
  y 軸に対しては半グリッドずれている場合.
  call def_var_xy(var, on_off_none_grid)

- 引数
  var：gms 型変数
  grid1, grid2, grid3：on or off or none を代入.

- 備考
  on には ”0”, off には ”1” というグリッド情報が与えられる. これにより, スタッガード格子点配列の変数についてサポートしている.

- 関連項目
  - 関連ルーチン (def_var)

4.17　position_all

- 機能
  gms 変数の座標位置を配列型変数, または gms 型変数として求める.

- 書式
  output1 = pos_all_<dir>(var)
  output2 = pos_var_all_<dir>(var)
  output1：配列型変数

output2：gms 型変数
<dir>：方向. x or y or z
ex) output1 = pos_all_x(var)
ex) output2 = pos_var_all_x(var)

- 引数
  var：gms 型変数
- 備考
  糊代を含む.
- 関連項目

4.18 size_all

- 機能
  gms 型変数の物理空間でのグリッド数
- 書式
  output = size_all_<dir>(var)
  <dir>：方向. x or y or z
  ex) output = size_all_x(var)
- 引数
  var：gms 型変数
- 備考
  size と違うところ：糊代も含めたデータを扱う.
- 関連項目

4.19 lower_boundary_noslip

- 機能
  境界条件に関するサブルーチン. 境界の中でも最小座標の部分を粘着壁条件にする.
- 書式
  call lower_boundary_noslip_<dir>(var)
  <dir>：方向. x or y or z
  ex) call lower_boundary_noslip_x(var)
- 引数
  var：gms 型変数
- 備考
  asym と違うところ：asym は最大・最小座標両方の定義をしている.
- 関連項目

4.20　lower_boundary_slip

- 機能
  境界条件に関するサブルーチン. 境界の中でも最小座標の部分をすべり条件にする. 温度については断熱壁になる.

- 書式
  call lower_boundary_slip_<dir>(var)
  <dir> : 方向. x or y or z
  call lower_boundary_slip_x(var)

- 引数
  var : gms 型変数

- 備考
  sym と違うところ : sym は最大・最小座標両方の定義をしている.

- 関連項目

4.21　same_value_boundary

- 機能
  境界条件に関するサブルーチン. 境界の最小・最大座標それぞれに値を代入する. 等値境界条件. 等温壁にすることができる.

- 書式
  call same_value_boundary_<dir>(var, input1, input2)
  <dir> : 方向. x or y or z
  call same_value_boundary_x(var, input1, input2)

- 引数
  var : gms 型変数
  input1 : 境界の最小座標での値
  input2 : 境界の最大座標での値

- 備考
  なし

- 関連項目

4.22　upper_boundary_noslip

- 機能
  境界条件に関するサブルーチン. 境界の中でも最大座標の部分を粘着条件にする.

- 書式
  call upper_boundary_noslip_<dir>(var)
  <dir> : 方向. x or y or z
  call upper_boundary_noslip_x(var)

- 引数
  var：gms 型変数
- 備考
  asym と違うところ：asym は最大・最小座標両方の定義をしている.
- 関連項目

4.23　upper_boundary_slip

- 機能
  境界条件に関するサブルーチン. 境界の中でも最大座標の部分をすべり条件にする. 温度については断熱壁になる.
- 書式
  call upper_boundary_slip_<dir>(var)
  <dir>：方向. x or y or z
  call upper_boundary_slip_x(var)
- 引数
  var：gms 型変数
- 備考
  sym と違うところ：sym は最大・最小座標両方の定義をしている.
- 関連項目

以上で説明を終わります.