第4章　データハンドリングと可視化

4.1　数値予報システム周辺で用いられるデータ形式 [1]

4.1.1　データモデル

(1)　概説

電子計算機の発達と分野間連携の拡大により、数値予報システムおよびそのプロダクト提供などの場で用いられるデータ形式は多様さを増している。これらを整理して記述するためには、データモデルすなわち我々が住む時空間をどのように離散化し、時空間における位置以外の情報と結びつけるかの方式によって分類するのが簡便である。地理情報科学では全てのデータモデルをベクタデータ (vector data) およびラスタデータ (raster data) に大別する整然とした体系 (図 4.1.1) があって便利なので、以下主にこれら用語を用いる（地理情報科学では関心が水平 2 次元に偏るが、適宜補足する）。

(2)　ベクタデータ

ベクタデータとは地上に描かれた一又は多数の幾何学的図形のそれぞれに時空間位置以外の情報を結びつけるものである。幾何学的図形の形状に応じて、多角形データ（2 次元）、線データ（1 次元）、点データ（0 次元）と呼ばれる。たとえば東京の気温であれば、東京管区気象台の位置が点で与えられ、これに気温という実数値のデータが結び付けられている点データと考える。予報区に対して発表される予報文は多角形データである。

(3)　ラスタデータと気象分野用語「格子点データ」

地理情報科学でラスタデータとは、地上の矩形領域上で 2 つの直交する座標軸に沿って等間隔の格子を配置して、その各点に数値を与えるものである。一方気象分野で格子点データと呼ばれているのはこれより幅広い概念で、次のような場合を含む。対応関係を図 4.1.2 に示す。

- 水平面以外の 2 次元データ
- 3 次元以上の高次元データ（地理情報科学では 3 次元が稀に使われるのみ）
- 不等間隔の規則格子：直交する座標軸に沿うが格子間隔が等間隔と限らないデータ。ガウス緯度（特定のルジャンドル多項式のゼロ点、佐藤 1982）と等間隔の経度とで作るガウス格子 (Gaussian grid) はその例
- 不規則格子：経緯度空間の矩形領域上だが、ひとつの軸に沿った格子数が極に近づくにつれて少なくなるようなデータ。緯度が等間隔なものを間引格子 (thinned grid) と呼び、ガウス緯度によるものを適合ガウス格子 (reduced Gaussian grid) と呼ぶ (宮本 2005)
- 非矩形格子：領域が矩形ですらないもの。正二十面体格子データや、球面調和関数の波数空間での三角形領域内の展開係数配列など

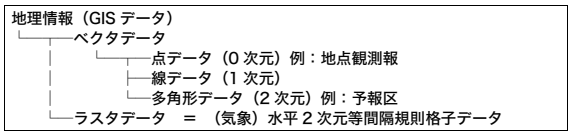

図 4.1.1　地理情報科学におけるデータモデルの分類。

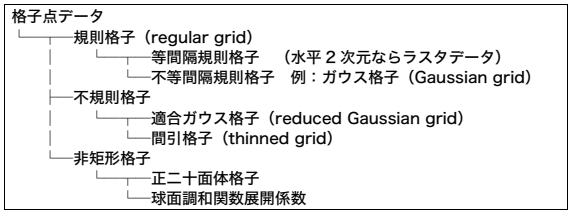

図 4.1.2　気象データにおける格子点データの分類。

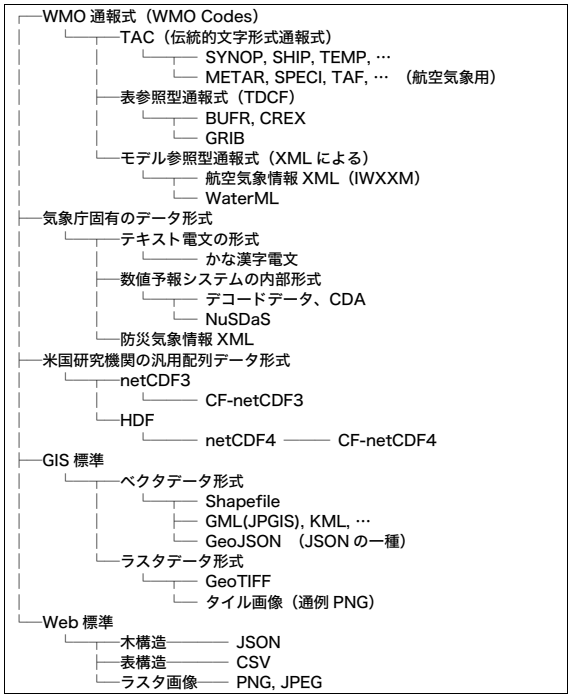

図 4.1.3　多種多様なデータ形式の分類。

4.1.2　データ形式各論

(1)　概説

数値予報システムの周囲で用いられる代表的なデータ形式を図 4.1.3 に示す。以下、策定主体による分類に沿って概観する。

[1] 豊田 英司（総務部企画課）

(2) WMO 通報式

WMO（世界気象機関）では世界気象監視計画（World Weather Watch Programme, 山本 1966）にしたがって国際的に交換される観測・解析・予報などのデータの交換形式である WMO 通報式 (WMO Codes) を定めている。WMO 通報式マニュアル (WMO 2015) は技術体系の異なる通報式をまとめた 3 分冊からなる。

ア　伝統的文字形式通報式

最も古い第 I.1 巻の伝統的文字形式通報式 (TAC: Traditional Alphanumeric Codes) は英数字などしか送れない電報（Alphanumeric を略して A/N 電文と俗称される）でデータを送るための通報式である。電報の利用は WMO 設立以前にさかのぼるが、現行 TAC 通報式の多くは 1970 年代に現在の形に整理された。文字種、行の長さ、電文長（15000 字）など、電報の技術制約を前提としている。

TAC はデータの種類に応じて地上実況気象通報式 (SYNOP)、海上実況気象通報式 (SHIP)、地上高層実況気象通報式 (TEMP) などの種類がある。いずれも決められた項目を固定桁数表示や符号表などを用いて英数字列（殆どは数字列）で表現するものであり、種類を特定しない一般のベクタデータを表現する枠組みを欠く代わりに、人手で作成解読しやすくなっている。殆どのデータモデルは点データであるが、線データを含む解析気象通報式 (IAC) や格子点資料気象通報式 (GRID) など例外もある。

イ　表参照型通報式

次の第 I.2 巻は表参照型通報式 (TDCF: Table-Driven Code Form) に分類される GRIB, BUFR 及び CREX を収録する。格子点データ用の GRIB とベクタデータ用の BUFR は両方ともバイナリである。バイナリが選ばれた背景には 1980 年代になって有線国際通信によるバイナリデータ伝送が可能になったことがある。

BUFR は Binary Universal Form of Representation の略であり、1988 年に第 1 版 (Edition 1) が制定された後幾多の改正を経ている。名称の universal とは TAC に欠けていた「一般のベクタデータを表現する枠組み」を指す。すべてのデータは数値型、符号表、又は文字列型のデータ要素に分解され、気温や風速などのデータ要素に 16 ビットの番号（記述子 descriptor）が与えられる。BUFR 報（バイナリ電報が前提なのでファイルではなく報 message という）に記述子列と対応するデータのビット列が含まれているので、記述子の表を参照すれば多種多様なデータが解読できる。TDCF が表参照という由来である。

CREX (Character form for the Representation and Exchange of data) は BUFR のデータを電報で送るための代替表現で、BUFR を構成する各構成要素を文字列で表現するようになっている。

WMO は全ての TAC による通報を段階的に BUFR 又は CREX に移行する方針で、航空気象情報（後述ウ）以外について 2014 年 11 月を移行期限としていた。上述 SYNOP, SHIP, TEMP などによる通報は現在対応する BUFR で行うのが本則であるが、実際には TAC+BUFR 並行通報または TAC だけの通報がいまだに多く見られるのが実情である。高層気象観測報の BUFR 移行が進まない現状や TAC との相違点による混乱については太田 (2015) や Ingleby et al. (2016) を参照願いたい。

TAC からの移行を支援するため、WMO では各 TAC 通報式から移行するための BUFR データ要素のリストをテンプレートとして取りまとめている。本来データ構造を自由に設計できるはずの BUFR であるが、同種の観測データを各国がそれぞれの都合で異なる構造で配信しないように統制することは自動処理に資する一方で、テンプレートは稀にしか用いられない要素も含んで設計されるため、BUFR 報の長さが対応する TAC 報より長くなりがちである。BUFR 制定当初は電文長が短いことがしばしば利点と宣伝されたが、現在では正しくない。TAC を BUFR に移行する意義はデータ管理や処理プログラムの簡素化と説明すべきである。

GRIB は Gridded Binary の略である。1989 年に制定された第 1 版 (Edition 1) は水平 2 次元格子データを表現するものであったが、2001 年に制定された第 2 版では多次元データを表現できるようになった。現在気象庁から部外配信される格子点データは原則として GRIB 第 2 版によっている（顕著な例外は、過去の互換性のための GRIB 第 1 版や、先進的取組として netCDF を用いたひまわりデータである）。

GRIB 第 2 版の文書にも膨大な表が含まれる。特に重要なのは気温や風速などの物理量など（GRIB ではパラメタという）の表と、データ構造を記述する表（テンプレート）である（BUFR でも同様）。ひとつの GRIB 報が第 0 節から第 8 節からなる構造からなっており、第 3 節が座標系、第 4 節がプロダクト定義（鉛直位置、時間、統計処理などのメタデータ）などを表すが、これら節の先頭の番号によって節の構造（テンプレート）が規定される。座標系の記述を選択することで不等間隔規則格子、不規則格子、非矩形格子などが表現でき、節の繰り返し（たとえば第 4 節以降の繰り返し）で同一座標系の鉛直系列や時系列が表現される。

時点ではなく時間範囲で定義されるデータ（たとえば降水量や発雷確率）は GRIB 第 2 版の第 4 節を適切に選ぶことによって初めてその時間がきちんと表現できるようになった。しかしながら、示強性の量の時間積算による表現（たとえば降水量 [kg m^{-2}] を降水強度 [kg m^{-2} s^{-1}] の時間積算と称する類）を導入したのにパラメタ表の単位を秒をかけたものに読み替えるという規定が欠けていたため解釈に混乱を生じた (Toyoda 2011)。

テンプレートの自由度を向上するなどの改善のため

GRIB 第 3 版が 2017 年から試行利用される。改善点は小規模に留まっており、義務性もない試行であることから現在のところ気象庁で採用する動きはない。

以上、TDCF 全体に共通する思想として、新しいデータ要素（数値型、文字列の両方）や構造を随時に追加させず、全て手続を踏んで表に登録し、番号を与えてから流通を許すという考え方がある。業務の新設・変更を急ぐ場合にはわずらわしいが、国際調整の立場からはこうでもしなければ世界での情報集約が図れない。ほとんどの表には国際共通部分の他に私用領域（8 ビットならば通常 192–254 が reserved for local use とされる）が用意されており、従来日本では往々にしてこの枠が活用されるが、私用番号は国際配信すべきでないものとされており、国内用として作成開始した資料を後で国際配信できなくなる問題がある。

ウ　XML を基にした通報式

最後の第 I.3 巻はモデル参照通報式 (model-driven code forms) と呼ばれており、XML を基にした通報式が収録されている。航空気象情報用の IWXXM (The ICAO Meteorological Information Exchange Model) を掲載するために 2015 年に新設され、2017 年には水文データ用の WaterML が追加される予定である。IWXXM は OGC (Open Geospatial Consortium) で作られた GML (Geographical Markup Language, ISO 2007) を基にしている。

XML を基にした通報式は、データ構造の記述[2]や符号表[3]がオンラインで提供されていることが特色である。TDCF ではデータ処理のプログラム構造を簡素化することはできても、表の正式な提供形態は WMO 通報式マニュアル（紙または PDF）により、各国で打ち込むべきという整理であることから比べると大きな省力化となる。IWXXM は XML 名前空間を多用した設計の複雑さはあるが、利点をうまく活用できるか、今後が注目される。

エ　CAP

厳密には通報式ではないが、WMO 情報システム (WIS) マニュアル (WMO 2016) には緊急又は警戒情報（alert and warning information. 定義は確立されていない）の交換のために CAP (Common Alerting Protocol, ITU-T 2007) に対応しなければならないという規定がある (1.7.2 節)。実際には CAP の利用は各国内が主で、国際間通信ではまだ伝統的なテキスト電報が多く使われているのが実情である。

(3)　気象庁固有のデータ形式

気象庁で内部的に用いられ部外配信されないデータについては独自のデータ形式が用いられることがある。多くは WMO 通報式の影響を受けており、類似した分類ができる。

ア　テキスト電報の形式

気象庁においても他の WMO 加盟国気象機関と同様、最も初期の電子データ交換はテキスト電報を前提としていた。観測報が日本国内だけで流通する場合「国内気象通報式」として標準化された。注意報・警報・気象情報・予報文などのテキスト情報は日本語文字を用いるため「かな漢字電文」と呼ばれ、これらの中にも数値データが混在するものが多い。国内気象通報式も 2015 年から気象庁ホームページで公開されている。

イ　数値予報システム内部で用いられるバイナリ形式

気象庁の数値予報システムでは BUFR 及び GRIB の影響を受けたデータ形式が内部的に利用されている。諸外国では BUFR や GRIB をそのまま内部形式としてデータベースに格納する例もあるが、気象庁で内部形式を用いるのには、大量のデータの扱いの標準化、あまりにも多様な外部形式への対応を簡素化すること、そしてデータ発信元・提供先の都合に振り回されないようにするなどの意義がある。

様々な形式で入電する観測報を解読（デコード）した結果はデコードデータと呼ばれる形式で保存される。これらの形式は次のような実用的な変更を加えた簡易 BUFR といえる。

- BUFR 報は節からなり、各節の長さは先頭 3 バイトで示されるのに対し、デコードデータはレコードからなり、レコードの先頭の 2 バイトでレコード長が示される
- BUFR は単独の通報の形式であり多数の BUFR 報を保存する方法は標準化されていないが、デコードデータは多種の構造のデータを連結して大きなファイルとして保持できる
- BUFR 報のデータ要素は必要最小限のビット数で表現されるため、ビットシフト演算を多用する必要があるのに対し、デコードデータのデータ要素は Fortran で扱いやすいように 2 バイトの倍数で格納され、個々の 2 バイトは 16 ビット符号付整数として扱われる
- BUFR 報は記述子列により無限のデータ構造を持ちうるが、デコードデータは種別番号ごとに固定的構造をもつ（後に BUFR でも過剰な自由度を制約するため任意のテンプレートが導入された）

格子点データについては NuSDaS（NWP Standard Dataset System, 豊田ほか 2012）と呼ばれる形式が用いられている。GRIB 第 2 版と比較した特徴は次のとおり。

- GRIB は単独の通報の形式だが、NuSDaS は多種大量のデータを格納できるデータベースであり、ディレクトリ構成とファイル形式の規定からなっている
- GRIB 報には単一の発信者からの単独の参照時間

[2] http://schemas.wmo.int/
[3] http://codes.wmo.int/

（初期値日時）のデータしか格納できないが、NuSDaS データセット（ディレクトリ構造）は 6 項目のキー（データセット名、基準時刻、対象時刻、アンサンブルメンバー、鉛直面、要素）で 2 次元配列を整理するデータベースであり、スーパーコンピュータシステムの全格子点データバンクを格納できるだけの自由度がある
- GRIB 報は節からなり、各節の長さは先頭 4 バイトで示されるのに対し、NuSDaS データファイルは Fortran の書式なし順番探査ファイルであり、なおかつ特定のレコードのバイト位置を直接アクセスできるように索引レコードが用意されている

ウ　気象庁防災情報 XML

気象庁ではかな漢字電文の近代化のため、防災情報 XML を策定した。策定経緯や設計理念は杉山ほか (2012) で詳述されている。緊急メッセージのための XML 形式としては CAP としばしば対比される。CAP はテキスト情報を伝達することに特化しているのに対して、防災情報 XML はデータを表現することもできるようになっている点が大きく異なる。かな漢字電文全体を通した字句構造の規則は存在しなかったため場当たり的な解析を強いられたのに対して、XML の採用により数値データを抽出する部分のロジックは汎用の XML 処理系が使えるようになり、処理ソフトウェア開発コストを低減している。

数値予報地点ガイダンスの庁外配信形式 (気象庁予報部 2010, 2014) は防災情報 XML と同じ名前空間を用いているが、若干拡張があるため厳密には防災情報 XML の枠組外とされている。

(4)　米国研究機関発の汎用配列データ形式

配列データについては netCDF (network common data form) や HDF (hierarchical data format) がしばしば用いられる。UCAR Unidata で開発された netCDF が気象・海洋分野で、NASA で開発された HDF が衛星リモートセンシング分野でよく用いられる。これらはどちらも、任意個の次元をもちうる配列に名前をつけ、これらを組み合わせて 1 ファイルのデータを構成する枠組みだけを規定するものである。データ表現だけに徹する姿勢は木構造データに対する XML と相似する。HDF はファイル内にディレクトリのような分類を設けることができる点、若干機能が多い。バージョン 3 以前の netCDF と HDF は別個の形式で、相互に読み書きはできなかった。統合を図るため netCDF バージョン 4 は HDF5 に基づいたものになったが、データ圧縮などの新機能が必要でない場合、依然として netCDF バージョン 3 は広く使われている。

気象学固有の概念を表現するため、配列名などの約束（規約 conventions という）を利用者間で決めておく。広く普及しているのは CF (Climate and Forecast) 規約 (Eaton et al. 2011) であり、netCDF バージョン 3 と CF 規約が OGC 標準となっている。CF 規約は TDCF と対照的に、テキストで与えられる情報は符号表ではなくそのテキストそのものを記載する方針をとる。新たなデータを扱い始める際に符号表の登録を待つ必要がなく、可読性も向上する反面、英語を強要することになり（学術界では当然でも国連機関では難しい）、利用実態の全貌が把握しにくくなるきらいがある（現業機関では受け入れがたい）。

CF 規約におけるテキスト主義の例外は標準名 (standard name) である。変数名（配列の名称）は自由につけられるかわり、気温や風速のような物理量を識別するため、登録済みの標準名を示すことになっている。公開のメーリングリストで提案後 3 週間異論がなければ採択、という手続の速さと透明性を謳っているが、実際をみると独特の整理学になじまない提案について延々と議論が続くということも散見される。一定期間内に新規データの取扱を始めねばならない時の国際調整の容易さについては TDCF 通報式と大差ないというべきで、中央で表を管理して意味論的一貫性を維持するという方式をとる以上避けられないコストと考える。

CF 規約では GRIB ほどではないが特殊な格子系（不等間隔規則格子や不規則格子）を扱えるようになっているが、利用実態としては緯度経度座標系の規則格子がほとんどであるとみられる。

(5)　GIS 標準

気象情報の利活用促進や、幅広い機関と連携した観測情報の活用を考えるとき、連携先パートナーの多くが地理情報システム（GIS: Geographical information system、要は地図を描くソフトウェア）の利用者である点を考慮する必要がある。彼らの利便性に配慮して地理情報分野固有のデータ形式の提供・解読が求められることが多くなってきている。

ア　ベクタデータ形式

ベクタデータについて圧倒的な影響力をもつのは ESRI 社の Shapefile 形式である。本来同社の商用 GIS の専用形式でありオープンフォーマットとはいえないが、他社・オープンソースを含めサポートしていない GIS は事実上存在しない。ひとつのデータ（レイヤー）が多数のファイルで構成されるのが特徴で、拡張子 .shp のファイル（仕様は ESRI 1998 参照）が点・線・多角形の位置情報を示し、図形に結び付けられた情報は dBase 形式 (拡張子 .dbf）のデータベースで示される。

特定の企業に従属しない業界標準データフォーマットを作る試みが幾度か行われた。古くは 1990 年代に開発された米国 ANSI 標準 SDTS (Spatial Data Transfer Standard)、2000 年代では OGC の GML がその代表例である。Google Earth で有名になった KML (Keyhole Markup Language) も管理が OGC に寄託されたので、この分類に該当する。

近年急速に影響力を増しているのは GeoJSON (But-

ler et al. 2016) で、JSON 形式（後述）に位置情報を付加したものである。ウェブブラウザとの親和性に加えてデータが 1 ファイルで表現できることもウェブ時代には非常に有利である。

日本での状況をみると、政府機関の地理データ提供は Shapefile と GML を採用する例が多かった（例：国勢調査結果、国土数値情報）が、国土地理院は近年ウェブ展開を強化するため GeoJSON を推進する姿勢を示している (出口・伊藤 2016)。

イ　ラスタデータ形式

ラスタデータについて留意したいのは気象業界との時空間のとらえ方の差である。まず空間について、GIS 業界でラスタデータという場合は矩形領域上の等間隔の規則格子だけ、ほとんどの場合、地図投影面上での等間隔（メートル単位）が用いられる。一方で気象業界ではもっぱら経緯度座標系が用いられ、度角単位での格子が多い。

次に時間についてみると、GIS は地図調製（数年から数十年間隔で編纂される）から発達してきたため、データに内在する時間軸を取り扱えないことがほとんどである。一方で気象情報では多くの場合 2 つの時間軸（参照時刻又は発表時刻と、解析や予報の対象時刻）がある。NuSDaS のデータベース的機能と同様に、アンサンブルメンバーや鉛直面などもあわせて粒度の差を吸収する機構が必要となる（たとえば OGC 2016）。

GIS 業界では共通のデータ流通フォーマットとしてはしばしば TIFF や PNG などの一般用の画像データ形式が用いられる。グレイスケール画像とは 2 次元格子に明暗の数値を割り当てたものだから、格子位置を別途与えてやればラスタデータ形式として使えるわけである。特に衛星リモートセンシング分野では（気象業界と対照的に）等値線図や矢印表示のような線画表示が情報量を欠落させるものとして忌避され、もっぱら明暗として画像表示することをよしとする文化的背景もある。画像ファイルに格子位置を付加するためには別ファイルを添付することもあるが、TIFF 形式の中に情報を書き込む機構を活用した GeoTIFF も用いられる。

GIS 業界の努力は特殊なファイル形式を作ることより、むしろ画像形式データを提供する様態に向かっている。利用者が希望する経緯度範囲のデータまたは画像を切り出す OGC Web Map Service という標準 API が 2000 年代に流行した。近年はサーバ負荷を軽減するため、あらかじめ決められた位置のタイルに分割した画像を提供し、利用者側で組み合わせる技法（タイリング tiling）が主流となっている（出口・伊藤 2016、前掲）。コンテンツ配信ネットワーク（CDN、キャッシュサーバをインターネットに展開するサービス）と組み合わせることによって、利用者増による提供経費増を大きく抑えることができる。

(6)　ウェブ標準の台頭と未来

CDN とウェブに基づくオープンデータの普及により、ウェブブラウザで無理なく取り扱えるデータ形式が、多少表現力が貧弱であったとしても選好される風潮が強まっている。具体的に言えばデータ構造によって木構造は JSON (Bray 2014)、表構造は（多少の異論はあるものの）CSV、ラスタ画像は PNG 又は JPEG を可能な限り使おうとするのである（念のため、JPEG2000 は全く使われない）。ブラウザ上ではプログラム言語として JavaScript しか選択肢がない状況であり、特に木構造データについては JavaScript のコードをそのままデータ形式とした JSON に優位性がある。

これは別の言葉でいえば GIS 業界に限らず業界固有フォーマットが廃れて分野間情報流通がさかんになる歴史的潮流である。2000 年代半ばに著者は気象業界固有のデータフォーマットが GIS 標準に併呑されてゆくと予想したが、現実は予想を超えた速さで展開している。気象データについていえば多次元格子データがウェブ標準ではカバーできない固有の形式といえるが、それも画像や動画形式で流通するようになるかもしれない。今後が楽しみである。

参考文献

Bray, T. Ed., 2014: The JavaScript Object Notation (JSON) Data Interchange Format. *IETF RFC 7159.*

Butler, H., M. Daly, A. Doyle, S. Gillies, S. Hagen, and T. Schaub, 2016: The GeoJSON Format. *IETF RFC 7946.*

出口智恵, 伊藤裕之, 2016: 「地理院地図」の 3 つのオープン施策. 国土地理院時報　第 128 集.

Eaton, B., J. Gregory, B. Drach, K. Taylor, S. Hankin, J. Caron, R. Signell, P. Bentley, G. Rappa, H. Höck, A. Pamment, and M. Juckes, 2011: *NetCDF Climate and Forecast (CF) Metadata Conventions (1.1 Goals).* Version 1.6 ed., `http://cfconventions.org/`.

ESRI, 1998: ESRI Shapefile Technical Description. July 1998. `https://www.esri.com/library/whitepapers/pdfs/shapefile.pdf`.

Ingleby, B., P. Pauley, A. Kats, J. Ator, D. Keyser, A. Doerenbecher, E. Fucile, J. Hasegawa, E. Toyoda, T. Kleinert, W. Qu, J. St. James, W. Tennant, and R. Weedon, 2016: Progress toward High-Resolution, Real-Time Radiosonde Reports. *Bull. Amer. Meteor. Soc.*, `http://dx.doi.org/10.1175/BAMS-D-15-00169.1`.

ISO, 2007: Geographic information Geography Markup Language (GML). *ISO 19136:2007.*

ITU-T, 2007: Common Alerting Protocol (CAP 1.1). *Recommendation X.1303.*, `https://www.itu.int/`

rec/T-REC-X.1303-200709-I/en.

気象庁予報部, 2010: GSM ガイダンスの変更について. 配信資料に関する技術情報（気象編）第 316 号.

気象庁予報部, 2014: MSM ガイダンスの提供開始について. 配信資料に関する技術情報（気象編）第 389 号.

宮本健吾, 2005: 適合ガウス格子. 数値予報課報告・別冊第 51 号, 気象庁予報部, 39–42.

OGC, 2016: OGC Best Practice for using Web Map Services (WMS) with Ensembles of Forecast Data. Seib, J. et al. ed. OGC draft Best Practice 16-086r2.

太田行哉, 2015: 従来型観測データの利用の現状と課題. 数値予報課報告・別冊第 61 号, 気象庁予報部, 3–8.

佐藤信夫, 1982: スペクトルモデルの数学的基礎. 電子計算室報告・別冊第 28 号, 気象庁予報部, 38–66.

杉山善昭, 竹田康生, 山腰裕一, 清本真司, 中村政道, 長谷川昌樹, 平原隆寿, 横井貴子, 2012: 気象庁防災情報 XML フォーマットの詳細と策定経緯. 気象庁測候時報, **79**.

豊田英司, 原旅人, 横井信太郎, 田浦俊太郎, 長谷川昌樹, 2012: NuSDaS（数値予報標準データセットシステム）バージョン 1.3. http://www.mri-jma.go.jp/Project/cons/data/nusdas13.pdf.

Toyoda, E., 2011: Units of Measurement of GRIB2 parameter and Accumulation. *WMO/CBS/IPET-DRC-III Doc.2.2(3)*, http://www.wmo.int/pages/prog/www/ISS/Meetings/IPET-DRC_Melbourne2011/Documents/IPETDRC-III_Doc2-2_3_AccumulationUnits.doc.

WMO, 2015: Manual on Codes. *WMO Pub.*, No.306. 2011 ed. Updated in 2015. ISBN 978-92-63-10306-2.

WMO, 2016: Manual on the WMO Information System: Annex VII to the WMO Technical Regulations. *WMO Pub.*, No.1060. 2015 ed. Updated in 2016. ISBN 978-92-63-11060-2.

山本孜, 1966: 世界気象監視（World Weather Watch WWW）について. 天気, **13**, 1–10.

4.2 格子点値データフォーマット [1]

数値予報システムでは、解析に用いる観測データ、その観測データの品質管理情報、解析やモデル予測の格子点値など、さまざまなデータが用いられている。本節では、気象庁の数値予報システムで利用されている格子点値データフォーマット NuSDaS を中心に、他の格子点値フォーマットとの比較も含めながら、利用する観点からそれらの論理（概念）モデル、物理（実装）モデル、そして NuSDaS という独自のフォーマットを気象庁で利用する背景について紹介する。また、NuSDaS の歴史とそれから得られた教訓についても触れる。

4.2.1 格子点値データのフォーマットの比較

第 4.1 節で紹介されたように、格子点値の格納には、世界気象機関 (WMO) 基礎システム委員会 (CBS) によって標準化されている GRIB/GRIB2、米国 Unidata で開発されている NetCDF (Network Common Data Form)[2] などの汎用のフォーマットが知られている。また、格子点値の可視化ツール GrADS ではコントロールファイルと呼ばれるテキストファイルと、データを単純に並べたバイナリファイルで構成されたファイル群を標準の入力フォーマットとしている。一方、気象庁では NuSDaS (Numerical Prediction Standard Data-set System) と呼ばれる独自フォーマットを使用している。

NuSDaS の詳細について紹介する前に、まず、これらの格子点値データフォーマットをいくつかの観点で分類してみる（表 4.2.1）。

(1) API の提供の有無

それぞれのデータにフォーマットが規定されていることは共通であるが、フォーマットによっては、データの作成や読み取りを行う関数（API と呼ばれる）がフォーマットとともに提供されており、フォーマットの内部仕様（物理モデル）を詳しく知らなくても、API の利用を習熟することでデータの作成や読み書きができる。また、API を提供しているフォーマットでは、何らかの事情によってフォーマットを変更する必要がある場合にも、（API に変更がない限りは）ユーザはライブラリだけを取り替えれば、そのフォーマットの変更を意識する必要がなく、ユーザのプログラムに修正を加える必要がないことも API が提供されることの利点の一つである。NuSDaS や NetCDF がこれに該当する [3]。一方、API が提供されないフォーマットについては、データの読み書きの際にフォーマットに基づいて専用のプログラムを自ら作成する必要があり、その段階で誤りが混入する場合もある。

なお、NetCDF と NuSDaS は API の提供はされているが、NetCDF は次元や変数の定義についても API が提供されているのに対し、NuSDaS ではこれらの定義を NuSDaS 定義ファイルと呼ばれるテキストファイルを通じて行うことから、これに対応する API がないという違いがある。

(2) データの基本単位・データの基本格納単位

データを読む際には、開くファイルまたはディレクトリを指定する必要がある。GRIB/GRIB2, NetCDF, GrADS では、ファイルに対して開く操作を行うのに対し、NuSDaS はディレクトリに対して行う。

数少ないファイル、またはディレクトリを扱うのであれば、取得したいデータを得るためにどのファイルやディレクトリを開けばよいかを決めることは難しくないだろう。しかし、気象庁の数値予報システムでは、非常に多くの種類のデータセットや、初期時刻・対象時刻が違うデータセットを取り扱う。これら多数のデータをトラブルなく扱うためには、開くファイルやディレクトリを機械的に特定できる仕組みが必要である。

データセットにおける主キー

格子点データには、一般にデータセット名（モデルの種類なども含む）、時間（初期時刻、対象時刻）、面、要素などの属性があり、属性の組によって格子点データが一意に定まる。一意にデータを定める属性、またはその組のことをデータベースの世界では「主キー」（または単に「キー」）と呼ぶ。一般には、主キーの一部から開くファイルやディレクトリの名前が特定され、残りの主キー（ファイル名と関連する主キーが重複してもよい）によってファイルまたはディレクトリのどの部分のデータに対応するのかが特定される。どの主キーをファイル名・ディレクトリ名に対応させるかは、フォーマットの仕様やユーザの設計に依存する。

NetCDF の場合

NetCDF では、任意の次元のデータを取り扱えるので、すべての主キーを次元に対応させれば、データ変数配列のインデックスによってデータを一意に定めることができる。そのため、原理上は一つのファイルだけですべてのデータを保持することが可能である（ファイルが一つであれば、その名前は何であってもよく、主キーによるファイル選択の必要はない）。しかし、一つのファイルですべてのデータを扱うのは物理的なファイルのサイズ制限、必要な一部のデータだけをファイル単位で取り出すことができないこと、あらかじめ主キーとなる属性の種類数（次元の数に相当）を決めておく必要があり途中で変更することが面倒であること、あらかじめ決めた次元数のデータ格納領域が最初から確保されてしまうことなどの困難を伴い、現実的では

---

[1] 原 旅人

[2] `http://www.unidata.ucar.edu/software/netcdf/`

[3] API が提供されているフォーマットでは、内部仕様を意識せずに読み書きができるとされる。NuSDaS ではファイルそのものの内部仕様を意識する必要はないが、データセットがファイルではなくディレクトリとなっているため、定義ファイルやデータファイルのディレクトリ内での配置については一定の知識が必要な場合がある。

表 4.2.1 代表的な格子点値データフォーマットの比較

	API の提供	データの基本形式	データの基本格納単位
GRIB/GRIB2	公式にはない （ECMWF によって開発された API はあり）	ファイル	水平 2 次元
NetCDF	あり	ファイル	任意の次元
GrADS	なし	ファイル	水平 2 次元
NuSDaS	あり	ディレクトリ	水平 2 次元

ない。そこで、データセット種別、初期時刻、対象時刻などでファイルを分割することが多い。つまり、主キーのうち、データセット種別、初期時刻、対象時刻などがファイル名と一意に対応するように設計するのである。ファイル名と対応させる主キーの選択については NetCDF のファイルフォーマットそのものはなにも制約を課しておらず、ユーザの裁量で決める必要がある。

GRIB/GRIB2 の場合

GRIB/GRIB2 の場合は、水平 2 次元データが格納の基本単位であり、任意の次元の変数を格納できる NetCDF とは事情が異なるが、各水平 2 次元データに上に挙げた主キーの情報を属性情報として付加している。そのため、GRIB/GRIB2 においても、1 つのファイルにすべてのデータを保持することが原理的には可能である。しかし、NetCDF と同様の事情に加え、標準の GRIB/GRIB2 データには各データの位置を示すインデックスがないので、必要なデータを先頭から探査する必要があるという困難がある。それはデータ量が大きくなるほど検索の効率を落とすことになり、データ量が多くなると実用に耐えない。したがって、GRIB/GRIB2 の場合も、データセット名、初期時刻、対象時刻などの主キーの一部をファイル名に割り当てているのが通常である。

GrADS データの場合

GrADS データファイルの場合には、記述できる時刻列は 1 つであり、ファイル名に初期時刻の情報を入れて、データファイルに記述できる時間の次元に対象時刻を対応させる使い方が多いようである。いずれにせよ、この場合も開くファイルが特定できるような命名規則をユーザが別途定める必要があることは、NetCDF, GRIB/GRIB2 と同じである。

NuSDaS の場合

上の 3 つのフォーマットに対し、NuSDaS ではユーザが NuSDaS データのディレクトリ（任意の名前で可）を `NUSDASXX`（`XX` は 2 桁の数字）という名前でシンボリックリンクを張るだけでよく、データセットの命名規則は必要ない。このことは、データセットの命名規則にユーザの裁量の余地を与えないようにしていることの反映である。

他のフォーマットに比べ柔軟性には欠けるが、なるべくユーザに裁量の余地を与えず、誰がやっても同じ結果が得られるようにしている。これが気象庁の数値予報ルーチンで用いられるフォーマットやツールの共通の特徴の一つである。

4.2.2 気象庁の格子点値フォーマット: NuSDaS

すでに述べたように、気象庁の数値予報ルーチンでは、格子点値データフォーマットとして、NuSDaS と呼ばれる独自フォーマットが利用されている。以下では NuSDaS のデータモデル、および独自のフォーマットが必要とされた背景について説明する。なお、NuSDaS の利用方法等については、マニュアル [4] を参照していただきたい。

(1) NuSDaS API

すでに述べたように、NuSDaS はさまざまな読み書きのための API を用意しており、フォーマットの仕様の詳細をユーザは知る必要はなく、API の使い方を習得すればよい。これには、API を正しく利用してデータの読み書きを行えば、誰もが同じ結果が得られることを目指していることが背景にある。

ただし、NuSDaS はディレクトリを単位としたデータセットであり、NuSDaS 定義ファイルと呼ばれる設定ファイルやデータファイルのディレクトリ内での配置については一定の知識が必要となる場合がある。これらの知識があれば、たとえば、初期時刻ごとに別の NuSDaS データセットとして格納された複数のデータセットを、ファイルの移動だけで一つのデータセットに統合することも可能となる。

(2) NuSDaS の論理（概念）モデル

NuSDaS は、以下の項目を主キーとしている。

- 種別 1（8 文字）（例:`_GSMRGET`, `_MSMLMLY`）
- 種別 2（4 文字）（例:`FCSV`, `AASV`）
- 種別 3（4 文字）
- 基準時刻（4 バイト整数）
- メンバー（4 文字）
- 対象時刻（4 バイト整数）
- 面名（6 文字）（例:`SURF`, `925`）
- 要素名（6 文字）（例:`PSEA`, `T`）

[4] `http://www.mri-jma.go.jp/Project/cons/data/nusdas13.pdf`

- データセットの ID 番号（01〜99、上の項目だけでは区別できない場合に指定）

種別 1 にはモデル名（1〜4 文字目）、水平座標種別（5〜6 文字目）、鉛直座標種別（7〜8 文字）の情報が、種別 2 にはデータの種類（1〜2 文字目、解析値、予報値などの種別）、データの時間種別（3〜4 文字目、瞬間値、積算値などの種別）が記述されている。種別 3 は任意の文字列をユーザが設定できる。メンバーはアンサンブル予報のメンバーの名前やレーダーのサイト名などの指定に活用できる。また、基準時刻はモデル予測値であれば初期時刻に、対象時刻は予測対象時刻に対応する。これらは、1801 年 1 月 1 日 00:00UTC からの経過時間（分単位）によって記述する。これらの項目によって 2 次元のデータが一意に指定されることは、1 つの 2 次元データを読み出す関数である `nusdas_read` のインターフェース（引数）でこれらを指定する必要があることからも分かるだろう。

NuSDaS は通常、面のデータを単位として格納している。これはキーに「面名」が含まれていることから理解できる。これは GRIB/GRIB2 と共通する部分である。一方、NetCDF では、データを一意に指定するためのキーはファイル名と変数名であり、変数の次元数はユーザが任意に設定できる。面は変数の一つの次元によって表現されるのが普通である [5]。また、時間の情報を変数の一つの次元によって表現してもよいし、ファイル名によって指定されるようにしてもよい。NetCDF と比べると、NuSDaS はファイル名や変数の次元数など、ユーザの裁量範囲が小さいことが分かるであろう。目的の用途を満たした上で、ユーザの裁量範囲を小さくすることが、数値予報ルーチンで用いられるツールに求められることである。これは、これまでに説明してきた気象庁独自のフォーマットや JCL, PBF（第 3.2 節）などの記述方法に共通することであり、一般的なフォーマットのように汎用性を持たせて適用範囲を広くすることとは対照的に、ユーザの裁量をあえて小さくすることで誰がやっても同じ結果になることを重視している。

(3) NuSDaS の物理（実装）モデル

すでに述べたように、ユーザは NuSDaS の物理モデルの詳細について知る必要は基本的にはないが、その知識があると利用の際の助けになることもある。

先に述べた論理構造を実現するための物理モデル（ファイルの配置、ファイルのフォーマットなど）は以下のようになっている。

- NuSDaS は単一のファイルではなく、ディレクトリデータセットと呼ばれるディレクトリを単位としたものである。その最上位のディレクトリを NuSDaS Root Directory (NRD) と呼び、`NUSDASXX`（`XX` は 01〜99 の数値）という名前にしておく。
- 種別 1, 2, 3 については、 NRD の直下にある `nusdas_def` というディレクトリに格納されている NuSDaS 定義ファイルに記述された `type1`, `type2`, `type3` によって、その NRD に格納されている種別 1, 2, 3 を特定する。
- 基準時刻はディレクトリ名、またはファイル名で指定される。基準時刻をキーにしないようなデータセット（解析値、観測値など）では省略できる。この設定は、NuSDaS 定義ファイルの `path` でユーザが指定する。
- メンバー、対象時刻については、それらを指定するための実装をユーザが選択することができる。面名や要素名のように 1 つのファイルの中に複数を格納できるようにしてその中から選択するようにしてもよいし、ディレクトリ名やファイル名によって指定できるようにしてもよい [6]。
- 面名、要素名については 1 つのファイルに複数のものを格納することができる。ファイル名やディレクトリ名によっては指定されない。すなわち、開くファイルは、面名、要素名以外のキーで指定される必要がある。
- データファイルはビッグエンディアン [7] の Fortran 順番探査ファイルになっている。レコードには一般情報（ファイルサイズ、レコード数、作成元情報など）を格納する `NUSD` レコード、メンバー・対象時刻・面名・要素名のリストや地図投影情報が格納された `CNTL` レコード、メンバー、対象時刻、面名、要素名で一意に指定されるデータが格納されている `DATA` レコードのファイル上での位置（ファイルの先頭からのバイト数）を格納している `INDX` レコードまたは `INDY` レコード [8]、1 つの面のデータが格納された `DATA` レコード、ファイルの終端に付加される `END` レコードなどがある。

この物理モデルを踏まえて、NuSDaS ではキーで指定されたデータを読み出す際には以下のような動作をしている。

- `NUSDAS01`, `NUSDAS02` のように `XX` で示される数値の順に探索を行い、ディレクトリが存在した場合

[5] 変数名に面の情報を入れる方法もあり得るが、一般的ではないだろう。

[6] この選択は、NuSDaS 定義ファイルの `path`、`member` や `validtime` に記述する `in` または `out` によって指定する。

[7] リトルエンディアンマシンでファイルを読み書きする場合は、ライブラリがバイトオーダーの変換を行うので、ユーザは実行マシンのエンディアンを気にする必要はない。

[8] `INDX` レコードは 1 つの `DATA` レコードに対して、その位置を 4 バイト整数で表現している。NuSDaS1.0 のときには、その 4 バイトを符号つき整数として解釈していたため、ファイルサイズの上限が約 2 GB であったが、NuSDaS1.1 で符号なし整数で解釈するように修正し、ファイルサイズの上限が約 4GB に拡張された。その後、4 GB を超えるファイル出力が必要になったため、4 バイトで表現していたファイル上の位置を 8 バイトに拡張した。それが `INDY` レコードである。

にはそのディレクトリの直下にある nusdas_def というディレクトリに格納されている NuSDaS 定義ファイルを読み（複数格納されている場合はすべて読む）、指定された種別 1, 2, 3 のデータを格納されているデータセットであるかの判定を行う。

- 種別 1, 2, 3 で指定された NRD が見つかった場合には、その定義ファイルから、ディレクトリやファイルの格納方法を読み取る。具体的には、path、member や validtime の in 指定または out 指定を見る。
- その格納方法情報と要求されたデータのキーから開くべきファイルが格納されたディレクトリ名とファイル名を特定して、そのファイルを開く。一度ファイルを開いたあとは、NuSDaS 定義ファイルの情報は参照されず、ファイル内部に格納された情報を参照する。ファイルが特定された時点で、キーのうち、種別 1, 2, 3、 基準時刻はファイルの選択で反映されたとして、以後参照されない。
- 要求されたデータのキーのうち、メンバー名、対象時刻、面名、要素名から特定されるデータのファイル上での位置を INDX レコードまたは INDY レコードから読み取る。
- 得られたファイル上の位置から DATA レコードを読み出し、そのデータのパッキング（圧縮の有無や方法）や欠損値の取扱方法を解釈して、格子点値配列をユーザに返す。

データをファイルに書き出す際は、すでにファイルが存在している場合は、書き込むファイルの選択の過程は上の読み取りの場合と同じである。ファイルが存在していない場合には、読み取りの場合と同じアルゴリズムで書き出すファイルを特定したのち、定義ファイルに書かれたメンバー名、対象時刻、面名、要素名、地図投影方法などの情報をファイルに転記する。転記後は定義ファイルの内容は参照されない。あとは、ファイルが存在していた場合と同じである。

(4) 独自フォーマットを採用する背景

一般的なデータフォーマットでは、さまざまなデータを汎用的に格納できることを求めることが多いだろう。NetCDF はまさにその代表的なもので、気象データに限らず、さまざまな分野での格子点データの格納に用いられている。NetCDF は、データの次元数や属性情報をユーザが任意に設定できるため、たとえ API を用いたとしても、人によってその格納方法は様々になり得る。つまり、NetCDF は “箱” を与えているだけで、どのようにデータを格納するのかについては何も規定していないのである。そのため、どのようにデータを格納するのかについては、フォーマットと別に規約 (conventions) がデータ利用コミュニティごとに定められていることが多い。気象のデータについては CF conventions[9] がよく使われている。この規約に基づいたデータの格納は、NetCDF API を駆使してデータの作成者が行う必要があり、規約に基づく実装を行うには NetCDF の API や規約に精通する必要がある [10]。

NuSDaS は、データを読み書きする API を提供しているという面では NetCDF と共通点を持つ。一方、次元の扱いの裁量の範囲を小さくするという観点から面を単位とする読み書きをするのは GRIB や GRIB2 の流れを汲むものである。さらに、データの選択に使われるキーの情報の一部（モデル、データ種別、基準時刻）を柔軟にユーザが設定したり変更することができてしまうファイル名に入れることを求めず、そのキーの情報もデータセット内部に内包している。つまり、NuSDaS は他のフォーマットに比べて、データベース的な機能が強化されているということができる。

他の汎用フォーマットが持つ柔軟性を廃して数値予報ルーチンの運用での利用を想定したものとし、API を使う限り、想定したルールでデータの読み書きが確実にできるようになっているのが NuSDaS であるということができる。

(5) NuSDaS の歴史とそれからの教訓

NuSDaS の誕生

NuSDaS は、数値予報ルーチンでの利用に特化した格子点値フォーマットとして 2001 年の第 7 世代スーパーコンピュータシステムへの更新とともに誕生した。数値予報ルーチンにおける格子点値データの格納を NuSDaS で行うことが規定されたため、自然と気象庁本庁のモデル開発においても NuSDaS を使うことになった。

NuSDaS の誕生当初は、数値予報ルーチンの環境での動作が確認されれば充分であると認識され、たとえば、当時の計算機システムのバイトオーダーであるビッグエンディアンにのみ対応し、x86 マシンなどのリトルエンディアンの環境では利用できなかった。また、気象研究所の計算機システムでは NuSDaS ライブラリのコンパイルさえもできない状況であり、その結果、気象研究所で NuSDaS はほとんど利用されなかった。そのために、NuSDaS 形式データの利用は現業数値予報システムを開発および運用している気象庁本庁の計算機システム上に限定されてしまい、気象庁本庁と気象研究所という気象庁の組織内でさえも格子点値データフォーマットが統一されないという不幸な状況を生み出してしまった。

---

[9] http://cfconventions.org

[10] CF conventions に基づいた NetCDF データであるかをチェックするウェブサイトがある:http://puma.nerc.ac.uk/cgi-bin/cf-checker.pl。このようなチェックサイトが存在していることは、CF conventions に基づいた実装が必ずしも容易ではないことの反映とも言える。

NuSDaS の移植性の向上と全面的なソースコードの書き換え

2002 年ごろからリトルエンディアンマシンでの動作などの移植性を高めた NuSDaS である pnusdas の開発が庁内のシステム開発の専門家とユーザの一部によって進められ、2003 年にほぼ完成した。しかし、数値予報ルーチンで利用されているライブラリとは開発が別々に行われており、一方からバグの発見や機能拡張があっても他方に反映されることは少なかった。

2006 年 3 月から運用された第 8 世代スーパーコンピュータシステムにおいては、その構成機器の一部にリトルエンディアンマシンがあり、数値予報ルーチンの運用でもリトルエンディアン対応に迫られた。そこで、その機会に乗じて、別々に開発されてきた数値予報ルーチン版 (NuSDaS1.0) と pnusdas を統合した（さらに最適化を行い、より高速に動作するようにしている)。これが NuSDaS1.1 と呼ばれているもので、第 8 世代スーパーコンピュータシステムにおいて広く使われた。

NuSDaS1.1 がリリースされる以前から、NuSDaS のソースコードの可読性の低さとそれに伴う維持管理の困難が懸念されていたが、その後、機能はほぼそのまま温存しつつ（一部の拡張は行われた)、オブジェクト指向的な考えを取り入れながら可読性を高め、ソースコードの完全な書き換えが行われたものが NuSDaS1.3 としてリリースされた。NuSDaS1.3 は 2012 年 6 月から運用中の第 9 世代スーパーコンピュータシステムにおける NuSDaS 標準ライブラリとなっている。

このように、移植性の向上によって利用できる開発環境が広がるなどユーザの利便性が高まるとともに、維持管理がしやすいライブラリへと改良された。

NuSDaS の歴史を踏まえた教訓

現在では、NuSDaS だけでなく、JCL, PBF などの数値予報ルーチン向けに開発され維持されているツール類について、モデル開発にも活用できるように必要な拡張が行われている。このように、ルーチンシステムのためのツール等が開発でも広く活用できるようになって、ルーチンシステムとは別のモデル開発環境を整備・維持する必要がなくなり、効率的なモデル開発に寄与している。

また、気象庁の数値予報システムで汎用フォーマットではない独自のフォーマットを用いていることは、部外とのデータ交換を阻害しているとの批判が庁内外にある。NuSDaS が当初の数値予報ルーチンに特化したものであるという立場から、移植性の確保や詳しいドキュメントの整備が積極的に行われてこなかったために、ライブラリのコンパイルができない、API や使い方のドキュメントが乏しいという状況になっており、そのような批判が出るのはやむを得ない部分があった。しかし、このような状況が効率的なモデル開発や部外とのデータ交換を阻害していることが認識されるようになって、ライブラリの移植性が高められ、ほとんどの環境で利用できるようになったり、詳しい API の解説書が作成されるなど、状況は変わりつつある。

これまでの数値予報ルーチンシステムとモデル開発を振り返ると、ユーザの裁量の自由度を制約するために数値予報ルーチンシステムのための独自のフォーマットやツールを作成する際には、数値予報ルーチンシステムに特化したものとして開発されることが多かった。しかしながら、それをモデル開発にもできる限り利用したほうが効率的であることから、開発にも利用されるようになったと言える。それを踏まえると、モデル開発者、システム開発者が連携して、数値予報ルーチンの維持管理にも効率的なモデル開発にも有用なツールを開発していくことが重要になっていると考えられる。

一方、モデル開発者がモデル開発に専念できるように、システム面をその専門家に任せるという流れの中で、ツールやライブラリの開発そのものは専門家に任せるとしても、それらを使いこなすことはモデル開発者が習得すべき基礎的な技術である。解説書があれば提供されたツールやライブラリの使用方法を習得できるといった基礎的な技術研鑽にモデル開発者が努めるとともに、システム開発者側でもその支援の一つとしてツール・ライブラリの移植性の向上と解説文書の整備がより必要になると考えている。

4.3 数値予報システム周辺でよく使われる可視化ツール・ライブラリと気象庁での利用 [1]

数値予報システムの入出力に用いられるデータには格子点データや観測データなどがある。これらの膨大なデータから有効な情報を取り出す手段の一つとして地図上でのプロットやグラフの描画などの可視化は広く行われ、数値予報システムの運用や開発には必要不可欠である。

4.3.1 気象分野でよく利用される可視化ツール

すでに述べたように、数値予報システムが扱うデータのフォーマットは多種多様なものが使われているが、可視化ツールも多種多様である。気象の分野でよく用いられる可視化ツール・ライブラリの一覧を表 4.3.1 に示した。

4.3.2 可視化ツールの開発

表 4.3.1 のツールの中には、欧州中期予報センター (ECMWF) が開発している Magics や MetView、英国気象局 (UKMO) が開発している IDL の拡張ライブラリや Iris、米国大気研究センター (NCAR) が開発している NCAR Graphics や NCL のように、数値予報センターが自ら使うために開発したものがある。原・高谷 (2013) で紹介したように、ECMWF や UKMO においては、組織で統一した可視化ツールを採用し、それを便利に利用できるようにするためのライブラリを自ら開発している [2]。可視化ツールを組織で統一することによって、そのツールやライブラリの開発に集中的に資源を投入することができ、また、利用方法について組織内で研修を行うこと、開発元に直接質問や要望を送ることができること、ユーザ間で密に情報交換をすることができるなどの利点がある。

一方、気象庁では ECMWF や UKMO のように組織で統一して利用したり開発したりしている可視化ツールはなく、表 4.3.1 に挙げたツールのいくつかが開発コミュニティや用途に応じて利用されている。ECMWF や UKMO のように組織的に機能の拡張に資源を投入することは難しいものの、既存のツールをその特徴を踏まえて用途、データの容量、フォーマット、ディスクやネットワークの資源に応じて適切に使い分けることで、開発コストを減らしている。また、世界中のユーザが公開している各ツールの利用法についての情報を参考にできる点は利点の一つと言える。

4.3.3 数値予報課における可視化ツールの利用

数値予報課では、格子点データの描画には GrADS (第 4.4 節)、GMT (第 4.5 節)、PANDAH (第 4.7 節)、NCL などが使われている。最近では、ウェブブラウザによる格子点データの表示を高速に行うためのツールである TAG が数値予報課で開発され (第 4.6 節)、それを活用したウェブモニタが多数開発されている。また、グラフの描画には、GrADS や GMT の他に、gnuplot、R などが使われている [3]。なお、ECMWF や UKMO では利用が盛んな Python を用いた描画は気象庁ではあまり利用されていないが、最近になって一部で活用が始まっている（第 4.7 節）。

以下の節では、数値予報課で活用されている代表的な格子点データの可視化ツールについてその特徴、用途、利用例、課題などを紹介する。

参考文献

原旅人, 高谷祐平, 2013: 海外数値予報センターの開発管理の例. 数値予報課報告・別冊第 59 号, 気象庁予報部, 195–199.

加藤輝之, 2004: PostScript コードを生成する描画ツール “PLOTPS” マニュアル. 気象研究所技術報告第 44 号.

[1] 原 旅人

[2] モデル開発者とは別の可視化の専門チームが開発している。

[3] R は可視化だけでなく、数値予報課アプリケーション班によるガイダンスの開発で大いに活用されている。

表 4.3.1 気象の分野でよく使われる主な可視化ツール・ライブラリ

	特徴・用途など	利用できる入力データ	公式ウェブサイト、文献など
格子点データ・グラフ描画ツール			
TAG	数値予報課で開発されているラスタ形式での高速描画ツール。設定ファイルは TAG によってアプリケーション的に記述する。	NuSDaS, （一部の）GRIB2	—
PANDAH	数値予報課で開発されている描画ツール。モニタ図やメソモデル開発などに利用。描画ライブラリに PLOTPS を用いている。	NuSDaS、テキストファイルで記述された地点データ（観測データ、台風進路データなど）	—
kplot, mplot	気象研究所で開発されている描画ツール。描画ライブラリに PLOTPS を用いている。	NuSDaS (kplot)、JMA-NHM の独自フォーマット出力（MRI 形式）(mplot)	—
GMT	ハワイ大学で開発されている描画のためのツールおよび描画のためのデータセット処理ツールの集合。多くの場合、複数のツールを組み合わせて利用する。詳しくは第 4.5 節を参照	テキストデータ、バイナリデータ、NetCDF	http://www.soest.hawaii.edu/gmt/
GrADS	COLA で開発されている気象関係のデータの扱いに特化したデータ描画ツール。詳しくは第 4.4 節を参照	バイナリデータ、GRIB1/2、NetCDF	http://cola.gmu.edu/grads/
NCL	NCAR で開発されているデータ解析、描画のためのスクリプト言語。描画ライブラリに NCAR Graphics を用いている。	NetCDF, GRIB1/2, HDF など	https://www.ncl.ucar.edu
MetView	ECMWF で開発されている GUI を併せ持った描画ツール。スクリプトによるバッチ処理も可能。描画ライブラリに Magics を用いている。	GRIB1/2, NetCDF, BUFR など	https://software.ecmwf.int/wiki/display/METV/Metview
gnuplot	有志によって開発されている 2 次元および 3 次元のグラフの作成ツール	主にテキストデータ	http://www.gnuplot.info
R	有志によって開発されている統計解析ツール。グラフ等の描画にも利用可能。	テキストファイルなど。拡張ライブラリによる拡張可	https://www.r-project.org
MATLAB	商用の技術計算言語。可視化だけでなく、強力な計算機能、データ解析機能を持つ。	テキストファイル、NetCDF、HDF など。外部ライブラリによる拡張可。	https://www.mathworks.com/products/matlab/
IDL	商用の技術計算言語。可視化だけでなく、強力な計算機能、データ解析機能を持つ。UKMO で利用されている。	NetCDF, HDF, GRIB など。外部ライブラリによる拡張可。	http://www.harrisgeospatial.com/ProductsandSolutions/GeospatialProducts/IDL.aspx
ライブラリ			
PLOTPS	気象研究所で開発されている PostScript を出力する描画ライブラリ。FORTRAN77 で記述されている。	他のデータ読み出しライブラリを組み合わせる。	加藤 (2004)
Magics	ECMWF で開発されている描画ライブラリ。C、C++、Fortran、Python のインターフェースを持つ。	GRIB1/2, NetCDF, BUFR などの読み出し機能を内包。他のデータ読み出しライブラリを組み合わせることで他のフォーマットも入力可能。	https://software.ecmwf.int/wiki/display/MAGP/Magics
matplotlib	有志によって開発されている Python の描画ライブラリ。MATLAB と似た使い方を目指している。	他のデータ読み出しライブラリを組み合わせる。	http://matplotlib.org
Iris	UKMO で開発されている地球科学データ向けの Python のライブラリ。描画部は matplotlib を利用。	GRIB, NetCDF, PP（UKMO の独自フォーマット）はデータの読み出しおよび投影法情報などのメタデータの自動設定に対応。その他についても、他のデータ読み出しライブラリを組み合わせてデータを読み出して自らメタデータを設定すれば利用可能。	http://scitools.org.uk/iris/
NCAR Graphics	NCAR で開発されている描画ライブラリ。C、Fortran のインターフェースを持つ。	他のデータ読み出しライブラリを組み合わせる。	http://ngwww.ucar.edu
PyNGL, PyNIO	PyNGL は NCAR で開発されている NCAR Graphics の Python インターフェース。	PyNIO と組み合わせることで、NetCDF, GRIB などの読み出しが可能。	https://www.pyngl.ucar.edu
DCL	地球流体電脳倶楽部で開発されているライブラリ。C、Fortran、Ruby のインターフェースを持つ。	他のデータ読み出しライブラリを組み合わせる。	http://www.gfd-dennou.org/library/dcl/

4.4 可視化ツール (1)–GrADS[1]

4.4.1 はじめに

GrADS (Grid Analysis and Display System) は、ジョージ・メイソン大学海洋陸面大気研究センター (COLA: Center for Ocean-Land-Atmosphere Studies) で開発されているオープンソース、クロスプラットフォームの描画ツールである。コンパイル済み実行形式が用意されていて、多くのプラットフォームで比較的簡単に動作させることができる。Doty and Kinter III (1995) で述べられているように、科学者が大量の地球科学データを一般的なフォーマットで容易に扱えるように設計された。さらに、当時の一般的な（地球科学向きではない）商用ソフトウェアでは難しかった、データの解析にも重点が置かれている。これは、GrADS の省略しない名称に “Analysis” が含まれることからも想像できる。なお、最新の情報やマニュアルについては公式ウェブサイト [2] を参照されたい。

GrADS では多次元のデータを容易に扱うことができる。また、X Window System による対話的な表示を行うことができ [3]、描画結果を基に図の表示領域を移動・拡大したり、断面図を取得したりするなどの作業が容易である。描画要素同士の演算（例えば東西風と南北風から風速を求めるなど）も中間記法で容易に記述できる。このため、GrADS は数値予報課でよく使われる描画ツールの一つである。本節では、GrADS の入出力ファイルや描画の仕様などについて述べた後、その特徴について説明する。

4.4.2 バージョンと互換性

安定版であるバージョン 2.1.0 が 2016 年 6 月にリリースされている。2013 年 12 月にリリースされたアルファ版の 2.1.a1 以降、X Window System での描画及びファイル出力は cairo ライブラリ [4] によってハンドリングされており、2012 年 11 月にリリースされた一つ前の安定版であるバージョン 2.0.2 とはフォントの扱いや出力フォーマットに大きな変更があるが、多くのコマンドは上位互換性を有している。バージョン 2.1.0 では、cairo ライブラリを使用した実行プログラムが `grads`、使用していないものが `xgrads` として提供されている。

4.4.3 ユーザインターフェイス

GrADS はユーザインターフェイスとして CUI (character user interface) を提供する。ユーザはコマンドによる対話型処理を行い、X Window System による描画表示を行うことができる。対話型処理ではコマンド履歴を利用することができ、設定により、履歴をファイル出力・再利用することができる。

また、スクリプト言語 GSL (GrADS scripting language) をファイルに記述することにより、関数として利用したり、バッチ処理を行うこともできる。GSL では対話型処理と同様の処理のほか、算術演算、関数定義、条件分岐、反復処理などのプログラミングを行うことができ、GSL で記述されたスクリプトが GrADS Script Library として公式ウェブサイトで公開されている。シェルの環境変数 `GASCRP` の設定により、あるディレクトリに配置したユーザ定義のスクリプトを関数として読み込むことができるので、あらかじめ汎用的なスクリプトを作成しておけば再利用することも可能である。

さらに、GSL ではマウスでのポイント・アンド・クリックにより X Window の座標を取得することができる。これを利用して、簡単な GUI (graphical user interface) を作成することもできる。

4.4.4 入力データ

描画用の入力データのファイルフォーマットは以下に対応している。

- バイナリ格子データ（4 バイト浮動小数点数、4 バイト符号付き整数、2 バイト符号付き整数、2 バイト符号無し整数、1 バイト符号無し整数のいずれか）
- GrADS station データ（地点データ用の GrADS 独自バイナリフォーマット）
- GRIB (GRIdded Binary or General Regularly-distributed Information in Binary form) 第 1, 2 版 [5]
- NetCDF (Network Common Data Form)
- HDF (Hierarchical Data Format) バージョン 4, 5
- BUFR (Binary Universal Form for the Representation of meteorological data)[6]
- SHP (shapefile)

---

[1] 江河 拓夢

[2] `http://cola.gmu.edu/grads/`。なお、2016 年 2 月にドメインが `iges.org` から変更となっている。COLA はもともと IGES (Institute of Global Environment and Socienty) の下にあったが、ジョージ・メイソン大学 (GMU) に移管された。マニュアルや古いウェブ資料などを参照する時には注意されたい。

[3] X Window System が使えない環境でも、バッチモードで利用することができる。

[4] フリーソフトである 2 次元画像描画ライブラリ。X Window System, Quartz, Win32, image buffer, PostScript, PDF, SVG などでの出力をサポートしている。

[5] いわゆる GRIB 及び GRIB2。

[6] BUFR データを読み込むためにはデコード表ファイルが必要である。GrADS に同梱されているデコード表ファイルは長い間更新されておらず、BUFR バージョン 4 や、BUFR バージョン 3 のマスター表バージョン番号 11 以降などには対応されていない。デコード表ファイルを自前で用意する必要がある。

OPeNDAP (Open-source Project for a Network Data Access Protocol)[7] サーバにアクセスし、リモートデータを取得することもできる。

描画用の入力データファイルを読み込むためには、データの内容について記述したテキストファイル（コントロールファイル）も別途必要となる[8]。コントロールファイルには、描画用のデータファイル名、次元の数、変数名、未定義値の設定などを記述する。さらに、通常の緯度経度直交座標とは異なる座標系のデータを入力とする場合には、格子情報を格納したバイナリファイルが別途必要となる場合もある。

このほか、通常のテキストデータを読み込むこともできるが、読み込んだデータを直接描画することはできない。GSL による文字列処理、画面座標に対する変換処理等が必要となる。

4.4.5 出力データ

バージョン 2.1.0 では以下の画像ファイルフォーマットでの出力がサポートされている。

- EPS (Encapsulated PostScript)
- PDF (Portable Document Format)
- PNG (Portable Network Graphics)
- PS (PostScript)
- SVG (Scalable Vector Graphics)

地図情報システム (GIS: Geographic Information System) データとして、以下を出力できる。

- GeoTIFF (Geospatial Tagged Image File Format)
- KML (Keyhole Markup Language)
- SHP (shapefile)

cairo ライブラリを使用しない場合には以下のファイルフォーマットでの出力も可能である。

- GIF (Graphics Interchange Format)
- JPEG (Joint Photographic Experts Group)
- XWD (X Window Dump)

過去のバージョンでは GrADS metacode format file あるいは GrADS metafile と呼ばれる中間ファイルフォーマットでの出力が EPS もしくは PS での最終出力のために必要であったが、バージョン 2.1 ではこの中間ファイルは廃止され、直接 EPS もしくは PS で出力できるようになっている。

このほか、ユーザが定義した変数を 4 バイト小数点数のバイナリで出力したり、NetCDF で出力したりすることが可能である。

[7] インターネットを用いた（特に科学的な）データ転送クライアント・サーバシステム及びプロトコルの総称。フリーソフトとして利用可能。

[8] COARDS (Cooperative Ocean-Atmosphere Research Data) 規約に準拠した NetCDF 及び HDF-SDF (Scientific Data Sets) であれば、これらのファイル形式は自己記述的であるため、コントロールファイルなしにデータを表示することができる。

4.4.6 描画の種類

1 次元データでは折れ線、棒グラフ、誤差棒など、2 次元格子データでは等値線、格子の塗りつぶしなどが描画可能である。2 変数で流線、矢羽根、矢印によるベクトル、散布図の表示、地点データでは天気記号の表示ができる。

空間 3 次元での描画には対応しておらず、基本的には任意の 2 つの次元での断面、あるいは 1 次元での線の描画を行う。ただし、GSL により内挿等を行うことで斜めに断面を取ることもできる。また、デフォルトでは時間軸を変えながら順に表示することでアニメーション表示させることができる。アニメーションは任意の次元に対して設定することができる。なお、アニメーションとしてのファイル出力はできない。

4.4.7 地図投影と地図データ

以下の座標系もしくは投影法での描画が可能である。

- 緯度経度直交座標（アスペクト比可変）
- 北極点もしくは南極点を中心とした極平射（ポーラーステレオ）図法
- ランベルト正角円錐図法
- モルワイデ図法
- 正射図法
- ロビンソン図法

なお、入力格子データが既に何らかの座標系で投影されている場合に、投影法の情報もしくは格子点の座標を適切に与えることにより、緯度経度直交座標で描画することもできる。このほか、非地図の座標系として時系列、対数座標での描画が可能である。

地図データ（海岸線や国境など）としては、3 つの解像度のデータが GrADS に付属している。これらのファイルフォーマットは公開されていない。異なる地図データを表示したい場合には、GrADS 標準のデータ描画と同時の地図描画ではなく、shapefile による描画が必要となる。

4.4.8 色空間

デフォルトで 16 色が用意されているほか、ユーザは最大 2032 色を RGB で新たに定義することができる。アルファチャンネル（透過色）も利用できる。また、外部から PNG ファイルを読み込み、タイルとして重ねることもできる。

4.4.9 フォント

6 種類の GrADS 独自フォーマットのフォントファイルが用意されている。そのうちの 1 つにはギリシャ文字や図形が含まれている。これらのフォントファイルを用いた表示では文字は多角形の塗りつぶしとなり、PS ファイルや PDF ファイルで出力したときに文字情報は埋め込まれない。なお、フォントファイルのフォーマットについてはマニュアルに記載があり、さらにフォ

ントを追加することもできる。一方、FreeType[9] でサポートされている TrueType や OpenType などのフォントファイルを外部から読み込み、文字情報を埋め込むこともできる。これにより、ベジェ曲線で表現されたより滑らかなフォントの出力が得られる。ただし、日本語のようなマルチバイト文字での出力には対応していない。

4.4.10 付属実行プログラム

以下の実行プログラムが GrADS に付属している。それぞれの機能を記す。

- `bufrscan`: BUFR データの記述子などの情報やデータの中身のテキストでの表示
- `gribscan`: GRIB データの情報表示、データのテキスト、バイナリ、GRIB での出力
- `grib2scan`: GRIB2 データの情報表示
- `gribmap`: GRIB データを `grads` で開くために必要な map ファイル（データの索引情報などを含む）を作成
- `stnmap`: GrADS station データに必要な map ファイルを作成

このほか、`gribscan` よりも高機能な GRIB デコーダである、NCEP (National Centers for Environmental Prediction) で開発された `wgrib` も同梱されているが、そのバージョンは 1.8.1.2a である。NCEP によるウェブページ [10] では 2013 年 9 月にバージョン 1.8.1.2c が公開されており、この間に JRA-55 (Japanese 55-year Reanalysis; Kobayashi et al. 2015) への対応等が行われているので注意されたい。また、同様に `wgrib2` という GRIB2 のデコーダも NCEP から公開されているが、こちらは GrADS に同梱されていない。これら以外にも、NCEP では GrADS 用のプログラム（例えば GRIB データから GrADS のコントロールファイルを作成する `grib2ctl` など）が公開されているので適宜参照されたい。

4.4.11 特徴

GrADS の大きな特徴は、X Window System による対話的な描画にある。ここで、NOAA (National Oceanic and Atmospheric Administration) によって公開されている全球の陸地・海底地形標高データ ETOPO1 (Amante and Eakins 2009) を入力として、X Window で表示した例を図 4.4.1 に示す。ファイルのフォーマットは NetCDF である。また、図 4.4.2 に描画実行時の GrADS コマンドを記す。

GrADS では、描画時の色の塗り分け方や、X 軸・Y 軸のラベルの間隔などがある程度自動で設定される。このため、比較的少ないコマンドの設定で描画することができる。一方で、GSL を駆使して凝った描画をすることもできる。図 4.4.3 はその例であるが、これだけの描画をするためには数百行の GSL が必要となる。だが、凝った描画を行うのであれば第 4.5 節で述べる GMT の方が使いやすいことが多いだろう。

GrADS のもう一つの特徴は、バイナリ格子データの

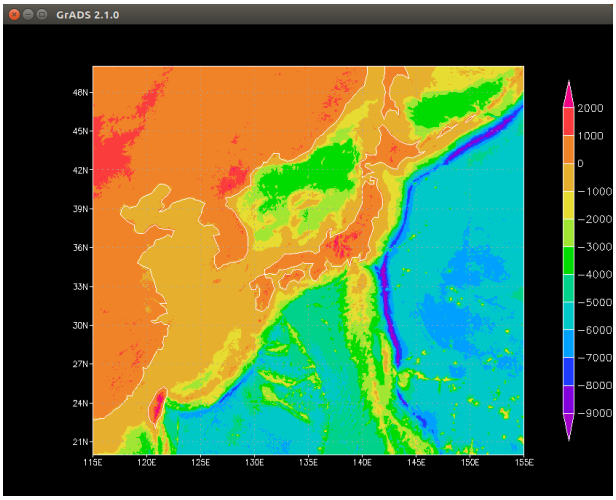

図 4.4.1 GrADS による X Window での表示例。1 分格子の標高データ (ETOPO1) を入力として、日本周辺を緯度経度直交座標で描画している。

```
set grads off
sdfopen ETOPO1_Bed_g_gmt4.grd
set lon 115 155
set lat 20 50
set gxout grfill
display z
run cbarn
```

図 4.4.2 図 4.4.1 を作成するための GrADS コマンド。なお、コマンドの意味については公式ウェブサイトのマニュアルを参照されたい。

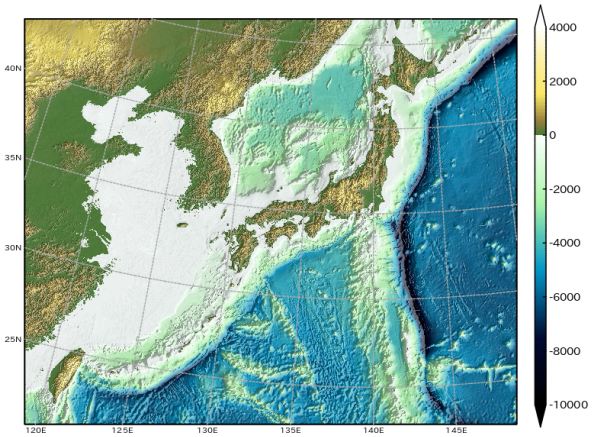

図 4.4.3 GrADS による GSL を駆使した描画の例。入力データは図 4.4.1 と同様であり、標高の描画に加え、標高から計算した地形の勾配と向きを陰影で重ねている。投影法はランベルト正角円錐図法を用いている。

[9] フォントデータのインターフェイスを提供するオープンソースのライブラリ。

[10] `http://www.cpc.ncep.noaa.gov/products/wesley/links_grads.html`

扱いが容易であるというところにある。GrADS では、複数の変数（例えば気温や風速など）の 5 次元のデータ（空間 3 次元と、時間、アンサンブルメンバーの次元）を格納したファイルを扱うことができるが、描画時には X Window で断面をスライドさせながら表示することができ、データの切り出しを予めプログラムで行う必要がない。また、描画用データの作成時に、異なる時間もしくはアンサンブルのデータを異なるファイルで出力しておき、GrADS で多数のファイルを同時に開いて読み込むこともできる。

バイナリ格子データ以外にも、気象データに用いられる多種多様な格子データのフォーマットを扱うことができることは、GrADS の利点の一つである。モデル開発者は必ずしもこの様なファイルフォーマットに精通する必要はなく、ファイルフォーマットの変換無しにデータの中身を確認することができることは開発効率を高めることにつながる。

一方で、GrADS では地点データの扱いは難しい。バイナリの GrADS station データのフォーマットは非常に特殊であり、マニュアルの記載も乏しい。また、テキストのデータを読み込むためには大量のスクリプトを作成する必要があるだろう。

GrADS では、描画コマンドを実行すると、X Window に表示されると同時に、描画の情報がバッファメモリに追加格納されていく。ファイル出力する時には、蓄積した情報をまとめて処理する。このため、大きなサイズのデータを扱ったり、1 つの画面に複数の図を描画したりするような場合にはメモリが不足する場合がある。特に GrADS station データでの使用メモリ量は大きく、一千万地点を超えるようなデータは描画ができない可能性がある。

参考文献

Amante, C. and B. W. Eakins, 2009: ETOPO1 1 Arc-Minute Global Relief Model: Procedures, Data Sources and Analysis. *NOAA Technical Memorandum NESDIS NGDC-24.* National Geophysical Data Center, NOAA, 19 pp.

Doty, B. E. and J. L. Kinter III, 1995: Geophysical Data Analysis and Visualization Using the Grid Analysis and Display System. *Visualization Techniques in Space and Atmospheric Sciences,* eds. E. P. Szuszczewicz and J. H. Bredekamp, National Aeronautics and Space Administration, 209–217.

Kobayashi, S., Y. Ota, Y. Harada, A. Ebita, M. Moriya, H. Onoda, K. Onogi, H. Kamahori, C. Kobayashi, H. Endo, K. Miyaoka, and K. Takahashi, 2015: The JRA-55 Reanalysis: General specifications and basic characteristics. *J. Meteor. Soc. Japan*, 5–48.

4.5 可視化ツール (2)–GMT[1]

4.5.1 はじめに

GMT (Generic Mapping Tools) は、ハワイ大学の Pål Wessel 教授らによって開発されているオープンソース、クロスプラットフォームの描画ツール、及び描画のためのデータセット処理ツールの集合である (Wessel et al. 2013)。GMT では、データ解析及び印刷用の品質での画像出力が可能である。また、様々な図や投影法での描画は GMT の大きな特徴である。最新の情報やマニュアルについては公式ウェブサイト [2] を参照されたい。GMT のユーザの多くは地球科学者であり、このほかにも医学研究・工学・物理学・数学・社会科学・生物学・地理学の科学者、漁業機関、石油会社、広範囲の政府機関、多くの愛好家により利用されていると主張されている。

GMT は数値予報課でよく使われる描画ツールの一つである。GrADS（第 4.4 節参照）と比べると地点データの扱いが容易であり、例えば日々実行される数値予報ルーチンにおいても、観測データの分布図を作成するために利用されている。本節では、GMT の入出力ファイルや描画の仕様などについて述べた後、その特徴について説明する。

4.5.2 バージョンと互換性

バージョン 5.3.1 が 2016 年 10 月にリリースされている。一つ前のメジャーバージョン [3] である GMT 4 シリーズについてもバージョン 4.5.15 が同月リリースされているが、GMT 6 の開発着手までは GMT 4 についてはバグ修正のみ対応するとされており、2015 年 11 月の GMT 5.2.1 リリース以降は GMT 5 の利用が公式に推奨されている [4]。

GMT 5 では幾つかの実行オプションに変更が加えられているので、使用する GMT のメジャーバージョンを 4 から 5 に変更した場合、古いバージョンに対して書かれたスクリプトを実行するためにはスクリプトの修正が必要となる場合がある。ただし、GMT コンフィグファイル (gmt.conf) の設定により、GMT 4 の実行オプションで GMT 5 を実行することもできる。

以下では、基本的に GMT 5 について説明する。

4.5.3 ユーザインターフェイス

GMT は、Unix などの環境から呼び出し可能なコマンド集として提供されている。ユーザはシェルスクリプトなどから GMT のプログラムを実行することが多い。描画プログラムの実行結果は標準出力に PS (PostScript) ファイル、もしくはその一部がテキストで書き出されるため、通常はファイルにリダイレクトする必要がある。このため出力結果を確認するためには別途 PS ファイルを表示できるビューアが必要となる。

GMT には 100 以上の実行プログラムが付属している。そのうち 4 つは bash スクリプトファイルである。バージョンの異なる 2 つの GMT を共存させて使用するための `gmtswitch`、一時ファイル [5] をカレントディレクトリではなく環境変数 `TMPDIR` で指定されたディレクトリに出力し前後の描画コマンドに影響を及ぼさないための `isogmt` など、GMT の実行設定等に関わるスクリプトがある。

GMT 5 では、メインの実行プログラムは `gmt` であり、残りの実行プログラムは全て `gmt` へのシンボリックリンクとなっている。これらのシンボリックリンクは、過去のバージョンとの互換性のために残されている [6]。例えば、海岸線等を描画する `pscoast` は `gmt pscoast` というように `gmt` からコマンドとして起動することが推奨されている。この変更は、例えば `surface` や `triangulate` などといった実行プログラムの名前が GMT 以外のものと競合することを避けるために行われた。

GMT では、C/C++で記述された GMT API (Application Programming Interface) ライブラリも提供されている。GMT のコマンド実行時にはこのライブラリが動的に呼ばれる。ユーザは自分で作成した C/C++ や Fortran のプログラムから GMT API のモジュールを直接利用したり、Julia, Python, MATLAB/Octave などのスクリプトから、それぞれのための API を通じて利用することもできる。これらを利用することにより、GUI (graphical user interface) を作成することも可能となっている。

4.5.4 コマンドの機能

前述のとおり、非常に多くのコマンドがあるため、全てについては説明しない。GMT のマニュアルにある用途別の分類を基に概説する。

- 1 次元、2 次元データのフィルタリング（外れ値の除去や、最頻値推定など）
- 1 次元、2 次元、3 次元データの描画（図枠、海岸線、文字列、記号、等値線、塗りつぶし、ベクトル場、3 次元遠近画像、ヒストグラム、鶏頭図などの描画、カラーパレットファイルの作成など）
- 地点データの格子化（最近傍法、グリーン関数による内挿など）
- 1 次元、2 次元データのサンプリング（間引き、

[1] 江河 拓夢

[2] `http://gmt.soest.hawaii.edu/`

[3] バージョン番号の一番上の位をメジャーバージョン番号と呼ぶ。また、メジャーバージョン番号が 5 である GMT を GMT 5 と呼ぶ。GMT 4, GMT 6 も同様。

[4] `http://www.soest.hawaii.edu/gmt/`

[5] GMT のコマンドの多くは、実行時に `gmt.history` というファイルにコマンドの実行履歴を保存する。例えば描画領域の設定を省略した場合には、前回の領域設定を引継ぐ。また、描画設定としてカレントディレクトリの `gmt.conf` を（ファイルが存在すれば）参照する。これらは GMT 4 では隠しファイル（`.gmtcommands4`, `.gmtdefaults4`）であった。

[6] 例えば Ubuntu 16.04 LTS (Long Term Support) の GMT 関連の配布パッケージではこれらのシンボリックリンクは含まれていない。

異なる格子への再サンプリングなど）
- 地図投影、座標の変換（地点もしくは格子データの変換、線または大円に沿った投影など）
- 情報検索（描画パラメータの表示・設定、格子データの情報表示など）
- データへの算術演算（逆ポーランド記法を用いた四則演算を含むデータ演算処理、時系列データのスペクトル推定、球面調和係数から格子データの算出、ドロネー・ボロノイ境界の作成など）
- データのサブセットの変換、抽出（バイナリとテキストの変換、サブ領域の切り出し、セグメントへの分割、地点・格子データの相互変換など）
- 1 次元、2 次元データの傾向の決定（もっともよく当てはまる大円または小円の算出、線形回帰、多項式や有限フーリエ級数による当てはめやトレンドの除去など）
- 2 次元格子データに対するその他の操作（格子データからカラーパレットの作成、周波数領域における格子データの操作、格子データの勾配の算出、ヒストグラム均一化、海岸線データからマスク格子ファイルの作成など）
- その他（KML (Keyhole Markup Language) データの処理、PS ファイルのラスタファイルへの変換）

このほかにも、補足的なパッケージ（発震機構解の描画、ホットスポットデータ、測線データの処理など）が用意されている。

4.5.5 入力データ

入力データのファイルフォーマットは基本的に以下に対応している。

- テキストデータ
- バイナリデータ（4, 8 バイトの浮動小数点数、1, 2, 4, 8 バイトの符号付き/無し整数のいずれか）
- NetCDF (Network Common Data Form)[7]

このほか、幾つかのコマンドがそれぞれ以下のデータを読み込むことができる。

- AGC (Atlantic Geoscience Center format)
- EPS (Encapsulated PostScript)
- Esri Arc/Info ASCII Grid Interchange format
- GDAL (Geospatial Data Abstraction Library) を通じたラスタデータ
- Golden Software Surfer format
- GRD98 (GEODAS (GEOphysical DAta System) Gridded Database Format)
- KML (Keyhole Markup Language)
- MGD77 (Marine Geophysical Data Exchange Format)
- PS (PostScript)
- RAS (Sun Rasterfile)
- raw RGB file
- SEGY (Society of Exploration Geophysicists format Y)

4.5.6 出力データ

個別のコマンドでの描画データの出力としては、GMT 5 では PS 形式のみがサポートされている。GMT 4 でサポートされていた EPS (Encapsulated PostScript) での直接出力は GMT 5 ではサポートされていない。

GMT コマンドの `psconvert`[8] では、GhostScript[9] を呼び出すことにより、PS ファイルを以下のいずれかに変換することが可能である。

- BMP (Microsoft Windows Bitmap Image)
- EPS (Encapsulated PostScript)
- JPEG (Joint Photographic Experts Group)
- PDF (Portable Document Format)
- PNG (Portable Network Graphics)
- PPM (Portable Pixmap)
- SVG (Scalable Vector Graphics)
- TIFF (Tagged Image File Format)

データ処理コマンドの出力は、コマンドによって多少異なるが、テキストのほか、バイナリデータ、NetCDF で可能である。

このほか、`gmt2kml` などで KML で出力することができる。

4.5.7 地図投影と地図データ

非常に多くの投影法での描画が可能である。特に、モデル開発やプロダクト配信で用いられるランベルト正角円錐図法と極平射（ポーラーステレオ）図法[10] がサポートされていることは重要である。

- 円筒図法（カッシーニ図法、メルカトル図法、ミラー図法、斜軸メルカトル図法、横メルカトル図法、正距円筒図法、ユニバーサル横メルカトル図法、正積円筒図法、平射円筒図法）
- 円錐図法（アルベルス正積円錐図法、ランベルト正角円錐図法、正距円錐図法、多円錐図法）
- 方位図法（ランベルト正積方位図法、正距方位図法、心射方位図法、正射方位図法、一般遠近図法、一般平射図法）
- その他の図法（ハンメル図法（正積）、正弦曲線図法（正積）、エケルト第四図法（正積）、エケルト第六図法（正積）、ロビンソン図法、ヴィンケル第三図法、モルワイデ図法（正積）、ヴァン・デル・グリテン図法）

[7] マニュアルには COARDS (Cooperative Ocean-Atmosphere Research Data) 規約及び Hadley Centre 規約に準拠している必要があるとの記載がある。前者の規約は CF 規約 (Climate and Forecast Metadata Conventions) のサブセットであり、CF 規約では COARDS 規約の次元順序の制約が緩和されている。後者の規約は CF 規約を指していると考えられるが、詳細は不明。

[8] バージョン 5.2.1 以前では `ps2raster` という名称であった。

[9] PS や PDF の変換ソフトウェア及び画像ライブラリからなるフリーソフト。

[10] 一般平射図法で投影中心を極に設定することで描画可能。

このほか、非地理的な座標系として直交座標（線形、対数、指数）、極座標での描画が可能である。

地図データ（海岸線、河川、国境など）としてはGSHHG (Global Self-consistent, Hierarchical, High-resolution Geography Database; Wessel and Smith 1996) が用いられる。このデータセットは 5 段階の解像度で提供されており、描画する地図の縮尺によって適切な細かさを選択することが可能である。

4.5.8 色空間

GMT では CPT (color palette table) と呼ばれるファイルを入力とし、これを基に多色での出力が可能となる [11]。40 種類の CPT が用意されているほか、ユーザが自作した CPT を読み込むことができる。

CPT で利用する色空間は RGB（赤・緑・青）、HSV（色相・彩度・明度）、CMYK（シアン・マゼンタ・イエロー・ブラック）のいずれかを指定することができる。例えば RGB では、3 色各々を 0 から 255 の整数で指定するため、実質 256^3 = 約 1677 万 色の指定が可能であるといえる。また、アルファチャンネル（透過色）も利用できる。

4.5.9 フォント

GMT 5 では、欧文基本 35 フォント (Base35) のほか、4 つのカスタムフォント (Ryumin-Light-EUC-H, Ryumin-Light-EUC-V, GothicBBB-Medium-EUC-H, GothicBBB-Medium-EUC-V) がデフォルトで設定されており、EUC-JP (Extended UNIX Code Packed Format for Japanese) で書かれた日本語での文字の出力が可能である。また、デフォルト以外のフォントを設定することも可能である。

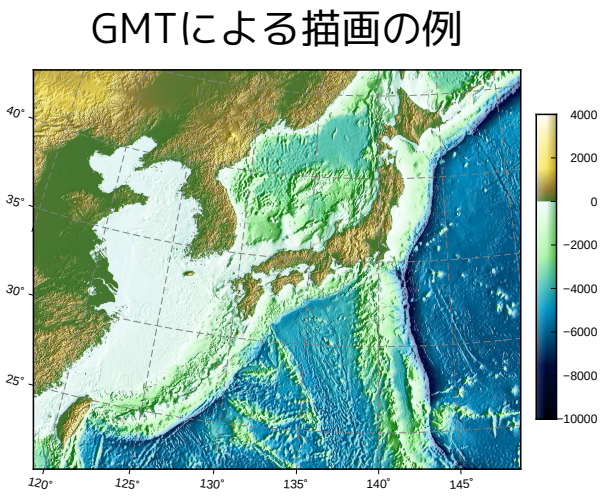

図 4.5.1 GMT による描画の例。入力データは図 4.4.1 と同様であり、ランベルト正角円錐図法で、標高（カラー）とその勾配と向き（陰影）を描画している。図上部の日本語は GMT で出力したものである。

[11] 単色の場合は色をコマンドラインで直接指定することもできる。

4.5.10 特徴

GMT を利用するときには、コマンドライン上で対話的に作業するよりも、シェルスクリプトファイルなどに処理を記述しバッチ処理を行うことが多い。GMT API を用いた対話的なプログラムの開発も可能ではあるものの、そのような利用は数値予報課では行われていない。

GMT の大きな特徴は、これまで挙げてきたような非常に多種多様な機能にある。これらを利用して、きれいな描画を比較的簡単に行うことができる。図 4.5.1 に、GMT による描画例を示す。使用したデータは図 4.4.1 と同様である。また、描画に使用したシェルスクリプトを図 4.5.2 に示す。GrADS ではきれいな描画には大量のスクリプトを記述する必要があったが、GMT では組み込みの機能を利用することができる。また、

```
#!/bin/sh

outfile=etopo

# 格子データ（NetCDF）から一部分切り出し
gmt grdcut ETOPO1_Bed_g_gmt4.grd \
  -R100/155/20/50 -G${outfile}.grd
# 切り出したデータを使って勾配を算出
gmt grdgradient ${outfile}.grd -Ne0.8 -A100 \
  -fg -G${outfile}_i.grd

# カラーパレットファイルを作成
gmt makecpt -Crelief -T-10000/10000/1250 -Z |
  head -n 8 > ${outfile}.cpt
gmt makecpt -Crelief -T-4000/4000/500 -Z |
  tail -n +9 >> ${outfile}.cpt

# 描画の設定
gmt gmtset PS_MEDIA=600x500 \
  FONT_TITLE=40p,GothicBBB-Medium-EUC-H,black \
  MAP_GRID_PEN=0.8p,128/128/128,7_3.5:3 \
  MAP_TITLE_OFFSET=5p
# 切り出したデータで描画
gmt grdimage ${outfile}.grd \
  -I${outfile}_i.grd -JL140/30/30/60/480p \
  -R119.393562/20.439766/152.363220/45.912659r\
  -P -BWS+t`echo GMT による描画の例 | nkf -e` \
  -Ba5g5 -C${outfile}.cpt -X1 -Y1 -K \
  > ${outfile}.ps
# 枠の描画
gmt psbasemap -Bne -Ba0 -R -J -O -K \
  >> ${outfile}.ps

# 描画の設定
gmt gmtset MAP_TICK_LENGTH=-5p
# カラースケールの描画
gmt psscale -DjTC+w300p/20p+o265p/44p -R -J \
  -C${outfile}.cpt -I0.4 -Ba2000g0f0 -O \
  >> ${outfile}.ps

# GMT 設定ファイル、履歴ファイルを削除
rm -f gmt.conf gmt.history

exit 0
```

図 4.5.2 図 4.5.1 を作成するためのシェルスクリプト。

描画コマンドだけでなく、データについて様々な統計処理を行うことができるのも GMT の利点と言えるだろう。

他方で、機能が多いということは、求める機能を利用するときにどのようなコマンド、オプションを使用するべきか覚えることが難しいということでもある。例示したシェルスクリプトの内容を見ても、GMT のマニュアルと付き合わせないと、それぞれのオプションがどのような意味を持っているのか理解することは難しいだろう。また、描画コマンドの出力のほとんどが PS 形式であり、描画に失敗した時に、描画時の GMT 実行コマンドに問題があるのか、あるいはそもそも入力データに問題があるのかの判断が、特に PS 形式に慣れていないユーザには難しい。

一方で、PS 形式での出力では一つ一つの描画コマンドが独立したプロセスで実行されるので、一度にまとめてファイル出力するような他の描画ツールと比べると、使用メモリ量は比較的少ないといえる。

GMT では、テキストデータを直接読み込んで描画することができる。GMT 5 では、空白区切りのテキストデータについて、どの列を何番目に読み込むかを指定できるように改良されたので、`awk` コマンドなどでのテキスト整形が不要となる場合もあり、利便性が増している。バイナリデータを直接読み込むことも可能であるが、一度 NetCDF 形式に変換した方がデータの扱いが容易である。

参考文献

Wessel, P. and W. H. F. Smith, 1996: A Global Self-consistent, Hierarchical, High-resolution Shoreline Database. *J. Geophys. Res.*, **101**, 8741–8743.

Wessel, P., W. H. F. Smith, R. Scharroo, J. Luis, and F. Wobbe, 2013: Generic Mapping Tools: Improved Version Released. *EOS Trans. AGU*, **94**, 409–410.

4.6 可視化ツール (3)–TAG [1]

4.6.1 背景

予報業務や各種事例調査などでの気象データの確認に、印刷された資料だけでなくウェブブラウザを利用する機会が近年増加している。例えば、ひまわり 8 号による高頻度観測 (横田・佐々木 2013) や、高解像度降水ナウキャスト (気象庁予報部 2014) などでは、数分単位での観測や予測が実用化され、高頻度なデータが利用できる。こうした高頻度データの確認では刻々と変化するデータを随時表示するために、ウェブブラウザを利用したリアルタイムな画像表示システムが利用されている。

数値予報システム開発の現場でも開発成果の結果確認にウェブアプリケーションを利用するなど、ウェブブラウザでの表示を目的とした画像作成の機会が増加する傾向にある。近年の地図表示ウェブアプリケーションでは、マウスドラッグによる表示領域の移動や拡大縮小が自在に利用できるインタラクティブな表示が一般化している。GUI による設定変更でユーザが容易に表示設定を変更することも可能なことから、開発業務でウェブアプリケーションによる画像表示が可能になると、資料確認効率が向上することが期待される。地図上に画像を表示させるウェブアプリケーション自体は、無料で提供されているものを含め多数のサービスが存在し、独自開発の必要性は低い。しかし、こうしたウェブアプリケーション上に気象データを表示させるには、気象データから表示用の画像を作成する必要があり、このためのアプリケーションとして TAG を開発するに至った。

ウェブブラウザで表示させる画像の描画手法は、大きく静的描画法と動的描画法の二つに分けられる。

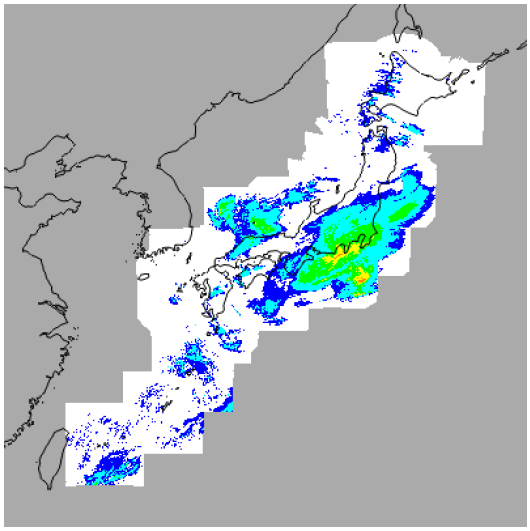

図 4.6.1 TAG の表示例。解析雨量を用いて 2016/08/20 00:00(UTC) の前 3 時間積算雨量を表示。

静的描画法は、数値予報データから予め表示させる画像を作成しておき、ユーザからリクエストがあった際に事前作成された画像をクライアントの画面に表示させる手法である。この手法はリクエストに応じて画像を作成する必要がないため、高速な応答を行うことが可能であるが、事前作成した画像しか表示することができないため、表示要素や描画領域は固定となり柔軟性に欠ける [2]。また、様々な要素の画像表示を行うためには、大量の画像を保存する必要がある。

動的描画法は事前に画像を作成せず、ユーザからのリクエストを受けて画像作成を開始する手法である。この手法ではユーザのリクエストに応じて要素や領域を変更した柔軟な描画が可能であるとともに、作成元データが存在する場合は、表示用の画像を事前に保存しておく必要がなく、画像用にわざわざストレージ領域を確保する必要がない。しかし、リクエストに応じて描画を行うため応答に時間がかかるとともに、アクセスが集中すればサーバ負荷の増加にもつながる。

こうした特徴から、アクセス数が多く表示させる要素が固定的なら静的描画法を、利用者は少ないが調査用に様々な要素・領域での描画が重要なら動的描画法を利用するといった、特徴に応じた使い分けが行われ、特に数値予報システム開発では後者の利用形態が多い。

4.6.2 用途と設計思想

TAG は動的描画法で利用することを念頭に数値予報課が独自に開発した、高速気象データ描画ツールである。ウェブアプリケーションの画像表示に TAG を利用することで、気象データをユーザのリクエストに応じて要素や領域を自由に設定して閲覧することができ、図 4.6.1 のような平面図や、断面図などの二次元画像を描画することが可能である。最新バージョンでは、例えば第 4.2 節の GRIB2 形式のうち気象庁が作成したデータや NuSDaS 形式のデータを表示可能である。

TAG が想定する動的描画法で重要となるのが、描画速度の向上である。一般的な開発業務で利用される気象データの場合、第 4.4 節、第 4.5 節、第 4.7.1 項で紹介している各種描画ツールを用いてウェブブラウザに表示すると数秒〜十数秒程度の時間が必要となる [3]。ウェブブラウザでの動的表示を目的に据える場合、数秒〜十数秒という時間はユーザビリティ面で決して望ましくない。ユーザビリティにおける応答速度の影響は古くから調査されており、ユーザの行動がページに瞬間的に反映されたと感じさせるためには 100 ミリ秒以下

[1] 雁津 克彦

[2] 画像をタイルに分割して用意することで描画領域が固定的となることを回避可能である。ただし、拡大縮小等にも耐えるためには、大量のタイル作成が必要であり、巨大なストレージを用意する、あるいは表示する時刻を限定して過去データを逐次削除するといった対処が必要になる。

[3] 多くの場合、描画そのものよりウェブ表示のための画像変換に時間がかかる。

で、作業中のユーザの思考を邪魔しないためには 1 秒以下で応答を行う必要がある (Nielsen 1993)。このため、動的描画法による表示を実用上十分といえる速度で提供するためには遅くても 1 秒以下、可能なら数十〜数百ミリ秒で描画することが求められる。TAG は一般的な開発業務で利用される気象データを、スーパーコンピュータシステムの業務処理サーバや支線 LAN 上のサーバを利用して数十〜数百ミリ秒程度で表示することを目標に開発し、多くの画像表示でこの目標を実現している。

なお、TAG 自身はウェブブラウザ上のユーザインターフェース機能を提供しない点に注意されたい。TAG はサーバ上などで気象データを画像に変換する機能のみ提供する。ウェブブラウザ上で動作するユーザインターフェースが必要な場合には、開発担当者が必要な実装を行うことを想定している。このため TAG が想定する主要な利用者は画像の閲覧を行う気象庁内外の一般ユーザではなく、数値予報モデル開発者などの開発担当者となる。

4.6.3 特徴と実装方法

(1) 高速な描画処理

TAG の一番の特徴は、ウェブブラウザでユーザビリティの制約を満たした動的表示を可能にする高速な応答であり、これを実現するためにラスタ形式 [4] での描画法を採用している。気象データの描画ツールは論文や現業用の印刷物作成で利用する機会が多いこともあり、ベクタ形式 [5] での出力を標準としてサポートするものが多い。しかし、気象データの多くは地点形式データを除くと格子データが大半で、ラスタ的なデータとして格納されている。このことはベクタ形式の画像出力処理を行うためには、処理が複雑なラスタからベクタへの変換が必要であることを意味する。しかも、ウェブブラウザがサポートしている画像形式の多くはラスタ形式であることから、一度ベクタ形式で作成した画像をラスタ形式に再変換するという処理も必要になる。既存の各種描画関係ツールを用いて動的描画を行う場合、こういった画像変換処理に数秒以上かかるため、ユーザビリティの確保が難しいのである。それに対し、最初からラスタ形式での出力を念頭において描画ツールを開発する場合、内部処理はラスタからラスタへの変換のみ対応すればよい。この変換は、入力格子位置から出力ピクセル位置を割り出す座標変換処理と、物理量を対応する色に置き換える色変換処理の二段階で実現され、処理が非常に単純であり高速に動作することが期待できる。

座標変換処理はランベルト正角円錐図法やメルカトル図法で三角関数や対数が利用され、比較的計算量が多い処理部分である。TAG では座標変換を行う際に格子位置から緯度経度を経由することなく、直接ピクセル位置を計算することで高速化を行っている [6]。また、描画で座標計算を呼び出す部分は格子ループの内部などの繰返し処理がほとんどであるため、座標変換単体ではなく格子ループも含めた関数として実装することで、関数呼び出しのオーバーヘッドが少なくなるようにしている。

色変換については気象データ特有のデータ形式である「レベルデータ」を活用することで処理の簡略化を行っている。レベルデータとは、格子に物理量を直接格納するのではなく、レベル 0 なら 0 [mm/h]、レベル 1 なら 0.5 [mm/h] といったレベルと物理量の対応を事前に決めておき、格子には物理量ではなくレベルを格納するデータ形式である。気象データの保存にレベルデータが利用されるのは、多くの場合実数を直接保存するより、ファイルサイズが小さくなるためである。実数を保存する場合、単精度浮動小数点数の場合は 1 格子あたり 4 バイトが必要で、格子数 n とするとデータサイズは $4n$ となる。しかし、例えばレベルの数を 256 以下にすれば、1 格子あたり 1 バイトで表現できるため、トータルサイズはレベル数 l として $4l+n$ となる。通常 $l \ll n$ であることから $4l+n < 4n$ となり、データサイズは小さくなる。

このレベルデータをうまく利用すると、描画処理の計算量を大幅に削減することができる。レベルデータではない通常の描画処理を行う際は降水量 0.5 [mm/h] 未満は水色、1 [mm/h] 未満は青といった実数の比較演算が格子数 n 回必要である。しかし、レベルデータの場合は先にレベル 0 に青、レベル 1 に水色……といった色の割り当てが可能で、これに必要な実数の比較演算は l 回と浮動小数点数の演算回数を大幅に減らすことができる。あとは求めた色を格子に割り当てるポインタ演算を n 回繰り返すことで画像が完成する。ポインタ演算は浮動小数点数演算より高速なため、多くの場合高速化が期待できる。

また、近年の技術動向として CPU 速度の向上に比べメモリバンド幅の向上が限定的である。このため、頻繁にメモリアクセスが発生する場合には全体のボトルネックとなることが懸念される。TAG では可能な限りメモリ使用量を削減するよう描画に不要な箇所の読み込みをスキップし、描画領域に関連する部分だけをメモリに保持している。先ほどのレベルデータの活用は

[4] 画像をドットの集合体で表現した画像形式。
[5] 画像を線分や領域の集合で表現した画像形式。

[6] 座標変換で緯度経度を介さないため、サポートする座標系の数が p の時、実装する座標変換の数は $O(p^2)$ と管理コストが急増する。そこで TAG は入力と出力でサポートする座標系を分離して管理コストを $O(p)$ に抑えている。例として (r, θ) 極座標データは読み込みに用いることはあっても、縦軸 r, 横軸 θ で画像出力してマウスドラッグする機会は稀であろう。TAG は入力サポートする座標系の拡張を容易にする一方、出力は緯度経度図法、ランベルト正角円錐図法（ポーラーステレオ図法も含む）、メルカトル図法（未実装）に限定し管理コスト低減を図っている。

データサイズ縮小にも繋がることからメモリ使用量削減においても効果的である。

こうした実装上の工夫により、TAG では高速な気象データの描画を実現している。

(2) 開発担当者向けの実装

TAG の想定する利用者は開発担当者であり、開発担当者向けのツール作成では機能の柔軟性が重要となる。描画ツールにおいても特定の固定的な機能だけ提供するのではなく、開発担当者のニーズに応じてある程度描画内容を自由に変更できることが望ましい。例えば、気象データの表示を行う場合、単に保存データをそのまま出力するということは稀であり、実用的な表示を行うため、

- 指定順序での実行（複数要素の描画順指定など）
- 条件分岐（ファイルが無い場合の動作など）
- 計算処理（差分や元データ値が格納されていないデータ（例えば相当温位）の計算など）
- 繰返し処理（積算やデータがないときの遡りなど）

といった処理が不可欠である。

こうしたことを実現するには何らかの言語的な仕組みを導入すればよい。簡単な解決方法としては、スクリプト言語を用いて処理を記述し、描画ツールと連携する方法が考えられる。この方法は汎用性が高く導入も容易なため、多くの開発場面で用いられるが、今回想定しているようなウェブブラウザによる動的利用の場合、追加でプロセスが起動されることで余分なオーバーヘッドが生じ、描画速度を損なう可能性があるため解決方法として望ましくない。

そこで、設定ファイル自体で言語的な記述である条件分岐、繰返し処理、関数といった機能を提供し、TAG 自身で処理を行うことで全体の速度が低下しないようにしている。例えば図 4.6.2 のような「スクリプト」を書くことで、図 4.6.1 にあるような 3 時間積算雨量の表示が可能である。

```
#!/[install_dir]/bin/tag

// 描画領域を指定
tag.setRegion({
  n:48deg, s:20deg, w:118deg, e:150deg
});

// 3 時間積算雨量を格納する変数
r3 = 0;
base = 20160820Z;
// ループで解析雨量読み込み
for(dt = -3h; dt < 0h; dt += 1h){
  ra_file = @Ra_Ra;
  ra_file.read({date:base + dt, retry:0});
  ra_data = ra_file.gpv;
  if(tag.isNull(ra_data)) continue;
  r3 += ra_data;
}
if(tag.isInt(r3)) exit 1;

// 400px 四方の灰色なラスタを用意
raster = tag.createRaster({
  x:400, y:400, color:#AAA
});

// r3 を描画する
raster.fill({
  data:r3,
  color:[#FFF,#00F,#0FF,#0F0,#FF0,#F80,#F00],
  scale:[ 0.01, 1, 5, 10, 20, 50],
  undef:#AAA
});

// 地図を描画する
raster.map({
  fileName:"world.4map", color:#000
});

// 標準出力に PNG で出力
raster.savePng(tag.STDOUT);
```

図 4.6.2 TAG で用いるスクリプトの例。この例は図 4.6.1 を表示させるためのスクリプト。

4.6.4 活用例

TAG はウェブブラウザでの利用を念頭に開発を行ってきた経緯により、主にウェブアプリケーションによるインタラクティブなツールの描画処理部分として活用されている。

図 4.6.3 は数値予報課アプリケーション班が開発したガイダンスモデルモニタと呼ばれるツールであり、GSM 及び MSM の予測結果並びにそれらのガイダンスの結果を一画面で表示できるようになっている。ユーザはマウスドラッグによる領域移動とホイールによる拡大縮小で任意の領域の画像を簡単に表示できる。データによってはマウスでクリックした任意の二点間の断面図や特定の地点を拡大して周辺の格子値を確認するといった利用も可能である。さらには、初期値ごとの予測結果の変化を確認するためのスパゲッティ図や MSM と GSM の比較など、非常に多機能なツールとして整備され、リアルタイムな予測結果の確認などに用いられている。

図 4.6.4 は開発中のメソアンサンブル予報システム（MEPS；小野 2016）の結果確認用ツールとして数値予報課メソモデルグループ向けに開発した表示ツールである。このツールでは MEPS で出力された各メンバーの結果を一覧で表示できるとともに、指定したメンバーのアンサンブル平均やスプレッドといった統計情報もその場で計算して表示する機能を提供している。

この他に第 2.5.2 項で紹介した数値予報課全球・台風グループ開発のリアルタイムモニタである DynaMo や航空予報室が開発した航空悪天 GPV モニタなど、TAG を活用したインタラクティブな表示ツールが各開発担当者のニーズに応じる形で利用されている。

4.6.5 課題

TAG は試用版を公開して 1 年強しか経っておらず発展途上のツールである。現在までに開発担当者からの

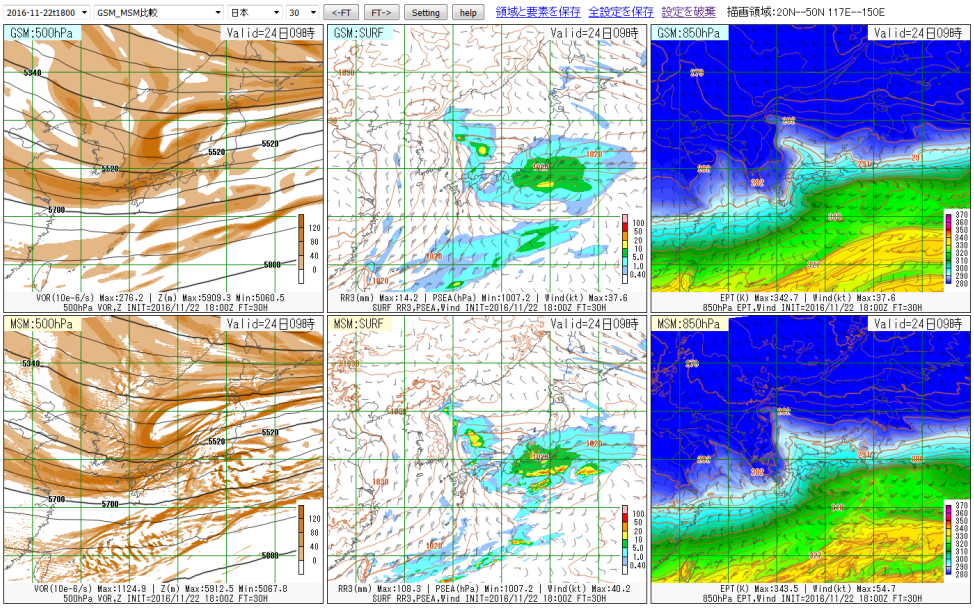

図 4.6.3　ガイダンスモデルモニタ。ユーザは画面上の 6 画面に任意の要素を表示させることができる。マウス操作により領域や時刻を自由に変更でき、表示状態の保存も可能である。この例では上段に GSM を、下段に MSM を表示させている。

要望も複数寄せられており課題も多い。2017 年 1 月現在の主要な課題は以下のとおりである。

(1)　地点データの表示機能強化

TAG は地点データ表示に対応しているものの、設定での取り扱いに不便な箇所がいくつか存在する。こうした取り扱いについて標準的なルールを定め、開発担当者が簡単に地点データを扱えるよう整備が必要である。また、第 4.2.2 項でも紹介した NuSDaS 形式の読み込みでは地点データに対応しておらず、利便性向上のためサポートを行いたいと考えている。

(2)　メルカトル図法への対応

一般的に公開されている各種地図表示ウェブアプリケーションでは、正角で一覧性に優れたメルカトル図法を利用するものが多く見受けられる。こうしたツールと親和的に利用するためには、画像出力形式でメルカトル図法に対応する必要がある。第 4.6.3 項の脚注にもあるとおり、出力形式としてメルカトル図法をサポートする予定であり、引き続き開発を進めたい。

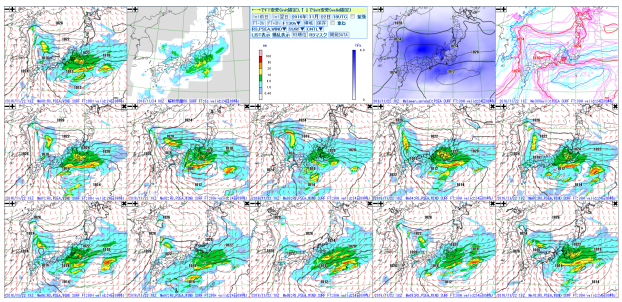

図 4.6.4　MEPS インタラクティブツールと呼ばれる MEPS の結果確認ツール。上段右から二番目の青みがかった画像は気圧のアンサンブル平均とスプレッドを表示しているが、各メンバー右上の×ボタンをクリックすることで対象メンバーを統計対象から外すことができ、統計結果をその場で動的に確認することができる。

(3)　NetCDF（特に CF 規約）への対応

NetCDF の CF 規約は大気、地上及び海洋の気候及び予報データに使われることを意図して設計された標準である (Eaton et al. 2011)。世界的に広く利用されるとともに、気象庁の数値予報システム開発でもしばしば利用されていることから今後対応を進めたい。

参考文献

Eaton, B., J. Gregory, B. Drach, K. Taylor, S. Hankin, J. Caron, R. Signell, P. Bentley, G. Rappa, H. Höck, A. Pamment, and M. Juckes, 2011: *NetCDF Climate and Forecast (CF) Metadata Conventions (1.1 Goals)*. Version 1.6 ed., `http://cfconventions.org/`.

気象庁予報部, 2014: 配信資料に関する技術情報（気象編）第 398 号　高解像度降水ナウキャストの提供開始について.

Nielsen, J., 1993: *Usability Engineering, (5.5 Feed-Back)*. 1st ed., Morgan Kaufmann.

小野耕介, 2016: メソアンサンブル予報システムの開発状況. 数値予報課報告・別冊第 62 号, 気象庁予報部, 100–113.

横田寛伸, 佐々木政幸, 2013: 静止地球環境観測衛星「ひまわり 8 号及び 9 号」の紹介. 気象衛星センター技術報告第 58 号, 121–138.

4.7 数値予報課で利用されているその他の可視化ツール [1]

4.7.1 PANDAH

PANDAH (Plotter for Any Numerical Data Arranged Horizontally) は NuSDaS で格納された格子点値データを可視化するためのツールであり、数値予報課で開発、維持されている。PANDAH の本体は Fortran (FORTRAN77 と Fortran90 が混在) で記述されており、描画ライブラリとして PostScript コードを生成するライブラリである PLOTPS (加藤 2004) を用いている。

PANDAH で描画をする際はあらかじめ図 4.7.1 のような描画要素設定ファイルを用意して、PANDAH に入力する。必要に応じて、描画要素拡張設定ファイル（風の矢羽の大きさや描画間隔などが設定できる）、描画要素コンター設定ファイル（等値線の描画間隔、太さ、ラベルなどが設定できる）、描画領域設定ファイル（このファイルに指定された水平座標系に変換される）などを用意する。

PANDAH は、数値予報課の現業作業で利用するモデル予測のモニタ図 (図 4.7.2) をはじめ、オンラインの各種モニタ、モデル開発（特にメソモデル）の際の描画に利用されている。

開発当初はその名前の通り、水平データの描画に限定されていたが、現在では鉛直断面図、時間鉛直断面図、等高度面データなどの描画機能もある。また、地点データ（テキストファイル）の描画機能があり、観測点の情報をプロットすることができる。

PANDAH は水平座標変換機能を有しており、指定した座標系に変換してデータをプロットすることが可能である。また、温位と気圧から気温に変換するといった要素変換機能、データ間の加減演算機能、複数のデータの積算や平均、2 つのデータの差分の描画を行うこともできる。

PANDAH の前身となるツール（PLT, NEWPLT, PLGVD, hplt_nusdas と呼ばれた）は 1995 年から数値予報課で開発されており、20 年以上の歴史を持つツールである。鉛直断面図や地点データの描画機能、さまざまな要素変換機能が追加実装されて今日に至るが、コーディングスタイルが異なる拡張が繰り返されたためにソースコードの可読性が低下し、現状では内部構造をよく知った者しかコードに手を入れることができない状況に陥っている。また、描画要素設定ファイルをはじめとするパラメータ設定ファイルが直感的にわかりにくいものが多く、初学者には使いにくいこと、GrADS などのように対話型ではないため、描画結果を見て別の図を描くというプロセスが効率よくできないなどの指摘がされている。さらに、以前は印刷が目的であったので PostScript による出力は望ましいものであったが、現在ではウェブブラウザで見ることが非常に多くなってきており、出力された PostScript ファイルを Ghostscript[2] などで PNG に変換するコストが無視できなくなってきている。

そのような状況の中、コードの整理を行い開発・維持を続けるか、新しいツールに移行するかを検討しなければならない岐路に立たされている。

```
! PANDAH element file
! ---------------------------------------------------------------------
    1,         : maximum number of output sheets
    4,    2,   : number of chart : NCLM * NROW <= 24
  TOP,  CLM,   : order of plot : (TOP, BTM) -> (CLM, ROW)
   10,  10,  10,  10, : (LEFT, RIGHT, TOP, BOTTOM) margin(unit = mm)
! ----------------------------+----------------------------------------
'20kmGSM FCST 18, 24H 2016/11/29/00': page title(upper of the sheet) (a30)
! ----------------------------+----------------------------------------
  -99, -99, -99, 60 : KTSTART, KTEND, KTCYCL, KT_UNIT
! KTSTART, KTEND, KTCYCLE : available if >= -50
! KT_UNIT = 60 : unit = hour, = 1 : unit = minute
! ---------------------------------------------------------------------
!   type1,    type3,   nmap,     level,      sp,      color, area,
!         type2,   member,    kt,         elem,    contr,  file, date,
 _GSMLLPP,FCSV,STD1,    , 1, 18,700   ,OMG   ,#, 1.e+30,-11, 0,10,-1,
 _GSMLLPP,FCSV,STD1,    , 1, 18,850   ,T     ,#, 1.e+30,  1, 0,10,-1,
 _GSMLLPP,FCSV,STD1,    , 2, 18,700   ,TTD   ,#, 1.e+30,-11, 0,10,-1,
 _GSMLLPP,FCSV,STD1,    , 2, 18,700   ,TTD   ,#, 1.e+30, 69, 0,10,-1,
 _GSMLLPP,FCSV,STD1,    , 2, 18,500   ,T     ,#, 1.e+30,  1, 0,10,-1,
 _GSMLLPP,FCSV,STD1,    , 3, 18,500   ,VOR   ,#, 1.e+30,-11, 0,10,-1,
 _GSMLLPP,FCSV,STD1,    , 3, 18,500   ,Z     ,#, 1.e+30, -1, 0,10,-1,
 _GSMLLLY,FCSV,STD1,    , 4, 18,SURF  ,RAIN  ,#, 1.e+30, -4, 0,10,-1,
 _GSMLLLY,FCSV,STD1,    , 4, 18,SURF  ,PSEA  ,#, 1.e+30, -1, 0,10,-1,
 _GSMLLLY,FCSV,STD1,    , 4, 18,SURF  ,WIND  ,#, 1.e+30,  2, 0,10,-1,
 _GSMLLPP,FCSV,STD1,    , 5, 24,700   ,OMG   ,#, 1.e+30,-11, 0,10,-1,
 _GSMLLPP,FCSV,STD1,    , 5, 24,850   ,T     ,#, 1.e+30,  1, 0,10,-1,
 _GSMLLPP,FCSV,STD1,    , 6, 24,700   ,TTD   ,#, 1.e+30,-11, 0,10,-1,
 _GSMLLPP,FCSV,STD1,    , 6, 24,700   ,TTD   ,#, 1.e+30, 69, 0,10,-1,
 _GSMLLPP,FCSV,STD1,    , 6, 24,500   ,T     ,#, 1.e+30,  1, 0,10,-1,
 _GSMLLPP,FCSV,STD1,    , 7, 24,500   ,VOR   ,#, 1.e+30,-11, 0,10,-1,
 _GSMLLPP,FCSV,STD1,    , 7, 24,500   ,Z     ,#, 1.e+30, -1, 0,10,-1,
 _GSMLLLY,FCSV,STD1,    , 8, 24,SURF  ,RAIN  ,#, 1.e+30, -4, 0,10,-1,
 _GSMLLLY,FCSV,STD1,    , 8, 24,SURF  ,PSEA  ,#, 1.e+30, -1, 0,10,-1,
 _GSMLLLY,FCSV,STD1,    , 8, 24,SURF  ,WIND  ,#, 1.e+30,  2, 0,10,-1,
!########,####,####,####, 1, 12,######,######,#, 1.e+30, -1, 0,10, 0,
end
! ---------------------------------------------------------------------
! end    : end line(necessary)
! type1  : NuSDaS parameter(######## represents same as previous line)
! type2  : NuSDaS parameter(#### represents same as previous line)
! type3  : NuSDaS parameter(#### represents same as previous line)
! member : NuSDaS parameter(blank usually,
!                           #### represents same as previous line)
! nmap   : order of plot(order is arbitrary, overwrite is possible,
!                        page return is impossible)
! kt     : available if KTSTART, KTEND, KTCYCKE < -50
! level  : NuSDaS parameter(###### represents same as previous line)
! elem   : NuSDaS parameter(###### represents same as previous line)
! contr  : contour interval(default value is set if >= 1.e+30)
!          RAIN : available if lower specification are absent
! sp     : = #   : plot usually
!        : = +/- : add/subtract element of next line and plot
!        : = i   : subtract element of next line,
!                  and plot by increment style
! color  : color
! file   : NuSDaS id(available if 0 < file < 100)
! area   : area file number(default value = 10)
! date   : = 0     : read base time from ft.01 only at first line
!        : = 1     : re-read base time from ft.01
!        : = other : read automatically
! ---------------------------------------------------------------------
    0,  8,  7, : interval of grid, wind(unit = grid), weather mark
    1,    1,   : smoothing(except for RAIN, RAIN) (unit = grid)
   10,         : intervals of latitude lines and longitude lines
   10,         : grid range to investigate maximum and minimum values
    6,         : rain accumulation time(unit = KT_UNIT)
    4,         : rain cut off(unit = 0.1mm)
! -------------- rain contour(unit = 0.1mm, max number is 10)
   10, 100, 200, 500,1000,2000,9999, -99,
! ---------------------------------------------------------------------
    1,         : convert WIND unit(1 : m -> knot, -1 : knot -> m)
    1,         : convert T    unit(1 : K -> C,    -1 : C    -> T)
 -999,         : topography cut off(unit = m)
! ---------------------------------------------------------------------
```

図 4.7.1 PANDAH の描画要素設定ファイルの例。エクスクラメーション (!) やコロン (:) より後ろはコメントである。

[1] 原 旅人

[2] http://www.ghostscript.com

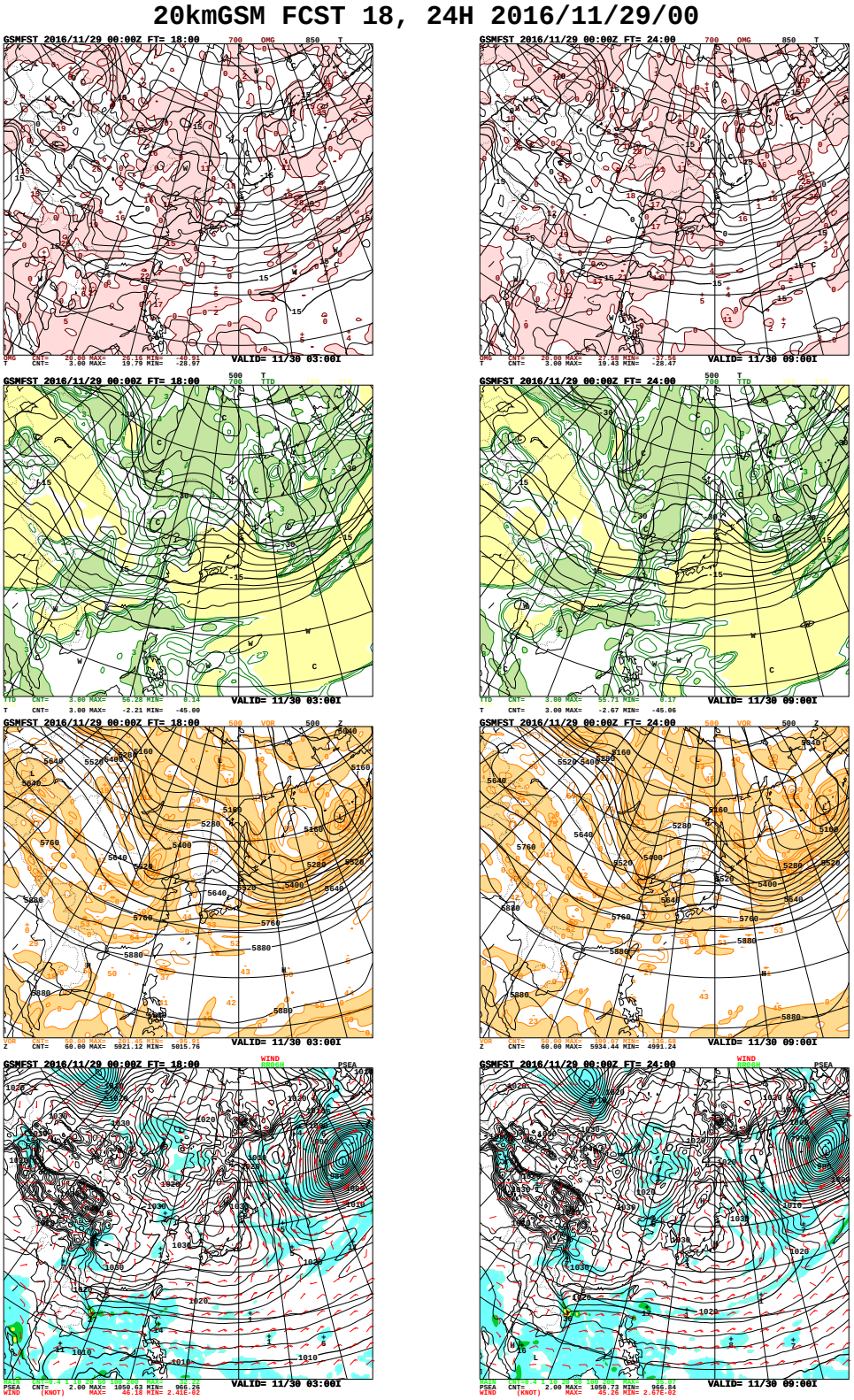

図 4.7.2　PANDAH で描画された GSM 予測のモニタ図の例。最上段： 850 hPa の気温（等値線。極大値、極小値をそれぞれ W, C で表示。）と 700 hPa の鉛直 p 速度（負の部分を桃色に着色。極大値、極小値をそれぞれ+, −で表示。）、2 段目： 500 hPa の気温（等値線。極大値、極小値をそれぞれ W, C で表示。）と 700 hPa の湿数（3 K 以下を緑で、15 K 以上を黄色で着色）、3 段目： 500 hPa のジオポテンシャル高度（等値線。極大値、極小値をそれぞれ H, L で表示。）と渦度（正の部分を橙色に着色。極大値、極小値をそれぞれ+, −で表示。）、最下段： 海面更正気圧（等値線。極大値、極小値をそれぞれ H, L で表示。）と前 6 時間降水量（着色。極大値を+で表示）、地上風速（赤矢羽）。左、右図は初期時刻からそれぞれ 18 時間、24 時間後の予測を示す。図 4.7.1 に示した描画要素設定ファイルを利用している。

4.7.2 Python

Python[3] はいわゆるオブジェクト指向のスクリプト言語に分類されるプログラミング言語である。スクリプト言語としては、数値予報課内ではシェルスクリプトや Ruby[4] が広く使われているが [5]、数値予報課での Python の利用は現時点ではほとんどない。しかし、Python には非常に豊富な科学技術計算ライブラリや描画ライブラリが提供されており、欧州中期予報センター (ECMWF), 英国気象局 (UKMO), 米国環境予測センター (NCEP), 米国大気研究センター (NCAR) などにおいても広く利用されている。特に `matplotlib` と呼ばれる描画ライブラリは非常に充実しており、気象関連の描画でも広く活用できる。そのような中で、著者は Python の気象庁での数値予報モデル開発への活用の可能性について探り始めたところであり、たとえば、原 (2016a) や原 (2016b) に掲載されている図（天気図を除く）は、`matplotlib` や UKMO によって開発・公開されている Iris を用いて、すべて Python を使って描画したものである。

また、C 言語や Fortran で記述された関数（サブルーチン）を Python から呼び出すことが簡単にできる。Ruby でも C 言語で記述された関数を呼び出すことはできるが、Ruby のメソッドとして登録するためのプログラムを C 言語で作成してコンパイルする必要があり、C 言語のプログラムを書く一手間が必要である。しかし、Python においては `ctypes` というモジュールを利用すれば、C 言語の関数群のソースコードには手を加えずに、利用したい関数のインターフェースの情報を Python で記述した上で、その関数がアーカイブされているライブラリ [6] を読み込めば、Python から利用することができる。NuSDaS ライブラリに対してそれを行い、NuSDaS API を C 言語や Fortran から利用している感覚で Python で使えるように、ラッパープログラムを記述したモジュールを付録 4.7.A に示した [7]。このモジュールを利用して NuSDaS のデータを Python で可視化した例が図 4.7.3 である。この図を作成するのに用いた Python スクリプトを付録 4.7.B に示した。

Python には Numpy[8] と呼ばれるベクトル計算に適した数値計算パッケージや、pandas[9] と呼ばれる統計計算パッケージもある。また、SQLite, PostgreSQL, MySQL などのリレーショナルデータベース管理システム (RDBMS: Relational DataBase Management System) の操作を行うことができる。最近、メソモデルの開発ではモデルの検証の一部に RDBMS を活用している。その際に、RDBMS のデータベースの作成、RDBMS からのデータの取得、データ処理、可視化を Python で行っている。たとえば、原 (2016b) では冬季の北西風の寒気移流に伴う気温の低下について、輪島の 925 hPa におけるゾンデ観測の風向に北風成分がある場合とない場合で条件付きサンプリング (原 2013) を行い、その 2 つの場合にゾンデ地点ごと、予測時間ごとに 925 hPa の気温の平均誤差を求め、地図にプロットしている。この図を作成するにあたっては、まず、ゾンデ観測値とモデル予測値を地点番号、観測時刻、予測時間などとともに RDBMS 上のそれぞれ観測値テーブル、予測値テーブルに格納する。そして、RDBMS に格納した観測値と予測値それぞれを pandas を通じて SQL 文でデータを取得して予測値と観測値の対応付けを行い、地点ごと、予測時間ごとといったグループ分けをした上で統計を行っている。

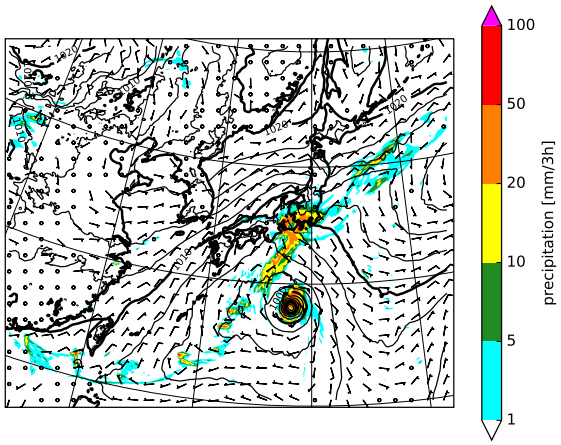

図 4.7.3 Python で描画した NuSDaS データの例。ある時刻の MSM 予測値の前 3 時間降水量（塗り分け、単位：mm/3h）、海面更正気圧（等値線、単位： hPa）、地上風（矢羽、短い矢羽は 5 m/s に対応）。

[3] `https://www.python.org`
[4] `https://www.ruby-lang.org`
[5] 数値予報システムの管理や NAPEX には Ruby が多くのところで使われている。また、ジョブスケジューラ ROSE も Ruby によって書かれている。
[6] 動的リンクができるようにコンパイルされている必要がある。
[7] 図 4.7.3 を描画するのに必要な関数だけ示したが、NuSDaS API のプロトタイプ宣言（`nusdas.h`）から、すべての公開 API について同様のコードをスクリプト言語でほぼ自動的に作成することが可能である。
[8] `http://www.numpy.org`
[9] `http://pandas.pydata.org`

参考文献

原旅人, 2013: Conditional Sampling. 数値予報課報告・別冊第 59 号, 気象庁予報部, 81–83.

原旅人, 2016a: 渦位の追跡によって見る MSM における境界値の影響. 平成 28 年度数値予報研修テキスト, 気象庁予報部, 89–99.

原旅人, 2016b: 寒気移流に伴う下層の気温低下の MSM による予測について. 平成 28 年度数値予報研修テキスト, 気象庁予報部, 100–104.

加藤輝之, 2004: PostScript コードを生成する描画ツール “PLOTPS” マニュアル. 気象研究所技術報告第 44 号.

付録 4.7.A　Python で NuSDaS ライブラリを利用するためのモジュール (nusdas.py)

```python
# -*- coding: utf-8 -*-

from ctypes import *
import numpy as np

# 動的リンクができるライブラリのロード
libnwp = cdll.LoadLibrary("libnwp.so")
libnus = cdll.LoadLibrary("libnusdas.so")

# 主な関数の返値と引数の仕様を記述
nusdas_read_ct = libnus.NuSDaS_read
nusdas_read_ct.restype = c_int32
nusdas_read_ct.argtypes = (c_char_p, c_char_p, c_char_p, POINTER(c_int32),
                           c_char_p, POINTER(c_int32), c_char_p, c_char_p,
                           np.ctypeslib.ndpointer(dtype=np.float32), c_char_p,
                           POINTER(c_int32))

nwp_ymdhm2seq_ct = libnwp.NWP_ymdhm2seq
nwp_ymdhm2seq_ct.restype = c_int32
nwp_ymdhm2seq_ct.argtypes = (c_int32, c_int32, c_int32, c_int32, c_int32)

nusdas_grid_ct = libnus.NuSDaS_grid
nusdas_grid_ct.restype = c_int32
nusdas_grid_ct.argtypes = (
    c_char_p, c_char_p, c_char_p, POINTER(c_int32), c_char_p, POINTER(c_int32),
    c_char_p, np.ctypeslib.ndpointer(dtype=np.int32),
    np.ctypeslib.ndpointer(dtype=np.float32), c_char_p, c_char_p)


# 直感的に利用できるような wrapper を定義
def nusdas_read(type1, type2, type3, ibase, member, ivalid, level, elem, data,
                dtype, dnum):
    icond = nusdas_read_ct(type1, type2, type3,
                           byref(c_int32(ibase)), member,
                           byref(c_int32(ivalid)), level, elem, data, dtype,
                           byref(c_int32(dnum)))
    return icond


def nwp_ymdhm2seq(year, month, day, hour, mi):
    iseq = nwp_ymdhm2seq_ct(
        c_int32(year), c_int32(month), c_int32(day), c_int32(hour),
        c_int32(mi))
    return iseq


def nusdas_grid(type1, type2, type3, ibase, member, ivalid, npro, gsize, ginfo,
                mean, getflg):
    icond = nusdas_grid_ct(type1, type2, type3,
                           byref(c_int32(ibase)), member,
                           byref(c_int32(ivalid)), npro, gsize, ginfo, mean,
                           getflg)
    return icond
```

付録 4.7.B　図 4.7.3 の描画に用いた Python スクリプト

```python
# -*- coding: utf-8 -*-

from nusdas import *
from ctypes import *
import numpy as np
import iris
import matplotlib.pyplot as plt
import iris.plot as iplt

# 降水のカラーとレベルの設定
rain_colors = ['#FFFFFF', '#00FFFF', '#228B22',
               '#FFFF00', '#FF8000', '#FF0000', '#FF00FF']
rain_levels = [1, 5, 10, 20, 50, 100]
rr = 3  # 降水の積算時間（単位:時間）
bint = 30  # 矢羽を描く格子間隔
kt = 12  # 描画する予報時間

# 描画する面と要素
elvs = (('SURF  ', 'RAIN  '), ('SURF  ', 'PSEA  '), ('SURF  ', 'WIND  '))

#初期時刻、NuSDaS TYPE1, 2, 3 の設定
year, month, day, hour, minute = 2015, 9, 7, 15, 0
type1, type2, type3, member = '_MSMLMLY', 'FCSV', 'SFC2', '    '

# 初期時刻を通算分に変換
ibase = nwp_ymdhm2seq(year, month, day, hour, minute)

# nusdas_grid の結果を格納する配列の確保
npro = create_string_buffer(4)
mean = create_string_buffer(4)
gsize = np.empty([2], dtype=np.int32)
ginfo = np.empty([14], dtype=np.float32)

# nusdas_grid の実行による格子情報の取得
icond = nusdas_grid(type1, type2, type3, ibase, member, ibase,
                    npro, gsize, ginfo, mean, 'get')
if (icond != 0):
    raise Exception("nusdas_grid error")

# nusdas_grid で得た情報の転記
nx, ny = gsize
xi, xj, xlat, xlon, dx, dy, slat1, slon1, slat2, slon2 = ginfo[0:10]
dnum = nx * ny

# numpy 配列の確保
data = np.empty([2, ny, nx], dtype=np.float32)

# 地図投影法情報の設定（ここでは Lambert 座標であることは前提にする）
# 座標の単位は m
# 緯度経度が central_lat, central_lon である点を原点とした座標系を
# false_easting, false_northing だけ平行移動した座標系
prj = iris.coord_systems.LambertConformal(
    central_lon=xlon,
    central_lat=xlat,
    secant_latitudes=(slat1, slat2),
    false_easting=xi * dx,
    false_northing=(ny - xj + 1) * dy)

# 格子点の定義
x_dim = iris.coords.DimCoord(
    np.linspace(1, nx, nx) * dx,
    standard_name='projection_x_coordinate',
    long_name='x coordinate of projection',
    units='m',
    coord_system=prj)
y_dim = iris.coords.DimCoord(
    np.linspace(1, ny, ny) * dy,
    standard_name='projection_y_coordinate',
    long_name='y coordinate of projection',
    units='m',
    coord_system=prj)
dims = ((y_dim, 0), (x_dim, 1))

# 予報対象時刻の通算分
ivalid = ibase + kt * 60
```

```
76
77  # 描画キャンバスの用意
78  fig = plt.figure()
79  # Axes を作成
80  ax = fig.add_subplot(1, 1, 1, projection=prj.as_cartopy_projection())
81
82  for plane, elem in elvs:
83      if (elem == 'RAIN  '):
84          # ivalid と ivalid - rr * 60 のときの RAIN を nusdas_read
85          for ii, ktrr in enumerate((0, rr)):
86              icond = nusdas_read(type1, type2, type3,
87                                  ibase, member, ivalid - ktrr * 60,
88                                  plane, elem, data[ii,:,:], 'R4', dnum)
89              if (icond != dnum):
90                  raise Exception("nusdas_read error")
91          #前 3 時間降水量に変換
92          data[0, :, :] = data[0, :, :] - data[1, :, :]
93
94          # cube の作成。cube のデータの南北方向は南側が起点なので、南北を入れ替えてセット
95          cube = iris.cube.Cube(data[0, ::-1, :], dim_coords_and_dims=dims)
96      elif (elem == 'WIND  '):
97          # 風の場合は、cube を配列として用意する。
98          cube = []
99          for ii, ee in enumerate(('U     ', 'V     ')):
100             # U と V をそれぞれ読み出し、cube 配列に入れる。
101             icond = nusdas_read(type1, type2, type3,
102                                 ibase, member, ivalid,
103                                 plane, ee, data[ii,:,:], 'R4', dnum)
104             if (icond != dnum):
105                 raise Exception("nusdas_read error")
106             cube.append(
107                 iris.cube.Cube(data[ii, ::-1, :], dim_coords_and_dims=dims))
108     else:
109         icond = nusdas_read(type1, type2, type3,
110                             ibase, member, ivalid,
111                             plane, elem, data[0,:,:], 'R4', dnum)
112         if (icond != dnum):
113             raise Exception("nusdas_read error")
114         cube = iris.cube.Cube(data[0, ::-1, :], dim_coords_and_dims=dims)
115
116     if (elem == 'PSEA  '):
117         # 等値線を描画。色は黒（'k'）で、900〜1048 までを描画。
118         # 5 本に 1 本、等値線を太くする。
119         cnt = iplt.contour(cube, levels=np.arange(900,1050,2),
120                            colors='k', linewidths=(2,1,1,1,1), axes=ax)
121         # 940〜1030 について、10 ごとに等値線の値を描画する。
122         ax.clabel(cnt, np.arange(940, 1040, 10), fmt="%d", fontsize=8)
123     elif (elem == 'RAIN  '):
124         # 塗り分け。レベルと色を設定。
125         cnt = iplt.contourf(cube, levels=rain_levels,
126                             colors=rain_colors, extend='both', axes=ax)
127         # カラーバーを描画
128         cbar = fig.colorbar(cnt)
129         cbar.set_label("precipitation [mm/3h]")
130     elif (elem == 'WIND  '):
131         # 矢羽を間隔 bint で描画
132         ax.barbs(
133             cube[0].coords()[1].points[::bint],  # x 座標の値
134             cube[0].coords()[0].points[::bint],  # y 座標の値
135             cube[0].data[::bint, ::bint],  # U
136             cube[1].data[::bint, ::bint],  # V
137             color='k',
138             transform=prj.as_cartopy_projection(),
139             length=4,
140             linewidth=1.0)
141
142 #海岸線を描画
143 ax.coastlines(resolution='50m', linewidth=2)
144 #緯度経度線を北緯 0〜90 度、東経 100〜180 度の範囲に 10 度ごとに描画。
145 ax.gridlines(
146     xlocs=np.arange(100, 180, 10), ylocs=np.arange(0, 90, 10), linestyle='-')
147
148 #描画した図を PDF で保存する。
149 plt.savefig("out.pdf")
```