数値予報課報告・別冊第 63 号

数値予報モデル開発のための基盤整備および開発管理

平成 29 年 3 月
March 2017

気　象　庁　予　報　部

はじめに*

数値予報課報告・別冊では例年特定のテーマを掲げて刊行している。従来は数値予報モデルやデータ同化、観測データ利活用など数値予報プロダクトの仕様や品質に直接かかわる題材に焦点をあてて最新の技術開発の成果や進捗を報告することが多かった。今回は少し趣を変えて、そうした技術開発の現場における基盤環境の整備や開発管理の実際について初めて網羅的に取り上げることとした。

数値予報業務の発足から暫くの間はソフトウェアとしての数値予報システムは比較的小振りで、例えば往年の 4 層北半球プリミティブモデルのプログラムは 2000 行ほどであった。モデル技術開発に携わる職員も少人数で、手工業的な開発方式でも十分通用する規模であったといえる。その後の数値予報システムは電子計算機の発展と共に飛躍的な精緻化・高精度化を果たし、これに伴ってそのプログラムも極めて複雑かつ巨大なものへと進化した。同時に、新たな知見や技術を数値予報システムに導入するためには、正当な手順によって調査実験を企画し、細心の注意を払ってこれを実施し、その結果を多面的かつ客観的に評価することが不可欠となっている。今後も継続的に数値予報の精度向上を図っていくには、大規模で多岐にわたるプログラムを複数の担当者が分担・協調して効果的かつ効率的に技術開発を進めていかなければならない。このため気象庁では、数値予報システムの開発工程の可視化・共有化を推進し、モデル技術開発担当者の知見の共有、技術力向上、相互点検の実施等を通じて、機能的で組織的な開発体制の確保を図っている。さらに、プログラムの履歴管理や実験制御、結果の可視化や検証など基盤的な業務を支援するシステムを構築し共通化すると共に、可能な限り自動化と客観化を図り開発効率の向上に努めている。

こうした取り組みは従前から地道に積み重ねられてきたところであるが、近年特にその重要性が再認識されたことから、気象庁技術開発推進本部モデル技術開発部会の下、部局横断的に実施されてモデル技術開発の促進に大きく貢献している。

本報告によって、こうした数値予報モデル開発のための基盤整備や開発管理の重要性と気象庁における精力的な取り組みについて理解が広まると共に、将来モデル技術開発に携わる職員の良い助けとなれば幸いである。

---

* 松村 崇行

数値予報モデル開発のための基盤整備および開発管理

目 次

第1章　序論

1.1　はじめに [1]

1.1.1　数値予報モデル開発を支援する基盤環境の整備の必要性

数年前から、数値予報課における数値予報モデル開発[2]は、単なるスキームの取り替えやパラメータチューニングを中心としたものから、モデル予測の誤差の原因やモデルの挙動を既知の事実や知見を論理的に積み上げることによって理解し、その理解に基づいてモデルの修正を試みる科学的なアプローチを重視するように変化しつつある。数値予報課報告・別冊第 58 号 (気象庁予報部 2012) や第 59 号 (気象庁予報部 2013) では科学的なアプローチによるモデルの問題発見とその解決に向けた手法についてまとめ、その後の開発をより科学的なものにする一つのきっかけとなった。

科学的なアプローチによる開発では、事実や知見を論理的に組み合わせて新たな結論を導き出し、その結論を別の知見などとさらに組み合わせることを繰り返し進める。そのため、科学的なアプローチによるモデル開発を行うためには、論理的な思考力を強化するとともに、現象の動態を把握すること、モデルの予測データを検証すること、現象とモデル予測に関する物理的考察を行うこと、気象学の知見を習得すること、また最新の研究動向を把握することなどを通じて、開発に利用できる事実や知見を増やす努力が日々必要で、大きな労力を要する。その上、モデルが高度化、複雑化している中で、モデルのさらなる精度向上を目指すためには、より高度な科学的知見が求められる。

一方、数値予報システムは多くのプログラムから構成されており、プログラム間で必要なデータをもれなく受け渡したり、依存関係を考慮したりしながらプログラムを実行し、その終了監視をして次のプログラムを実行するといったジョブ制御が必要となる。また、出力結果の検証、可視化のために膨大な出力データを取り扱う必要もある。計算機の計算能力の向上とともに、数値予報システムで扱うデータの量や種類は入力、出力ともに膨大になってきており、その扱いには高度な技術が要求されるようになってきている。また、モデルの問題点を見出すために、様々な観点で可視化を行うには、出力データが大きくなる中で、そのデータをより短時間で効率よく描画できる技術や、問題点を浮かび上がらせることができる表現方法の考案とその実装などが必要となっている。

従前の数値予報課では、上で述べた 2 つのスキル、すなわち、モデル開発に必要な気象学や物理学の知見などのサイエンス的なスキルと、実験システムの構築、データの可視化や検証のためのデータのハンドリング、ソースコードの最適化、並列化などのエンジニアリング的なスキルの両方がモデル開発者には求められた。しかし、近年、それぞれが高度化、複雑化する中で、両方の性質の異なるスキルを同時に求めることが難しくなってきている。その結果、モデル開発者のスキルが中途半端な状態となり、データの可視化や実験システムの構築に時間が取られ、モデル改良のためのサイエンス的な考察や思考に十分な時間を費やすことができず、効率のよいモデル開発を阻害している面があった。

原・高谷 (2013) で紹介したように、欧州中期予報センター (ECMWF) や英国気象局 (UKMO) では、開発環境を組織全体でできるだけ統一した上で、モデル開発にかかわる基盤環境の整備を専門に行う部署があり、モデル開発者はその基盤環境を利用して、科学的な思考に集中できる環境が整えられている。当庁においてもこのような部門の必要性が認識されて、実験システムの構築や可視化環境の整備といったモデル開発を支援するための基盤環境構築を担う基盤整備グループが、2011 年に数値予報課数値予報班の一つのグループとして発足した。第 3 章で紹介する数値解析予報実験システム (NAPEX) は、以前は数値予報モデルの開発者が自ら構築して、スーパーコンピュータの更新のたびに新しい大きな変更が加えられてきた。第 9 世代スーパーコンピュータシステムの運用開始とともに導入された NAPEX は、スーパーコンピュータシステムで実行する数値予報ルーチンの管理システムを構築した数値予報課プログラム班が構築に参加し、システム面の専門性が活かされた実験システムとなっている。

本報告では、このようなモデル開発環境の変化を踏まえ、モデル開発を支援するための基盤環境とその整備をテーマの一つとする。

1.1.2　開発管理

基盤整備の動きとともにモデル開発の効率を高めることを目的に進められたのが、開発管理のための環境構築とその実行である。

すでに室井 (2013) で詳細に論じられているように、数値予報モデル開発が必然的に大規模かつ横断的なものになったことから、充分な議論を経て開発計画を作成の上、その工程管理や情報・成果の共有、ソースコードのバージョン管理を行うこと、すなわち「開発管理」が必要となった。開発管理は、開発の進捗状況を管理者の立場から把握することが主目的ではなく、どのような開発が進行しているのかを開発者間で把握しやすくすること、そして、開発の際に議論したこと、考えたこと、作業などを記録して、現在の開発者はもちろんのこと、未来の開発者への説明責任を果たせるように

[1] 原 旅人

[2] 本報告では、便宜上、解析システムなど、数値予報モデル以外の数値予報システムを構成する部分の開発を含めて「数値予報モデル開発」または「モデル開発」と呼ぶことにする。

することが大きな目的である。近年のモデル開発においては、組織横断的な開発が行われることも多くなり、開発者のコミュニティが大きくなりつつあるが、その結果、他の組織の開発の進捗は把握しにくい場合がある。また、モデル開発は継続的に行われるものであるが、その開発を未来永劫にわたって同じ開発者が行うことはなく、開発者の入れ替わりがある。そのような中で、開発者が替わったことで過去の知見が継承されないようなことがあれば、新規の開発者のスムーズな参入を妨げるとともに、すでに分かっている知見を再度、新しい開発者が自ら見出すために時間を費やすことになるなど、開発が非効率になる。このように、成果のまとめだけでなく、開発の記録を残しておくことは、継続的なモデル開発を実施するために必須のことである。

開発管理のための基盤になるシステムが第 2.2 節、第 2.3 節でそれぞれ紹介するプロジェクト管理システムとバージョン管理システムである。

バージョン管理システムは、ソースコードをはじめとするファイルの変更履歴を管理することが主たる役割の一つである。バージョン管理システムを利用する際には、ユーザがリポジトリと呼ばれるファイルの保管庫に変更内容を登録することで、ファイルに加えた変更がいつ、誰によって、どのような内容で行われたのか（変更差分）をそれに対する変更者のコメント（ログ）とともに記録する。また、変更記録に基づいて、最新版ではなく任意の時刻のファイルの状態を取り出すことも可能である。

バージョン管理システムを用いることで、ソースコードそのものの変更履歴を追跡できるようになる。しかし、その意図や背景、テストの結果などはバージョン管理システムに残された記録だけを見てもわからないことが多い。すでに述べたように、バージョン管理システムには変更者のコメント（ログ）を記録することができるが、記入できるのはテキストだけであり、図を用いるなどした詳細な説明をログに書き込むことは困難である。

さらに、モデル開発はソースコードの修正だけにとどまらない。実験や日々の予測結果から気がついた問題点の記録とその調査、実験結果の検証とその考察、それらに対する開発者間での議論もモデル開発における重要な過程の一つである。また、実験の結果、現業化に至った開発事項については、報告や刊行物などにその成果がまとめられることもあるが、結果が期待したものではなかった実験の情報もその後の開発においては重要な情報になりうる。しかし、従来、これらの情報は、そのときの開発者間のみで共有されることが多く、それ以外に共有されたり、将来の開発者のために記録として残されることが少なかった。また、添付資料なども開発者個人の保管場所に保存することが多かったために、人事異動とともに失われることも多かった。このような状況は、時間が経過してから、当時の結果や議論を追跡することには困難を伴い、過去の開発者の知見も活かした効率的なモデル開発を阻害する一つの要因でもあった。

そのような背景のもと、開発過程の記録を残すことを目的に導入したのがプロジェクト管理システムである。プロジェクト管理システムは、ソースコードの修正はもちろんのこと、モデル開発におけるさまざまな過程を記録し、また、それを一覧にすることで進捗の把握にも利用できる。バージョン管理システムと併用することで、ソースコードの変更とその詳細な説明を一体として扱うことができる。また、作業にとりかかる際には、やるべき作業のリストアップにも用いることができる。このように作業過程、調査資料、議論を一か所に集約しておくことで、将来の開発者がそれらを見直すことも容易となる。その記録は庁内からは制限なく閲覧できるようになっているので、他の部署のモデルの開発状況を把握したり、それを基に共同開発につなげやすくなった。

現在、気象庁におけるモデル開発では、バージョン管理システム、およびプロジェクト管理システムを利用した開発管理が推進されている。こうした開発管理の実際を報告するのが、本報告のもう一つのテーマである。

本報告の 2 つのテーマである基盤整備と開発管理の強化は、冒頭で述べた科学的なモデル開発の推進に大きく寄与している。すでに述べたように、科学的なモデル開発のためには既知の事実や知見を論理的に組み合わせることが必要であるが、その論理的考察に用いた事実や知見、結論を導き出した論理を開発管理ツールに記述することで、開発記録だけにとどまらず、論理の確認、開発者間での共有にも活用している。また、NAPEX や汎用的な描画ツールの整備により、科学的な思考に費やす時間をより確保できるようになった。科学的な開発姿勢と、それを支援する基盤整備と開発管理の強化の集大成が 2016 年の全球モデルの改良 (米原 2016) や、次世代非静力学モデルの開発 (気象庁予報部 2014) と言えよう。

1.1.3 本報告の内容・構成

本報告では、近年、気象庁のモデル開発で推進されてきた基盤整備、開発管理に焦点をあてて、その背景とこれまでの取り組みについてまとめる。

本節では、これまでに基盤整備や開発管理の必要性とその背景を説明してきた。開発管理の強化は数値予報モデル開発のプロセスの明確化とともに進められたものである。そこで、本章の後半では、数値予報課における開発プロセス、およびそのプロセスにおける開発管理の活用について論じる。また、最近の英国気象局における開発や現業化の判断などのプロセスの最新事情について、最近まで英国気象局に派遣されていた

モデル開発者から報告する。

第 2 章では開発管理について取り上げる。気象庁における開発管理に対する取り組みを概観したのちに、そのためのシステムであるプロジェクト管理システムやバージョン管理システムの一般的な事項について簡単に説明する。その説明を踏まえて、数値予報課における開発管理ツールの活用について紹介する。加えて、検証およびそれを実施するツールのあり方についても触れる。検証ツールは数値予報システムの開発において行うべき評価を具体化したものであり、開発経験から得られた科学的知見が集積されていくものである。そのため、数値予報システムだけではなく検証ツールも開発管理の対象に含むことは重要である。

第 3 章では気象庁の数値予報モデル開発の重要な基盤の一つである NAPEX について述べる。NAPEX の必要性とその歴史を概観したのち、現在の NAPEX の設計思想と実装、利用について紹介する。

第 4 章では数値予報モデル開発で必要かつ重要な基盤技術であるデータハンドリングと可視化について取り上げる。気象庁の数値予報システムで用いられているデータフォーマット、数値予報課で用いられている可視化ツールについて概説する。

本報告の発行を通じて、開発基盤整備の重要性や開発管理への理解の深まりとそのさらなる普及を期待したい。

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.

気象庁予報部, 2012: 物理過程の改善にむけて (I). 数値予報課報告・別冊第 58 号, 気象庁予報部.

気象庁予報部, 2013: 物理過程の改善にむけて (II). 数値予報課報告・別冊第 59 号, 気象庁予報部.

気象庁予報部, 2014: 次世代非静力学モデル asuca. 数値予報課報告・別冊第 60 号, 気象庁予報部.

室井ちあし, 2013: 開発管理の必要性. 数値予報課報告・別冊第 59 号, 気象庁予報部, 192–194.

米原仁, 2016: 全球数値予報システムの物理過程改良の概要. 平成 28 年度数値予報研修テキスト, 気象庁予報部, 1–3.

1.2 数値予報システム開発のプロセス [1]

1.2.1 はじめに

数値予報は、国内外の観測データの収集に始まり、次にそのデータの品質管理（誤差の大きいデータの除去や補正）を行い、観測値と第一推定値を用いてデータ同化システムにより初期値を作成する。そして、初期値から数値予報モデルを用いて将来の大気状態を予測し、その結果は統計的手法を用いて補正される（ガイダンス）。これら一連の処理はスーパーコンピュータシステム等で自動で実行され、そのための数多くのプログラムが用いられる (室井 2012)。

本節では上記の流れの中で観測データの品質管理から数値予報モデルによる予測までを数値予報システムと呼ぶこととし、その開発プロセスについて述べる。数値予報システムは気象学、物理学、計算機科学、数値流体力学、数学、統計学等の様々な科学分野の知見が必要となる。そのため、多くの開発者の知見を集める必要がある。また、室井 (2013) に述べられているように、近年、数値予報システムの目的が多様化し、開発は必然的に大規模かつ横断的なものになっている。そのため、開発管理が重要となり、その中の一つである開発プロセスの策定が重要となっている。

気象庁で現業運用される数値予報システムは気象業務の根幹を支えるものであり、まず第一に定められた時間内に処理を終え、障害を起こさない（万一の場合は速やかな復旧や代替手段を取る）ことが重要である。その上で、予測精度の向上を図る必要がある。

そこで、本節ではこれまで数値予報課で検討してきた数値予報システムの開発プロセスについて述べる。まず、予測精度向上のためにはそれを阻害する誤差要因を把握する必要があり、伝統的に認識されてきた誤差要因を第 1.2.2 項 に簡単にまとめた。そして、これまでにあまり認識されてこなかった誤差要因を含めて今後の開発で検討すべき事項について第 1.2.3 項 でまとめた。第 1.2.4 項ではこれらの検討すべき事項を踏まえて実際の開発の進め方（フロー）についてまとめた。

1.2.2 数値予報システムの誤差要因

精度向上のためには、誤差を軽減させる必要がある。誤差要因には様々なものがあるが、伝統的に認識されてきた主たるものだけでも以下が挙げられる。

- 初期条件の誤差
  - 観測の時間・空間分解能に起因する誤差
  - 観測機器の持つ誤差
  - 観測の代表性に起因する誤差
  - 品質管理手法に起因する誤差
  - データ同化手法に起因する誤差（データ同化システムの分解能による制限や観測演算子に起因するものを含む）
- 数値予報モデルの誤差
  - 空間・時間分解能の制限による誤差（離散化手法に起因する誤差を含む）
  - 物理過程の不確実性に起因する誤差
  - 数値予報モデルが考慮しない大気中の素過程による誤差
- 境界条件の誤差
  - 地形の表現の制限に起因する誤差
  - 地表面状態の不確実性に起因する誤差
  - 海面・海氷状態の不確実性に起因する誤差
  - 人為的な上部境界条件に起因する誤差
  - 領域モデルにおける側面境界条件に起因する誤差

有限の数値演算による数値予報システムではこれらの誤差（及びここに記載しない要因による誤差）を全てゼロにすることができないものの、全ての誤差を可能な限り小さくしていくことにより予測精度向上が期待できる。

効率的な開発のためにはこれまでの開発において誤差がある程度減少してきた要因に対する開発よりも相対的に大きな誤差要因に対する開発により注力することが望ましい [2]。しかし、どの誤差要因が大きいか特定することは困難であり、それを発見するための開発も必要である。その際には、伝統的には認識されていなかった新たな誤差要因が見つかることもある。次項では、ここで述べた誤差要因を念頭において、今後の開発において考慮すべき事項について述べる。

1.2.3 数値予報システムの開発において今後考慮すべき事項

現業数値予報システムの開発は気象庁において 50 年以上の歴史があり、その開発の歴史の中で様々な誤差が減少している。しかし、全ての誤差が一様に減少してきたわけではないため、時代によって相対的に大きな誤差を生じる要因は異なっている。また、これまでの多くの先人の努力により現業数値予報システムが成熟してきた現状では、これまでは想定しなかったことが誤差の要因となることや相対的に小さかった誤差が大きくなることがあるだろう。

数値予報課では第 1.2.2 項で述べた問題意識から、数値予報課における開発を通じて得られた教訓や海外の数値予報センターの動向を基に、今後の開発で着目すべき事項について議論を行ってきた。現時点における議論を通じて得られた知見を以下の通りまとめた。

(1) 科学的根拠と統計的根拠

現業数値予報システムの開発において真の予測精度向上を担保するためには、科学的根拠が重要なことはいうまでもない。問題は、どのように真の予測精度向上を担保するかである。ここで、真の予測精度向上と

[1] 石田 純一

[2] ただし、それぞれの誤差を減らすための開発は、開発者の人材育成までを考慮すると数年から十数年は必要となるため、現時点では相対的に小さな誤差要因に対しても継続的に開発を行っていく必要があることにも留意しなくてはならない。

は、開発成果が実用化された後の事例（すなわち、未来の事例）に対する精度向上を指す。

後述の開発フロー（図 1.2.1）にあるように基礎開発の段階で様々な手法により科学的根拠を積み上げていく。その後、変更の導入前後で比較実験を行い、統計検証を通じて検証スコアが向上する結果が得られたことも真の予測精度向上に対する統計的根拠の一つとなりうる。ここで、事前の基礎開発を通じて積み上げた根拠に対して比較実験により得られた統計的根拠をどのように扱うべきか議論の余地がある。まず、比較実験は過去のデータを用いて実施するより他はないが、現業数値予報システムの改善で求められるのは前述の通り、未来における予測精度向上であり、本質的な制約がある。例えば、パラメータ等について実験期間に過剰に合わせることにより、実験期間では精度が向上してもそれ以外の期間ではむしろ精度が悪化するというオーバーフィッティングの問題が生じる懸念がある。また、これまでの開発により数値予報システムの誤差は非常に小さくなってきたため、真の改善の影響（シグナル）は予測精度の変動（ノイズ）と比べて小さくなってきた（即ち、S/N 比が低くなった）。このような問題意識に対して、Geer (2016) による欧州中期予報センターにおける検討結果があり、以下に簡単に抜粋して紹介する。

- 数値予報の開発で典型的な 0.5%のスコア向上を信頼できる精度で検証するには半年程度のサイクル実験が必要だが計算機資源的にどの予報センターでも困難。
- そのため、伝統的なスコアは予測における科学的な開発の質の指針に常に用いることができない。
- 比較実験前の基礎開発のみを根拠とする（即ち比較実験を行わない）ことはバグの可能性や基礎開発の科学的根拠の不完全性を考えると不適切。
- 従って、より少ない実験設定で多くのサンプルによる実験を行うことが最適であろう。
- 様々な変更のそれぞれに対して異なる実験を行い、予測精度（スコア）により良い変更を選択する戦略は無駄が多い。
- 実験設定の選定においては長期間の実験に先立って個々の変更を注意深く評価し、良い変更に対してフルの実験を行うのが良い。

即ち、従前と比して比較実験を通じて統計的根拠を得るためには個々の変更に対して事前の基礎開発の段階での科学的根拠を重視する必要があり、基礎開発の段階で妥当と考えられる変更をまとめた（パッケージ化した）比較実験が必要である。スコアの良し悪しはその大小で判断できる [3] であろう。しかし、基礎開発の段階での科学的根拠の妥当性は自明でないこともありうることから、議論を積み重ねる必要がある。

今後の開発においてさらに予測誤差が減少する（さらに S/N 比が低くなる）ことを期待すれば、益々このような観点での開発が重要となると考えられる。

(2) compensating errors を考慮することの重要性

compensating errors とは堀田・原 (2012) に述べられている通り、過程 A の持つ誤差と過程 B の持つ誤差の符号が逆で、合計の誤差が小さくなることを意味する。このことは、合計の誤差が小さく見えている場合に、仮に過程 A の誤差が小さくなるような修正を施した場合に、合計の誤差がかえって増大することを意味する。従って、ボトムアップ・アプローチ (堀田・原 2012) により個々の過程の問題点を見つけ出して修正することは精度向上の必要条件であっても十分条件ではなく、これに加えて別の検討が必要であることを意味する。

このことは数値予報モデルの個々の物理過程の間で生じうるだけでなく、物理過程と境界条件の間、あるいはデータ同化サイクルにおいて、数値予報モデルとデータ同化システム（観測データの品質管理を含む）の間等でも生じうる。

過去においても、compensating errors は存在していたと考えられるが、数値予報システムが成熟し、合計の誤差が小さく見えることが多くなってきたことから、近年になってこの誤差が顕在化してきたと考えられる。

(3) 実装の細部まで検討することの重要性

数値予報システムの開発では最終的に解くべき方程式系（多くの場合は時間・空間方向に連続な偏微分方程式系となる）を導出する必要がある。例えば、サブグリッドスケール現象のパラメタリゼーションではサブグリッドスケールの現象がそれより大きいスケールの現象に与える影響をモデル化し、それに沿って方程式系が決まる。この方程式系の導出段階では如何にして適切なモデル化を行うかが重要であると共に計算機資源等の制約を念頭において何らかの近似を用いることも検討する必要がある。続いて、この方程式系を離散化し、最終的には Fortran 等のプログラムコードに落とし込んでいく。この離散化やプログラムコードへの変換においては多くの任意性がある。また、方程式系では単なる定数として表されるものが実際には限られた観測事実から得られた幅を持つ値しか得られない場合であっても、数値予報システムに実装する際には一意に定める必要がある。このような例として、物理過程の様々なパラメータ、観測データの品質管理における間引き手法や間引き間隔、初期値や境界値等の作成時の水平・鉛直内挿手法等多くのものが挙げられる。さらに、実装においては処理順序やループ範囲等まで一意に定めなくてはならない。

従前はパラメタリゼーションにおけるモデル化や方程式系の妥当性が予測誤差に大きな影響を与えるとの認識の下、これらについての検討に多くの注意が払われてきた。この検討の重要性は引き続き残るものの、

[3] ただし、複数のスコアがある場合に、その優先度を決定することは困難が伴う。

これまでにあまり検討されなかった実装の細部におけるわずかな違いが予測誤差に与える影響が大きいことがありうることが認識されてきた。このことを指して、「Minor treatment の重要性[4]」と呼ぶこともある。

これも、数値予報の開発の黎明期ではシステム全体の誤差が大きく、例えば、モデルで考慮されていなかった過程を追加する、といった開発により誤差が減少してきたところ、近年ではシステム全体の誤差の減少に伴い、実装の細部に至るまで検討することが重要となってきたものである。

注意すべき事項として、論文等においてはモデル化や定式化について触れられていても紙幅の制限から実装の細部まで触れられていないことがある。しかしながら、実装の細部の重要性に鑑みて、何らかの形で検討結果を記録する必要がある。このために第 2.2 節で述べられているプロジェクト管理システムを重要なシステムとして位置づけている。

(4) スケール分離の認識の重要性

一般に格子間隔の 5–8 倍のスケールの現象は数値予報モデルでは陽に表現され、格子間隔より小さいスケールの現象はサブグリッドスケールとしてパラメタリゼーションで表現する。そして、その間に位置する格子間隔と同程度から 5 倍程度のスケールは陽に表現することもサブグリッドスケールとしてパラメタライズすることもできないため、新たな扱いが必要となり、これは Grey Zone と呼ばれる。Grey Zone については原 (2012a) で触れられており、境界層、対流さらには地形性重力波についての検討 (Vosper et al. 2016) がなされている。この記述で分かるように、どの現象が Grey Zone と呼ばれるかは数値予報モデルの格子間隔によって異なってくる。すなわち、数値予報モデルの格子間隔を小さくして分解能に起因する誤差を減少させようとする場合は、これまでにパラメタリゼーションとして扱ってきた現象が部分的に陽に表現されるようになり、新たな扱いを検討する必要がある（当然、この場合も第 1.2.3 項 (3) で述べた通り、実装の細部に至るまで検討が必要である）。

(5) 不確実性の大きいパラメータ・予報変数を極力増やさないこと

数値予報システムの開発はこれまでの先人の開発の上に成り立っており、多くの過程が精緻化・高度化されてきた。このことにより、数値予報モデルにおいては特に精度に大きな影響を与えうる大気中の多くの現象がパラメタライズされて取り込まれるようになってきた。即ち、前述の「数値予報モデルが考慮しない大気中の素過程による誤差」や「物理過程の不確実性に起因する誤差」は大きく減少したことを意味する。

そこで、既存の過程の精緻化・高度化に取り組むこととなるが、不確実性の大きいパラメータを増やすと精度向上につながるどころか精度が悪化することがありうることに注意する必要がある。パラメータを追加した場合には何らかの観測値（あるいは観測値や他の予報変数による関数）でその値を与える必要がある。予報変数を追加する場合には、その時間変化を記述する方程式（予報方程式）を追加する必要がある。しかし、第 1.2.3 項 (3) で述べた通り、観測値が限られた観測事実から得られた幅を持つ（不確実性を持つ）値しか得られないことがあり、予測誤差に与える影響が大きいことがある。また、予報方程式は適切な離散化を行わないとノイズが生じて予測精度を悪化させる可能性もある。

不確実性を持つパラメータを決定する際にはパラメータをわずかに変更した複数の比較実験を行うことがある。しかし、これは前述の Geer (2016) が報告した無駄が多い戦略となりうる。特に不確実性の大きいパラメータを多く導入すればするほど組み合わせの数は膨大となる。このような戦略では、多くの開発期間を要することになるか、少数の事例による比較実験に基づいたオーバーフィッティングの問題が生じうる[5]。また、仮にこれらの問題を解決したとしても複数の不確実性の大きいパラメータによる compensating errors の懸念を払拭することは容易ではない。

このことは時としてパラメータや予報変数を減らすという仕様のダウングレードが精度向上をもたらすという逆説的なことがありうることを意味する。実例として、原 (2015) で示されたメソモデルにおける境界層過程の改良が挙げられる。これは、メソモデルにおける境界層過程を Mellor-Yamada-Nakanishi-Niino (MYNN) レベル 3 モデルから MYNN レベル 2.5 モデルに変更したものである。これは名前からも分かる通り、仕様としてはダウングレード（実際にも予報変数を減らしている）であるが、精度向上につながっている。この問題はパラメータの不確実性ではなく予報方程式の解法に起因していたものであるが、闇雲にパラメータや予報変数を増やすことが精度向上につながらないことがありうる例であろう。

精緻化・高度化に向けた開発においては、極力不確実性の小さい物理量を用いてモデル化・方程式系の導出を行っていくこと、不確実性の大きい物理量を導入せざるを得ない場合はいかにして不確実性を減らすかを（非常に困難であるが）検討することが重要である。また、時には精緻化・高度化ではなく新しい知見を活かして既存のパラメータの不確実性を減らすことに注力することが良いこともあるだろう。

[4] http://www.mpimet.mpg.de/fileadmin/atmosphaere/WCRP_Grand_Challenge_Workshop/presentations/GC_Kawai.pdf

[5] この最たる例がいわゆる champion case によるパラメータの推定である。champion case により得られたパラメータの組み合わせが普遍的でないことは直感的にも理解しやすいが、一つの事例でなく複数の事例であっても少数である場合は問題が生じうることに注意が必要である。

(6) 分解能向上のための開発期間

数値予報システムの分解能向上への社会的要請は大きい。これは、より詳細な分布の情報が欲しいという要望もあれば、分解能の制限によりこれまで表現できなかった現象をも予測対象にして欲しいという要望もある。しかし、分解能の制限による誤差は、これまでの数値予報システムの歴史において、大きく改善された誤差要因であり、誤差要因の観点では相対的に誤差の大きさは低下してきていると考えられる。

すなわち、社会的要請を満たしつつ効率的に誤差を軽減させなくてはならない。分解能を向上させるためには、スーパーコンピュータシステムの更新等により得られた計算機能力の向上を利用することとなる。単純に分解能を向上させるだけであれば容易であるが、これまでに述べた事項を念頭におくと精度向上を得るには、相当の開発期間を要する可能性がある。新しい計算機での分解能の仕様が決まってから導入までの 5, 6 年の期間を持つ開発計画として位置づけて開発を行うことが重要であろう。

1.2.4 数値予報課における数値予報システムの開発プロセス

第 1.2.3 項で述べた考慮すべき事項を踏まえて検討した、現在の開発のフローを図 1.2.1 に示す。

開発のフローは数値予報課において「ルーチン変更のガイドライン」（以下、「ガイドライン」という）としてまとめてきた。それに基づいて、数値予報課と気象研究所との連携を促進するため、2015 年度に気象庁技術開発推進本部モデル技術開発部会メソグループにおいてメソ・局地モデルの開発を対象として議論したものを同じく 2015 年度に開催された気象庁予報部・観測部と気象研究所の研究懇談会で数値予報システムの開発全体に対象を広げた上で懇談したものである。

数値予報課における数値予報システムの開発を担当する数値予報班には複数のグループ（全球・台風グループ、メソモデルグループ、観測データ処理グループ、基盤整備グループ）がある。グループの中にもいくつかの開発コミュニティがあり、取り扱う開発課題や開発コミュニティの規模が異なっている。さらに、開発課題により適切な開発プロセスが異なりうる。ここで示した開発フローはいわば最大公約数として集約したものであり、特に基礎開発の段階では具体的な開発プロセスは開発コミュニティにより異なっている。これらについては第 2 章の活用例をご覧いただきたい。

以下では、開発フローの各フェイズで実施すべきことを述べる。なお、前述の通りこのフローは数値予報課と気象研究所の共同開発を念頭において作成したものである。数値予報課が単独で実施する場合でもフローそのものは同じであるが、一部違いがあるため、適宜補足する。

(1) 開発課題検討

開発課題の設定においては、現業数値予報における課題を確認することが必要である。ここで、開発に着手する場合に何をすべきか自明でないが何らかの開発が必要である課題と、何に着手すべきか自明であるものの必要性が必ずしも明確でないものの両者がありうる。

前者には、例えば数値予報のユーザにとって望ましくない予測特性の解決といった課題や安定運用を確保するための計算安定性の向上といった課題が挙げられる。このような課題に対しては、人的・計算機資源とユーザにとっての利益を勘案して優先すべき課題を定め、その上で、具体的な開発課題に落とし込むことが必要である。

一方、後者には、例えば、現在の数値予報システムで用いられている定式化や離散化手法の不十分な点の改善及び新たな観測データの利用に向けた開発等が挙げられる。さらに、国内外で研究ベースで開発された手法を利用することもある。これらの開発課題に対しては、ユーザにとっての利益につながるかどうか、あるいは compensating errors も考慮した上でそもそも予測精度が向上するかどうかを精査する必要がある。その上で、課題として設定するかどうかを決定する。

続いて、開発工程を作成する。開発課題に対して、その段階で十分な科学的根拠が積み上げられているか、開発により想定される予測特性の変化をどれくらい事前に把握できるか、変更により影響の及ぶ範囲等を考慮し、必要な工程を作成する。その場合に関連する文献のレビューがこれらを考慮する際に有益である。ここでいう文献のレビューとは必ずしも論文に限らず、これまでに開発で得られた知見も重要であることから、第 2.2 節で述べられているプロジェクト管理システムでの知見の集積が有益である。特に、持続的な開発においては、集積された過去の知見は欠かせない。

(2) 基礎開発及びその評価

開発課題が設定されれば基礎開発及びその評価のフェイズに移る。ここでは、基本的な開発として、開発プログラムの性能・インパクトを評価する。ルーチンと同等の環境で開発する必要はない（ガイドライン）。このフェイズは科学的根拠を積み上げるもっとも重要なフェイズである。従って、この段階で多くの開発時間を要することが通常である。

この段階での開発手法には開発コミュニティや開発対象により様々なものがある。特に実データでの実験に先立って検討すべきことが多い。例えば、力学過程の開発ではフルモデル（実際に現業数値予報システムで用いられるモデル）への組み込みに先立った理想実験（一様流による移流のテスト (石田ほか 2014) や浅水モデルによるテスト等）やフルモデルへ組み込んだ理想実験（山岳波 (石田ほか 2014) 等）がある。また、物理過程では SCM (Single Column Model) によるテストといった理想実験 (原 2012b) がある。このような

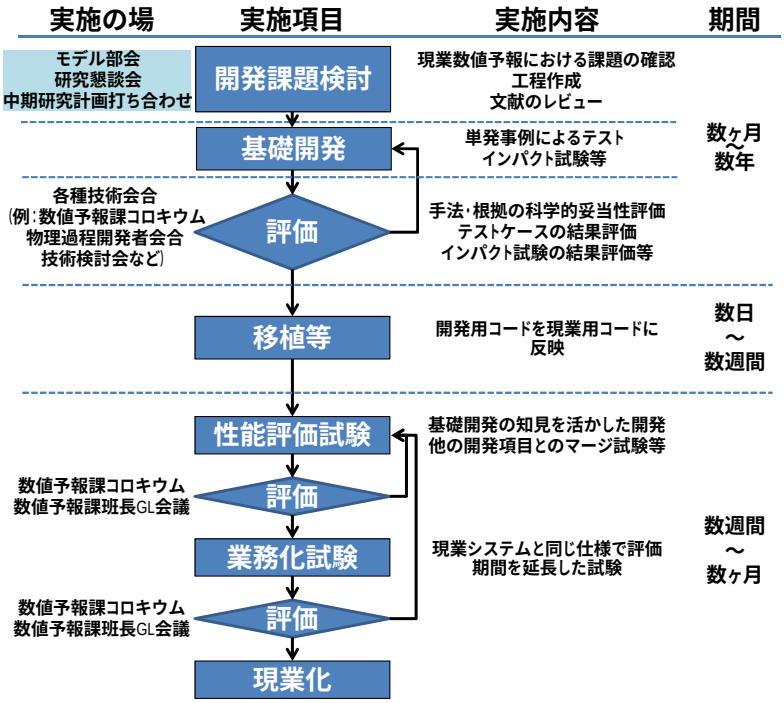

図 1.2.1　現業化までの開発フロー

テストでは解析解や特別観測等との比較が行われる他、国内外の多くの数値予報システムの開発者が参加するモデル相互比較プロジェクトにおいてモデル間の比較を行うこともある。データ同化システムの開発でも簡略化したモデルを用いた新しいアルゴリズムの効果の確認や 1 点観測データ同化実験といった理想実験が行われることがある。また、観測データの品質管理においては、異常データや同化手法の定式化より要請される仮定（観測データ同士に誤差相関がない、観測データにモデルに対するバイアスがない等）を満たさないデータの除去が必要である (佐藤 2011)。品質管理は多くのプロセスから成り立っており、実データでの実験を実施する前に不適切なデータが除去されているか、あるいは利用すべき適切なデータが誤って除去されていないか入念な確認が必要となる。

実データでの実験でルーチンと同等の環境でない実験の例として、数値予報モデルの開発におけるデータ同化サイクルを実施しない実験、全球モデル (GSM) 開発における低分解能モデルによる実験（事例数を増やす、もしくは予報期間を延長する）等が挙げられる。さらに、計算機資源が許す範囲で、開発フローの次のフェイズをにらんで性能評価試験と同等の期間の実験を行うこともある。

実データの実験による結果から、重要な一部の指標（例えば、全球数値予報システムにおける台風進路予測やメソ・局地数値予報システムにおける降水予測）もしくはそれらを含む複数の指標での精度の良し悪しのみをもって何らかのパラメータを決定するいわゆるチューニングはなるべく行わないことが望ましい。これは、第 1.2.3 項 で述べた問題点があるためである。とはいえ、観測により一意に定まらないパラメータもあり、実モデルでの評価に先立った基礎開発の段階でも決められないことがある。その場合は、比較実験により定める他ないことから、それに備えて観測の幅の範囲でパラメータの変更による影響の度合いを調査して知見を積み上げておくことも行われる。

このフェイズでは、評価とその結果に基づいた基礎開発のサイクルを入念に行うことが必要である。この評価のため、数値予報課コロキウム [6]、物理過程開発者会合 [7]、グループミーティング [8] が開催され、議論が行われる。その他に、開発コミュニティにより様々なミーティングが開催されている。また、ミーティング以外にも開発管理サーバ (第 2.4 節) の Redmine (第 2.2 節) のチケット上で議論を行うこともあり、ミーティング

[6] 原則、週に 1 回開催し、各自の開発課題の途中経過報告や性能評価試験・業務化試験の報告等を行う。
[7] 技術開発推進本部モデル技術開発部会物理過程グループが年に数回程度開催し、物理過程の詳細な議論を行う。
[8] 数値予報班の各グループがそれぞれ月に 1 回主催し、進捗報告等を行う。

を実施した場合でも、その議論結果をチケットに記載しておくことも有益である。

(3) 移植等

このフェイズでは、別のシステムで開発が行われた開発課題をルーチンと同等の仕様に移植すること等を想定している。基礎開発の段階で移植まで済んでいれば特に実施する必要はない。

従前は移植に要する開発コストは大きかった。しかし、近年は数値予報システムで用いられる言語が FORTRAN77 から Fortran90（もしくはそれ以降）となり、Fortran90 以降で利用可能となる module 文を利用したモジュール化・パッケージ化が進んできたこと、数値予報システムの運用で使われるスーパーコンピュータシステム以外のシステムやコンパイラの利用がしやすくなったこと等により、数値予報システムとは別のシステムからの移植の困難性は低下している。また、バージョン管理システムのマージ機能も移植に係るコスト低減につながっている。数値予報システムは複数のプログラムから成り立つシステムであるが、NAPEX (第 3 章) の整備により開発したプログラムをシステムに移植するコストも低減している。

(4) 性能評価試験及びその評価

ガイドラインでは「基本的な開発で成果が得られた場合、ルーチンと同等もしくは準ずる仕様で性能評価試験を行う。」としている。性能評価試験はシステムにより異なっており、全球数値予報システムでは、「現業システムと同等の仕様（ただし、全球速報解析は行わず、初期値として全球サイクル解析値を用いて 1 日 1 回（12UTC 初期時刻のみ）の予報を行う）による、原則として夏期と冬期の各 1 か月程度のサイクル実験」とし、メソ数値予報システムでは、「現業システムと同等の仕様による、原則として夏期と冬期の各 1 週間程度のサイクル実験」としている。局地数値予報システムは毎時実行されることから、1 日あたりの実行回数を減らして様々な事例に対する影響を評価することとしている。また、これらにおいて、実験の比較対象（コントロール）は原則として最新の現業システムによる実験結果とするとしている。

ここでは、NAPEX を利用した実験が行われる。複数の開発課題がある場合、独立した試験を行うこともあるが、前述の compensating errors の存在や次のフェイズを念頭におき、複数の開発課題を統合して試験を行うと、より手戻りが少なくなることが期待される。このフェイズでは、第 1.2.3 項 (1) で述べた通り、信頼できる統計的根拠を与えるだけの事例数が十分でないことから、統計検証の結果を過度に重視せず、基礎開発での知見との整合性を重視することが望ましい。

評価にあたっては、数値予報課コロキウムにおける主として技術的な観点での議論と数値予報課班長会議[9]における主としてマネジメントの観点での議論・意思決定が行われる。マネジメントの観点とは他の開発課題を含めた全体の開発工程の確認・調整等を指す。次のフェイズでより多くの計算機資源及び時間を費やすことからその評価は慎重にする必要があり、数値予報課全体として複数の議論を経て次のフェイズに進むこととしている。

[9] 数値予報課長を筆頭に、課内の班長や各グループのリーダー等が参加する会議。原則週に 1 回開催される。

(5) 業務化試験及びその評価

業務化試験は性能評価試験で期待される改善を提示した上で、実験期間を延長する（全球数値予報システムにおいては夏期・冬期の各 3 か月、メソ数値予報システムにおいては夏期・冬期の各 1 か月）ことを主とする。局地数値予報システムにおいては、1 日 24 回のテストを行い、現業運用に対して問題がないか確認する役割を持たせている。

ここでは、後続のシステム（ガイダンスや全球数値予報システムにおけるメソ数値予報システム）の評価が必要となる場合に備えて、必要なデータ（境界値等）の出力を行う。

このフェイズでは実際に現業化したことを想定した予測精度の評価が重要となる。近年、数値予報システムの用途が広がり、多くの要素・指標等が利用されている。開発において、全ての要素・指標が精度改善することは困難であるが、様々な観点で評価を行う。

評価にあたっては、性能評価試験同様に数値予報課コロキウムと数値予報課班長 GL 会議において議論・意思決定が行われる。

(6) 現業化

現業化の意思決定が行われた後は実際に現業化するためのフェイズに入る。ここでは、変更による予測特性の影響を庁内のユーザへ説明する。総合的な改善が現業化に必要であるが、前述の通り、全ての要素・指標において精度を改善することは困難である。また、精度が改善した場合でも特性が異なる場合は、ユーザの利用方法が変わりうることを考慮する必要があり、幅広く解説する必要がある。また、庁外のユーザへ技術情報の提供も行う。

その他に、現業化に向けた作業として、精度に影響のない高速化、必要な出力データの精査、時にはジョブ構成の見直し等も行われる。

ここで述べた開発フローは順調に開発が進んで、実際に現業化される場合を表している。しかし、開発課題によっては評価フェイズをパスできないこともあり、この場合は仕切り直すこともありうる（ただし、フロー図には明記していない）。

1.2.5 諸外国の動向

開発プロセスの検討においては、諸外国の現業気象機関における動向も参考にしている。英国気象局における開発プロセスは第 1.3 節や原・高谷 (2013) を、欧州中期予報センター及び英国気象局における物理過程

開発におけるプロセスは高谷・原 (2012) を、参照いただきたい。

まず、米国環境予測センター (NCEP) の動向について簡単にまとめる。これは、数値実験作業部会 (WGNE)[10] の第 31 回会合（平成 28 年 4 月開催）で報告 [11] された。NCEP では、数値予報システムの精度改善をなるべく速やかに実施して欲しいという要望と、改良後にポストプロセス等の開発が必要なことから、なるべくゆっくり改善して欲しいという相反するユーザの要望がある。そこで、NCEP では開発プロセスを以下のように定めている。まず、開発サイクルの当初にワークショップを開催し、優先すべき開発課題を費用対効果の面から検討し、スケジュールを策定する。このワークショップでは開発者だけでなく、カスタマー（ユーザ）も参加する。続いて、開発（2–4 か月）と評価（2 週間）を 1 つのサイクルとして繰り返す。なお、評価にはユーザも参加する。最終的にこのサイクルを繰り返して得られた変更が承認されると導入される。

次に WGNE 会合の動向も報告する。これまでの会合では各数値予報センターにおける改良の網羅的な概略報告が行われていた。これは、ある数値予報センターで導入された手法が改善につながるという情報だけでもその他の数値予報センターにおいて、同じ手法を導入することによる改善が期待されることから、このような情報交換は有益であったと考えられる。しかし、近年では、第 1.2.2 項で述べた通り、compensating errors の存在や実装の細部の検討の重要性が増したことにより、他の数値予報センターで改善が得られた手法をそのまま導入するだけでは精度向上につながらないことが多い。そこで、第 31 回会合から網羅的な概略を報告するよりはむしろ、何らかの開発項目に焦点をあてて開発で得られた知見を共有するように変更された。これは、数値予報課における問題意識が WGNE と共通していることを意味していると考えている。

1.2.6 まとめ

本節で述べた事柄はここ数年間で数値予報課の多くの開発者が検討・議論して得られた知見をとりまとめたものである。

以下では、今後検討すべきと考える項目について述べる。まず、多くの課題に対する開発が個別に行われているが、早い段階で統合して評価することを促進する必要があるだろう。個々の開発課題の単独の評価では精度改善の効果が見えにくいという問題への対応、compensating errors への対応が図れることが期待される。また、必要となる計算機資源を減らすことも期待される。現在は、予報モデルの開発では基礎開発のフェイズで統合した評価が行われている。一方で、観測データの利用においては業務化試験のフェイズで統合することが多い。計算機資源を減らすことや第 1.2.3 項 (1) で述べた S/N 比が低くなったことを踏まえると、性能評価試験のフェイズから統合することも視野に入れたい。さらに、数値予報モデル・データ同化システム・観測データの利用の開発を統合した評価も今後の検討課題である。なお、改良をまとめて現業化することにより、ガイダンスの開発コストや特性の変化に対応するユーザ負担が減少するといった効果も見込める。また、ユーザーニーズをどのように汲み取り、開発プロセスにどのように反映させていくかの検討も必要である。

数値予報システムの開発プロセスは今後も継続して検討していく必要があると考えている。引き続き議論を行い、より良いものにしていきたい。

[10] 数値予報モデルと大気気候モデルによる数値実験に関わる研究開発を推進するため、世界気象機関大気科学委員会と世界気候研究計画合同科学委員会の合同部会として設置されている専門部会。世界の主要数値予報センターの開発者が委員となっている。

[11] http://www.wmo.int/pages/prog/arep/wwrp/new/documents/5_wgne_ncep_M_Ek_April2016v2.pdf

参考文献

Geer, A., 2016: Significance of changes in medium-range forecast scores. *Tellus A*, **68**, 30229.

原旅人, 2012a: 数値予報モデルにおける物理過程の役割. 数値予報課報告･別冊第 58 号, 気象庁予報部, 2–7.

原旅人, 2012b: 理想実験による物理過程の評価. 数値予報課報告・別冊第 58 号, 気象庁予報部, 130–137.

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.

原旅人, 2015: 境界層過程の改良. 平成 27 年度数値予報研修テキスト, 気象庁予報部, 24–27.

堀田大介, 原旅人, 2012: 物理過程開発のボトムアップ・アプローチとトップダウン・アプローチ. 数値予報課報告・別冊第 58 号, 気象庁予報部, 120–122.

石田純一, 河野耕平, 松林健吾, 2014: 理想実験を通じたドライモデルとしての評価. 数値予報課報告・別冊第 60 号, 気象庁予報部, 62–87.

室井ちあし, 2012: 数値予報の流れ. 平成 24 年度数値予報研修テキスト, 気象庁予報部, 2–3.

室井ちあし, 2013: 開発管理の必要性. 数値予報課報告・別冊第 59 号, 気象庁予報部, 192–194.

佐藤芳昭, 2011: 数値予報データ同化システム. 数値予報課報告・別冊第 57 号, 気象庁予報部, 1–6.

高谷祐平, 原旅人, 2012: ECMWF および UKMO における物理過程開発. 数値予報課報告・別冊第 58 号, 気象庁予報部, 123–128.

Vosper, S. B., A. R. Brown, and S. Webster, 2016: Orographic drag on islands in the NWP mountain grey zone. *Quart. J. Roy. Meteor. Soc.*, **142**, 3128–3137.

1.3 英国気象局における全球モデルの開発プロセス [1]

第 1.2.4 項では、数値予報課における数値予報モデル開発プロセス、およびそのプロセスにおける開発管理の活用について述べてきた。本節では、海外の数値予報センターにおけるモデル開発プロセスの例として、英国気象局における全球モデルの開発プロセスについて紹介する。本節に書かれている内容は英国気象局科学諮問委員会 [2] 等の公開資料や、筆者が 2014 年 6 月から 2 年間、英国気象局に滞在した際に実際に見聞きしたこと・経験したことに基づいている。

英国気象局では、短期予報から気候予測まで複数の時間スケールをカバーする高精度の全球モデルを開発する目標を掲げている (Brown et al. 2012) 。その目標を達成するために、モデルの開発プロセスや必要な試験を明確にしている点、プロジェクト管理システム（第 2.2 節）を有効活用している点、実験システム・標準評価ツール等の開発基盤整備にも力を入れている点など、モデル開発コミュニティが開発管理を強化する上で参考にすべき点が多い。実際に、プロジェクト管理システムについては英国気象局の取り組みを参考に、気象庁でも利用が進んでいる （第 2.2.2 項)。

1.3.1 英国気象局の全球モデルの標準設定：GA (Global Atmosphere)

英国気象局では、全球・領域統一モデルである、UM (Unified Model) を開発・維持している。UM は共通の入出力システムのもと、全球モデル、領域モデル、移流拡散モデル等多種多様な用途に使われる基盤モデルとしての位置づけになっており、様々なオプション設定が含まれる。その中でも、UM の全球大気モデルとしての標準設定は GA (Global Atmosphere) と呼ばれる (Walters et al. 2011)。具体的には、使用する力学過程や物理過程の選択、パラメータの設定、入力データとしての地形や気候値などが GA に含まれる。英国気象局内での全球モデルを使った研究開発は、全て GA をベースに行われる。力学過程、物理過程等の改良のパッケージの導入によって、全球大気モデルとしての予測性能が上がると GA のバージョンが上がっていく。GA に関連するコンポーネントの設定として、陸面モデル GL (Global Land) 、 海洋モデル GO (Global Ocean) 、海氷モデル GSI (Global Sea Ice) もあり、それらを結合したものが GC (Global Coupled) と呼ばれる。2017 年 1 月時点での最新バージョンは GA7, GC3 である。領域モデルについても全球モデルに倣い、RA (Regional Atmosphere) と呼ばれる標準設定を開発・管理するよう、準備が進められている。また、各バージョンのトランクに相当する標準設定であり、次期バージョンの開発の雛形になるような設定であることを強調する際は、例えば “GA7.0” のようにバージョン番号の後に “.0” をつける。

GA の実行プログラムは UM 本体そのものである。そのため、GA の開発は UM 本体と並行かつ連動して行われる。開発者は何らかのモデル変更を GA に導入したい場合、まず UM にその変更を導入する。変更と前後して全球モデルとしての試験を行い、後述（第 1.3.4 項）のプロセスを経て GA の設定として正式に採用される。その変更が反映されるバージョンのリリースは UM 本体と GA でタイミングが異なる。UM, GA のバージョンが別に存在することは一見奇異に感じるかもしれない。これは、UM のバージョン番号は全球モデルとしての性能を識別する番号ではないためである。前述したように UM は多様な用途で使用されており、多数のオプションが存在する。そのため、UM の各用途での設定は UM 本体の管理とは別に管理する必要がある。また、英国気象局全体として全球モデルの開発・改良を進めるためにも、GA のような標準設定を厳密に管理することが必要となる。全球モデルにおいては、UM, GA のバージョンアップはそれぞれ、ソフトウェア、モデルの予測特性の更新を示している側面が強い。そのため、例えば「UM10.2 での GA6.0 設定」、「UM10.3 での GA6.0 設定」といった呼称が存在する。しかし、この例の場合、GA のバージョンが同じであるので、全球モデルとしての予測特性は変わらない。深刻なバグ修正等がなければ、異なる UM バージョンでの同じ GA 設定の結果は厳密一致または無視できる範囲の差となる。

2017 年 1 月における UM 本体の開発プロセスは原 (2013) による解説から大きく変化していないため、本節の以降の項では GA の開発プロセスに絞って解説する。

1.3.2 GA と現業数値予報モデル

GA は現業数値予報モデルの設定としても使用される。図 1.3.1 は GA の更新と現業数値予報モデルの更新の関係を表した模式図である。GA を現業数値予報モデルとして採用する場合、パラメータやオプションについて小規模な変更を施すことがある。この場合、現業数値予報モデルとしての設定は GA のバージョン番号を x として、“GA x.1” のように、GA x.0 から分岐させる形で区別して呼ばれる。現業数値予報モデルが GA x.1 からさらに分岐またはアップデートされることはなく（不具合修正や UM のソフトウェアとしてのバージョンアップ等を除く)、現業数値予報モデルを更新する際はより新しいバージョンである GA y.0 （y は x より新しいバージョン番号）から分岐された GA y.1 が採用される。GA y.0 を現業モデル向けに微調整を行う必要がない場合、GA y.1 は GA y.0 と全く同じものになる。GA が現業数値予報モデルとして採用されたのは、2011 年 3 月に導入された GA3.1 (Walters

[1] 氏家 将志

[2] Met Office Scientific Advisory Committee (MOSAC) http://www.metoffice.gov.uk/learning/library/publications/science/met-office-scientists/mosac

et al. 2011) が最初である。さらに、2014 年 7 月に力学コアの更新と物理過程の大規模な改良を含む、GA6.1 が現業化された。原則的に GA のバージョンが進むごとに基本的な性能は向上していくが、予報スコアやプロダクトへの悪影響の問題等で現業数値予報モデルとしては採用を見送られることもある。実際、GA4.0 と GA5.0 は気候平均場の再現性は良いものの、予報スコアの面で GA3.1 を上回ることができず、現業数値予報モデルとしては採用されていない。しかし、その間も GA の開発は最新バージョンをベースとして行われていた。

GA の更新の意思決定と、その現業導入の可否の意思決定は別途行われている点に注意したい。これは GA の利用形態のひとつは現業数値予報システムであるが、そのほかにも気候予測や研究などの用途にも利用されるためである。GA の更新の意思決定は本節で述べるプロセスを経てなされる。一方、現業数値予報システムへの導入の最終判断は、モデル・解析を問わず、科学部門、顧客サービス部門、IT 部門の長によって構成される WGOS (Working Group on Operational Suites) が行う。WGOS で承認された変更内容は PS (Parallel Suite) と呼ばれる並行運用を経て、現業数値予報システムに導入される。

1.3.3 GA 開発におけるプロジェクト管理システムの利用

英国気象局における技術開発、例えばモデル開発やデータ同化システム開発、数値予報ルーチンシステム開発等においては、プロジェクト管理システムのひとつである、Trac（第 2.2.1 項）を用いて開発の情報の記録・共有や進捗管理がなされている。GA および関連するコンポーネントの開発においても、GMED (Global Model Evaluation and Development) と呼ばれる専用の Trac が用いられており、各開発項目の記録と管理、開発の進捗管理、ミーティングの記録の共有などに活用されている。UM のソフトウェアとしての管理は UM system team と呼ばれる専門のチームが UM 用の Trac 上で行うが、GA の設定の管理や実験環境、評価・検証ツールの整備は全球モデル開発評価セクションが行う。GA の開発において、管理者（全球モデル開発グループのマネージャー）は次のバージョンに導入される開発項目、各項目のステータスを漏れ無く把握するために Trac を活用している面が強い。そのため、Trac のチケットの属性（第 2.2.3 項）のうち「マイルストーン」（チケット終了の目標）、「ステータス」（チケットの段階）を適切に記載することが重視されていたり、意思決定の記録として Trac の Wiki 上に各種会議の議事録が記載されている。

GA の開発に関わる人数は、バージョンにもよるが、力学過程、物理過程、実験システム、評価・検証システムの開発者、モデルの評価者を含めて 40–50 人程度である [3]。気象庁の全球モデル開発（第 2.5.2 項）より規模は大きく、セクションをまたいだ開発も行われている。そのため、プロジェクト管理システムを活用した開発管理やモデル変更のより厳密な手続が必要となる。

1.3.4 GA シリーズの開発サイクル

GA シリーズの開発は概ね 1 年、長くても 2 年を 1 サイクルとして行われ、そのプロセスや必要な実験などの手続は明確に示されている。開発の 1 サイクルは、図 1.3.2 のような流れになっている。まず、サイクルの最初の「“Prioritisation Meeting”」で、次のバージョンの GA への導入項目やその優先順位付けがなされる。その後、「個々の開発」の中で各項目の性能が評価される。仕様が固まった項目については、「“Science Assurance Meeting”」 で各開発項目の導入の承認を諮る。承認された個々の項目については、取りまとめられ、「パッケージテスト」としてさらに総合的な性能評価が行われる。最終的な変更内容が固まれば「“Code Freeze”」とし、最終テストを経て、新バージョンとして「リリース」される。リリース後はユーザも交えた「“Global Model Evaluation Workshop”」で新バージョンの性能や課題が議論される。議論された内容は次々期の Prioritisation Meeting にも反映される。また、各フェイズで標準実験とその評価が行われており、次々期の課題設定にもフィードバックされている。開発サイクルに関連して、開発内容の UM のトランクへのマージや現業数値予報システムへの導入等も行われる。以下では個々のフェイズとその中で行われる標準実験について、詳細に説明する。

(1) Prioritisation Meeting

Prioritisation Meeting は開発サイクルの最初に行い、既存のモデルの問題点やこれまでの進捗、モデル開発の将来性の観点を勘案して、どの項目が優先事項かが議論される。この会合には力学過程・物理過程・全球モデル開発・部外連携セクションの各ヘッドやマネージャー [4] が出席し、議論および意思決定を行う。次期バージョンに導入される項目は “Implementation cycle” というカテゴリに所属し、このミーティングで各開発項目に対して、“critical”, “high”, “normal” などの優先順位付けがなされ、Trac に記録される。次期バージョンに導入できるほどの進展が得られていない開発項目については “Research cycle” というカテゴリに属し、基礎開発が継続され、次々期バージョンでの Prioritisation Meeting で再度議論される。

[3] GA6.0/6.1 の解説論文である、Walters et al. (2016) の著者として記載されている人数。さらに GA6 の開発において、UM のシステム管理者、大気・海洋結合モデルの開発者、現業数値予報システムへの導入に関わったデータ同化システムの開発者等を含めると 100 人を超える。

[4] 英国気象局の科学部門は各セクションごとに「ヘッド」と呼ばれる長がおり、セクション内のグループごとに、グループを取りまとめる「マネージャー」を置いている。

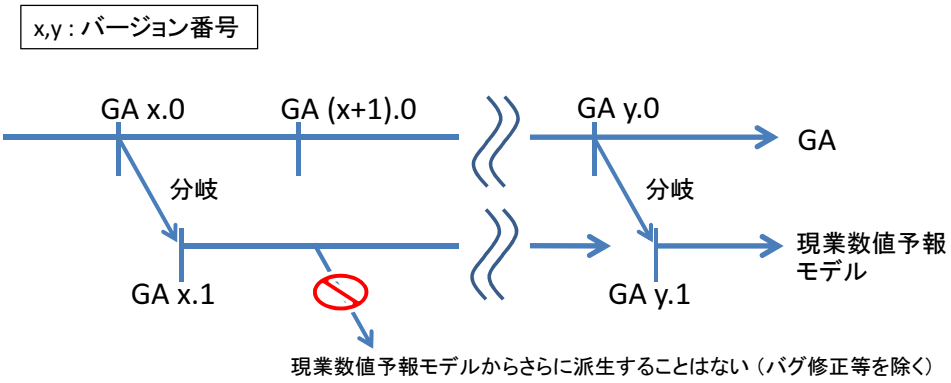

図 1.3.1　GA と現業数値予報システムで使われる全球モデルの関係を表した模式図。図中の水平の矢印はバージョンが上がる方向を、斜めの矢印は現業数値予報モデル用のブランチとして分岐することを表している。x, y は GA のバージョン番号 (y>x) を示している。

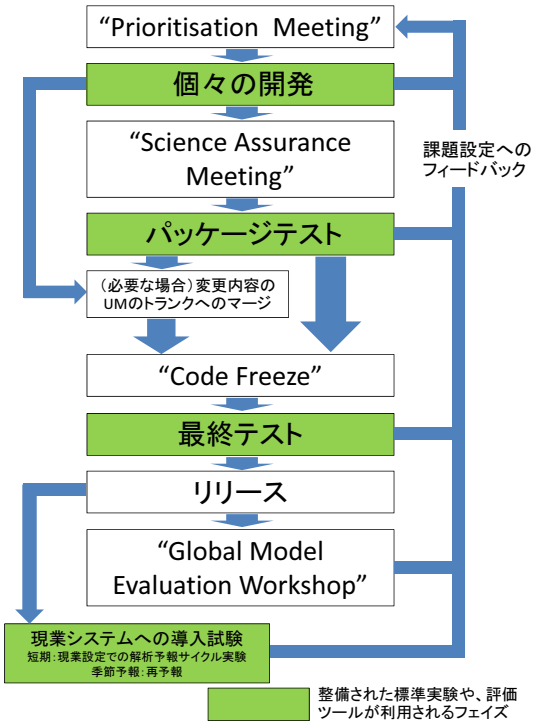

図 1.3.2　GA の開発 1 サイクルの流れ。図中の緑で塗られたボックスは、標準実験や評価ツールを利用するフェイズを示している。

(2)　個々の開発

個々の開発項目については、UM 本体の開発（原 2013）と同様、その内容が Trac 上のチケットに必ず登録されていなければならない。また、UM 本体の開発と GA の開発は並行かつ連動して行われているため、GA のチケットから、UM の関連チケットにリンクが張られていることも多い。チケットには開発の目的を記述し、進捗や使用した UM のブランチの所在、実験結果等を記録する。また、変更内容が UM のトランクに既に反映されているか否かも記録する。変更内容が固まり、パッケージテストに導入する準備が整った場合は、開発者は各チケットにリンクされた、“TicketDetails” と “TicketSummary” と呼ばれる Wiki ページを作成する。TicketDetails では、変更の概要や、物理的基礎（参照論文がある場合はそのリンク）、解像度依存性の有無を記述する。TicketSummary には、標準実験の結果を掲載し、変更のインパクトを改善点、改悪点含めて記述する。また、計算安定性・計算時間などに影響があればその旨も記述する。TicketDetails, TicketSummary を作成したら、レビューの段階に進み、開発者または管理者から指名されたレビューアによって変更内容が評価される。通常、レビューアは変更内容の分野に詳しい者が指名される。レビューアは変更内容が適切であるか、適切な実験がなされているかを審査する。レビューアからコメントや指摘があった場合、開発者はそれに対応する必要がある。レビューにおいては、精度向上とともに、物理的に適切な変更であるかも評価の対象となる。単体の変更で精度向上が見られなくても、物理的に正しい変更であり、さらに他の変更との組み合わせで精度向上が見込まれる場合は、レビューを通過することもある。

(3)　標準実験とその評価

標準実験は、短期予報実験と気候予測実験の 2 種類があり、開発者は両方を行わなければならない。短期予報実験では、水平格子間隔約 55 km のモデルを用いた、“Case Study” と呼ばれる ECMWF の解析値を初期値とした、夏冬合わせて 24 事例が標準事例として整備されている。気候予測実験については水平格子間隔約 130 km のモデルを用いた、AMIP ラン（Atmospheric Model Intercomparison Project：大気モデルの相互比較プロジェクトで指定された実験で、大気モデルに海面水温解析値を与える）での 20 年積分が標準実験として行われる。短期予報実験と気候予測実験では、エーロゾルの扱い（短期予報実験では気候値、気候予測実験ではエーロゾルスキームで計算）や積雲対流スキー

ムのパラメータの１つ、質量・エネルギー補正スキーム適用の有無などの違いはあるが、その他は原則的に共通の設定を用いる。

評価・検証についても標準パッケージが準備されており、短期予報実験については TRUI (Trial User Interface) と呼ばれる平均誤差, 平方根平均二乗誤差等の標準的な予報スコアや平均予報場を計算する検証パッケージが、気候予測実験については Validation Note（気候平均場の検証や、放射収支の確認)、Auto Assess（モンスーン、 マッデン・ジュリアン振動、赤道波、ブロッキングなどの現象に応じた評価パッケージ）と呼ばれるパッケージが整備されている。これらの評価・検証ツールは標準実験を行えば誰でも簡単に実行できるよう整備されており、図 1.3.2 の緑で塗られたボックスのフェイズで使用される。また、開発者は必要に応じて追加の実験や、追加の解析結果も TicketSummary に記載することができる。解析予報サイクル実験の実施は個々の開発の過程では任意だが、大規模なモデル変更や陸面過程の変更など、解析場への影響が大きいことが予想される場合は、この段階で解析予報サイクル実験が行われることもある。

(4) Science Assurance Meeting

各チケットのレビューと前後して、各チケットで検討された変更をパッケージテストに導入してよいかの判断を Science Assurance Meeting と呼ばれる会合で行う。この会合には力学過程・物理過程・全球モデル評価開発・部外連携セクションの各ヘッドやマネージャー、データ同化・アンサンブル予報のセクションのヘッド、領域モデル開発グループのマネージャー等が出席し、意思決定を行う公式な会合である。パッケージテストへの導入項目はチケット単位で議論される。通常、チケットの担当者が会合の出席者に対してプレゼンテーションを行い、導入可否を諮る。モデルの予測精度に影響のある変更は原則的にすべて Science Assurance Meeting にかけられる。Science Assurance Meeting で議論されたすべての項目が承認されるとは限らず、次期バージョンに導入するには事前調査や準備が不十分であると判断され、“Research cycle” のカテゴリに差し戻される項目も実際に存在する。Science Assurance Meeting はパッケージの最終仕様が固まるまで随時開催される。この会合の議事録、決定事項、次回の会合までの課題等は Trac 上の Wiki に記録され、英国気象局の職員及び共同研究契約を結んでいる外部の協力者に公開される。

(5) パッケージテスト

レビューを通過し、Science Assurance Meeting で承認された項目はパッケージテストに組み込まれる。この段階ではレビューを通過した項目を統合して、性能評価試験が行われる。パッケージテスト自身も Trac のチケットの一つとして扱われ、実験結果を記録していく。性能評価のための標準試験は個々の開発の中で行われるものと同じであるが、パッケージテストでは解析予報サイクル実験も行われる。パッケージテストの最初のバージョンに取り込まれる内容は優先度や進捗を考慮し、議論して決定される。その後、準備ができた項目やパッケージテストの中で発覚した問題の対処版を順次導入し、テストを行う。GA7 の開発では、最終的に GA6 に対して約 40（チケット数）の変更がパッケージに取り込まれた。パッケージテストの段階で、新バージョンが前バージョンに対して予測精度が向上しているかどうかが強く問われるようになる。もちろん、ただ精度が向上していればそれで良し、ということではなく、どの変更がどの要素の精度向上に寄与しているかの切り分けも行う。パッケージテストの実行は全球モデル評価開発セクション内の全球モデル開発グループが取りまとめて行うが、テスト結果については個々の変更の開発者も交えて頻繁に議論が行われる。必要に応じて、パラメータチューニングもこの段階で行われる。

(6) Code Freeze

パッケージが固まったら、Code Freeze とし、問題が発覚した場合を除き、原則的にそれ以後の変更は次々期のバージョンの開発に反映させる。また、新バージョンに導入される変更内容は、この段階で UM のトランクに反映されていなければならない。

(7) 最終テスト

Code Freeze 後は全球モデル評価開発セクションが 1 年程度の長期解析予報サイクル実験、さまざまな水平分解能での大気・海洋結合モデルでの長期積分、季節予報の再予報等を実行し、総合的な評価を行う。また、現業数値予報システムに導入するための試験（現業数値予報システムと同じ水平分解能での解析予報サイクル実験）もこの段階で開始される。ただし、現業数値予報システムに導入するための判断は GA の開発サイクル本体とは分岐して行われる。

(8) リリース

Code Freeze 後、x をバージョン番号とすると、GA x.0 の名称で新しいバージョンがリリースされ、英国気象局内および外部のユーザが利用可能となる。新バージョンを現業システムに反映させる場合、現業用に細かい修正を加えたものはブランチとして GA x.1 の名称でリリースする。また、新しいバージョンがリリースされると、英国気象局内外のユーザがそのバージョンのモデルを使った研究に関する論文で引用できるよう、GA x.0/x.1 の概要や性能を解説した査読付き論文を投稿・出版する。2017 年 1 月現在、大気モデルについては GA3.0/3.1 (Walters et al. 2011), GA4.0 (Walters et al. 2014), GA6.0/6.1 (Walters et al. 2016) が、大気・海洋結合モデルについては GC2.0 (Williams et al.

2015) の解説論文が出版または投稿されている [5]。

(9) Global Model Evaluation Workshop

リリースと前後して、外部ユーザによる評価結果も含め、Global Model Evaluation Workshop が開催される。この会合では英国気象局の職員に加え、外部の研究協力者も出席し、新しい GA の性能や改善点・問題点が議論および共有される。また、優先して改善すべき点もリストアップされ、次々期の Prioritisation Meeting に向けた基礎資料となる。

1.3.5 GA シリーズの開発プロセスに関する分析

前項までは、英国気象局における、GA の開発プロセスを概観してきた。この項では、まとめにかえて、英国気象局の取り組みに関して、モデル開発コミュニティが参考にすべき点、気象庁における全球モデル開発との共通点など筆者なりの見解を述べたい。

まず、開発サイクルの初期段階で、重要開発項目の関係者間での認識合わせ、開発項目の具体的な優先順位付けをしている点は特筆すべきである。モデルの既存の問題点、その問題点に対するアプローチの進捗、さらにユーザからの要望を反映した上で、次の全球モデルの更新で何を改善するかを明確にしている。このことは、現業気象機関においてより良い気象・気候情報サービスを提供し続けるために非常に重要であるといえるだろう。気象庁においては、台風進路予測の精度向上や、予報現業を通じて見られる問題点（例えば、原 (2015) で指摘された南岸低気圧事例におけるトラフの遅れなど）に対するアプローチ、その他全球モデルとしての精度向上（対流圏・成層圏の気温や風のバイアス、降水分布、熱帯循環の維持）、前バージョンで残された（または副作用として顕在化した）問題などを勘案して、開発サイクルの初期段階で全球モデルの次期バージョンの目標を明確にすることが、開発サイクルを活性化する上で必要であろう。

次に、短期予報・気候予測両方での実験を義務付けていること、さらにそれら実験の検証・評価を標準化している点も注目すべきである。この点には、短期予報から気候予測まで複数の時間スケールをカバーする全球モデルを開発するという、英国気象局の方針が強く反映されている。また、(i) 複数の時間スケールの数値実験から、多角的にモデルの評価を行っている、(ii) 目標を達成するための開発環境の基盤整備がなされている、という意味で、モデル開発コミュニティが参考にすべき点でもある。

(i) について、筆者が英国気象局に滞在した際に、積雲対流過程における固体降水の融解過程の改良を行った。この際、短期予報実験と気候予測実験を両方行うことで見える改善点（例えば、対流圏中層における気温のバイアス改善が時間スケールを問わず見られること）や問題点（例えば、北西太平洋の循環場のバイアス拡大が気候予測実験で顕著になること）などが見つかった。複数の時間スケールでの数値実験は、モデルの改良に問題がないかの確認や、compensating errors がないかを探すきっかけ [6] となり、改良を進める上で役立つことを実感した。短期予報モデルの開発においても、長期積分や気候予測実験による評価からバイアスの特性を探るのは有益である。

(ii) について、英国気象局においては

- 複数の時間スケールをカバーする統一的な全球モデルを開発する

という目的を達成するために

- 共通のモデル設定による、短期予報・気候予測実験システム
- 気候平均場や大気現象の再現性の評価に重点をおいた評価ツール

が整備されている。これは、英国気象局は気候研究を行うハドレーセンターや応用気候のセクションを有しており、気候研究や気候サービス業務の規模が大きいこと、英国内の天気予報作成支援は領域モデルが主な役割を担っていることから、全球モデルの評価は気候予測モデルとしての評価に重点を置いている傾向があることの反映であろう。

気象庁においては、全球モデル・全球データ同化システムを含む全球数値予報システムは、天気予報や台風進路予報、週間天気予報の作成支援が重要な役割であること、全球モデルは全球 EPS (経田 2016) 等のアンサンブル予報システムの予報モデルとしても利用されること、さまざまな水平分解能での実験や長期積分によって見えるモデルのバイアスの確認の重要性も認識されつつあることから、

- 天気予報、台風進路予報、週間天気予報等を支援する、全球数値予報システムを開発する
- 多角的な評価を行い、全球モデルの総合的な予測精度を向上させる

ことを目指すために

- 解析予報サイクル実験システム
- 気候予測実験システム
- 上記システムの評価・検証ツール

を整備・拡充することが重要になるだろう。実際に、第 2.5.6 項で述べられているように全球モデルの開発と並行して評価・検証ツールの開発も進められているところである。

また、最終的な予測精度はパッケージテストの中で評価し、個々の開発項目単体では、科学的妥当性や基礎調査が十分行われているかを重視している点も注目すべきである。このことは、第 1.2.2 項で述べられている、数値予報システムの誤差要因が数値予報の発展と

[5] GA5.0 の論文は未出版であるが、力学コアの更新が最も大きな変更であったため、GA5.0 の代替として Wood et al. (2014) が引用されることがある。

[6] 最終的に積雲対流スキームの他の部分の改良や、地形性重力波スキームの調整との組み合わせで大きな問題とならなくなることがわかった。

ともに変わってきていることと関係していると考えられる。かつては、モデルの中で考慮されていない重要な過程（例えば、1980 年代以前における、全球モデルの重力波抵抗など）が数多くあり、一つの変更で劇的に精度が向上するということもあったが、現在はそのようなことは稀で、一つの項目を改善すると別の隠れていた問題が顕在化することのほうが多い。そして、多くの場合、その問題は Minor treatment（第 1.2.3 項）と関連している。最終的な予測精度をパッケージで評価するということは、大気現象をある程度再現するのに必要なスキームが出揃い、一つのスキームの追加や変更だけで精度が向上する時代はすでに終わったこと、今後は各過程を洗練させつつ、複数の変更や過程間の相互作用を考慮したうえで精度改善を図る時代になったことの反映であると考える。第 2.5 節に詳しく述べられているように、数値予報課での全球モデル開発もここ数年は同様のことを行っており、英国気象局での取り組みと整合している。パッケージテストをモデルの各パーツの基礎調査の次に行うステップとして重視することは、今後、継続的な予測精度向上を進める上で重要であると言える。もちろん、複数の変更を単純に組み合わせて精度が向上すれば良い、ということでなく、精度向上の原因の切り分けや、どのエラーが他のエラーを隠蔽しているのかの正しい見極めが必要になる。そのためには、担当者が自らの専門性を高めるだけでなく、周辺分野の相互理解やモデル開発全般に関する広い知識と経験が必要になる。また、モデル開発と評価を融合してモデルの総合的な性能を上げていくアプローチは、数値予報課コロキウムをはじめとした各種会合や数値予報研修テキストおよび数値予報課報告・別冊（付録 A）において、数値予報に関して幅広く理解し、議論する機会が多い気象庁では、より効果を発揮するのではないかと筆者は考えている。

気象庁と英国気象局では、数値予報システムの開発規模や関連する業務を取り巻く環境に違いはあるものの、英国気象局でのモデル開発の取り組みのうち、合理的と思われるものは気象庁の事情に応じてアレンジした上で反映することで、開発サイクルの活性化や数値予報システムの継続的な精度向上に資することができると考えている。

参考文献

Brown, A., S. Milton, M. Cullen, B. Golding, J. Mitchell, and A. Shelly, 2012: Unified Modeling and Prediction of Weather and Climate: A 25-Year Journey. *Bull. Amer. Meteor. Soc.*, **93**, 1865–1877.

原旅人, 2013: 英国気象局の開発管理. 数値予報課報告・別冊第 59 号, 気象庁予報部, 195–197.

原旅人, 2015: 事例検討. 平成 27 年度数値予報研修テキスト, 気象庁予報部, 82–99.

経田正幸, 2016: 全球アンサンブル予報システムの開発. 数値予報課報告・別冊第 62 号, 気象庁予報部, 52–57.

Walters, D., M. Brooks, I. Boutle, T. Melvin, R. Stratton, S. Vosper, H. Wells, K. Williams, N. Wood, T. Allen, A. Bushell, D. Copsey, P. Earnshaw, J. Edwards, M. Gross, S. Hardiman, C. Harris, J. Heming, N. Klingaman, R. Levine, J. Manners, G. Martin, S. Milton, M. Mittermaier, C. Morcrette, T. Riddick, M. Roberts, C. Sanchez, P. Selwood, A. Stirling, C. Smith, D. Suri, W. Tennant, P. L. Vidale, J. Wilkinson, M. Willett, S. Woolnough, and P. Xavier, 2016: The Met Office Unified Model Global Atmosphere 6.0/6.1 and JULES Global Land 6.0/6.1 configurations. *Geosci. Model Dev.*, in review.

Walters, D. N., M. J. Best, A. C. Bushell, D. Copsey, J. M. Edwards, P. D. Falloon, C. M. Harris, A. P. Lock, J. C. Manners, C. J. Morcrette, M. J. Roberts, R. A. Stratton, S. Webster, J. M. Wilkinson, M. R. Willett, I. A. Boutle, P. D. Earnshaw, P. G. Hill, C. MacLachlan, G. M. Martin, W. Moufouma-Okia, M. D. Palmer, J. C. Petch, G. G. Rooney, A. A. Scaife, and K. D. Williams, 2011: The Met Office Unified Model Global Atmosphere 3.0/3.1 and JULES Global Land 3.0/3.1 configurations. *Geosci. Model Dev.*, **4**, 919–941.

Walters, D. N., K. D. Williams, I. A. Boutle, A. C. Bushell, J. M. Edwards, P. R. Field, A. P. Lock, C. J. Morcrette, R. A. Stratton, J. M. Wilkinson, M. R. Willett, N. Bellouin, A. Bodas-Salcedo, M. E. Brooks, D. Copsey, P. D. Earnshaw, S. C. Hardiman, C. M. Harris, R. C. Levine, C. MacLachlan, J. C. Manners, G. M. Martin, S. F. Milton, M. D. Palmer, M. J. Roberts, J. M. Rodríguez, W. J. Tennant, and P. L. Vidale, 2014: The Met Office Unified Model Global Atmosphere 4.0 and JULES Global Land 4.0 configurations. *Geosci. Model Dev.*, **7**, 361–386.

Williams, K. D., C. M. Harris, A. Bodas-Salcedo, J. Camp, R. E. Comer, D. Copsey, D. Fereday, T. Graham, R. Hill, T. Hinton, P. Hyder, S. Ineson, G. Masato, S. F. Milton, M. J. Roberts, D. P. Rowell, C. Sanchez, A. Shelly, B. Sinha, D. N. Walters, A. West, T. Woollings, and P. K. Xavier, 2015: The Met Office Global Coupled model 2.0 (GC2) configuration. *Geosci. Model Dev.*, **8**, 1509–1524.

Wood, N., A. Staniforth, A. White, T. Allen, M. Diamantakis, M. Gross, T. Melvin, C. Smith, S. Vosper, M. Zerroukat, and J. Thuburn, 2014: An inherently mass-conserving semi-implicit semi-Lagrangian discretization of the deep-atmosphere global non-hydrostatic equations. *Quart. J. Roy. Met. S.*, **140**, 1505–1520.

第2章　開発記録とバージョン管理

2.1　気象庁における開発管理の取り組み [1]

気象庁の数値予報モデル開発において、開発管理についての全庁的な取り組みは 2011 年度に気象庁技術開発推進本部 [2] モデル技術開発部会（モデル部会）に「開発管理グループ」が設置されたことに始まる。これは、欧州中期予報センター (ECMWF) や英国気象局 (UKMO) への派遣経験者からの報告で、これらのセンターにおける開発管理の取り組みが報告され (原・高谷 2013)、その重要性が庁内に認識されたことがきっかけであった。

開発管理グループは数値予報課、気候情報課、気象研究所などの庁内のモデル開発にたずさわる職員から構成され、庁内の各モデル開発の現状とあるべき姿や、諸外国の状況についてのレビューを行った上で、バージョン管理やプロジェクト管理システムを用いた情報共有が必要であることを提言した。

それ以前にも、開発コミュニティ（あるモデルやシステムを開発するための開発者の集まり）によってはバージョン管理システムやプロジェクト管理システムをそれぞれのコミュニティが管理するサーバにインストールして活用してきたが、そのようなサーバ機材や構築技術、維持体制がすべてのコミュニティにあるわけではないことが課題となっていた。開発管理グループによる提言では、庁内のモデル開発にかかわるすべてのプロジェクト管理システム、バージョン管理システムの運用を全庁的に統一することを求め、その提言に基づき、翌年には数値予報モデル開発管理情報共有装置（通称：開発管理サーバ）が導入され、庁内のどの開発コミュニティでも開発管理に関係するツールが利用できるようになった [3]。

2012 年度以降は、開発管理グループに代わって開発管理調整グループがモデル部会に設置され、開発管理サーバの運用ルールの具体化などが行われた。その運用ルールでは以下のようなことが定められた。

- プロジェクト管理システム、バージョン管理システムを活用する。
- プロジェクト管理システムとしては Redmine を用いる [4]。
- バージョン管理システムとしては Subversion または Git を用いることとし、どちらを使うかについては開発コミュニティの裁量とする [5]。
- バージョン管理システムにおける開発本流には業務用または共用最新版のソースコードを登録することとし、開発用のソースコードをバージョン管理システムのブランチ（本流からの枝分かれ）に登録することを推奨する。
- 開発本流の内容に変更を加える際には、必ずプロジェクト管理システムにその説明を記録する。

これらは最低限のルールであり、各開発コミュニティではこれらのルールに沿った上で、それぞれが利用しやすい開発ルールを定めることとしている。

さらに、この開発ルールをより具体的にした上で「統一環境における数値予報モデルの開発管理の指針」が 2014 年 3 月に気象庁技術開発推進本部によって制定された。現在では多くの開発コミュニティがこの指針に基づいて開発管理サーバーを利用し、その開発過程の記録をしている。

これまでに述べた動きは、気象庁技術開発推進本部モデル部会が主導して行ったモデルの開発管理に関するものであったが、開発管理の必要性はモデル開発にとどまるものではなく、開発管理のためのツールはモデル開発の基盤整備、アプリケーション開発、数値予報ルーチンシステム管理でも活用されている。

本章では、プロジェクト管理システム、バージョン管理システムの一般的な事項について簡単に説明したのち、開発管理サーバの仕様と運用、そして、数値予報課内でのプロジェクト管理システム、バージョン管理システムの活用について、各開発コミュニティから報告する。すでに述べたように、最低限のルールに従えば、これらのシステムの利用方法は各開発コミュニティーが自由に定めることができるため、その利用方法は多様である。その多様な利用方法は、これから開発管理にかかわるツールを使い始めようとする開発コミュニティにも大いに参考になると期待する。

なお、本稿では開発管理に関するツールの利用についての数値予報課内での取り組みを取り上げたが、これらのシステムは数値予報課以外にも、気候情報課、海洋気象情報室、環境気象管理官、気象研究所、気象衛星センターなどで利用されている。

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.

[1] 原 旅人、永戸 久喜

[2] 気象庁長官を本部長とし、モデル技術開発部会、豪雨監視予測技術部会、静止衛星データ利活用部会が設置されている。この 3 つの技術開発間における情報共有と相互連携により一体的な推進を図るとともに、これらの技術開発に共通する外部機関との連携・協働の促進等に戦略的な方針と計画の作成を通じた総合的な技術開発体制を構築することを目的にしている。

[3] 開発管理サーバーの管理は数値予報課で行っている。

[4] 当時は開発コミュニティによっては Trac と呼ばれる Redmine と類似したツールを使っていたが、管理コストの低減、利用方法の共有の促進の観点から、機能がより豊富で柔軟に運用しやすい Redmine に統一することとした。

[5] プロジェクト管理システム同様、一つのシステムに統一したいところであったが Subversion と Git で設計思想が大きく違う中で、すでに使用を開始しているものを一方に変更させるのは負担が大きい、ということから、どちらかを選択すればよいようにした。

2.2 プロジェクト管理システム [1]

2.2.1 プロジェクト管理システムとは

ソフトウェア開発の現場では、その開発をプロジェクトと呼ぶことが多い。一般に「プロジェクト」とはある目標を達成するための（定常業務とは異なる）期限がある特別な計画を指すが、ソフトウェアの開発はある明確な目的と期限を設けて行うことが多いので、まさにプロジェクトと呼ぶにふさわしいのであろう。

ソフトウェア開発などでは多くの人が様々なタスクを分担して行い、その集合としてプロジェクトを構成する。ソフトウェアは多くの部品で構成され、それぞれに要件定義（仕様を定めること）、詳細設計（定めた仕様に基づき詳細な挙動を決めること）、実装（ソースコードのプログラミング）、テストなどのステージがあり、それぞれのステージの専門家が担当するのが一般的である。それらの部品が一つでも欠ければそのソフトウェアは完成しないため、それぞれ部品を漏れなく工程管理することが必要となる。

Trac[2] や Redmine[3] に代表されるウェブベースのプロジェクト管理システムは、これらの多くのタスクを登録してリスト化した上で、それぞれについてその過程や関連する議論を記録するとともに、そのタスクの現在の進捗状況（ステータス）を把握しやすくするためのシステムである。プロジェクト管理システムと呼ばれるものは以前から商用のものが多くあったが、Trac や Redmine は無償で利用することができて、ユーザはウェブブラウザ以外の特別なソフトウェアを必要としないことから、急速に利用が広まっている。元来はソフトウェア開発のプロジェクトを管理するためのシステムとして開発されたが、作業過程の記録、現在の進捗状況の把握は、期限が定められたプロジェクトとしてだけではなく、継続的に開発を進めるソフトウェア（モデル開発はこれに該当）、ソフトウェア開発以外のプロジェクト、さらにはプロジェクト以外の日常業務でも必要であるため、さまざまな場面で活用されている。

2.2.2 プロジェクト管理システムの気象庁での利用

モデル開発においてもプロジェクト管理システムにおけるタスクの登録、作業過程の記録、進捗状況の把握は、現在および将来の開発者に自らが行った開発についての説明責任を果たす上で必須のことであり、プロジェクト管理システムを大いに活用することができる [4]。

数値予報課でのプロジェクト管理システムの本格的な利用の始まりは、2008 年ごろの asuca (気象庁予報部 2014) の開発への Trac の導入であった。その後、いくつかの開発コミュニティで Trac、さらには Redmine の利用も始まった。

2012 年以前はさまざまなサーバに分散して Trac や Redmine が運用されていたが、すでに述べたように、開発管理サーバを整備して、数値予報課だけに限らず、庁内の数値予報モデル開発に関するプロジェクト管理システムの運用を集約した。開発管理サーバでは Redmine を標準的なソフトウェアと定め [5]、Trac を利用していたプロジェクトは Redmine に移行した [6]。

2.2.3 プロジェクト管理システム Redmine の機能

以下では、プロジェクト管理システムの一つである Redmine について、本章の後半で紹介される活用例の理解に必要最低限の事項を説明する。詳細については、Redmine の公式ウェブページを参照されたい。

(1) チケット管理

Redmine では、タスク登録のための「チケット」を作成する。そのチケットの管理がプロジェクト管理システムの最も重要な機能の一つである。プロジェクト管理システムにチケットとしてタスクを登録し、それに基づいて進めるソフトウェア開発の手法はチケット駆動開発と呼ばれている。なお、その機能を強調して、プロジェクト管理システムのことを課題管理システムと呼ぶこともある。

それぞれのチケットは以下のような属性を持つ。

- チケット番号（チケットを作成した際に自動的に付加される。この番号がチケットを特定するキーとなる。）
- トラッカー（チケットの分類。初期設定では「バグ」「機能」「サポート」が登録されている。管理者が設定を変更することもできる。）
- チケットのタイトルとその概要
- 担当者
- ステータス（初期設定では「新規」「進行中」「解決」「フィードバック」「終了」「却下」のステータスが登録されている。管理者が設定を変更することもできる。）
- 優先度
- 期日

属性の種類は管理者によって追加・変更することができる。初期設定におけるチケットの新規作成画面の例を図 2.2.1 に示す。

---

[1] 原 旅人

[2] `https://trac.edgewall.org`

[3] `http://www.redmine.org`

[4] 「統一環境における数値予報モデルの開発管理の指針」（第 2.1 節参照）では、役割を明示するために「プロジェクト管理システム」を「情報共有ツール」と読み替えている。

[5] Trac では一つのプロジェクト管理システムに一つのプロジェクトしか登録できなかったが、Redmine では複数のプロジェクトを登録することが可能である。また、プロジェクトの間に親子関係を持たせることもできる。後発の Redmine のほうが機能が充実していた点がいくつかあったので、Redmine を選択し、統一することにした。

[6] Trac から Redmine への移行ツールは、開発管理サーバの管理を担当している数値予報課基盤整備グループによって開発された。

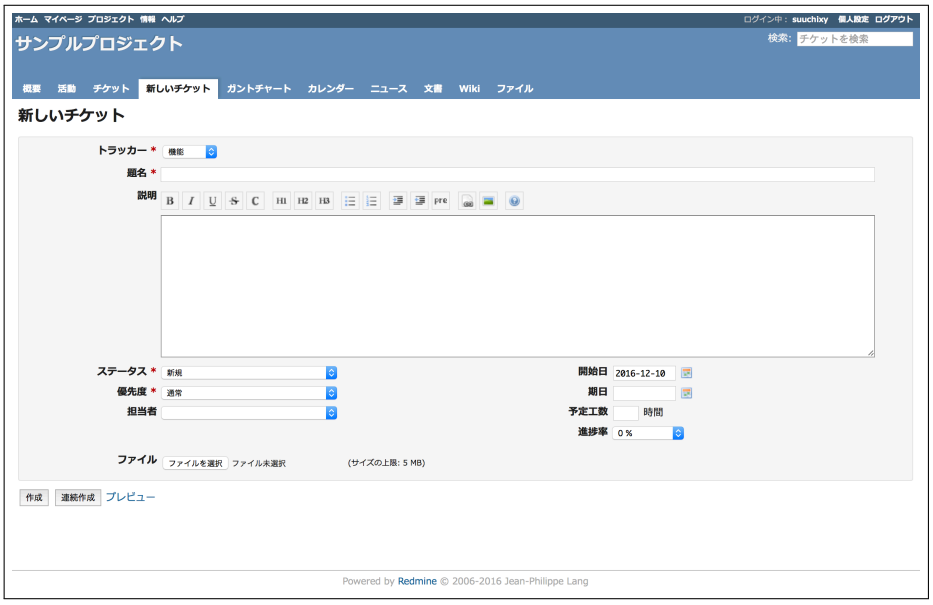

図 2.2.1　Redmine における新規チケット作成画面の例。

図 2.2.2　Redmine におけるチケット表示画面の例。プロジェクト名やユーザ名は架空のものである。

図 2.2.3　Redmine におけるチケット一覧表示画面の例。プロジェクト名やユーザ名は架空のものである。

登録された各チケットは、図 2.2.2 のように表示され、チケットのタイトル、内容、ステータス、担当者などが上部に、チケットに対するコメントの履歴がその下に表示される。

また、このチケットを一覧表示することで、各タスクの現在の進捗状況が一目瞭然となる。その例を図 2.2.3 に示す。

チケットの担当者やステータスは進捗に応じて変更する。また、それぞれのチケットには作業の記録、コメントの記入、ファイルの添付などができて、これらの記述が、現在および未来の開発者への説明責任を果たすことに対応する。

(2) ワークフロー

ソフトウェアの開発においては、ソースコードの作成、そのレビュー、バージョン管理システムへのコミットなど、開発コミュニティによって手順が定められていることが多い。その手順を Redmine 上に実装したのがワークフローである。ワークフローはチケットの属性の一つである「ステータス」の遷移を規定するもので、トラッカーごとに異なるワークフローを定義することができる。トラッカーが「機能」の場合の開発者のワークフローは図 2.2.4 のようになっている（「却下」は「機能」のトラッカーでは利用されていない）。たとえば、現在のステータスが「進行中」の場合に、遷移できるのは「解決」「フィードバック」「終了」であり、「新規」には遷移できない。このようなワークフローは、Redmine ごとにその管理者が設定できる。

これらの機能を活用することで、開発者にその開発コミュニティの開発フローに沿ったステータスの変更を行わせることができる。その活用については、この章で紹介する各開発コミュニティでの利用例を参照されたい。

(3) バージョン管理システムとの連携

ソフトウェア開発のプロジェクト管理はソースコードのバージョン管理と密接に関係している。そのため、プロジェクト管理システムはバージョン管理システムとの連携機能を持っている。例えば、そのプロジェクトに関連するとして登録したリポジトリの変更をブラウザで閲覧でき、そのチケットに関連する変更を見やすく表示することができる。

(4) 情報共有のための機能

Redmine にはいくつかの情報共有のためのツールが実装されている。例えば、各プロジェクトのトップページを含め、Wiki を使うことができる。Wiki はウェブブラウザを利用してウェブサーバ上の文書を書き換えることができるシステムである。Redmine 内部のページを含むウェブページへのハイパーリンクを記述することもでき、プロジェクトに関するお知らせや様々なリンク集を Wiki に作成して公開することができる。そ

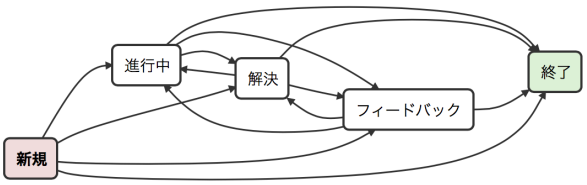

図 2.2.4 Redmine の初期設定におけるワークフロー。トラッカーが「機能」の場合。

の他に、掲示板機能としてフォーラムと呼ばれる機能もある。

また、チケットが新規に作成されたり、編集された場合に、プロジェクトの関係者にメールを送信する機能もあり、Redmine を能動的にチェックしなくても、変更情報などを受け取ることができる。

さらには、Redmine には RSS 機能が搭載されており、RSS リーダーを利用してチケットやリポジトリの更新情報をまとめて閲覧することもできる。

2.2.4 プロジェクト管理システム利用にあたっての留意点

プロジェクト管理システムを利用することのメリットを享受するためには、その開発コミュニティの開発ルールを明確化して、開発に参加している関係者全員がそのルールにしたがって、タスクをチケットとして登録して、そのチケットに開発過程を記録し、ステータスを進捗に応じて正しく変更することなどが必要である。本章で紹介する数値予報課内での活用例は、これから Redmine を使い始める開発コミュニティに参考になるであろう。

Redmine の利用にあたって、チケットに登録する際のタスクの規模、すなわち、タスクをどこまで細かく分割してチケットにすべきかについて悩むユーザが庁内には多いようである。それぞれの開発コミュニティでルールを設けることが必要であるが、タスクの分割の仕方は主観的な面もあり一律に判断するのが難しい。そのような場合は、チケットを作成することを躊躇せずに、まずはチケットを作成してその規模について試行錯誤をしてみることをお勧めしたい。その試行錯誤の中で、開発コミュニティの中での使い方が定まっていくことが多い。また、チケットの統合、分割は頻繁に行われることであり、作業を進める上で統合または分割したほうがやりやすいと判断される場合は、そのときにその統合または分割を行えばよい。ぜひ、利用しながら自分の開発コミュニティにとって便利で有効な使い方を模索していただきたい。

参考文献

気象庁予報部, 2014: 次世代非静力学モデル asuca. 数値予報課報告・別冊第 60 号, 気象庁予報部.

2.3 バージョン管理システム [1]

2.3.1 バージョン管理システムの機能

バージョン管理システム (VCS: Version Control System) は、ソースコードの変更履歴を記録するとともに、並行したいくつかの開発を効率良く行うための支援機能を提供する。

有名な VCS はいくつかあるが、共通していることは以下のことである。

- リポジトリとよばれるデータベースを用意し、リポジトリにファイルの変更履歴などを格納する。
- ユーザはリポジトリからそのコピー（作業コピーと呼ばれる）をローカルのディスクやネットワークを通じて取り出し、その作業コピーのファイルに対して変更を加える。そして、その変更をリポジトリに登録する。多くの VCS ではその登録作業をコミットと呼んでいる。その変更内容の登録の際には、変更者、時刻もあわせて記録される。
- ユーザは任意の時点におけるファイル群をリポジトリから取り出すことができる。
- リポジトリには開発本流があり、ユーザはその本流の枝分かれであるブランチを作成して、同様にブランチへの変更内容をコミットしてリポジトリに登録できる。また、ブランチからさらにブランチを作成することもできる。ブランチへの変更は、マージと呼ばれる操作で枝分かれ元に反映させることができる。これらの流れの模式図を図 2.3.1 に示す。もし、マージの際にすでにコミットされた他の開発者からの変更と衝突する場合はそれを検知してユーザに知らせる。

以下では、数値予報課で以前 VCS として広く使われてきた CVS (Concurrent Versions System) [2]、そして、現在、開発管理サーバで利用可能となっている Subversion[3] および Git[4] について簡単に解説する。なお、開発管理サーバの利用にあたって、Subversion と Git のどちらを使うかは開発コミュニティが選択することができることとしているが、開発過程を共有するためのツールという観点からの運用ルールの留意点について触れる。それぞれの詳細については、各システムの公式ウェブページを参照されたい。

2.3.2 CVS

2000 年代まで VCS として広く使われていたのが CVS である。数値予報課でも 2000 年ごろから利用が始まり、全球モデル、メソモデル、NuSDaS ライブラリなどの変更履歴が CVS で管理されていた [5]。CVS は

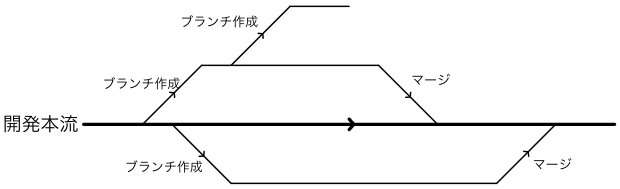

図 2.3.1 バージョン管理システムにおける開発本流とブランチ、マージの模式図。左から右へと時間が流れる。

バージョン管理システムが求められる基本的な機能を有していたものの、ファイル名やディレクトリ名の変更履歴が追跡できないこと [6]、バイナリファイルの扱いが得意ではなかったこと、ファイル単位でバージョン番号が与えられていたので、複数のファイルの変更が同時にコミットされてもそれが分かりづらかったり、ファイルによってコミット時刻が微妙に異なったりするために [7]、指定されたある時刻における状態を確実に取り出せている保証がなかった [8] ことなどの欠点があった。これらの欠点を解決しようと開発されたのが、現在では広く利用されている Subversion である。

2.3.3 Subversion

Subversion は CVS の操作性を継承しつつ、CVS ではファイルごとにつけられていたバージョン番号による履歴の管理ではなく、ファイル群にリビジョン番号を付加して管理することで [9]、リビジョン番号を指定すれば、確実にそのときのファイルの状態を取り出せるようになった。また、ファイル名やディレクトリ名の変更の際にも変更したことを履歴に残せるようになり、変更履歴の連続性を確保できるようになった。また、CVS ではリポジトリと作業コピーの間の差分を調べる際にもリポジトリとのネットワーク通信が発生していたが、Subversion ではリポジトリの内容がローカルにもコピーされているため、リポジトリと作業コピーの間の差分の取得にネットワーク通信が必要なくなり、高速に動作するようになった [10]。

CVS のリポジトリから Subversion のリポジトリへの移行は `cvs2svn`[11] と呼ばれる移行ツールを利用することで行うことができる。数値予報課では、2008 年ごろから一部のモデル開発で CVS から Subversion への

---

[1] 原 旅人
[2] `http://www.nongnu.org/cvs/`
[3] `http://subversion.apache.org`
[4] `https://git-scm.com`
[5] CVS が利用される以前のほとんどの変更履歴は、現在の開発者から見られるようになっていない。
[6] ファイル名などを変更する場合は、一旦削除した上で、新規のものとして登録することになるので、そこで履歴の連続性を失っていた。
[7] 複数のファイルを同時にコミットしても、ごく短時間の間にファイルが一つずつコミットされる扱いになる。
[8] アトミック性を満たさない、と言われる。
[9] Subversion では、ファイルを一つでも変更するとリポジトリのリビジョン番号が 1 つ増える。
[10] ただし、最新のリポジトリの内容との差分を取得するためには、`svn update` によってリポジトリの内容とローカルに保存されている情報を同期させておく必要がある。
[11] `http://cvs2svn.tigris.org`

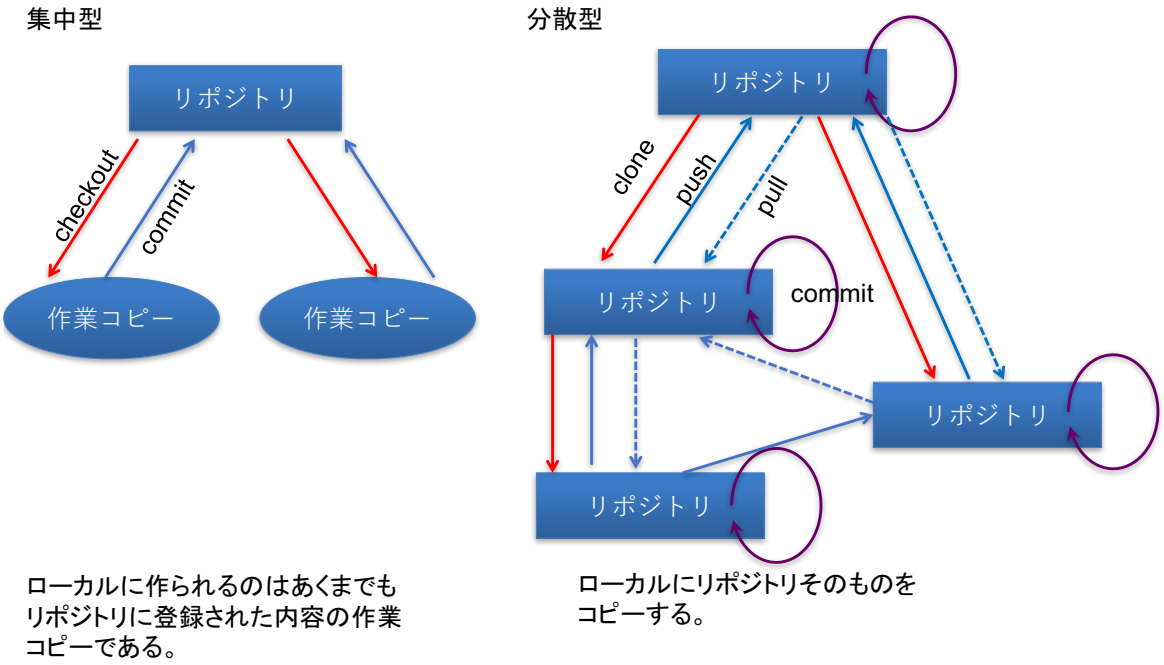

図 2.3.2　集中型 VCS と分散型 VCS のリポジトリとその関連操作の模式図。

移行が行われたり、新しくリポジトリを作成する場合には Subversion が利用されることがあった。現在、数値予報課で CVS のリポジトリが利用されているのは、既に開発がほとんど行われていない古いソフトウェアばかりであり、数値予報課内で利用されているほとんどのリポジトリは Subversion か、後述の Git になっている。

Subverion では開発本流のことを trunk と呼んでいる。リポジトリの最上位のディレクトリに `trunk`, `branches`, `tags`[12] というディレクトリを作成することを推奨し、開発本流は `trunk` 以下に格納することが多い。このことがあくまでも「推奨」であることからもわかるように、ソフトウェア上で trunk を特別扱いする機能はなく、リポジトリの運用ルールのなかで決められるものである。

2.3.4　Git

これまで紹介してきた CVS や Subversion ではリポジトリをあるサーバ上に設置し、ユーザはそのリポジトリに接続して、作業コピーの取得や修正内容のコミットを行う。すなわち、リポジトリは 1 つであり、そのリポジトリは中央リポジトリと呼ばれることがある。このような性質を持つ VCS を集中型 VCS と呼んでいる。

一方、分散型 VCS と呼ばれるものもあり、その代表的なものが Git である。その他にも、Mercurial[13], bazaar[14] などが知られている。集中型では中央リポジトリに格納されているファイルのコピーをユーザは取得するが、分散型ではリポジトリそのもののコピーをローカルやネットワークを通じて取得する。集中型では、ユーザの変更をコミットすることで中央リポジトリに登録するが、分散型ではユーザが取得するのはリポジトリであるので、ユーザは自らの変更内容のコミットを自分が取得したリポジトリに行う。

分散型 VCS におけるコミット操作はローカルの自分のリポジトリに対するものであるので、自分の変更を他のリポジトリに反映させたり、他人の変更を自分のリポジトリに反映させるには、それぞれプッシュ、プルと呼ばれる操作を行う。

分散型においては、コピー元とコピー先のリポジトリはソフトウェア上は対等であり、特に区別するものはない。しかし、VCS の主要な機能の一つは、多くの開発者が行った変更の開発本流への取り込みであり、その取り込みを行うために運用ルールとして中央リポジトリを一つ定めていることが多く、各ユーザが自らのリポジトリに行った変更をその中央リポジトリに反映させる操作を行う。

集中型と分散型のバージョン管理システムのリポジトリとその関連操作の模式図を図 2.3.2 に示す。

集中型 VCS である Subversion と比較したとき、分散型 VCS である Git は、リポジトリをコピーしてしまえば、自らのリポジトリの更新そのものにはネットワーク接続は必要なくオフラインで利用できること（集中

[12] CVS にはリポジトリのある時刻の状態にタグというものを付加できて、そのタグを指定することである時点でのファイル群を取り出すことができた。Subversion では CVS のようにタグを付加する機能はないものの、ある時点の `trunk` を `tags` にブランチを作成するのと同じ要領でコピーすることで、タグの付加と同等の機能を実現している。

[13] `https://www.mercurial-scm.org`

[14] `http://bazaar.canonical.com/en/`

型では変更内容のコミットには中央リポジトリへの接続が必要となる)、中央リポジトリに自らの変更をプッシュをするまでは中央リポジトリに影響をあたえることなく利用できることが大きな特徴であると言えよう。また、リポジトリを簡単にコピーできるため、中央リポジトリとは関係なく、開発本流と少しだけ機能を変えた派生版を作りやすいことも特徴の一つであろう。一方で、ブランチでの開発も含めて共有したい場合には、これらの特徴がデメリットになり得るので注意する必要がある。

2.3.5 バージョン管理システムの運用ルールの留意点

すでに述べたように、開発管理サーバではSubversionとGitを標準的なVCSと位置づけ、利用ができるようになっている。開発が完了した開発本流の変更履歴を管理していくという点では、どちらを使っても大きな差はない。一方、開発途上のソースコードの扱いという面では違いが生じうる。

集中型であるSubversionでは、開発途上のソースコードの変更履歴を残すためには[15]中央リポジトリにコミットする必要がある。そのために、中央リポジトリにブランチを作って、開発途上のソースコードの変更内容をそのブランチにコミットすることが多い。その結果として、その変更履歴を他の開発者が見ることができて、他の開発者の開発内容や進捗を把握することができる。

一方、分散型であるGitでは、変更履歴を残したりバックアップのためにリポジトリにコミットすることは集中型と同じであるが、そのリポジトリは自らのリポジトリであり、他の開発者からも見ることができるリポジトリにプッシュしなければ、自らの変更内容を他の開発者からは見えないようにすることもできてしまう。

一般的なオープンソースのソフトウェアの開発においては、開発途上のソースコードに興味を持たれる場面は多くないかもしれない。しかし、モデル開発においては、最後の完成形だけでなく、開発途中のソースコードにも重要な情報が含まれている場合もある。VCSとしてSubversionを使う場合はブランチに開発途上のソースコードをコミットするタイミング、Gitを用いる場合には、他の開発者からも見えるリポジトリにプッシュするタイミングを開発コミュニティで決めておく必要があるだろう。

[15] 変更履歴の記録とともに、バックアップにもなる。

2.4 数値予報モデル開発管理情報共有装置（開発管理サーバ）[1]

2.4.1 はじめに

気象庁では第 2.2 節で述べた開発管理システム及び第 2.3 節で述べたバージョン管理システムを実際に提供するための環境として、数値予報モデル開発管理情報共有装置（以下、開発管理サーバ）を整備し、気象庁内の数値予報モデルの開発・研究や維持管理に携わる職員（以下、数値予報モデル開発者）が実施する開発について、情報の管理を行っている。

本節では開発管理サーバの設置目的とシステム構成を概説した後、実際どのように運用を行っているか、その方法について解説する。

2.4.2 装置の設置目的

気象庁内で数値予報システムを利用する分野が拡大するのに伴い、気象庁本庁の複数の課室や気象研究所の研究室が数値予報システムの開発に参加するようになっている。しかし、開発項目が拡大するにつれ、他の課室・研究室で行われている開発の内容や進捗状況の情報共有といった連携が十分にとれないという弊害も発生しやすくなる。このような状態に陥ると、開発内容の重複が生じるなど非効率的な開発が発生することに繋がりかねない。また、既知の技術的な手法で解決可能な問題にも関わらず、情報不足によって未解決のまま開発が停滞するなど、業務全体の遅延をもたらす可能性がある。これらの懸念を解決するためにも統一した環境のもとで開発管理を行い、数値予報モデル開発者同士が互いに開発内容を把握するとともに成果を相互利用できることが望ましい。

こうした基本認識を前提に、気象庁技術開発推進本部に設置されたモデル技術開発部会開発管理調整グループでは、2011 年度から統一的な開発環境の整備について継続的に議論し、2014 年 3 月 25 日に「統一環境における数値予報モデルの開発管理の指針」（以下、指針）をとりまとめた。この中では、

- モデル技術開発部会が取り組む、すべての開発内容を開発管理の対象とすること
- 管理の対象とするコンテンツは、現業モデルの開発や維持管理に必要なソースコードや定数[2]などのデータ、開発中のソースコード等の成果とすること
- モデル本体だけではなく、必要により周辺ソフトウェア（検証、可視化など）の管理も実施することを推奨すること
- 関連グループ長[3]は、開発計画をプロジェクト・サブプロジェクト[4]上に公開し、数値予報モデル開発者はこれを遵守して開発を行うこと

などが定められるとともに、

- プロジェクト管理システムとして第 2.2 節で述べた Redmine を利用すること
- バージョン管理システムとして第 2.3 節で述べた Subversion 又は Git を使用すること

が決定された。この指針を基に開発管理環境を提供するシステムとして、開発管理サーバが設置された。

表 2.4.1 開発管理サーバの機器構成

サーバ	主・副 2 台
	CPU: 3 GHz (6 core) × 2
	メモリ: 48 GB (8 GB × 6)
	HDD: 900 GB 2.5 型 SAS × 4
	RAID: RAID5 + spare
大容量ストレージ装置	2 台 + 待機 1 台（非接続）
	HDD: 3 TB SATA × 6
	RAID: RAID6

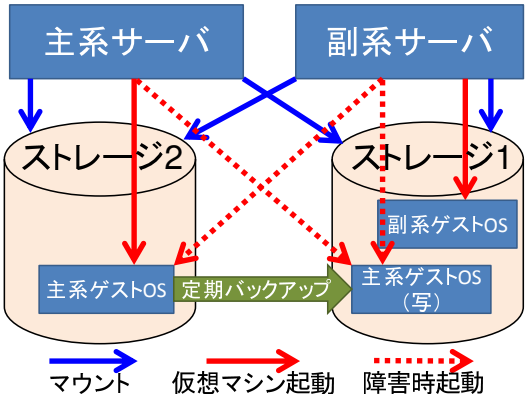

図 2.4.1 開発管理サーバの構成図。両サーバは 2 台の大容量ストレージ装置をマウントしており、それぞれに主系・副系のゲスト OS を記録したイメージファイルを格納している。

2.4.3 システム構成

開発管理サーバは表 2.4.1 のような、主・副のサーバ 2 台と、3 台の大容量ストレージ装置によって構成されている。大容量ストレージ装置のうち 1 台は予備機のためサーバと接続していない。サーバは主・副いずれもホスト OS として CentOS Linux を採用し、KVM (Kernel-based Virtual Machine) による仮想環境を用いたゲスト OS 上に開発管理環境を構築している。ゲスト OS のイメージファイルは大容量ストレージ装置に保存しており、サーバ本体のディスクにはホスト関係のファイルのみが保存されている（図 2.4.1）。現在は開発管理サーバが開発業務の中核を担っていることもあり、これらのデータが保管されているイメージファ

[1] 雁津 克彦
[2] 実行プログラムが利用するデータのうち、標高や気候値のように内容が滅多に変わらないデータ。
[3] 関連グループとは、気象庁技術開発推進本部モデル技術開発部会に設置された各グループを指す。
[4] プロジェクト・サブプロジェクトは第 2.4.4 項 (2) で取り上げる。

イルを毎日 1 回バックアップするとともに、ゲスト OS に保存されているデータのうち、開発管理に直接関係するファイルは個別にバックアップを実施している。

利用者は主系のゲスト OS へアクセスすることで、開発管理環境を利用することができる。このとき、アクセス可能なプロトコルを HTTP(S) に限定して許可している。SSH や RSH といったターミナルによるリモートアクセス、FTP や CIFS などのファイル転送・共有プロトコルは一切受け付けていない。通常、数値予報システム開発では SSH 等を利用して直接サーバにログインを行う。開発作業における各種業務を実施するにあたって、サーバ間のファイル転送、コンパイル作業、画像作成、統計的な検証などでサーバ上のコマンド利用が実務上不可欠なためである。これに対して開発管理サーバの利用では、このようなサーバ上でのコマンド利用を行うことなくほぼ全ての機能を提供可能である。プロジェクト管理システムである Redmine は、Ruby on Rails で構築されたウェブアプリケーションであり [5]、ユーザは通常のウェブブラウザを用いることでほぼ全ての機能を利用することが可能である。また、バージョン管理システムの Subversion と Git についても、利用形態はコマンドライン主体であるが、通信プロトコルとして HTTP(S) を選択可能であり (Sussman et al. 2011; Chacon and Straub 2014)、ユーザの端末でコマンドを実行し、HTTP(S) 経由で開発管理サーバのバージョン管理システム（第 2.3.1 項を参照）を利用することが可能である。こうしたことから、開発管理サーバではユーザが直接サーバにログインすることなく HTTP(S) によるアクセスのみで十分機能するようになっている。

管理面では、ターミナルによるリモートアクセスをサポートする必要がないため、サーバに必要なソフトウェアを限定することができる。また、サーバへの通信手段も限定されることから障害調査時に原因切り分けが容易になる効果も生まれ、結果として管理コストの低減にも繋がっている。

開発管理サーバは気象庁外部から直接アクセスできないようになっており、併せて適宜セキュリティパッチを適用するなどセキュリティ対策を行っている。また、Redmine についてはプラグインにより機能を強化することも可能であることから、必要があればプラグイン導入の作業を実施している [6]。ただし、気象庁全体の情報共有という目的を逸脱しない範囲の利用に留めており、不要なプラグインの導入は実施しない方針としている。

[5] `http://www.redmine.org/projects/redmine/wiki`

[6] Redmine は Ruby で実装されているため、プラグイン管理も Ruby を用いる。Ruby は数値予報ルーチン関連のツールでも利用されている言語であり、管理者育成でも比較的少ない人的教育コストで管理可能という面を持ち合わせている。

表 2.4.2　2017 年 1 月時点で運用中の Redmine 一覧。末尾の「お試し Redmine」は Redmine に不慣れなユーザ・管理者が機能を自由に試用し、慣熟できる環境として公開している。

全球モデル
NHM
asuca
物理過程ライブラリ
海洋気象情報
数値予報事例 DB
共通基盤
化学輸送
同化・観測・QC 関連
結合系
海洋
ガイダンスグループ
お試し Redmine

2.4.4　開発管理サーバの管理体制

(1)　各コミュニティによる利用

開発管理サーバの特徴として、一つのサーバ上に複数の Redmine を運用している点が挙げられる。指針が定める開発管理サーバの利用範囲は、気象庁本庁と気象研究所を含めた複数の課室を想定していることもあり、管理する開発対象が非常に多岐にわたっている。こうした背景からサーバ上で複数の Redmine を運用し、各 Redmine の管理方法の細目は利用するコミュニティに委ねる方式を採用している。これによって、各コミュニティの既存の開発プロセスを踏襲しつつ、第 2.2.3 項で述べた Redmine を活用した開発が随時導入できるよう配慮している。

現在、開発管理サーバでは表 2.4.2 に挙げる Redmine を運用しており、モデル技術開発部会が取り組むほとんどの開発課題は、関連する Redmine 上で開発管理が実施されている。

(2)　各 Redmine の管理

Redmine は内部に複数のプロジェクトを設置することができ、さらにプロジェクト内部にサブプロジェクトを設置することが可能である。指針では開発管理サーバ上の Redmine に設置するプロジェクトとサブプロジェクトについて、数値予報モデル開発の中程度の単位をプロジェクトに、数値予報モデル開発の小さな単位をサブプロジェクトに割り当てることとされている [7]。非常に概念的な指定方法と思うかもしれないが、一方でプロジェクトやサブプロジェクトの単位を各コミュニティが実情に沿った方法で選択可能であることを意味し、柔軟な運用を可能にしている。開発管理サーバ自体の管理業務は数値予報課数値予報班基盤整備グループが担

[7] 数値予報モデル開発の大きな単位は Redmine に割り当てる。

表 2.4.3 Redmine、プロジェクト及びサブプロジェクトの設置、変更及び廃止に必要な手続

対象	必要な手続
Redmine	関連グループ長と開発管理調整グループの承認が必要
プロジェクト	関連グループ長と開発管理調整グループの承認が必要
サブプロジェクト	プロジェクト管理者の承認が必要

当しているが、このように運用面の多くをコミュニティに委ねていることもあり、各 Redmine、プロジェクト及びサブプロジェクトの管理についても、管理者権限を各コミュニティに委譲している。すなわち、開発管理サーバ自体の管理者の他に、各 Redmine に Redmine 管理者を、プロジェクトにプロジェクト管理者を、サブプロジェクトにサブプロジェクト管理者を設け、運用管理を分散して行っている。

開発管理サーバで管理する課題の内容は、業務上定められた各種の開発計画に沿ったものを想定している。このことは新規プロジェクトや新規サブプロジェクトの設置において、計画に沿った開発項目が必ず背景に存在していることを意味している。一方で Redmine の性質上、新規プロジェクトやサブプロジェクトを数分程度の簡単な作業で作成できる。しかし、プロジェクトやサブプロジェクトを乱立させることは、前述の想定を鑑みると決して好ましいことではない。そこで、Redmine、プロジェクト及びサブプロジェクトの設置、変更、及び廃止に際しては、それぞれ表 2.4.3 に示す手続が必要となっている。これによって、業務上必要な課題に必要な資源を投入し、開発に取り組むことを推奨している。

2.4.5 利用方法

(1) 利用開始時の作業

Redmine 上のチケット及びリポジトリは、開発管理サーバが設置されているスーパーコンピュータシステムの支線 LAN に HTTP(S) で接続できる端末から誰でも閲覧することができる。しかし、Redmine 上にチケットを作成したり、リポジトリにコミットするためには Redmine にユーザ登録を行う必要がある [8]。

ユーザ登録は登録を希望する Redmine の画面上で行うことができる。ユーザ登録を行うと Redmine 管理者にメールが送られ、Redmine 管理者が承認することで Redmine 上の各種機能を本格的に利用することができる。注意点として開発管理サーバでは複数の Redmine を運用しているため、異なる Redmine を利用する際にはその都度ユーザ登録が必要となる。例として全球モデル Redmine にユーザ登録済みであったとしても、海洋 Redmine を利用する際には海洋 Redmine にもユーザ登録を行い、海洋 Redmine の Redmine 管理者から利用の承認を受ける必要がある。

Redmine 管理者がユーザを承認する際は必要に応じてユーザにプロジェクト上の役割である「ロール」を設定する。ロールによってチケットが作成できる・できないなどの権限が変わる。どのようなロールが存在するかは、Redmine によって変更することができるため一概には言えないが、概ね、管理者・開発者・報告者の三つは用意されていることが多い。通常、同じ Redmine 内であってもプロジェクトによってユーザに与えられるロールは異なりうる。例えば、あるプロジェクト A では主体的に開発を行うので「開発者」ロールを、別のプロジェクト B では開発成果の利用がメインなので「報告者」ロールを与える、といった方法でユーザとプロジェクトとの関係が割り付けられる。各ロールで何ができるかは Redmine ごとに細かく設定できるため、詳細はシステム上で確認する必要がある。Redmine にログイン可能ユーザであれば Redmine 画面の「情報」⇒「権限レポート」で容易に参照できる。

(2) チケットの登録

指針ではコンテンツを登録する前に、まず課題をチケットに登録することを推奨している。このルールを数値予報モデル開発に導入すると、開発で取り組む予定の全ての課題をチケットとして登録することになる。メリットとして作業の可視化が実現されるとともに作業内容をプロジェクト内で共有できるため、作業の漏れがあればチケットを追加するといった作業抜け防止が期待できる。また、チケット上での各種議論を踏まえた上で開発成果をコミットするという方法をとることで、第三者のレビューによる業務信頼性の向上が期待できる。

指針では、チケットに登録する課題として基本的に開発計画に沿ったものが想定されている。ただし、現業システムの維持管理に必要な開発、軽微な不具合などの外的要因による修正、数値予報モデル開発者個人のアイデアによる成果などは、開発計画に明記されていなくても補完的に登録することができるようになっており、柔軟な開発にも対応している。むしろ、実際の開発ではトライアル・アンド・エラーで様々なアイデアを試すことや予期せぬ不具合への対応が非常に重要になってくることから、こうした柔軟性の確保は不可欠である。

チケットの登録では、プロジェクト内の数値予報モデル開発者が共通して理解できる程度の平易な内容で記載することが推奨されている。これはチケットが登録者だけではなく、相互レビューなどの議論によって第三者から閲覧されることを想定しているためである。また、チケット上の議論は後に過去の開発経緯を調べ

[8] チケット、リポジトリ、コミットなどの用語は第 2.2 節、第 2.3 節を参照のこと。

表 2.4.4　チケットとリポジトリの紐付け方法。以下のコメントを所定の記載箇所に記入することで紐付けできる。

コメント	例	記載箇所	効果
r[リビジョン番号]	r200	チケット	チケット本文から特定リビジョンへのリンク
#[チケット番号]	#1432	コミットログ	リポジトリブラウザからチケットへのリンク
refs #[チケット番号] issueID #[チケット番号] references #[チケット番号]	refs #1432 issueID #1432 references #1432	コミットログ	リポジトリブラウザからチケットへのリンクと、チケットからリポジトリブラウザへの相互リンク（どの書法も同じ効果）

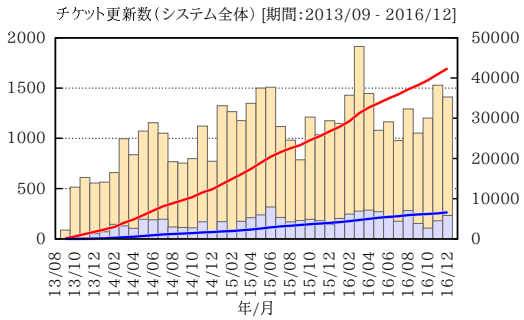

図 2.4.2　月ごとのチケット更新数（棒：左軸）と累計（折れ線：右軸）。橙・赤は全体数を、水色・青は全体数のうち気象研究所の該当数を表す。ここで、2017 年 1 月時点で気象研究所のメールアドレスを登録しているユーザを気象研究所ユーザとし、気象研究所ユーザの更新した件数が気象研究所の更新件数としてカウントしている。

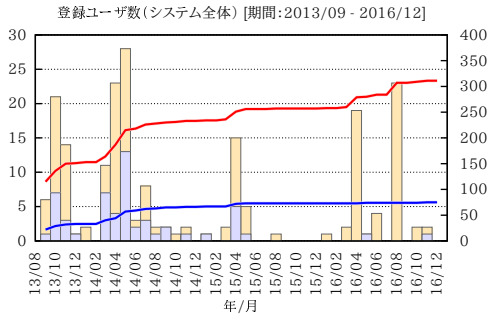

図 2.4.3　月ごとの新規ユーザ登録数（棒：左軸）と累計数（折れ線：右軸）。橙・赤は全体数を、水色・青は全体数のうち気象研究所の該当数を表す。ユーザ判定方法は図 2.4.2 と同じ。

る際に重要な情報源として活用できることから、検索性の面でもプロジェクト内の関係者が理解できる内容で登録することが望ましい。

以下に、開発管理サーバでのチケットを利用した典型的な開発プロセスを挙げる。もちろん、実際の開発手順は各コミュニティに委ねられているものであり、これは一例に過ぎない。

1. 課題発生（不具合修正、機能追加など）。
2. チケット作成。担当者割り当て。
3. 進捗を随時記録。
4. 進捗に応じて担当者を変更。
5. 第三者によるレビュー。
6. 課題解決。チケットを閉じる。
7. 一連の作業が記録に残ることで後から検索。

(3)　リポジトリとの連携

Redmine はチケット駆動開発に大変適している。チケット駆動開発ではチケットなしでコミットしてはいけないという「No Ticket, No Commit!」ルールがある (小川・阪井 2010)。これを導入することで、課題と成果の紐付けが可能になり開発情報が整理される。さらに、コミット前にチケットが作成されることで、作業内容が多くの関係者の目に触れることになり、作業に先駆けて問題点やアドバイスが受けられるなど開発の効率化が期待できる。また、レビューを経てからリポジトリを更新することを追加で義務付けることで開発成果の信頼性向上にも繋げることができる。

Redmine はチケットとリポジトリを連携する機能を提供しており、チケット駆動開発を強力にサポートする。一例として表 2.4.4 のようなコメントを入れることで、チケットとリポジトリを相互に参照することができる。最終的な開発成果は Subversion であればトランクへの、Git であればマスターへのコミットで開発が終了するが、開発管理サーバ独自の運用として、Subversion のトランクへコミットができるユーザをデフォルトではプロジェクト管理者に限定している。これによって、不用意にトランクが改変されることを防ぐとともに、開発計画と異なる改変がないかプロジェクト管理者が確認できるようになっている。

2.4.6　開発管理サーバの利用推進と利用実績

開発管理調整グループでは指針の取りまとめと前後して、開発管理サーバの積極的な普及のため、2013 年度から 2015 年度にかけての 3 年間、気象庁本庁と気象研究所で数値予報モデル開発者向けに利用者説明会を毎年開催し、開発管理サーバの利用目的、利便性、活用方法などについて精力的に紹介を行ってきた。こうした背景もあり、図 2.4.2、図 2.4.3 のように、2013 年度の利用開始以来チケット更新数、ユーザ登録者数が増加し、着実に気象庁内での利用が普及しているところである。特に利用開始からの 2 年間で急速に普及が進んでいることが見て取れる。なお、近年はユーザ登録者数の伸びも減少し開発への導入が定着している状況が伺える。こうした現状を踏まえ、2016 年度は数値予報モデル開発者全体への利用者説明会に代わり、新

規数値予報モデル開発者を対象とした研修で案内を行うよう変更するとともに、より効果的な活用方法についての検討会を気象庁本庁と気象研究所で開催する方針に変更している。こうした取組を通じて、開発管理サーバの運営方法の改善も含めた、効率的な開発方法の模索を継続している。

2.4.7 開発管理サーバの課題

開発管理サーバではユーザビリティを向上させるため、ユーザ登録メールやコミットメールの発出処理、自動バックアップ、管理者一覧情報の作成など、Redmine以外のツールも多数組み合わせて開発管理環境を提供している。また、Redmine自体の利便性を高めるため、サードパーティ製又は気象庁独自開発のプラグイン導入も行っている。こうした各種ツールは、ユーザの要望や障害対応などを踏まえて随時開発を行っているが、これらのツールの管理に開発管理サーバは一切用いられていない。

その最たる理由は障害対応である。開発管理サーバに関する開発情報は、開発管理サーバに障害が発生した際に利用できる必要があり、障害時でも参照できる場所に情報が存在しなければならない。そのため、開発管理サーバ自体の開発に関する情報は開発管理サーバの外で独自に管理を行う必要がある。これらの情報の多くは開発管理サーバ導入前の環境、すなわちRedmineによる開発プロセスが導入される前に整備された環境を利用して情報が集約されている。このため、現在では当然のように利用されているチケット駆動開発やレビューを行うための枠組みが存在しない。

開発管理サーバの導入から5年が経過し更新が予定されていることを鑑みると、今後開発管理サーバの移植に関する作業が増加することが予想される。こうした作業についても指針で定めたプロセスで管理ができるよう、既存の別サーバや開発管理サーバの副系を活用するなど検討が必要と考えている。

参考文献

Chacon, S. and B. Straub, 2014: *Pro Git, (CHAPTER 4: Git on the Server, The Protocols, The HTTP Protocols)*. 2nd ed., Apress, `https://git-scm.com/book/en/v2`, 127–129 pp.

小川明彦, 阪井誠, 2010: Redmineによるタスクマネジメント実践技法. 翔泳社, 190 pp.

Sussman, B.C., B.W. Fitzpatrick, and C.M. Pilato, 2011: *Version Control with Subversion: For Subversion 1.7: (Compiled from r5239), (1. Fundamental Concepts, Version Control the Subversion Way, Addressing the Repository)*. `http://svnbook.red-bean.com/en/1.7/svn-book.html`, 8–9 pp.

2.5 活用例 (1)–全球モデル [1]

本節では全球モデル (GSM) の開発における開発の管理手法について紹介する。まず開発管理の手法を発展させる必要性が高まった経緯と背景について解説した後、対応策として導入した開発の管理手法と検証環境について実例を交えながら紹介する。

2.5.1 背景

GSM は数値予報の中の基盤となるモデルであり、気象庁では 1988 年の運用開始以来継続的に開発・改良を実施している。2007 年 11 月には、その水平分解能を約 55 km から約 20 km に大幅に引き上げ、同時に運用を終了した領域数値予報モデル (RSM) と台風数値予報モデル (TYM) が担っていた役割を引き継いでいる (北川 2006)。この GSM0711[2] は開発の大きな節目であり、RSM, TYM の役割を統合した高分解能 GSM の運用を開始したことは数値予報システムとして大きな発展であった。

一方で、その後の GSM0808 (岩村 2008) において、並行して進めていた関連項目の開発が一段落して以降、GSM1212 (下河邉・古河 2012) まで、数年の間精度改善につながるモデル更新が停滞する期間が続いた（表 2.5.1）。停滞の原因を明確に特定することは不可能ではあるが、当時の開発で問題になっていたことの一つとして、単独の開発項目（あるスキームの更新など）を反映させた試験において、予測精度が大きく悪化する要素が見られ、その問題となる部分を克服できない例が続いたことがある。

例えば、GSM1212 以前の開発で問題になっていたが、その後解決に成功した例として陸面過程の開発が挙げられる。当時、新しい陸面過程を GSM の大気部分と組み合わせると、冬期に北半球高緯度の地表面付近で大きな低温バイアスが生じることが問題となっていた (平井・堀田 2009)。GSM には陸上で雲が少ないという問題があり、地表面へ入射する下向きの長波放射が過少であったため、本来なら旧陸面過程でも地表面では加熱が足りず低温バイアスとなるはずであった。しかし、旧陸面過程や旧境界層過程には、積雪の不適切な取り扱いが原因で地中から過剰な熱輸送を表現する、境界層での強安定時の拡散係数の下限値に非現実的な値を用いて大気から過剰に熱を輸送する、陸域の地表面の熱容量を大きく設定し温度変化を抑えるなど、長波放射過少を補償する不適切な取り扱いや非物理的な対策が多く存在していた。新陸面過程でより物理的に妥当な取り扱いをしても、長波放射不足の問題が顕在化して過剰な夜間の冷却、大きな低温バイアスが生じてしまう状況であり、地表面の放射収支に対してより正しく応答する新陸面過程が、従来の非物理的な調整を多く含む旧過程に対し予測精度で上回れない状況であった [3]。

表 2.5.1 GSM の変更履歴（GSM0711 以降）

	主な変更内容
GSM0711	水平分解能約 20 km、鉛直層 60 層、モデルトップ 0.1 hPa へ仕様変更 (北川 2007)
GSM0801	積雲過程の改良 (気象庁予報部 2007)
GSM0808	力学過程の改良、適合ガウス格子の採用 (岩村 2008)
GSM1011	入出力システムの刷新による高速化
GSM1108	出力専用ジョブ統合による高速化
GSM1212	層積雲スキームの改良 (下河邉・古河 2012)
GSM1304	放射過程（エーロゾル気候値、水蒸気吸収係数）の改良
GSM1403	物理過程改良（放射・境界層・重力波・積雲・陸面)、鉛直層 100 層とモデルトップ 0.01 hPa へ仕様変更、及び入出力システムなどの高速化 (米原 2014)
GSM1603	物理過程改良（積雲・雲・陸面・放射・海面）及び力学過程の高速化 (米原 2016)

このように、根本的な原因が解決されない状態で別の手段により打ち消されて「隠された」問題点は、compensating errors と呼ばれる (堀田・原 2012) [4]。GSM に compensating errors が多く内在しているであろうことは当時も推測されていたが、その要因は複雑に絡み合っており、単純に一つの部分だけを修正すれば全体的な精度改善が得られる状況ではなかった。各過程の担当者は、それぞれの過程の枠内での調査し改良を試みるものの、モデル全体を通した開発・調査は十分には進展しなかった。これには、計算機資源の制約により、複数の過程の変更を組み合わせた実験を十分に行えなかったことも関係している。項目の組み合わせにより必要な実験数は容易に増加してしまうが、水平格子間隔 20 km の高分解能 GSM を実行するには、当時の計算機では大きな資源を必要としたためである。実験結果が無い状態で不確実な推測ベースの議論を行うだけでは、開発者間の問題点に対する共通認識が十分に生まれなかったのである。

転機となったのが、2012 年 6 月 5 日の第 9 世代スーパーコンピュータシステムの更新 (西尾 2011) である。利用できる計算機の能力が飛躍的に向上したことによ

[1] 米原 仁

[2] 2007 年 11 月に運用を開始した GSM のバージョン。本節では、GSM の各バージョンを、改良を導入した西暦の下二桁と月を「GSM」の後ろに付けて呼ぶ。

[3] この問題の解決については本節の第 2.5.3 項で紹介する。

[4] 第 1.2.4 項にも解説がある。

り、それまでは計算機資源の制約のため実行できなかった改良項目の組み合わせ実験が可能になった。計算機の更新は、基礎開発を行い易くなったことも含めて、GSM 開発に非常に大きな発展をもたらしている。一方で、組み合わせ実験の計画と実施、結果の評価検証と分析、担当者間の議論を通じた相互のフィードバック、次の実験の設定といった開発サイクルを効率良く進めていくために、組織的・系統的な進捗の管理、評価検証手法の高度化、各種実験の実施をサポートするツールなど、開発管理手法を発展させる必要性が生じたのである。

このような背景があるため、GSM の開発プロセスでは、GSM を構成する各部分はお互いに強く関連しあっていることを意識し、全体を一つの「パッケージ」と考えて効率的に開発を行なうことに重点が置かれている。力学過程や陸面過程、積雲対流過程といった、GSM を構成する個別部品の開発をそれぞれ独立に行うのではなく、GSM 全体の問題意識を開発者間で密に共有し、組み合わせ実験と議論を繰り返している。パッケージとしての開発を成功させるためには、担当者がそれぞれの高い専門性を持つだけでなく、モデル全般に関する広い知識と経験に基づいて相互に活発な議論を行うことが重要であり、開発者間のコミュニケーションを促進することも開発管理手法の重要な役割の一つである。

第 1.2.4 項の図 1.2.1 に数値予報課における一般的な開発フローが紹介されているが、GSM の開発プロセスでは、その中でも特に性能評価試験を組み合わせ実験の対象として重視している。性能評価試験はモデルによる予測実験だけでなくデータ同化実験（全球解析）を含み、現業運用の形態にかなり近い試験である。全球解析では第一推定値や品質管理に GSM の予測値を用いるため、予測モデルだけを変更した場合でも、データ同化に利用される観測数をはじめとして、観測から引き出す情報量が変わり、解析値の精度自体が大きく変わり得る。同一の解析値を初期値に使用してモデルを比較する場合に比べて、データ同化実験を含めた試験では予測精度の差異がかなり強く現れるため [5]、変更のインパクトが性能評価試験を行って初めて分かることも多い。

また、評価検証は開発フローの一部であり、効率的な開発のためには検証システムの効率化・高度化も重要である。そのため、評価検証の点でも様々な取り組みを行っており、GSM 開発の標準検証環境である DPSIVS (Deterministic Prediction System Integrated Verification System) はその成果の一つである。評価検証についての取り組みも本節の後半で紹介する。

[5] これまでの開発の経験から、統計的なスコアの改善率が数倍程度異なることがあることが知られている。

2.5.2 開発の進め方

ここからは、GSM 本体の開発管理手法について具体的に説明していく。この手法は GSM1403 (米原 2014) の開発時から利用されており、GSM1603 (米原 2016) 及び開発中の次期 GSM の 3 世代の開発で用いられている。Redmine や各種ツール群は、第 1.2.4 項の図 1.2.1 でいうところの基礎開発の段階での活用や、第 2.2 節で解説されている一般的な観点でも利用されている。ただし、ソフトウェア開発として一般的な内容であるので本節では触れず、ソースコードに関するポリシーの紹介に留める。

(1) 開発の基本的な進め方

開発課題の進捗管理方法は、次の GSM へ導入を目指す短期課題とそれ以外の中長期課題で大きく異なる。中長期課題の計画と調整は、気象庁技術開発推進本部・モデル技術開発部会などで議論されている。短期より長い期間におよぶ開発計画は、基本的にはスーパーコンピュータシステムの更新に合わせ、次の世代の計算機で何ができるかを中心に組み立てられることが多い。ただし、複数年にまたがる開発項目である中長期課題の進捗管理は、基本的にそれぞれの開発者に任せられていることが多い。年度当初に各担当が開発会議で昨年度の計画からの変更点を示し、その後の進捗については定例ミーティングなどで報告するのが基本であり、Redmine やリポジトリも各開発者が使い易い方法で利用している。

一方、短期課題は Redmine 上でのチケット作成、共有リポジトリの利用が必須となる。図 2.5.1 に GSM の短期課題についての開発作業の流れを示す。GSM のバージョン更新には、1 年から 2 年程度の期間で区切りを付けている。はじめに、これまでの開発から得られた問題点などの知見を基に議論を行い、優先する開発項目を洗い出す。開発項目を次のバージョンの候補にする提案があった場合、その導入を目指すべきか関係者で議論して優先度を決める。このとき、科学的な正しさ、調査の進捗度合い、現在判明している問題点との関係などを議論するが、ミーティングにおいて議論が論争的であった場合は、その後もチケット上でフォローアップを繰り返す作業を行う。

各項目の担当者は、まずは基礎試験及び、必要な場合は単独で性能評価試験を実施して結果をまとめ、ミーティングで報告することが推奨されている。ミーティングは開催する機会が毎週決まった曜日に設けられており、希望者がいる場合に開催している。単独の試験結果に問題がまったくなければ良いが、通常そのようなことはあまりなく、想定していた部分での改善以外の、打ち消されて隠されていた誤差が顕在化する。そこで、開発者が相互にその問題点を共有し、組み合わせて考えるべき課題の選定や、問題解決に向けて必要な調査・開発項目の新規追加が議論される。そして再

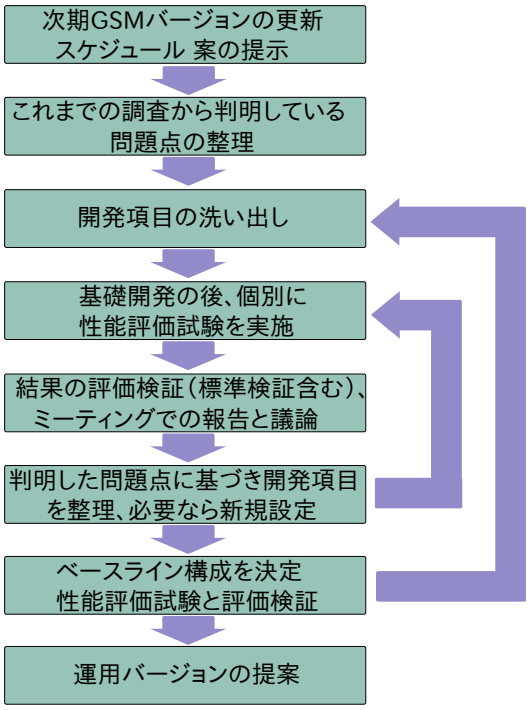

図 2.5.1　GSM の短期課題についての開発作業の流れ。全体が GSM の一つのバージョンの開発に相当する。

び、それぞれにおける基礎開発と性能評価試験を実施するステージに戻る。

開発と調査が進み、議論を経て運用バージョンとしての精度をある程度確保できる見込みが見えてきたら、その構成について性能評価試験を実施したのち、検証の結果を分析して、問題がなければ次のベースライン実験とする。ベースライン実験は、今後の開発のベースにすることが決まった構成による実験を意味し、更新後はその構成に改良を加える形で開発を進める。ベースライン実験への採用基準は変更内容に依存する部分も大きいが、

- 科学的に正しいアプローチか
- 変更の狙いとその影響が明確か
- 必要な精度評価がなされているか
- 予測結果に精度・特性上問題がないか
- 計算安定性や実行時間に問題がないか

といった点について説明し、開発者間のコンセンサスを得ることが基本になる。その時の議論に基づき、最終的には取りまとめの担当者が判断をしている。

ベースライン実験は開発者の共有アカウントで実施され、少なくとも標準的な検証が一通り実行される。その結果について開発者で議論し、問題点の整理と共有を行う。

この一連の手続に基づくベースライン実験の更新は複数回繰り返され、現業運用に問題がないと開発者間で認識を共有できる構成を作成した後、数値予報課内の承認プロセスへ進む。

このように、個々のコンポーネントの開発、取捨選択と組み合わせ、パフォーマンス調整という一本道の進行ではなく、評価検証によりそれまで隠れていた誤差を積極的に明らかにして問題点を開発者で共有し、柔軟に開発項目を入れ替えつつ全体パフォーマンスの調整を行うのが、近年の GSM 開発の特徴である。

(2)　Redmine プロジェクトの利用

解析システム、EPS を含む GSM 関連の開発において、プロジェクト管理・情報共有のためのソフトウェアとしての Redmine の利用は、当時の議論を受けて 2011 年 7 月から開始している (室井ほか 2013)。それ以前の開発情報の共有には、数値予報課が管理するサーバに Wiki を設置して利用するか、電子メールを用いて周知することが多かった。その後、試用期間を経て 2013 年頃から Redmine プロジェクト「全球モデル」として本格的な利用が始まっている。現在この Redmine プロジェクトは、GSM の開発だけでなく解析や全球アンサンブル予報システム、各種ポスト処理、検証システムの開発でも利用されており、2016 年度に利用実績のあるユーザ数は庁内他課室や気象研究所も含めて 30 名程度存在する。

プロジェクトは細かく分けず、GSM に関連する開発全体を 1 つのプロジェクト「GSM」としている。その他のプロジェクトには、数値予報の結果を用いたプロダクト作成処理の管理や、他課室を含めた気象庁全体での開発計画調整などが並ぶ。

一方で、サブプロジェクトは GSM 本体以外の大きな開発対象ごとに分けられている。項目を書きだすと以下のようなものが並ぶ。

- 全球解析
- 全球決定論的検証
- EPS
- アンサンブル検証

プロジェクトとサブプロジェクトはソフトウェアの機能的には同等であり（第 2.2 節）、両者の区分けに利用上の大きな違いはないが、GSM 本体の開発はサブプロジェクト全ての項目の基盤であると考えてプロジェクトに割り当て、その下に各種サブプロジェクトを配置している。

図 2.5.2 に全球モデル Redmine のトップページを示す。左側はよく利用されるページへのリンク集、右側はニュースの表示に利用している。

(3)　チケットの利用

開発における具体的な項目はチケットを用いて管理しているが、チケット利用のポイントはチケットを分類するカテゴリの使い方にある。例えば、モデルであれば GSM のバージョンごとにカテゴリを作成し、各開発項目がどのバージョンで完結予定かで分類している。カテゴリ名は、2017 年に完了予定の GSM 開発について「GSM17XX」と年単位で分けたり、全球 EPS の新規導入であれば「全球 EPS 導入」と開発の節目に

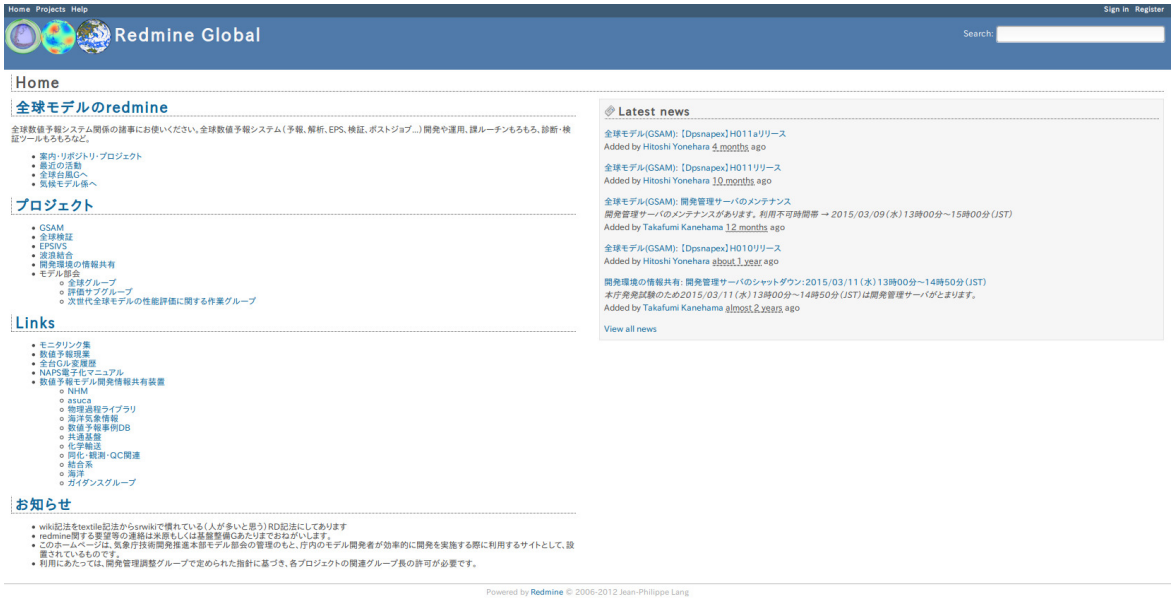

図 2.5.2　全球モデル Redmine のトップページ。

したりしている。具体的なスケジュールが未定のものは開発一般として分類しておき、次期版へ取り込みたい場合にはそのカテゴリへ変更する。ある時点で次期版での候補となっている変更内容・関連項目はカテゴリ別に一覧でき、開発者が相互に内容を共有しやすくなっている。

開発項目についてのチケットの粒度や書き方については特にルールは設けていない。大きいものでは「地表面摩擦の再考」など総合テーマが親チケットとなり、作業別に子チケットを多く持つものや、小さいものでは軽微なバグ修正など一作業に対応するものなどがあり様々である。ただし、実際には「GSM17XX における陸面モデルの改良」などのように、あるカテゴリに対応した大きめの開発項目ごとに作られるものが多い。チケットに関しては、カテゴリ設定が正確で、組み合わせ実験を行う時までに、組み合わせの単位となる変更内容が関係者に十分理解できるよう記載してあれば問題はない。

ステータスは「進行中」か「終了」、「却下」といったシンプルな分類を利用しており、ワークフローは用いていない。これは、バグ修正や機能追加といった状況が明瞭な項目以外の開発項目に関するチケットが多く、チケットの粒度は内容によって様々であるため、ワークフローを共通化しない方が良いからである。トラッカーはバグ修正、改良項目、話題 (TIPS) といった分類で大まかに利用しており、ほとんどの項目が改良項目に分類されている。日単位で管理する必要の無い項目がほとんどであり、緊急に対応すべきことも少ないため、開始日や終了日、優先度の項目を利用する必要性は生じていない。

個別の開発項目とは異なり、共有されるベースライン実験についてのチケット利用には一定のルールを設けている。夏冬の性能評価試験 1 セットに対して 1 チケットが割り当てられ、以下の項目が標準的な記載内容になる。

- ベースライン実験からの変更点
- ベースライン実験への変更内容の組み込み作業
- 内容のクロスチェック（レビュー）
- 実験のセットアップと実施
- 実験内容のレビュー
- 進捗の確認ツールのセットアップ
- 標準検証ツールの実施と結果のまとめ

変更内容が大規模で組み込み作業が多い場合は、その部分だけ別チケットにすることもある。ベースライン実験はカテゴリ名にバージョン番号を付けた名前で呼ばれる。例えば GSM1403 の開発では最初のベースライン実験が GSM1403v0 で、最終的には GSM1403v7 が現業運用版になっている。初回ベースラインには、現業運用版にその時の中心的課題の変更を加えたものが設定されるが、GSM1403v0 は鉛直層数増強（60 層から 100 層、モデルのトップを 0.1 hPa から 0.01 hPa）を中心として、放射過程や上部境界設定など関連する変更が適用されたものであった。

実際の利用例としては、GSM1403 カテゴリで 39 項目、GSM1603 カテゴリで 66 項目のチケットが立てられていた。

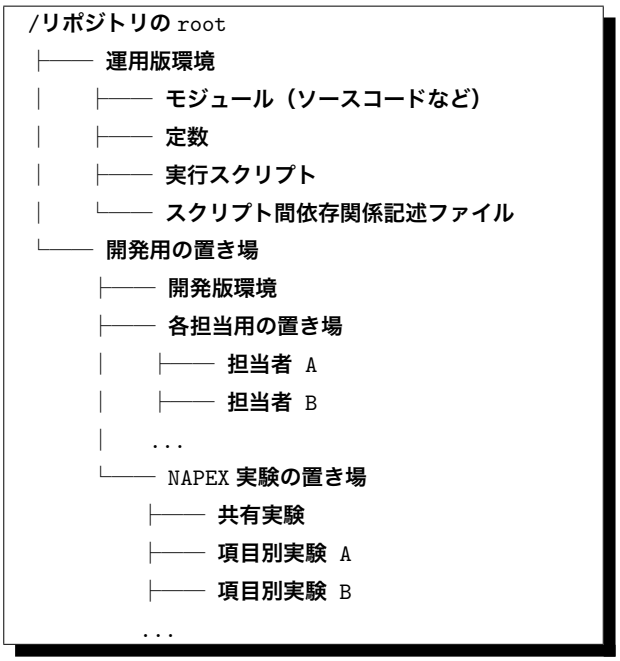

図 2.5.3 全球モデルリポジトリのディレクトリ構造。

(4) Wiki の利用

開発情報をまとめるため、カテゴリごとにポータルを Wiki で作成している。内容としては短期開発の大目標やスケジュールの記載、各種報告資料へのリンクの他に、組み合わせ試験の進捗や各課題の状況などがチケットへのリンクとしてまとめられている。現在は進捗管理の担当者が節目ごとに記載を追加しているが、記載漏れを防ぐ意味では作業の定型化をすすめて規則的に行うことも考えられる。

主な利用目的としては、定期的に開発作業に漏れがないかを確認することや、後で開発全体を振り返りやすくすることである。特に振り返りのための入り口を作成しておく価値は高い。最終的に採用された成果だけでなく、不採用となった項目や予測精度の悪化原因が十分に理解できなかった項目なども非常に重要な知見であり、振り返りが新しい示唆をもたらすことは多い。

(5) 実験内容の管理とリポジトリの利用

性能評価試験の実施には NAPEX（第 3.2 節）を利用しており、そのデータベースには全体の構成が記録されている。GSM の開発ではそれに加えて利便性向上のため、実験内容を構成するファイル群を数値予報モデル開発管理情報共有装置（以下、開発管理サーバ）の Subversion リポジトリ上で、GSM のモデルソースコード、周辺ツール類と同時に管理している。

GSM のリポジトリ構成としては、ソースコードだけでなく実験に必要なファイル全体を一つのブランチとして管理している点に特徴がある。そのディレクトリ構造は大まかには図 2.5.3 のようになっている。

ここで、「運用版環境」以下にある「モジュール」、「定数」、「実行スクリプト」、「スクリプト間依存関係記述ファイル[6]」のファイルセットが実験設定全体を構成する 1 単位である。これらのファイルは第 3.2.3 項 (2) で解説されている、数値予報ルーチン管理のツールに準ずる形式であり、NAPEX 実験にも対応している。この運用版環境へのコミットには開発コミュニティの承認が必要であり、原則として現業システムへ変更申請を行い、プログラム班のチェックを通過したものがコミットされる。

「開発用の置き場」の下には、「開発版環境」とそのブランチ置き場及び、「NAPEX 実験の置き場」が存在している。開発版環境とは、運用版環境に対してモニタ出力の追加や低水平分解能モデル、理想試験環境、サポートツールなどの開発用の変更を加えたものであり、最新の開発成果はまずそこへコミットされる。開発版環境へのコミットは、結果に影響のない修正についてはチケットなどへ報告したのち随時行われるが、結果に影響のあるものについてはミーティングなど関係者の議論と承認、コードレビューを得てコミットが行われる。それぞれの開発者は開発版環境を別ブランチとしてコピーして利用する。

基礎開発を行い変更内容が固まれば、次は性能評価試験へ進む。まずはベースライン実験を各担当者の実験の置き場にリポジトリ上でコピーしたのち、そこに開発項目についてのブランチから変更内容をマージすれば良い。こういった使い方を可能にするため、実験全体を構成するファイルセットを 1 単位としてリポジトリで管理している。図 2.5.4 にリポジトリの各ブランチの分岐と合流の流れを示す。この流れに沿って開発を行う限り、1 つのリポジトリ上に現業運用版、開発版、各開発者が試したもの、性能評価試験が行われたもの全てが記録されることになる。チケットの記載などと突き合わせて、過去の実験を再現したり内容を確認したりすることも容易である。また、この方式には NAPEX 実験間のマージにおいて Subversion の機能を利用できること、Redmine 上のリポジトリブラウザなど、リポジトリ可視化ツールを通じて組み合わせ実験の履歴や差分などを確認できることなどの利点もある。

(6) 開発を補助するツール群

効率的に開発を進めるために、共通して行う作業については補助ツールを整備して共有している。これらのうち、低水平分解能 GSM の実行環境とシングルカラムモデル[7] を含む各種の理想試験環境は、開発版環境に含めている。ツールの共有により開発効率を高めるだけでなく、実験時に単純なミスが発生するのを防止している。

低水平分解能 GSM としては幾つかの設定が用意さ

[6] 全球解析と全球モデルの実行を含むため、同時並列で実行される大量のジョブを含む。
[7] 鉛直 1 次元のモデルであり、何らかの外部境界条件の下でパラメタリゼーション手法の試験を行うもの。

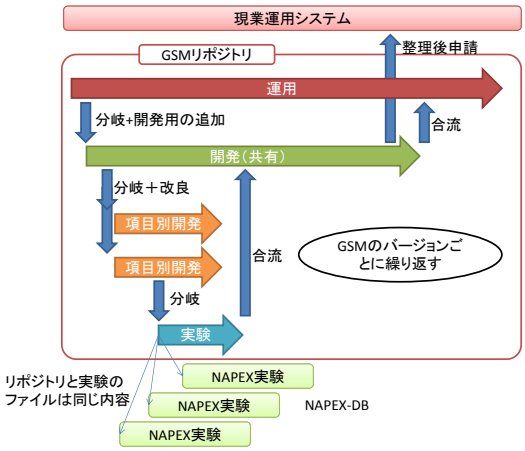

図 2.5.4　全球モデルリポジトリのブランチの分岐と合流の流れ。NAPEX-DB は NAPEX の実験情報を登録しているデータベースを指す。

れている。その中には、デバッグ用の TL31（PC 上で実行可能）、1 年積分実験や AMIP（Atmospheric Model Intercomparison Project）型実験の実施に用いる TL159、1 か月積分試験に用いる TL319、EPS のコントロールランに相当する TL479 が含まれる。鉛直層数は現業運用版と同じ 100 層で全て固定している。これら低水平分解能 GSM を含めて、GSM の試験実行ツール（TestHarness）が用意されており、スクリプト中の実験名と実験タイプなど数行を編集するだけで利用できる。TestHarness はジョブ間の依存関係や利用する実行モジュール、定数ファイル、初期値などを自動で判別する機能を持ち、環境をスーパーコンピュータ上に配置したのちジョブを登録・実行する。このツールを利用することにより、手元のファイルさえきちんと更新しておけば、実験を実行する計算機上へのコピー忘れや実験環境の構築ミスなどは防止できる。単発試験に必要な初期値（低水平分解能含む）は開発者で共通の事例を調べるため、性能評価試験の対象期間中で用意されているが、必要があれば現業運用データから望んだ日時の初期値データを作成するツールも用意されている。開発者は自分の課題の開発ステージや利用可能な計算機資源を考慮しつつ実験タイプを選んで基礎的な試験を進める。

結果を簡単に確認するための可視化には、画像作成スクリプトやウェブブラウザを利用した画像ビューアが用意されている。スクリプトを実行すれば海面更正気圧と降水量や上中下層雲量、対流圏の代表的な面における気温、湿度、高度場、風などの標準的な分布図が描画されると同時にそのビューアが設置され、手間をかける必要はない。より詳細に確認する場合は、TAG（第 4.6 節）を利用したリアルタイムモニタである DynaMo による描画が可能である。図 2.5.5 に DynaMo による描画例を示す。このモニタはセットアップ用のシェルスクリプトにデータの置き場所などを記載して実行するだけで利用でき、GSM から出力される要素を網羅的に水平面表示するだけでなく、実験間の差分や解析値に対する誤差、任意の点間の断面図も瞬時に表示可能である。また、ウェブブラウザ上からの操作で表示領域の変更や拡大・縮小、図の並び替えも行なうことができる。このツールは開発者が実行した予測実験だけでなく、現業運用 GSM の結果や性能評価試験の結果確認にも利用されている。ただし、DynaMo は動的なモニタであるためデータが保存されている間に限られる。この他、GSM の出力は気象庁独自形式である NuSDaS 形式（第 4.2 節）であるため、データを NetCDF など描画ソフトから利用可能な形式へ変換するツールも整備されており、必要に応じて GrADS などの各種描画ソフト（第 4.3 節）も利用している。

更には、リポジトリ上の単位ファイルセットから、NAPEX への実験登録を自動で行うツール (dpsnapex) が整備されており、円滑な実験セットアップを補助している。このツールは手元のファイル群から NAPEX へ登録すべき差分を自動で判別して設定ファイルを作成・登録する。利便性が向上するだけでなく、登録漏れが起こる可能性を排除している。NAPEX では実験名やデータ保存場所、保存するデータの種類など利用者が自由に設定できるが、dpsnapex では設定を簡素にすることによって利用状況が複雑になるのを防いでいる。NAPEX をそのまま使うより簡便なので、開発者は自然と dpsnapex を使い、結果として設定に秩序が生まれる。また、このツールを使うと GSM のリポジトリ上のどのパス、どのリビジョンの実験が登録されたかが、NAPEX のデータベースのコメント欄に自動で記載され、NAPEX 側と GSM のリポジトリの対応を明確にしている。

2.5.3　開発の進め方の実例

開発の進捗の流れの例として GSM1603 の開発履歴を紹介する。GSM1603 は 2014 年 4 月に開発が開始された。2014 年 4 月の全球・台風グループ開発会議で次期 GSM の開発予定案が提示されるとともに、GSM1403 の検証結果に基づいて、熱帯の対流圏中・下層を中心とした低温バイアスや台風の発生・発達の不足など、解決すべき課題が共有された。同時に、当時取り組んでいた開発項目の一覧が更新され、短期課題とする候補が選定された。このとき、新しい陸面過程の導入、新海氷スキームや開水面 (open water surface) と海氷の両方が混在する状態を扱う混在格子の導入、放射における雲量オーバーラップ手法の改良、雲粒の光学的特性の取り扱い精緻化、雲過程・積雲過程の大規模改良など多くの項目をリストアップしている。これら項目の多くは、GSM1403 での導入を目指していたものや、同時並行で開発が進められていたものであり、この時点で各項目のチケットは既に作成されていた。GSM1603 開発のカテゴリが新設され、それぞれのチケットはそ

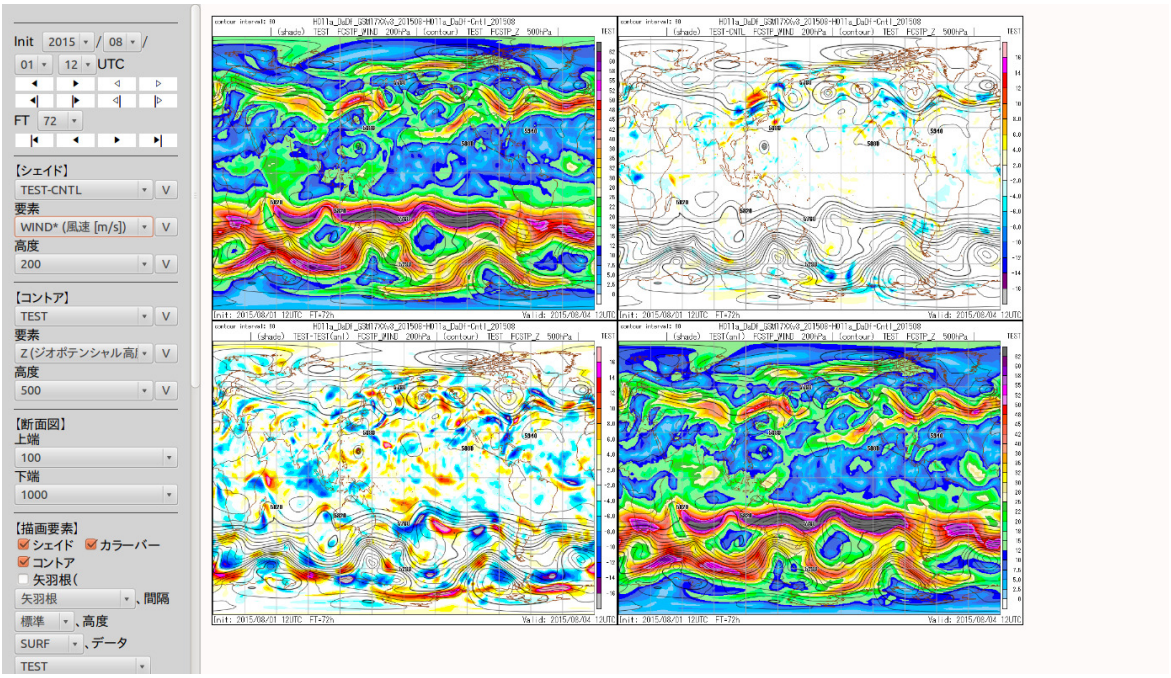

図 2.5.5　DynaMo の描画例。性能評価試験の結果について、等値線で 500 hPa 高度場、色の塗り分けで 200 hPa 風速を表示している。左上が予測値で、右上が対照実験との差分、左下が誤差（解析値との差分）、右下が解析値の図である。

のカテゴリに所属するように変更された。

2014 年 4 月以降、個別項目の基礎開発が進められるとともに、性能評価試験とその評価検証が行われた。性能評価試験の実施には dpsnapex を用いており、開発管理サーバ上で変更内容を共有する時には Subversion のマージ機能を活用している。評価検証では DPSIVS（第 2.5.6 項）を必ず実行し、ミーティングにおいて共通の物差しで議論するための資料にしている。この作業を繰り返してベースライン実験を更新していったのであるが、GSM1603 は最初のベースライン (v0) の構築に 1 年以上の期間を要し、2015 年 6 月に構成が決まっている。その v0 から始まり、v7 まで実験が行われ、最終的には v5 を現業運用版として採用している。この時に行った一連の議論で用いた資料はチケット上で共有しているため、詳細を振り返ることが可能である。

v0 は、GSM1403 に放射、雲、積雲、海面の諸過程の改良と、新しい陸面、海氷過程の導入を含む大規模な変更を施したものであった。v0 を構築するまでには大量の実験と議論が行われているが、地表面付近の低温バイアスに関する視点を中心にその流れを紹介する。

新しい陸面過程導入時の問題については前述したが、境界層過程の強安定時の熱の過剰輸送が GSM1403 で解決されるなど一部進展はあったものの、低温バイアスは依然として大きな問題であった。陸面過程自体もフラックス交換スキームや植生パラメータなど様々な改良と調整が試みられていたが、最終的な問題の解決には、海面・海氷過程と、積雪・土壌での長波射出率、雲氷落下スキームの 3 つの改良が大きく寄与している。

GSM1403 の海面・海氷過程、特に海氷のスキームには大きな問題があり、地表面付近の低温バイアスの原因となっていた。旧陸面過程は冬期に北半球高緯度で高温バイアスを持ち、両者でバイアスの方向が逆であったため表面化していなかったが、新陸面過程は低温バイアス傾向であり、相乗的に低温バイアスを拡大する状況になっていた。GSM1603 の海面・海氷過程では、開水・海氷混在格子、接地境界層における輸送係数の計算手法の改良、氷 4 層と表面を取り扱う新しい海氷スキームなどが導入されて、海面・海氷過程由来の低温バイアスが大幅に減少することが確認された。そのため、海面・海氷過程と陸面過程は一体の開発として扱うこととし、以降の実験は必ず組み合わせて行うことになった。

GSM1403 では地表面は黒体であり、長波の有効射出率は 1 として取り扱っていた。しかし、この近似は精度が良いものではなく、特に陸上では長波放射による冷却が過剰になる原因の一つになっていた。この点を変更するには、放射、陸面、海面、海氷の各過程をまたぐ改変が必要であり、これまで積極的な開発項目としては挙げられてこなかったが、低温バイアス対策のため各過程の担当が共同して改良を行っている。

雲氷落下スキームの改良は、そもそもは熱帯上層雲量が過少な問題と、スキーム自体が持つ時間積分間隔の依存性低減を目指したものである。この変更単独では熱帯対流圏上層の高温ドリフトなどの問題があり、積雲過程とセットで開発と調整が進められていた。一方で、この改良の結果、雲氷はよりゆっくり落下するようになるため、高緯度側でも雲量が増加し、地表面での長波放射収支が改善する。陸面過程の開発において

下向きの長波放射の過少は大きな問題とされていたため、積極的に組み合わせて試験を進めることになった。

上記を含めて多くの実験と評価検証、議論を行った結果、v0 の構成について開発者の認識が共有されたため、実験環境の構築を開始した。実験ブランチに各自の開発ブランチから変更点がマージされると同時にソースコードの変更点についてクロスチェックも実施しており、それらの構築作業に 2 週間程度、その後の実験実行にさらに 2 週間程度を要している。評価検証の結果、概ね想定されたインパクトが見えていることが確認されたが、一方で熱帯陸上の降水過剰、地表面への短波入射過剰に伴うユーラシア中央の地表面付近での高温バイアス、氷床での予測誤差の増加など、様々な問題が局所的ではあるが見つかったため、今後の課題に設定された。

次のベースライン実験となった v1 は、2015 年 8 月にその構成が決まっている。このバージョンでは課題への対応として、陸面過程のフラックス交換スキームの再調整、氷床アルベドの再設定などに加えて、陸上地表面への短波入射が過剰な原因の一つとなっていた、雲オーバーラッピングのパラメータについての再調整が取り込まれた。

2015 年 11 月に実施された v2 では、雲放射における雲粒サイズ診断と雲粒光学特性式を改良し、短波放射に対して水雲が光学的に厚くなる変更が取り込まれている。また、運用版とすることを念頭に、積雲過程の改良により増加した実行時間を運用上の制限の中に収めるため、ルジャンドル変換の高速化などが取り込まれている。変更内容に合わせて、正確な実行時間計測などの作業もチケットに記録されている。

2015 年 12 月から 2016 年 1 月の間に実施された v3 から v5 のベースライン実験では、台風の発達を抑制する幾つかの対策が取り込まれた。v0 の段階では、積雲・雲過程の改良により台風の発達がより強く表現されるようになっていることが確認されていたが、その後実験を進めるうち、過発達した台風周辺での計算安定性に大きな問題が存在することが判明し、その対策が緊急の課題となった。この時の作業と議論はかなり厳しいスケジュールで行われたのであるが、情報共有や実験の管理にはリポジトリとチケットが有効に利用されている。緊急時ほど進捗管理の正確さが求められるため、システマティックな開発管理の有効性は増す。また、このとき採用には至らなかった v7 も、その後問題点の修正や構成の整理が行われて次のバージョンのベースになっており、記録された内容は有効に利用されている。

以上 GSM1603 での実例を紹介したが、開発の履歴は Wiki やチケットの記載から辿れるようになっており、過去どのような点に問題意識が持たれ、どのような解決策が模索されたのかが確認できるようになっている。GSM1603 ではこの他にも積雲・雲・放射にまつわるものなど幾つかの compensating errors を巡って議論が行われている。一連の開発では、性能評価試験を実施した構成だけでも 50 以上の試験が存在しており、低水平分解能での試験や予報実験のみ行ったものも含めるとその数倍になる。これらの試験群は、思い付くものを総当たりで流れ作業的に試したわけではなく、モデル内の各プロセスの問題点を一つ一つ洗い出し、各開発項目を組み合わせた時に何が起こるかを想定しながら試験を進めたものであり、複雑な作業工程を伴っている。このスケジュールでの一連の開発プロセスは、開発管理手法の発展なくしては実現できなかったであろう。

2.5.4 ソースコードの管理

GSM のソースコードの管理方針やルールについて、その現状と狙いを解説する。

(1) リポジトリの利用

GSM のソースコードに対しては、2002 年から CVS (Concurrent Versions System) を使ってバージョン管理が行われていた。当時は、モデルの大きなバージョン更新ごとに履歴を引き継ぎながらリポジトリを乗り換えていた。2010 年 1 月から管理ソフトウェアを Subversion へ変更し、その後開発管理サーバ上へ引き継がれている。定数作成や格子変換を行う GSM の周辺モジュールに関しても、当初は実行モジュールごとに管理方法が異なっていたが、共有可能な Fortran モジュールは共有することとし、一括して GSM と共通のリポジトリで統一的に管理する方針に変更している (宮本 2009)。

GSM ではトランクという概念は明瞭には利用していない。開発ラインを束ねる意味では運用版 (ope) が存在し、共有される開発の先端は次期バージョンのブランチ (dev) が相当する。一連の開発終了時に dev がそのまま ope になるわけではなく、不必要な部分の絞り込みと現業運用モデルとしてのブラッシュアップが行われて差異が生じる。開発で膨らんだ冗長な部分の整理も兼ねて、次の dev は ope から分岐させて再構築している。

(2) システム間での住み分け

現状、最新の GSM をそのまま利用するのは、全球数値予報システムと全球アンサンブル予報システム (GEPS) の二つである。それ以外にも GSM を利用するものは多く存在するが、過去バージョンの GSM にそれぞれが必要とする改変を加えて利用しているものが多く、最新の GSM への追随までの期間は長い。このため、システム間の住み分けについては直接関連する 2 システムのみを考慮している。両システムでソースコードの開発時系列は同じであり、リリース時期が異なるのみである。通常は、まず全球数値予報システムへ新 GSM が導入された後、GEPS へ導入されるが、タイミングによっては GEPS が先行することもあり得

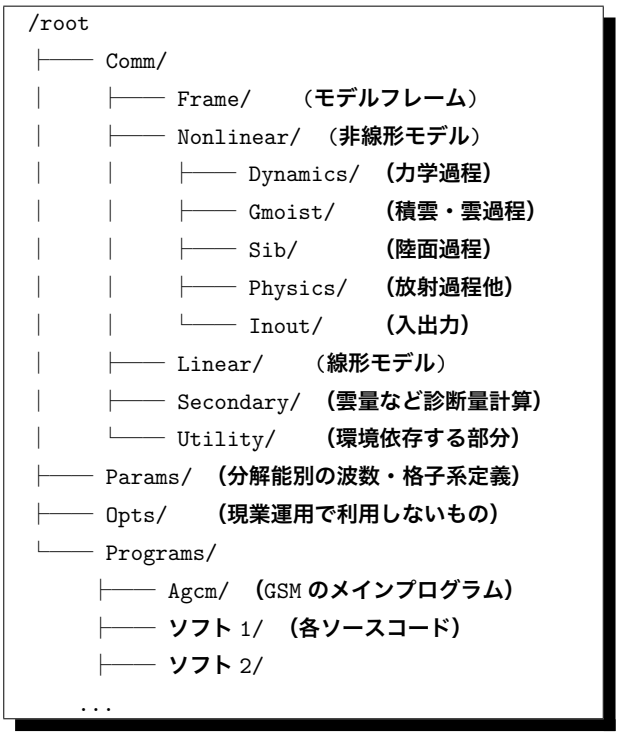

図 2.5.6 GSM のソースコードを格納するディレクトリ構造。

る。システム間の違いは、ネームリスト及び定数ファイルで制御するが、積分時間やプロダクト、モデルアンサンブル摂動に関わるものだけであり、時間発展演算には水平分解能以外に本質的な違いはない。

(3) ソースコードのディレクトリ構造

GSM のソースコードを格納するディレクトリ構造について解説する。基本的な方針としては、GSM の本体だけでなく関連ソフトウェア全体で分かりやすい配置になることを目指している。参考であるが、GSM1603 のソースコードは関連ソフトウェアを除いた本体部分のみでおよそ 10 万行、3.6 MB であり、このうち 3 割はプロダクト作成・出力部分である。図 2.5.6 に、現業運用版の構造の概略を示す。

Comm 以下に GSM のほとんどの部分（メインプログラム以外）が収められているが、これら以下の部分は変分法解析、格子変換、オフライン陸面モデルなどの別アプリケーションで共有することを想定しているためである。関連ソフトウェアのソースコードは、Programs の下のそれぞれの名前のディレクトリ以下に配置されている。

モデルフレームとは座標系、物理定数、通信用関数、球面調和変換用関数、時間情報等を一元管理する Fortran モジュール群のことであり (宮本 2009)、他のものとは別にまとめて配置されている。宮本 (2009) では、GSM0711 までの GSM では各過程の開発者が類似の変数を独自に利用しており混乱を招いていたが、それを教訓にモデルフレームの概念を明確化した上で整理を行ったことが報告されている。

Nonlinear の下には力学過程、物理過程、入出力など、大多数のソースコードが含まれる。力学過程は時間積分制御とコア部分、物理過程は放射・境界層・重力波のグループと、雲・積雲対流のグループ、陸面の部分に分けて配置されている。区分けからも想像できるが、ソースコードのサイズや行数について、陸面過程は物理過程全体の 6 割程度を占める。Inout にはモニタシステム及び、各プロダクト作成モジュールが配置されている。

開発版は現業運用版に理想試験用ソースコードなど様々な機能を追加したものになっている。ただし、現業運用版に含まれるものはほとんど変更せず、サブルーチンの呼び出し元が追加される形にする。これは開発版からの現業運用版を作る作業の時に、ソースコードの齟齬がなるべく発生しないようにするためである。

(4) GSM の実装に関するルール

GSM の実装について、採用されているルールを解説する。これらルールは、現業運用されるソフトウェアとしての品質を保つため、実装内容の維持管理を行いやすくすること、運用上の問題発生を抑制することを目的に定められている。関連するソフトウェアについても事情は同じだが、コード規模が小さいこともあり、GSM 本体ほど厳格ではない。

オプションは必要最小限とし、現業運用や開発で利用しないものは削除するのが原則である。開発で利用する場合でも、利用者が幅広く存在するものでなければならず、基礎試験で必須のものにほぼ限られている。仮にオプションを増やす場合は、必ず他のものを減らせないかを考慮する。また、特にプリプロセッサでの分岐についてはその利用方法を限定しており、I/O 周りの一斉切り替えと計算機・実行環境ごとの特殊な手当てのみに用いている。これは、プリプロセッサはその仕組み上、影響範囲が広がりやすい上に予想外の動作を引き起こしやすく、またコンパイラによるチェックが機能しにくいことが理由である。この方針はコードの可読性向上や想定外のトラブル防止には効果が高く、現業運用モデルとしては重要な方針である。一方で、開発時には不便な面があるため、開発版のブランチでは適用していない。ただし、運用開始時には整理する必要があるため、開発途中でもこまめに整理するのが通常である。

時間積分で引き継がれる予報変数は全体で共通管理とし、各モジュールが独自に持つことを許可していない。これは、時間積分やリスタートに関係する部分は分かりにくいバグの混入や各過程間の取り扱いの矛盾などトラブルが起こりやすいことから、各過程には任せないことにしているためである。また、必要な計算機資源を見積もるときに、入出力に関する部分の透明性を確保することが必要なことも理由である。ただし、この点については今後の GSM の高度化に伴って議論

が必要かもしれない。地表面過程などが多くのサブモデルを抱えた場合、独自に扱う方が簡便であるという意見はある。

ファイル入出力を行う部分は担当者の責任分解点を明確にするため、作成するプロダクトごとに一つの Fortran モジュールを用いることにしている。これは、かつて統一的な入出力機構を用いていた時代に、あるプロダクトのための変更が別のプロダクトへ副作用を起こしてしまい混乱が生じた反省に基づいている。ほとんど同じ処理であっても、ソースコードの共用は原則行わず、変更する可能性のある診断処理などは共用しない。ただし、内挿サブルーチンや一般的な気象要素変換など変更の見込まれない一部のものは共用している。入出力周りは現業運用や開発時に計算機利用上のトラブルが起こりやすいため、予測値の出力だけでなく、モデルのログ出力も全体で統一して共通に管理している。

(5) プログラミング作法

プログラミング作法（コーディング規約）については特定のものを採用していない。もちろん、無秩序ということではなく、

- Fortran90 の規格に従うこと
- 他人が読み、メンテナンスすることを念頭にソースコードを書くこと
- 現行ソースコードの慣習を尊重すること（全面改変は妨げない）
- モデルフレームで管理されている変数に類似した変数名は利用しないこと

といった基本的な事項は指定されており、またバグ混入を減らすための努力は当然求められる。抽象的な表現にとどまらず具体的に決めるべきという話もあるだろうが、コミュニティはオープンではなく限られた範囲に止まるので、不適切なソースコードが提案されたときにはクロスチェックの段階で指摘されるため問題にはならない。結局のところ、保守性、可読性や判読性をどうすれば確保できるかは文脈依存であり、汎用ルールで書き尽くせるものではないし、本質を理解していない作業者が細かいルールに従ったとしても、作業速度が落ちるだけで現業モデルとして品質の良いコードになるわけでもない。

この方針も、既存のコーディング規約にこだわりすぎて開発の効率が悪化した事例の反省に基づいている。現状ではコーディング規約を採用する予定はないが、採用する場合にはコーディング規約の改訂手順を明確化した上で、定期的なブラッシュアップの実施、確認コストを減らすための自動検査ツールの導入、柔軟な例外規定を定めて水掛け論発生を防止する工夫などが必要と考えている。

(6) ソースコードの整理とレビュー

あるバージョンの GSM の構成が決定された段階で、ソースコードが十分に整理されていることがもちろん望ましい。しかし、開発の節目で行われるクロスチェックやコードレビュー時にある程度整理されるとはいえ、開発の締め切り間際において時間的制約により対応を延期する項目も多い。例えば、必要なコメントの不足、変数の使い回しなど不必要な複雑さ、内容を想像しにくいファイル・サブルーチン名など計算結果や速度に影響のない部分は後回しにされやすい。そのため、新しいバージョンを立ち上げた時に、まずソースコード整理を行うようにしている。その他にも、他の計算機環境でのデバッグ結果がフィードバックされたタイミングなど、積極的に整理を行っている。近年では固定形式など FORTRAN77 までの制約に基づいて記述された部分についても整理が進み、次期 GSM ではほとんどの部分が Fortran90/95 以降では推奨されない文法を排除したものになる予定である。

(7) ドキュメント

GSM 及びその周辺プログラムに関するドキュメントは、開発管理サーバ上のリポジトリで管理している。リポジトリの構成には特殊なものは用いておらず、標準的な trunk, branches, tags 構成である。使用するファイルフォーマットは特に定めていないが、差分管理が可能である点や数式記述の利便性から LaTeX が用いられることが多い。

ドキュメントを作成するタイミングについて特に明確なルールは無いが、クロスチェックやコードレビューを受ける段階である程度のメモは作成しておく必要があるため、その内容が反映される GSM バージョンのソースコード凍結までには最低限のものは作成されている。GSM 全体のドキュメントも簡単なものが整備されているが、その更新はやや滞りがちであり、開発の進捗とドキュメント作成をどのように無理なく紐付けるかは今後の課題である。

2.5.5 開発の進め方のまとめと課題

ここでまでは GSM の開発管理について、その背景、進捗管理、プロジェクト管理ソフトウェアやリポジトリの利用、ソースコードの管理方針について紹介した。現在の手法は、GSM 全体を「パッケージ」として効率的に開発することに重点が置かれている点に特色がある。これらの開発プロセス、開発基盤環境はコミュニティ全体の開発効率の向上に貢献しており、今後もブラッシュアップしながら利用を継続することが重要と考えている。

今後の課題であるが、大きな枠のものとしては GSM の中長期的な開発課題と短期的な開発をどうすり合わせていくかという点がある。本節で解説したように、現在の開発の進め方は 1, 2 年単位の短期的な開発に特化しており、中長期的な開発課題については考慮され

ていない。今後この点について考え方を整理していく必要があるだろう。

開発項目の選定についても課題がある。研究コミュニティにおける自然科学の発展や他の現業センターの動向などを背景に、モデルの既存の問題点とユーザからの要望を整理し、対処するためのアプローチとその進捗を整理した上で行うのが理想的である。現状ではその部分が十分システマティックではないため、議論の進め方や共通認識の形成手続きなど工夫の余地があるように思われる。ただし、私見ではあるが、GSM は気象庁の基盤モデルとして多くの役割を担っているため、ユーザからの要望に優先度をつけるための議論には多くの困難があると思われる。GSM は短期予測から中期・延長予測まで様々な時間スケールを予測対象としている。数日先までの台風予測精度の向上など、日本の防災に直結する項目の優先度が高いことは言うまでもないが、その他にも様々な要望に対してどのように優先度をつけて開発（評価）を行なうかは非常に難しい問題であろう。複数の指標を何らかの整理に基づいた重みで組み合わせて、統一的な数値指数 (Index) を作成する方法なども考えられるが、整理をつけることは非常に困難であるし、数値指標の短期的な向上だけを過剰に重視した短絡的な開発に陥ってしまう危険性もあるため、慎重な検討が求められるであろう。

現在取り組んでいる課題として、実験の評価検証の標準化と拡充がある。開発フローの中で、性能評価試験へ至る前段階では、共有の実験・評価が決まっていない。もちろん、その部分で行うべきものは開発項目に強く依存するので、一律のものが定められるわけではないが、幾つか共有の実験設定、特に気象予測実験だけでなく気候予測実験が用意されているべきという議論がある。その一つとして、現在低水平分解能モデル (TL159) での一年積分を評価する実験・検証環境の作成や、AMIP 型実験・検証環境の整備が進められている。

また、気象予測実験についても直近の夏冬だけを対象にするのではなく、再解析を初期値とした実験を行い、GSM の改良内容のインパクトが、エルニーニョ・南方振動やマッデン・ジュリアン振動などスケールの長い現象の状況に依存しないかの確認を行なうべきといった議論や、事例依存性の強い台風進路予測などの現象については低水平分解能でもっと多くの実験を行うべきといった議論がある。最終的には利用できる計算機資源と開発へのフィードバックのバランス次第であるが、検討を続けていく予定である。

2.5.6 検証と検証システムのあり方

ここからは、GSM 開発における検証について、そのあり方と現在の標準検証システム DPSIVS を、特に開発管理の観点から紹介していく。

(1) DPSIVS

DPSIVS は、性能評価試験を主な対象とする、全球決定論予測についての標準検証環境である。近年の GSM 開発においては、複数の改良項目を組み合わせて、compensating errors を解きほぐしながら改良を進めることが鍵となっている。その開発プロセスでは、変更項目を色々と組み合わせた実験を実施するが、実験ごとに結果の評価検証と分析、担当者による議論を通じたフィードバック、次の実験の設定といった手順を踏む。これを繰り返すサイクルにおいて評価検証と分析の作業量は大きなウエイトを占め、検証システムが効率的であることは開発全体を効率的に進めていく上で大切なポイントである。DPSIVS は性能評価試験を主な対象としたツールであるため検証においては、変更された実験 (TEST) を、対照実験 (CNTL) と比較することが基本になっている。

(2) DPSIVS 開発の経緯

DPSIVS は、全球モデルについての評価検証への問題意識を基に 2013 年 3 月から開発を開始している。

GSM 検証ツールの歴史を大まかに振り返ると、初めは開発者が各自・各グループで独立に整備・利用しており、開発者ごとに利用するツールが異なる状況であった。その状況から、山下 (2013) で紹介されている全球実験標準評価モニタ、通称 SPV (Super Verif) が 2004 年に当時の検証ツールを取りまとめる形で整備され、必要最低限の項目について指標・図表の統一がなされた。SPV 作成時の問題意識は、最低限検証すべき項目の標準化や、検証モジュールを統合することによる信頼性の向上、ユーザの利便性向上などが挙げられる。SPV が整備された恩恵は非常に大きかったが、その後幾つかの問題が意識されるようになった。問題の一つが、SPV の検証内容は必要最小限 [8] であり、総合的な評価を目的としたものではなかったが、当初の想定を超えて開発における評価指標として過度に重視された点であった。本来、評価検証は様々な面から行われるのが望ましいが、過去には試験の検証を利用しやすい SPV で取り扱える評価指標のみで行うなどの例も散見された。そこで、問題意識を持った開発者により、補完的な追加検証のツールが複数作成されたが、それらの統合には至らず、開発者ごとに利用するツールが異なる状況であった。

一方で、気象庁内では、2010 年頃より全球モデル開発のための評価検証について議論が深まりつつあった。2010 年 1 月以降、全球モデルの誤差特性について議論する全球評価会合が数値予報課と気候情報課共同で定期的に開催され、モデル開発のための評価検証を充実させていくことの重要性が確認された。その流れは数

[8] どのような内容かは山下 (2013) に解説がある。予測スコアとしては、海面更正気圧や 500 hPa 高度、850 hPa 気温などの要素のアノマリー相関係数差や平方根平均二乗誤差の改善率が主な対象となっている。

値予報課報告・別冊の第 58 号、第 59 号にも繋がっており、特に第 59 号では「モデルの総合的な検証を通した物理過程の開発」が主要テーマの一つに掲げられ、気象庁内の評価手法についてのレビューや新しい検証手法の紹介が行われている (原 2013)。

このような状況のもと、新たな評価検証手法を取り込んだ統合的な検証環境の整備を目指して DPSIVS の開発は始まった。拡張性を始めから考慮して設計し、統一的な検証環境を提供することにより、開発者が利用するツールを共通化することも目標の一つであった。現在でも、数値予報課および気候情報課の開発者が連携しつつ、試験結果を検証する標準検証環境として開発・整備が進められている。

2.5.7 DPSIVS の考え方

ここでは、GSM の開発を効率的に進めるために、DPSIVS がどのような考え方を採用しているのかを紹介する。開発効率を高めるには、手間を減らすことと評価結果から得られる知見を増やすことの両面が求められる。

(1) 目的

GSM の開発は科学的方法に基づいているが、科学的方法には検証プロセスが不可欠である。検証結果は実験に適用された開発成果の正当性を示す資料であり、問題点の把握を通じて次の開発方針を決める材料にもなる。開発者は検証プロセスを通じて予測結果へのインパクトを適正に把握し、更なる改良のための調査も行う。同時に、検証結果は数値予報の利用者へ GSM の変更内容や改善の効果を説明するための材料にもなる。GSM は現業用途の実用モデルであり、開発者は利用者目線での評価検証を最終的には行い、改善内容について説明し理解を得る責任がある。

一方で、評価検証に対するニーズは開発する側と利用する側の両者で必ずしも同じにはならない。この意味で、検証には大きく 2 つのカテゴリが存在すると言えるが、DPSIVS では開発者の側に重きを置いている。その理由の一つとして、性能評価試験を実施する段階はまだ開発途中であり、開発する側のニーズがより重い点がある。無論、検証ツール単体としてみるとより多くの利用目的に適うことが良いのは確かである。ただし、DPSIVS は開発者自らが開発・保守を行っているので、ツールの目的を広げると管理コストが増える点には注意が必要である。現業運用版 GSM を変更する回数や、業務化試験（第 1.2.4 項）を実施する回数は性能評価試験より遥かに少なく、求められる図表もその時々の事情によって異なるため統一的なものは難しい。そのため DPSIVS では開発のためのツールであることに集中し、業務化試験については動作するもののサポートの対象外としている。DPSIVS の検証内容にはモデル開発者の今後の開発へのフィードバックが重視されるべきで、その結果により開発者の理解が深まり、より多くの気付きが引き出されるものが望まれる。

(2) 管理方針

DPSIVS 全体の構造としては、独立した検証ツール群の集合体であることを基本としている。これは、それぞれ独立に存在していた検証ツールをまとめる所から始まったという開発経緯も理由であるが、DPSIVS 自体の改良と維持管理コストを複数の担当者に分散させることが狙いである。また、管理コストを維持可能な範囲に収めるためには、需要が低下したツールをフェードアウトさせることが必要であるが、その基準としては維持管理の担当者として手を挙げる開発者がいるかいないかを用いている。GSM の開発は検証と一体不可分であるので、有用なツールは開発者の手により維持されている。

各ツールは完全に独立に実行できるよう、実行時の依存性は排除しており、ファイルフォーマット変換や格子系変換などの一部共有モジュールを除いて、一つの変更がツールの枠を超えて影響することはない。図 2.5.7 に DPSIVS により作成される結果閲覧用ウェブページのトップ画面を示す。ここにリストされている項目が、それぞれの検証ツールに対応している。DPSIVS ではそれぞれの検証ツールをパッケージと呼んでおり、それらパッケージの集合体に閲覧機能やパッケージ全体を実行するためのツールを加えたものが DPSIVS 全体を構成している。

ソースコードと定数ファイルの管理には数値予報ルーチン用のツールを利用している（第 3.2.3 項）。ルーチンのディレクトリ規則を踏まえると、パッケージごとに親ディレクトリが分かれ、ソースコードや定数の利用状況が分かりやすい。NuSDaS から NetCDF への変換ツールや描画における標準色設定など、共有されるものは Comm という名前のディレクトリ以下に配置している。一方で、JCL（第 3.2.3 項）は利用せず、シェルスクリプトをそのまま利用している。これは、検証処理ではシェルスクリプト内での分岐や繰り返しといった制御が多く使われており、JCL のステップ記述の利用があまり馴染まないためである。仮に利用したとしても、一つのジョブが一つのシェルスクリプトを実行するステップのみを持つ場合が多くなり非効率である。パッケージ内には実行時の依存関係をもつ複数のジョブが含まれるが、その記述にはパッケージの開発者がテストを行いやすいように考慮した結果、スーパーコンピュータシステムにインストールされている LoadLeveler（第 2.10 節）のジョブ記述ファイルを用いている。DPSIVS からジョブを実行する時には、依存関係を解釈し独自制御機構で並列数を制御しつつ順番に実行している。DPSIVS の開発と保守には、開発管理サーバ上のリポジトリおよびチケットを利用している。リポジトリの構成には特殊なものは用いておらず、標準的な trunk，branches，tags 構成である。各

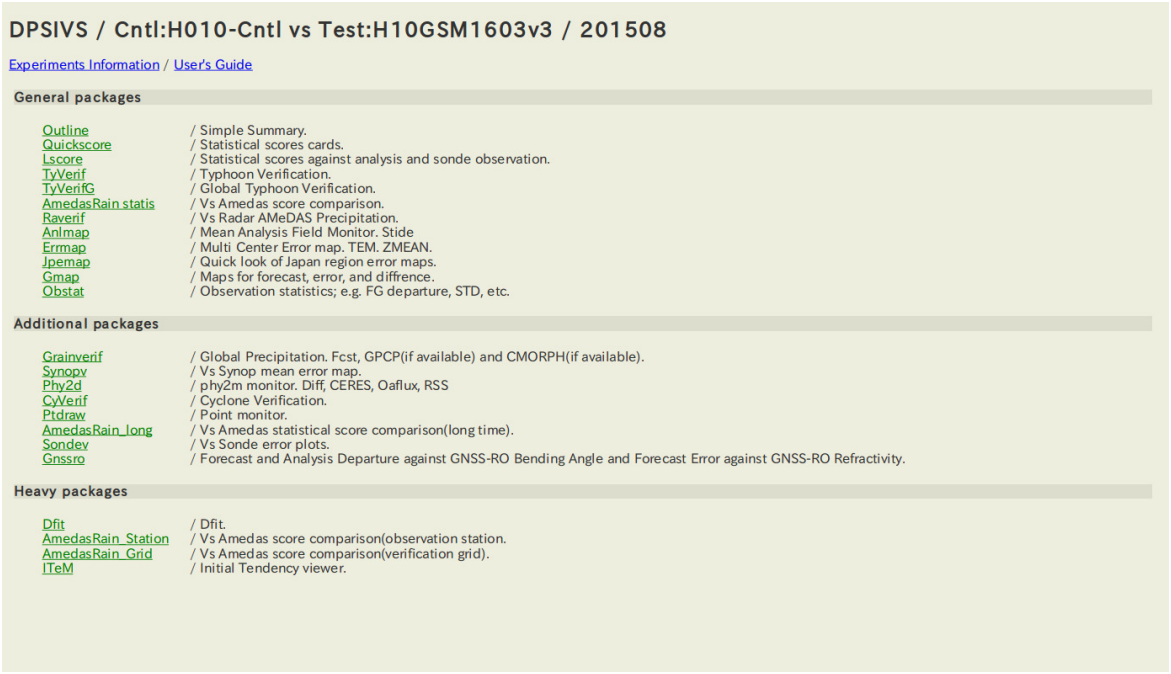

図 2.5.7　DPSIVS の実行により作成されるウェブページのトップ画面。検証パッケージごとに分類されている。

開発者は変更内容をチケットに記載した後、ブランチで作業を行い、管理者がトランクへマージする。ユーザードキュメントや開発者ドキュメントには Wiki を用いており、全球決定論的検証サブプロジェクト上に配置されている。パッケージの内容は各担当者が基本的には自由に決めるものであるが、移植性なども考慮して、作業ディレクトリや環境変数 PATH の設定、利用する外部ツールには制限を設けている。また、ツール内の絶対パスの利用は原則禁止であり、リファレンスデータのパスなども環境変数を通じて利用する。この方式により、例えばパッケージごとに利用する Ruby のバージョンが異なり、移植性に問題が生じる事態などのリスクを減らしている。それらの設定が記載されたテキストファイルが実行時の最初に作成されるため、各パッケージのシェルスクリプトには冒頭でそのテキストファイルを取り込むことが義務づけられている。

DPSIVS の実行はスーパーコンピュータの計算ノード以外のサーバで行うことにしている。他の検証ツールは必ずしもそうではなく、SPV のスコア計算や業務化試験の検証ツールでは計算ノードを利用するが、計算ノードを並列の大型計算が優先的に利用できるように DPSIVS では利用を避けている。また、性能評価試験の解析・予測の全てが終了してから一括で検証を実行することを基本にしている。これは、GSM の 11 日予測の検証において解析値や品質管理済み観測データを利用するためには、それらのデータを作成する解析ジョブが 11 日先まで終了している必要があり、予報ジョブの実行に合わせて検証を行おうとすると、解析ジョブと予報ジョブの実行タイミングに依存性を持ち込んでしまい、NAPEX（第 3.2 節）の利用上好ましくないためである。

(3)　総合的・網羅的な検証の必要性

開発者へのフィードバックの点では、まずは変更した項目に対するインパクトを幅広い観点で確認できることが重要である。自身が変更した内容に対して期待するインパクトが見られるかを確認することは当然だが、想定していない副作用がどの程度現れているのかも十分に把握しておく必要がある。

モデル内の各プロセス間の相互作用は複雑であり、基礎的な調査を十分行った上で科学的に正しい変更を加えたとしても、必ずしも予測結果の精度が向上するわけではない。大気モデルに限らないが、近似を含む方程式系の一部分を改良・精緻化しても、解の精度が向上するとは限らない。また、理想実験で良好な結果を得られていたとしても、理想実験が現実にあらわれる全ての状況を網羅することは不可能であるし、そもそも開発者が何か誤りをおかしている可能性もある。このため、試験結果を検証して副作用を確認することは開発における必須事項である。検証ツールの実行結果を一通り調べることにより、予測精度上の顕著な問題点はなるべく洗い出されていることが望ましい。

また、台風進路予測誤差など、現業運用に向けた重要指標に関する項目については最低限確認しておかないと、最終的な業務化試験で見過ごせない問題点が発覚するケースが増加し、開発における大きな手戻りが問題となり得る。これらの要請に応えるために、DPSIVS では総合的・網羅的な検証を行う幾つかの中核パッケージを用意している。中核パッケージ群は、SPV の検証内容や台風検証、降水検証などこれまで伝統的に行わ

れていた検証内容を含むだけでなく、多くの新規検証項目を取り込んでいる。一方で、網羅的であることと検証の処理時間はトレードオフの関係にあり[9]、バランスをどう取るかが問われる。実行にあまりにも時間がかかってしまうと結局利用されなくなってしまい、網羅的な検証が行われなくなる。この問題に答えはないが、自分たちで利用しながら妥協点を探っているところである。DPSIVS の実行は開発に利用している業務用サーバで行い、全体の実行にはおよそ半日程度の時間がかかる。ただし、実行が終了したパッケージから順にトップページへのリンクが張られ、数分で終わるものから順番に見ていけばあまり待ち時間を意識することなく利用できる。

(4) 効率的な利用

幅広い観点で様々な検証を行うには、大量のデータ・スクリプトのハンドリング技術が要求される。そのため、検証を少ない作業量で効率的に行うためには、検証ジョブを統合する環境整備が大切になってくる。特に、2 つの実験間を比較する場合には、組み合わせ実験の回数に応じて検証ツールの実行回数も増加しやすいため、せっかく検証ツール群が整備されていても、利用するために設定する項目が多く、時間と手間がかかってしまえば、大量の実験を評価することは難しくなる。

また、検証結果を評価していく時に、結果が網羅的でありつつも要点を絞って表示されていることや、ビューアなどでの閲覧のしやすさも開発効率を高めるには必要な要素である。開発者が確認に利用できる時間は有限であり、せっかく多くの検証図が作成されていても、見て分析・理解して知見を得なければ意味はないため、閲覧などの時間と手間は節約されるべきである。

DPSIVS では実行のためのエントリースクリプトを用意している。スクリプトは Ruby で記述されており、各開発者は標準のパスからスクリプトをコピー、必要な箇所を書き換えて実行する。最低限書き換えが必要な箇所は図 2.5.8 のような、TEST と CNTL 実験それぞれの実験番号・枝番[10] と図に表示する名前、検証対象期間である。

実験データの保存場所や、予測実験と初期値に用いた解析実験の関係などはエントリースクリプト側で NAPEX のデータベースから必要な情報を取得しており、利用者はそれを意識せずに必要最小限の設定だけで利用できる。

実際に実行する検証項目についても選択行をコメントアウトすることで取捨選択できるが、通常は全てを

```
# TEST 実験番号・枝番・略称
EXP_NO=9845 ; EXP_SUBNO=2
ABBR_TEST="GSM17XXv3"

# CNTL 実験番号・枝番・略称
EXP_NO_CNTL=9554 ; EXP_SUBNO_CNTL=2
ABBR_CNTL="H011a-Cntl"

# 検証対象期間（YYYYMMDD）
VERIF_SDATE=20150801
VERIF_EDATE=20150831
```

図 2.5.8 DPSIVS を実行するためのエントリースクリプトの例。書き換えが必要な部分のみを抜き出している。

実行して問題はなく、むしろ意図しない変更の副作用を把握するために網羅的な検証を行うことが推奨されている。DPSIVS には簡素ではあるが独自のジョブスケジューラが実装されており、順次 LoadLeveler を始めとする複数のジョブ管理ツールへのバッチジョブ投入を行う。ツールの管理コストを抑えるためには、第 3.2.3 項 (4) で紹介する ROSE を用いることが望ましいかもしれない。しかし、検証ジョブは様々なサーバで実行する可能性があり、ROSE でサポートされない環境も考慮して独自実装としている。今後のサポート状況によっては ROSE 利用へ切り替えることも念頭に置いている。

エントリースクリプトを実行すると、作業領域のパス、結果が保存されるパス、ウェブモニタへの URL が示される。ウェブモニタには、検証が終わった検証パッケージへのリンクが順次追加されていく。

ブラウザから利用するビューアには、共通で利用できるもの（図 2.5.9）を用意している。このモニタツールは HTML と JavaScript で実装されており、画像が入ったディレクトリの横に HTML ファイルを配置するだけで機能し、ファイル名の規則を解析して日付や要素名のリストを自動作成するビューア（リストの展開機能付）を提供する。検証パッケージの開発者はファイル名に気を遣うだけでよく、自分で HTML や JavaScript のファイルを作成または更新する必要はない。利用者もシンプルではあるが必要十分な機能を備えたビューアを様々な用途に対し一貫して利用できる。

(5) 知見の蓄積と拡張性

検証の実施者には、結果を評価する上で多くのことが期待される。例えば、次の開発へより良いフィードバックをもたらすには、モデルの変更点により生まれた予測結果の変化について、モデルの中で何が起きているのかを把握し、その差異を正しく理解することが求められる。リファレンスとする観測データや解析値に対しても、その誤差特性を理解し、検証結果につい

[9] 実行時間を短くするには検証ツールの高速化も重要である。ただし、高度な高速化を行うには開発コストが必要であり、こちらもトレードオフの関係にある。

[10] 記載するのは予測実験のもののみであり、対応する初期値を作成したサイクル解析実験の情報は NAPEX のデータベースから取得する。

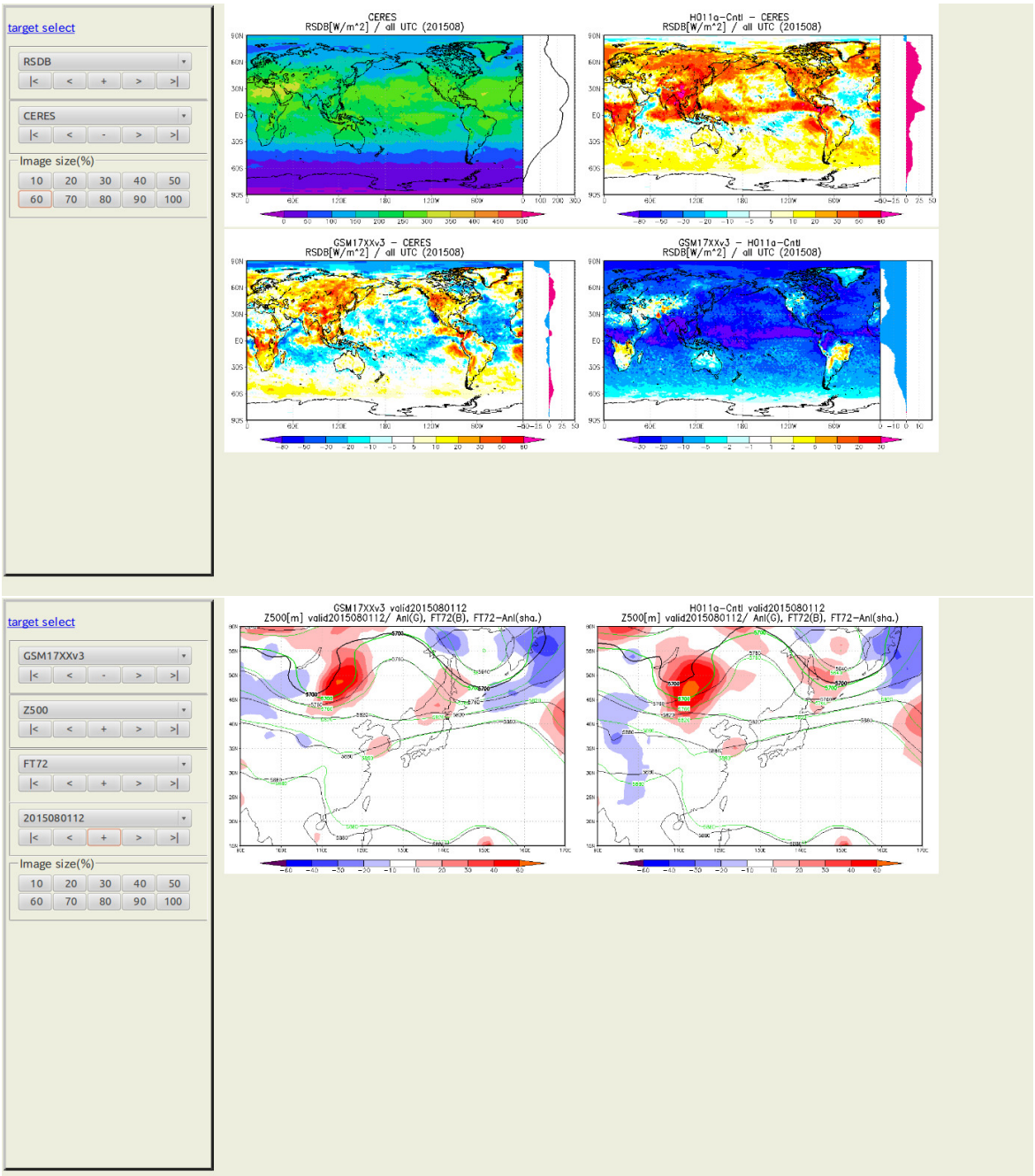

図 2.5.9　DPSIVS に装備されているビューアの表示例（上図と下図の 2 例）。自動リスト化とリストされた項目ごとの同時表示機能、画像表示の拡大縮小、キーボードショートカット機能を持つ。上図は phy2d パッケージの対 CERES 検証（後述）で、RSDB は地表面での下向き短波放射の月平均値を意味し、参照値、TEST と CNTL それぞれの誤差、TEST と CNTL の差分を表示している。下図は jpemap パッケージの 500 hPa 高度場の誤差表示で、黒線で予測値、緑線で解析値、塗りつぶしで誤差を表示している。（パッケージについては第 2.5.8 項を参照）

てどの程度の確からしさで議論が可能であるかに注意しなければならない。また、検証結果の統計性についても、統計学の知見を誤用せずに議論を進める必要がある。

検証ツールの立場から開発を助けるためにどのようなことが出来るであろうか。もちろん、開発者に代わって考えることなど出来ないし、結果を先回りして必要な図を作成することも不可能である。しかし、誰かが切り開いた道をフォローすることは可能であり、検証システム自体を評価検証の知見を蓄積する場と定義することはできる。評価検証における知見は、議論や開発における気付きにより蓄積されていくものである。実際の開発でも、モデルの予測に問題点が見つかり、その原因となるプロセスを議論した結果、新しく注目すべき量や指標などが提案されることはよくある。そのような知見に基づく新しい検証方法が検証システムに

追加されることによって、検証システム自体も成長していく。

そのため、新しい検証が将来追加されることを前提にして、柔軟性を持ちつつ管理コストが膨らみにくいソフトウェアとしての構造が求められる。DPSIVS がパッケージの集合体であることは、拡張性の面でも理に適っている。実際、最初のバージョンには現在の半分程度しかパッケージが含まれていなかったが、その後の開発において新たに必要となった検証ツールを新パッケージとして順次取り込んでいる。例えば、非地形性重力波スキームの新規導入に伴い中層大気を検証する目的で、GNSS 掩蔽観測を用いた検証が追加される、台風だけでなく全球的な熱帯低気圧へ着目することの重要性が認識された結果、熱帯低気圧検証のパッケージが追加されるなど、開発上の必要性から新しいパッケージが複数作成されている。DPSIVS は評価検証の知見を蓄積する場として今も成長している。

(6) 標準とカスタマイズ

評価検証を行うことは幅広い知識と深い思考を求められる作業である。そのため、利用者の思考を妨げる要因はなるべく排除すべきであり、混乱の原因が少ないことは大切である。図の描画に関する不整合や、同じ名前の指標を計算しているが微妙に手法が異なるといった手法の非一貫性、検証内容についての背景説明がない、論文などの解説資料へのアクセスが担保されないなど、利用者に無駄な注意力や知的負荷を要求する点はなるべく減らしたい。DPSIVS では、検証内容の説明には Redmine 上の Wiki を利用して情報共有を行っている。また、開発者一人の理解には限界があり、実験結果をより深く評価するには、検証結果をまとめ、ミーティングなどで議論することが大切になってくる。議論の立て方や資料のまとめ方などが重要ではあるが、同時に、検証結果の図表の視認性の高さ、理解しやすさは効率的な議論に欠かせない。多種大量の図が出てくる中で、図の説明を逐次していては時間が足りないので、色の使い方や目盛の間隔などが統一的であると良い。例えば、TEST と CNTL を示す色や、差分の色が図ごとに整合していない資料を考えると効率が悪いのは想像がつくだろう。検証図にある程度広い要望を満たせる統一的な基準（標準）が存在することの恩恵は大きい。そのため、DPSIVS では表示する要素やカラーレンジについてはなるべく可変でないものを推奨し、色の使い方に対する標準的なガイドラインを簡単ではあるが定めており、GrADS（第 4.4 節）上で利用できる標準カラー設定スクリプトなども提供している。また、パッケージ間で実験名の呼び名が異なったりしないように、環境変数を通じて一貫した値を提供している。ただし、最終的にはパッケージごとに事情があるため、判断は担当者に任せている。

一方で、変更内容によってどの程度のカラーレンジでの議論が必要なのかは異なる。ある要素の差が全体的な精度への影響としては無視できる範囲であったとしても、変更内容から想定される狙ったインパクトである場合には重要な注目点になる。このため、必要に応じて利用者が図を簡単に調整できることが望ましい。その設定がパッケージの奥深くに存在すると利用者では改変できなくなるため、描画する要素やレンジを記載したテキストファイルは Const の Param ディレクトリ以下に配置して改変を行いやすくすることを推奨している。設定変更が必要となった利用者は、自分で DPSIVS 環境をリポジトリから取得して書き換えて利用することが可能である。設定ファイルの内容はパッケージごとに異なるが、例えば図 2.5.10 のように単純なものが多い。

```
#element level int(shad) int(zmean) mask
PSEA 1013   1  0.5   0
T    1000   1  0.5   1
・・・(中略)・・・
Z     500   2   10   0
・・・(中略)・・・
PSI   850   1  0.5   0
PSI   200   1  0.5   0
```

図 2.5.10 描画する要素やレンジを記載したテキストファイルの例。これは平面で平方根平均二乗誤差を比較するための設定ファイルで、描画要素と誤差のレンジタイプ、気圧面が地面の下に潜る場合の特別処理の有無を毎行記載するだけのものである。

(7) 統計検証

DPSIVS が統計検証手法について採用している方針を解説する。モデルの精度向上を確認するためには、数値実験を実施しその結果を検討するのであるが、数値実験には多くの計算機資源が必要であり、精度向上を確信するのに十分な事例数の実験を通常は行えない。そのため、ある程度小さい事例数（通常 30 程度）の試験結果を検討することになる。つまり、地球のレプリカが取り得る状態分布に基づいて実験を行った場合（全数調査、母集団）の結果を、限られた事例数での結果（標本）から推定する必要がある。小さい標本について誤差の平均値などを比較する場合、推計統計学に基づかないと容易に結論を間違ってしまい、議論は往々にして多くの主観を含む非科学的なものになる。統計検証を十分注意深く行うことは実験の結果の分析において必須であるため、検証ツールでも十分に考慮されている必要がある。

統計検証には様々な手法が確立されているが、予測誤差については母集団の確率分布が何に従っているのかは明確ではないと思われる。また、利用者にとっては推定手法が明確であり、その特性を理解した上で誤解のないように使えることは大切である。そこで DPSIVS

では、成立していない可能性のある前提条件を利用者が意識せずに用いてしまうことを避け、また背景が整理されないまま手法が乱立することを防ぐため、必要な仮定が少ないシンプルな手法を全体で共通して利用することにしている。具体的にはブートストラップ法 (Efron and Stein 1981) を採用し、それを行う共有モジュール、サブルーチンを提供している。ただし、データ同化の領域では正規分布がある程度成り立っていると考えられるため、関連する検証には特に制限を加えていない。また、エラーバーの意味を明瞭にするために、95%信頼区間を採用することを義務付けている [11]。もし、複数のエラーバーを表示したい場合は、68%, 95%, 99%を利用することを推奨し統一化を図っている。

しかしそもそも、統計検証の解釈に関しては本質的に難しい問題がいくつか存在している。例えば、我々が本当に知りたいのは未来の現業運用において精度が向上するかであって、過去の期間から抽出した標本は条件が偏っていないといえるのか不明なことが挙げられる。特に、性能評価試験は特定の年の夏と冬を対象に実験を行なっているが、例えば熱帯海上のエルニーニョ・ラニーニャの状況によって、積雲対流過程変更のインパクトは異なって見えたりしないのか、といったことである。性能評価試験の構成では、無作為抽出の前提は疑わしく、偏った標本によりどのような評価の違いが生じ得るかに関する知見は現状では十分にあるとは言えない。

過去の期間を対象とした統計的な推定が示す有意性を、未来の運用での改善の可能性と安易に結びつけることは危険である。未来における再現性について我々は物理法則の普遍性を信じるのが原則であり、観測的事実に基づいて改良内容を検討し、変更が GSM 内にどのような変化をもたらすのか、モデル内のプロセスを十分に調査した結果がまず先にあるべきである。統計検証は、その結果を用いて変更内容の正当性を補強するための材料にするのが基本であり、統計検証自体が正当性の根元になるわけではない点には注意が必要であることは、強調しすぎることはないであろう。モデル改良の成果はどのような実験によって確かめられるべきなのかについて、今後とも調査・議論を通じて理解を深めていきたい。

2.5.8 DPSIVS の各パッケージ

DPSIVS に含まれる各パッケージの概要を簡単に紹介する。

(1) Quickscore

標準的なスコアを手早く確認することが目的である (図 2.5.11 がサンプル)。短い時間で結果を得るために、検証する対象要素を 6 種類 (海面更正気圧 [12]、500 hPa 高度、850 hPa 気温・風速、250 hPa 風速、700 hPa 相対湿度) に絞るとともに、アルゴリズムも冗長な計算・ファイル I/O を行わないよう工夫している。検証領域も北半球 (20°N 以北)、南半球 (20°S 以南)、熱帯 (20°S から 20°N) と日本周辺域 (20°N から 50°N、110°E から 150°E)、北西太平洋域 (0° から 60°N、100°E から 180°E) の 5 種類に絞っている。検証内容は、平方根平均二乗誤差 (RMSE) 、アノマリー相関係数 (ACC) 、平均誤差 (ME) の要素について、CNTL と TEST および両者の差分と、それぞれの区間推定である。差分について信頼度を変えて有意であるかどうかを確認するためのカテゴリ表示図も作成している。参照値は解析値とラジオゾンデ観測値である。

(2) Anlmap

平均解析場を比較するものである。解析場の変化が変更内容に対する応答として自然なものかを把握する。また、対解析値の検証は解析値自体の変化に影響を受けるため、それを丁寧に把握しておく必要がある。CNTL と TEST の比較においては予測精度の差が小さい場合もあり、その場合は特に解析値自体の変化に注意が必要である。また、各現業数値予報センターの解析値を用いた検証結果比較から、極域や熱帯域においては誤差の大きさと解析値のばらつきは同程度になることがあり、また中緯度においても 3 日目程度先までは誤差に対して解析値のばらつきが無視できないことが知られている (Langland et al. 2008; Jung and Matsueda 2016; Kanehama 2014)。このパッケージでは、TIGGE (The Interactive Grand Global Ensemble, Swinbank et al. 2016) データを利用して、他の現業数値予報センターの平均解析場に対して CNTL と TEST それぞれの差分も表示している。それぞれの解析値が分布する範囲から GSM の解析値が大きく外れるようだと特に注意が必要である。また、このパッケージには解析値にみられる太陽周期半日大気潮汐 (S_2) を比較する検証も含まれている。これは GSM の解析値には大気潮汐の位相ずれがあることが知られているためである (堀田・檜垣 2010)。

(3) Obstat / Dfit

データ同化における、第一推定値及び解析値と観測値との整合性評価を行うツールである (Lentze 2016; 計盛 2015)。衛星観測を含む全球解析で利用された観測データについて、観測値と解析値の差の標準偏差、観測値と第一推定値の差の標準偏差、品質管理を通過したデータ数の変化を観測種別、衛星搭載センサーのチャンネル別、領域別に網羅的に検証する。データ同化で利用される観測情報は膨大であり、かつ一定の品質管理も行われているためリファレンスとして非常に有用である。GSM を変更した時に、第一推定値と観測値との整合性が悪化していないかは必須の確認項目となる。

[11] 閾値に 95%を用いるのは慣習に従ったためである。
[12] ただし対ラジオゾンデ検証では代わりに 700 hPa 気温。

(4) TyVerif

台風検証のパッケージで、参照値には気象庁ベストトラックを用いている。進路予測誤差、中心気圧予測誤差についての統計的な検証と、個別の追跡結果比較図、中心気圧散布図を作成する。また、転向ステージごとに、位置誤差を進行方向について平行・直交する成分に分解する検証 (梅津・森安 2013) も行う。台風の誤差については、台風番号ごとに共通した特性を示すことが多いため、個々の台風ごとに統計検証を行ったものや追跡比較図をまとめたものも作成している。

(5) TyVerifG

台風の検証をサポートすることを目的とする熱帯低気圧検証のパッケージである。検証する項目は基本的に台風検証 (TyVerif) と同じであるが、参照値に B-deck[13] を利用して、北西太平洋領域を含む全球的な検証を可能にしている。台風を検証した結果は台風事例による依存性が高く、性能評価試験の範囲では統計的に十分な評価が難しいため、全球に適用範囲を広げて熱帯低気圧全体としてサンプル数を増やしている。また、台風ボーガスにより解析値が修正されない海域の方が、GSM 変更による熱帯低気圧への影響を明瞭に見ることが可能な場合がある。

(6) CyVerif

熱帯・温帯低気圧を含む全ての低気圧を対象とする全球低気圧検証パッケージである。低気圧予測全体の精度変化を把握する。独自の擾乱検出アルゴリズムで低気圧を追跡し (太田 2016)、解析値における追跡結果を参照値に予測値を検証する。位置誤差、中心気圧誤差、低気圧の存在そのものを予測できているかの捕捉率などの検証を含む。

(7) Lscore

解析値とラジオゾンデを参照値とする統計的スコアを表示するパッケージであり、線 (Line) で表示できる統計スコア全般を受け持つのが名前の由来である。実行時間は問わず、検証対象とする要素、面、領域を潤沢に取っているのが特徴である。RMSE, ACC, ME の鉛直プロファイル表示や、予報・解析活動度、絶対誤差、Taylor 図 (梅津ほか 2013)、ラジオゾンデによる検証と地点を揃えた対解析値検証などが含まれている。データ格納にはリレーショナルデータベースを利用しており、DPSIVS で作成される図表だけでなく、各自のアイデアに基づいて柔軟にデータを加工し利用できる。

(8) Errmap

解析値を参照値とする誤差統計について、面的な検証全体を受け持つパッケージである。RMSE と ME を気圧面、帯状平均での高度断面で比較できる。参照値とする解析値には、GSM のものだけでなく他センターの解析値も利用している。Lscore の結果と見比べて、予測誤差の変化を立体的に様々な要素について総合的に把握することが評価の基本となる。また、TEM (Transformed Eulerian Mean) に基づく残差循環の質量流線関数や Eliassen-Palm フラックスの子午面誤差検証も含まれている。

(9) Gmap

代表的な面・要素について、月平均場及び幾つかの初期日の予測結果を比較する。予測値そのものを描画するだけでなく、TEST と CNTL の差分や解析値との差分も描画している。予測結果がどのように変わっているのかについて、大まかな特徴を把握するのに利用する。Errmap と異なり単純な演算しか行わない代わりに、描画対象とする面・要素は多くなっている。

(10) Jpemap

スナップショットの予測誤差比較モニタである。日本周辺域を対象領域として、海面更正気圧、500 hPa 高度、925 hPa 気温、850 hPa 気温、500 hPa 気温、250 hPa 風速の対解析誤差を検証対象期間の全対象時刻、FT=24, 48, 72 について表示する。GSM の主要なターゲットの一つである日本域について、天気図の基本要素に関する予測誤差を確認するためのパッケージである。毎日の誤差マップと統計検証の結果を突き合わせ、どのような現象で予測誤差が変化しているのかを把握する。中緯度帯で誤差が大きく成長する条件は限られているため、モデル更新前後でも誤差のパターンが大きく異なる例は稀であるが、ジェットの速度、トラフの深まりや位相変化、擾乱の進行、高気圧の張り出しなどに着目しながら比較していく。統計スコアを解釈するときに、類似の気象現象で共通した誤差の変化が見られるのか、そうではなく事例ごとに異なるのかは重要な情報である。

(11) Synopv

参照値として SYNOP の 2 m 気温、2 m 比湿、海面更正気圧および前 24 時間積算降水量を用いる地上物理量検証を行う。SYNOP 観測の空間代表性に疑問があり、また計算負荷を抑えるために、全球の全地点に対して検証を行うのではなく 1.25° × 1.25° 格子ごとのメッシュ単位で検証を実施している。気温、比湿、海面更正気圧については 6 時間ごとに検証を行い、日変化を捉えることができるようにしている。実際、GSM の 2 m 気温には夜間と日中で逆符号のバイアスがある地点が多く、日平均で比較すると相殺されて誤差傾向が見えにくくなる。降水量については、後述する Raverif や AmedasRain と同様の検証を行っており、Grainverif の平均降水量評価と相補的に利用する。

[13] National Hurricane Center と Joint Typhoon Warning Center による熱帯低気圧解析。`http://ftp.nhc.noaa.gov/atcf/README`

(12) AmedasRain

アメダスを参照値とする降水検証を行うもので、バイアススコア (BI) 、スレットスコア (TS) 、エクイタブルスレットスコア (ETS) について統計検証と信頼区間推定を行うパッケージの一つである。検証対象とする降水量閾値、積算時間には 20 種類程度の組み合わせが設定されており、どのような降水でスコアが変わっているかを把握できる。また、アメダスの地点別に検証し、結果をマップで表示したものも含まれる。

(13) Raverif

解析雨量を参照値とする降水検証パッケージである。BI, TS, ETS だけでなく、誤検出率や適中率、降水量の RMSE, ME も計算する。また、面的な統計検証結果のマップ表示、日々の降水分布予測の比較図、検証対象期間中の毎日の各種スコアの時系列図、降水頻度分布図などを含む。解析雨量は海上域なども含み、AmedasRain とはお互いに相補的な関係にあるが、検証結果が整合的であることが多い。

(14) Sondev

ラジオゾンデを参照値とする、全球的な地点ごとの統計検証と散布図作成パッケージである。地点ごとに、気温、比湿、高度、東西・南北風の RMSE と ME が気圧面でマップ表示され、Errmap と同様に Lscore の結果と見比べながら利用する。散布図の要素は 850 hPa の相当温位、700 hPa の相対湿度など、解析値とラジオゾンデ観測値の差異が大きくなりやすい下層の量を対象にしている。

(15) Grainverif

期間平均した全球降水量の面的比較を行う。参照値は GPCP[14] (Global Precipitation Climatology Project) によるデータセットと CMORPH[15] (CPC MORPHing technique) プロダクトを用いており、CNTL と TEST で降水分布がどのように変化し、熱帯の加熱強制にどのような問題が考えられるかを評価する。予測時間が進むにつれて予測値の降水分布は参照値から離れていくが、最終的な落ち着き先だけではなく、離れていく速度がどう変化するかも確認する。

(16) Gnssro

GNSS 掩蔽観測を参照値とする統計検証であり、屈折角と屈折率に対して検証を行う。GNSS 掩蔽観測は高高度でも利用でき、観測のバイアスが小さい、鉛直分解能が高い、雲の存在に影響を受けにくいといった特性を持つ非常に有用な観測である。このパッケージでは対流圏上部から成層圏下部付近を対象に、RMSE, ME や利用データ数の変化などを検証している。検証内容は水平面、鉛直断面、時間高度断面など多岐にわたる。

(17) Phy2d

各種 2 次元モニタの統合検証パッケージである。CERES[16] (Clouds and the Earth's Radiant Energy System) データを参照値とする期間平均した全球放射バイアスの検証、OAflux[17] (Objectively analyzed air-sea fluxes) データを参照値とする期間平均した全球海表面フラックスバイアスの検証、RSS[18] (Remote Sensing Systems) によるデータを参照値とする期間平均した可降水量や液水量の検証と各種モデル出力の差分比較を含んでいる。GSM の放射バイアスの確認には CERES を参照値とする面的な比較がよく利用される。また、CNTL と TEST を比較して、気温や高度、風など基本的な要素に顕著な違いが見られなくても、モニタに出力される各種フラックスやパラメタリゼーション内の診断量では明瞭な違いが現れていることがあるため、基本的には全要素の確認を行う。

(18) Ptdraw

地点モニタの比較ツールである。GSM ではラジオゾンデ観測地点や特別観測地点を中心にその地点の予測値を地点モニタとして一時間ごとに出力している。このモニタは GSM 内のほとんど全ての予報変数、診断量が出力されており、モデルの挙動を詳細に把握することが可能である。このパッケージはそれら出力の値および差分を鉛直プロファイルと時系列で表示するものである。ただし、画像枚数が膨大になるため、地点及び初期時刻を間引いて描画している。統計検証の結果から気になった地点について、どのような変化が起きているのかを詳細に確認するのに利用する。

(19) ITeM

初期傾向診断法 (木南 2013) と FT=6 までの各過程の時間変化率を、様々な領域、断面上での平均値で比較するパッケージである。解析サイクルにおける第一推定値の修正量に着目し、その修正量の持つバイアスをモデルの中の各過程の時間変化率に分解して評価することにより、系統誤差の発生場所を診断する。また、このパッケージは鉛直方向にはモデル座標面を用いており、パラメタリゼーションの実装上の問題から発生する、特定の面で発生する時間変化率についての問題などの調査にも利用される。

---

[14] 全球降水気候値プロジェクト。http://precip.gsfc.nasa.gov/

[15] NOAA Climate Prediction Center による全球降水量解析。http://www.cpc.ncep.noaa.gov/products/janowiak/cmorph_description.html

[16] 全球放射収支計によるプロダクト。http://ceres.larc.nasa.gov/

[17] 全球の大気・海洋フラックスについての研究開発プロジェクトによる解析プロダクト。http://oaflux.whoi.edu/

[18] リモートセンシングシステム社。http://www.remss.com/

(20) Outline

基本的なパッケージの結果のうち代表的なものを一覧できるようにまとめ、クイックルックを提供する（図 2.5.11）。まとめる対象になるパッケージは、Quickscore, Anlmap, Obstat / Dfit, TyVerif, Lscore, Errmap, AmedasRain, Raverif, Grainverif である。

2.5.9 DPSIVS のまとめと今後の展望

GSM 開発における標準検証環境である DPSIVS について、特に開発管理の考え方を中心に簡単に紹介した。本検証システムは、近年の GSM 開発において中心的課題となっていた「compensating errors の解消」のため、改良項目の組み合せ実験の評価検証を効率的に実施し、開発者へのフィードバックを増やすことを大きな目的としている。また、検証システム自体の開発・保守を GSM 開発者自身が行い、問題点への理解や評価の知見が深まるにつれて、DPSIVS も成長していくところに特徴がある。本節では、それぞれのパッケージについて簡単に紹介したが、その詳細な内容と具体的な利用例の説明については別の機会に譲りたい。

最後に、今後の GSM 開発における検証環境の課題として、新しい検証手法の利用および、性能評価試験以外の標準実験の必要性に触れておく。

基本的な話として、参照値として利用できる観測データや観測プロダクトをさらに収集して、検証可能な要素を地道に増やす努力、及び新しく提案された検証手法を試しながら取り込んでいくことが重要なのは言うまでもない。同時に、現業数値予報センターである強みを最大限に活かし、データ同化で利用されている観測結果の評価検証でのさらなる有効活用も模索すべきであろう。

また、今後のパラメタリゼーション改良を考えた時に、雲についての 3 次元的な構造や日変化の検証は重要である。他の現業数値予報センターにおいても、衛星観測シミュレーションを利用した雲検証や、降水の日変化特性の検証などが日常的に利用されるようになってきている。今後の開発のためには、GSM においても早期にこれらの検証が実施できるよう環境整備を進める必要があるだろう。

性能評価試験以外にも、低解像度モデルを用いたより気候予測実験に近い構成での試験や、再解析を初期値として幅広い年を対象にした試験など計算コストを大幅に増大させることなく GSM の問題点をより鮮明に引き出せる試験構成の模索も必要である。特に、台風進路予測の検証結果は強い事例依存性を持つことが知られており、事例数を多く稼ぐために低解像度モデルを有効に利用できる可能性はある。ただし、それらの実験を有効に活用するためにも、GSM の分解能依存性についての調査と、Minor treatment（第 1.2.3 項）や数値計算上の問題から来る不必要な感度を持たないモデル開発が必要になってくるだろう。現在、これら低解像度モデルを利用した評価については、気候予測検証の経験をもつ気候情報課のアンサンブル予報システム (EPS) 開発担当者と連携しながら様々な取り組みを行っているところである。実際に 1 年積分実験や AMIP (Atmospheric Model Intercomparison Project) 型実験を行う環境の整備が進められており、それら新しい標準実験に対する検証システムにおいても、DPSIVS の思想とフレームワークは共有され開発が進んでいる。

EPS を対象とした検証システムとしては、EPSIVS (Ensemble Prediction System Integrated Verification System) が開発・利用されている。EPSIVS は 2009 年当時の、現業システムの予測結果検証と一体型であった EPS 検証システムから EPS 開発専用の検証環境として分岐し、その後多くの改良を取り込みながら発展し、全球 EPS (経田 2016) の開発において主要な検証環境として利用されている。EPS 開発が対象であるので、検証内容に確率的検証項目を多く含んでいる点は DPSIVS とは異なるが、利用者視点よりも開発者へのフィードバックを重視することや、開発での必要性や議論を取り込みやすいように拡張性を持った設計にしている点など背景としている思想は共通している。検証システムの管理コスト低減のためにも、お互いの良さは残しつつ、共有できるところは統一していきたい。

2.5.10 最後に

今回、代表して紹介させて頂く機会を得たが、GSM の開発管理手法や検証システムは、多くの開発者の協働により整備されているものである。また、利用しているサーバやソフトウェアの開発・管理については、気象庁内各課室にお世話になっている。ご尽力いただいた全ての方に感謝し、引き続き GSM の精度向上に努めていきたい。

【追記】ここで紹介した開発基盤ツールの開発者を付記しておく。DynaMo は齊藤 慧（数値予報課）、Super Verif は田内 利治（同）、DPSIVS の基盤部分は金浜 貴史（気候情報課）、関口 亮平（同）の寄与が大きい。

参考文献

Efron, B. and C. Stein, 1981: The Jackknife Estimate of Variance. *Annals of Statistics*, 586–596.

原旅人, 2013: 概論. 数値予報課報告・別冊第 59 号, 気象庁予報部, 1–5.

平井雅之, 堀田大介, 2009: 陸面過程. 数値予報課報告・別冊第 55 号, 気象庁予報部, 99–108.

堀田大介, 檜垣将和, 2010: 現業数値予報モデルにおける太陽周期半日潮汐の表現のセンター間比較. 大会講演予講集 98, 日本気象学会, 53.

堀田大介, 原旅人, 2012: 物理過程開発のボトムアップ・アプローチとトップダウン・アプローチ. 数値予報課報告・別冊第 58 号, 気象庁予報部, 120–122.

岩村公太, 2008: 高解像度全球モデルの改良. 平成 20

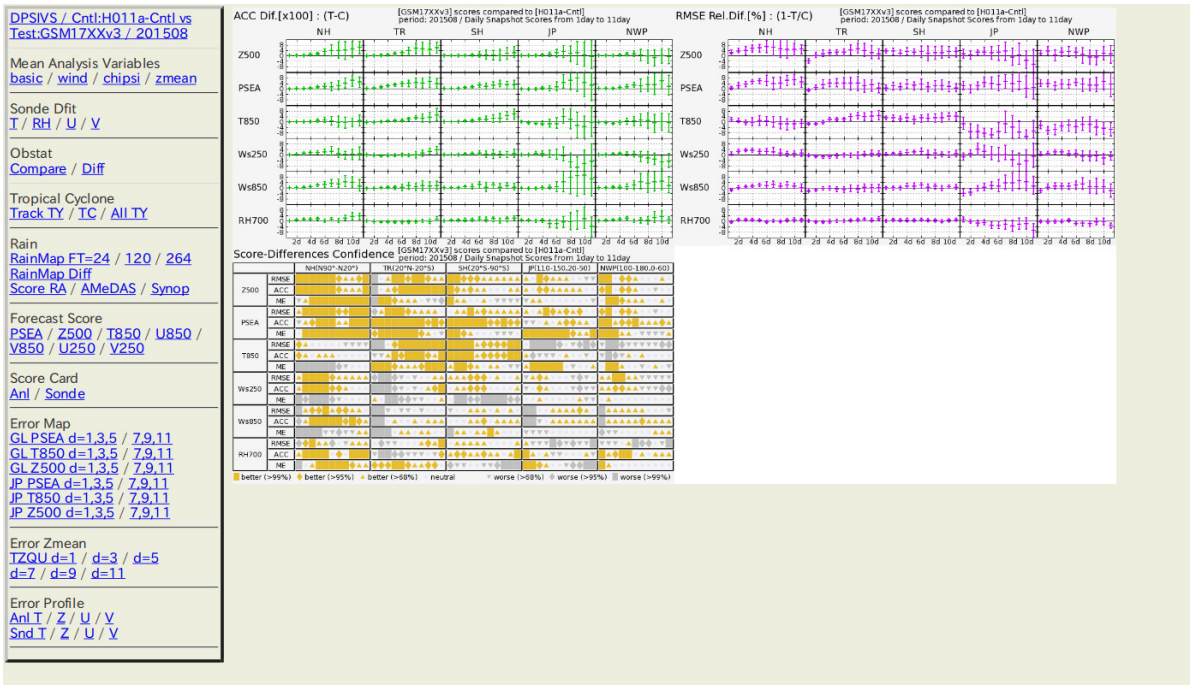

図 2.5.11　Outline の表示例。図は Quickscore パッケージの結果。

年度数値予報研修テキスト, 気象庁予報部, 1–6.

Jung, T. and M. Matsueda, 2016: Verification of global numerical weather forecasting systems in polar regions using TIGGE data. *Quart. J. Roy. Meteor. Soc.*, 574–582.

Kanehama, T., 2014: Verification against multiple analyses. *WGNE-29*, `https://www.wmo.int/pages/prog/arep/wwrp/new/documents/multicenter_verif_cmuroi.pdf`.

計盛正博, 2015: 衛星観測輝度温度データを使った同化サイクルにおける影響評価. 数値予報課報告・別冊第 61 号, 気象庁予報部, 82–85.

木南哲平, 2013: 初期傾向診断法. 数値予報課報告・別冊第 59 号, 気象庁予報部, 70–75.

気象庁予報部, 2007: 全球数値予報モデル (GSM) の積雲対流スキームの改良について. 配信資料に関する技術情報 (気象編) 第 275 号.

北川裕人, 2006: モデルの概要. 平成 18 年度数値予報研修テキスト, 気象庁予報部, 7–10.

北川裕人, 2007: 変更の概要. 平成 19 年度数値予報研修テキスト, 気象庁予報部, 1–4.

経田正幸, 2016: 全球アンサンブル予報システムの開発. 数値予報課報告・別冊第 62 号, 気象庁予報部, 52–57.

Langland, R. H., R. N. Maue, and C. H. Bishop, 2008: Uncertainty in atmospheric temperature analyses. *Tellus*, **60**, 598–603.

Lentze, Georg, 2016: Newsletter No.149 - Autumn 2016. 30–33.

宮本健吾, 2009: 適合ガウス格子版全球モデル. 数値予報課報告・別冊第 55 号, 気象庁予報部, 27–49.

室井ちあし, 藤田司, 大野木和敏, 2013: 具体的な取り組みの実例. 数値予報課報告・別冊第 59 号, 気象庁予報部, 202–203.

西尾利一, 2011: 計算機 (スーパーコンピュータシステム). 平成 23 年度数値予報研修テキスト, 気象庁予報部, 68–70.

太田洋一郎, 2016: 低気圧予測の精度. 数値予報課報告・別冊第 62 号, 気象庁予報部, 43–46.

下河邉明, 古河貴裕, 2012: 層積雲スキームの改良. 平成 24 年度数値予報研修テキスト, 気象庁予報部, 92–96.

Swinbank, R., M. Kyouda, P. Buchanan, L. Froude, T. Hamill, T. Hewson, J. Keller, M. Matsueda, J. Methven, F. Pappenberger, M. Scheuerer, H. Titley, L. Wilson, and M. Yamaguchi, 2016: The TIGGE Project and Its Achievements. *Bull. Amer. Meteor. Soc.*, 49–67.

梅津浩典, 森安聡嗣, 2013: WGNE 熱帯低気圧検証. 数値予報課報告・別冊第 59 号, 気象庁予報部, 98–111.

梅津浩典, 室井ちあし, 原旅人, 2013: 検証指標. 数値予報課報告・別冊第 59 号, 気象庁予報部, 6–15.

山下浩史, 2013: 現業化を判断する指標. 数値予報課報告・別冊第 59 号, 気象庁予報部, 28–32.

米原仁, 2014: 変更の概要. 平成 26 年度数値予報研修テキスト, 気象庁予報部, 1–3.

米原仁, 2016: 全球数値予報システムの物理過程改良の概要. 平成 28 年度数値予報研修テキスト, 気象庁予報部, 1–3.

2.6 活用例 (2)–メソモデル [1]

2.6.1 はじめに

2017 年 1 月現在、メソモデル（MSM, 水平格子間隔 5 km）の予測モデルとして気象庁非静力学モデル（JMA-NHM）が利用されている (Saito et al. 2006)。一方で、次世代の非静力学モデルとして数値予報課で開発してきた asuca(気象庁予報部 2014) の現業利用が、2015 年 1 月には局地モデル（LFM, 水平格子間隔 2 km）で始まり (原ほか 2015)、2017 年 2 月には MSM でも始まる予定である。

現在の MSM の開発は、力学コアとしての asuca と、asuca に物理過程を提供する物理過程ライブラリ (原 2012) を 2 本の柱として開発が行われており、本節ではそれらの開発管理とそのための関連システムの活用について述べる。また、開発の上で必須となる評価実験の管理についても紹介する。

なお、asuca や物理過程ライブラリの開発管理については、石田・藤田 (2014) でも触れられている部分があるが、本稿では開発プロセスに即して、より詳細に記述する。

2.6.2 物理過程ライブラリと asuca の開発における開発管理

物理過程ライブラリと asuca の開発においては、設計思想やプログラム構造を明確にした上で、新しい計算科学の知見の導入、計算安定性の確保、最近および将来の計算機の趨勢を踏まえた効率的なプログラム構造の導入を目指している (石田・藤田 2014)。その中で、加えようとする修正の必要性や科学的・技術的妥当性の検証を重視しており、そのプロセスにおいてプロジェクト管理システムへの開発過程の記録、バージョン管理システムのブランチの活用、そしてレビューが重要な役割を果たしている。

以下では、これらに焦点を当てながら、物理過程ライブラリと asuca の開発管理について述べる。

(1) 物理過程ライブラリの開発における開発管理

物理過程ライブラリの開発は 2010 年 8 月ごろから始まった。開発が始まるとともに、ソースコードに修正を加える際のプロセスとして、英国気象局における手法 (原・高谷 2013) を大いに参考にしながら、以下のプロセスを導入した。

チケットとブランチの作成、作業過程の記録

ソースコードに修正を加えようとする開発者（修正者）は、まず、プロジェクト管理システム（当初は Trac、後に Redmine に移行）にその修正について記録するチケットを作成する。合わせて、バージョン管理システム (Subversion) のリポジトリにその修正をコミットするためのブランチを作成する。修正者は、作成したチケットに自らの作業記録を残しながら、修正内容をブランチにコミットしていく。

これらの作業過程の記録は、後述のレビューや、後にその修正を振り返る必要が生じたときに、修正者の考えや実際の作業を追跡できる資料となっている。また、作業途中の内容も含めてブランチにコミットしながら作業を行うことで、その作業の途中経過を他の開発者が把握することが可能になり、その内容についての議論やテストなどが行われるようになった。

レビューに向けたドキュメントの整備、テストの実行

修正が終わったら、レビューへ向けて、修正の妥当性を示すドキュメントや作業記録を整備し、テストの実行と検証を行う。レビューは他の開発者（レビューア）に依頼して実施される修正内容のチェックで、

- 修正意図が、必要性や科学的・技術的根拠に基づく適切なものであること
- その意図に沿って実装が行われていること
- その実装がライブラリの設計思想やコーディングルールに合致していること
- 必要なテストが適切に行われていること

などの観点から行われる。

これは、レビューアだけではなく他の現在の開発者、未来の開発者への説明責任を果たすためのプロセスである。このようなプロセスがあることで、自らが行おうとするソースコードの修正が他の開発者に説明できるものであるかどうかを、各開発者が意識しながらモデル開発を行うようになった。

レビューの実施

レビューに必要な資料が準備できたら、レビューアにレビューの実施を依頼する。レビューアは先に示した観点などからブランチにコミットされた修正内容をチェックし、問題点が見つかったり、説明や記録が不十分で妥当性の判断がつかない場合には、修正者に差し戻して対処を依頼する。

このプロセスを通じて、修正者が気がつかなかった誤りが見つかったり、より望ましい方法がレビューアによって提案されたりするとともに、修正の必要性や科学的・技術的根拠の検討が修正者と別の開発者であるレビューアによっても行われ、より客観的に修正の妥当性を判断できるようになった。また、設計思想やコーディングルールへの合致をチェックすることで、無秩序な拡張によってソースコードが必要以上に複雑になることを防ぎ、継続的な開発の維持にも寄与している。さらには、新規に参入した開発者に対して設計思想やコーディングルールについての理解を促す機会にもなり、技術継承にも役立っている。

修正内容のブランチから開発本流へのマージ

レビューアから修正内容についての承認が得られたら、リポジトリの管理者がブランチにコミットされている修正内容を開発本流（トランク）にマージする。ト

[1] 原 旅人、荒波 恒平、松林 健吾、石田 純一

ランクへのコミット権限は管理者にのみ付与されており、これまでに述べた手続きを経ずにトランクに修正を加えることができないようになっている。

この仕組みによって、レビューアから承認が得られた修正のみがトランクに反映できるという手続きが明確化された。

次で述べるように、この開発管理プロセスは asuca にも導入され、asuca と物理過程ライブラリの開発管理プロセスは、ほぼ共通化されている。

(2) asuca の開発における開発管理

asuca の開発は 2007 年ごろからごく少人数で始まったが、モデルの大枠が定まってきた 2008 年には開発管理の支援のために Trac や Subversion が導入された。導入当初は、ソースコードに修正を加えようとするときには Trac にチケットを作成すること、このチケットと Subversion のリポジトリへのコミットを必ず対応させることがルールとして定められた一方、当初は Subversion のブランチ機能は活用されておらず、開発者が個別にトランクにコミットしていた。2010 年 1 月ごろにブランチの活用が始まり、修正はブランチに作成した上でトランクにマージするというスタイルが採用された。ただ、この時点では現在の開発プロセスの中で行っているレビューは行っていなかった。これは、開発が本格化して間もない段階で、ソースコードの新規追加や大幅な書き換えが頻繁に行われる中で、コストの方が得られる効果よりも大きいと考えられたこと、ごく少人数での開発であるため設計思想やコーディングルールについての認識が共有されており、レビューの大きな役割の一つであるそれらの観点からのチェックの必要性が低かったことなどが背景にある。しかし、良い開発のためにレビューが必須であることが asuca の開発者の中で共通認識となった今から振り返れば、設計思想やコーディングルールについての共通認識が得られやすい少人数の開発であっても、誤りの発見はもちろんのこと、レビューによる修正内容の相互理解の促進などのメリットは大きく、たとえ少人数でもレビューは行った方が良かったと思われる。

その後、物理過程の実装が始まるなど新規に参入する開発者が増えるにつれて、設計思想やコーディングルールを理解した上で修正が行われているかをチェックすることが従来よりも必要となったこと、また、asuca が物理過程として利用している物理過程ライブラリと開発管理プロセスを共通にしておくことは双方の開発者にメリットがあるとの判断から、既存の開発管理プロセスを拡張して、2010 年 12 月に物理過程ライブラリとほぼ同じ開発管理プロセスが asuca にも導入された。

次項では、現在メソモデルの開発における開発管理とその関連システムの活用について、asuca や物理過程ライブラリの開発に即して、より具体的に説明する。

2.6.3 asuca と物理過程ライブラリの開発（ソースコードの改変）における開発管理関連システムの利用

図 2.6.1 に、asuca や物理過程ライブラリの開発チーム内におけるソースコードの改変を伴う開発ワークフローの典型的な例を示す。

(1) 課題の設定

どのような開発も、課題の設定が起点となる。課題は

- 現在のモデルがもつ問題点を解決する可能性のある手法を考案した、または関連する文献を見つけた
- ソースコードやドキュメントを読んでいて誤りに気がついた
- モデルが異常終了した
- ある値をモニタしてみたら物理的に不自然な値や振動が計算されていることに気がついた
- モデルの評価、検証によるフィードバック
- 新しい機能を追加する必要が出てきた

など、様々なきっかけにより設定される。

(2) チケットの作成

課題が設定されたら、Redmine 上にチケットを作成する。チケットにはその内容を端的にあらわす「タイトル」を付け、課題の内容について「説明」を記述し、「担当者」を設定する。調査の進展具合に応じて、タイトルや説明と内容にずれが生じる場合もあるが、その場合はタイトルや説明を変更したり、関連チケットを作るなどして対応する。担当者は、業務分担や現在行っている仕事との優先順位、緊急度等を勘案して、開発チーム内のメンバーから割り当てられる（チケットを作成した本人とは限らない）。課題の優先度が低い場合には、亡失を防ぐために、当面担当者を割り当てずにチケットのみを作成しておく場合もある。

(3) ブランチの作成

トランクを改変する場合、機能拡張のために新たなソースコードを追加する場合や、ある程度規模の大き

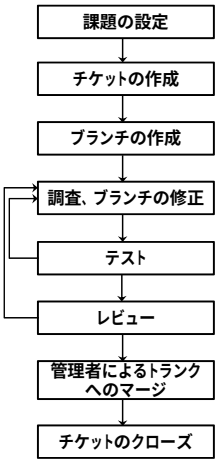

図 2.6.1 メソモデル開発でのソースコードの改変を伴うワークフローの例

な調査用のプログラムを作成する場合は、バージョン管理を行うために、開発ブランチを作成する。作成した開発ブランチの場所、および（可能であれば）その変更履歴をチケットに記録する。次に述べる調査の結果によっては、仕切り直しのために新しいブランチを作成することもある。

(4) 調査、ブランチの修正

次に実際の調査を行う。どのような目的で調査を行ったか、結果はどうだったか、結果を受けて次にどのような調査を行ったか、など、チケットを見た人が理解できるように記録する。現状、記録のやり方については特に制限を設けておらず、チケット上にテキストで記録したり、LaTeX 等でドキュメントを作成したり、スライドでまとめた資料を作成したり、画像ファイルを貼り付けたりするなど、調査の内容や進展状況に応じてやりやすい方法をとっている。調査に用いたプログラム等があれば、それも記録する。調査用の個別のプログラムについては、ブランチを調査用に作って適当なディレクトリを作成し、そこにコミットしていくことで、記録として分かりやすくする方法もしばしば用いられる。すでにあるソースコードの一部を変更する場合も、調査の過程では、作業内容を失わないために、ブランチについても随時更新していくことが多い。ただし、試行錯誤の結果、変更内容が複雑／膨大になった場合には、新しくブランチを作成し、必要な部分だけをブランチにコミットする形で整理しなおすこともある。

調査の内容は、試行錯誤の段階でうまくいかなかったものも含めて記録するようにしている。特に、期待される結果と違う結果が出てきた場合には、すぐに解決できない場合でも、今後の課題として認識しておくことが重要である。そうしておくことで、別の調査を行っている際に、原因を同じくする問題が顕在化してその解決法が考案され、それをきっかけに一連の課題が解決するケースもある。

(5) テスト

プログラムの改変を行い、それをトランクにマージする前には、十分なテストを行う。開発の目的（例えばあるスキームの持つ気温の不自然な時間変化率の改善など）に応じて、それが実現されているかどうかテストを行う。たとえば物理過程の変更では、1 次元モデルでの実験による評価、ある事例についての 3 次元モデルでの実験による評価、ある程度まとまった期間の実験による評価等を行い、力学過程の改良であれば、理想実験を通じた実験 (石田ほか 2014) 等を行って、結果を記録する。

また、asuca については、トランクへの変更がなかった場合も含めて [2]、テストハーネスと呼ばれる仕組みを毎日自動で実行している。これは

- 理想実験
  - 石田ほか (2014) で述べられている、2 次元定常山岳波、周期境界条件における重力波、暖気塊のテスト、重力流、St-MIP
  - 静止大気実験
  - 2 次元スカラー移流
- 3 次元モデル [3] による狭領域実データ実験
  - 局地解析
  - 接線形モデルチェック
  - 随伴モデルチェック
  - 局地予報
  - メソ予報

といった様々なケースについての実験 [4] が、最新のトランクを用いて行われる。入力データは毎日同じものを用いるため、入出力システムの変更等、予報結果を本質的に変える変更でない場合には、結果はビットレベルで一致する。自動実行されたテストハーネスの結果は毎日メールで報告され、結果が前日とビットレベルで合わない場合にはそのようなメッセージが付加され、開発者が覚知することができる。このため、トランクへのマージを行う改変については、テストの段階で開発者が自らテストハーネスを実行し、その結果についてもチケットに記録する。また、変更の前後で結果が一致しない場合には、その理由や差の妥当性（分布図で見た場合の評価等）も含めて、記録を行う。

(6) レビュー

レビューアは、第 2.6.2 項で述べた観点で変更の確認を行い、コメント等をチケットに記録する。その際、実行性能の観点から非効率なものになっていないか、コーディングスタイルが他と調和がとれているかなども含め、様々な観点から検討を行う。実際のレビュープロセスでは、多くの場合、そのまま通過することは少なく、レビューアから何らかのコメントがなされることが多くなっている。これにより、ドキュメントやソースコードの誤りの修正や、より分かりやすくするための変更などが行われ、これらの品質の向上に大いに寄与している。

なお、レビューのプロセスは Redmine 上の公開プロセスとして行われており、レビューアに指名されていない開発者も、チケットを見て変更内容をフォローし、疑問があれば積極的に議論に参加することが推奨されている。最終的に説明が不十分な場合、必要な実験が

[2] 毎日実行することで、ソースコードを変更していないにもかかわらず、100 回に 1 回程度の割合で結果が再現しない問題を検知し、OpenMP によるスレッド並列化に関するバグが特定された例もある。数値予報ルーチンとしては、再実行性の観点から、何度実行しても結果が一致することが重要である。

[3] 局地予報、メソ予報については、水平方向の格子数以外は現業のメソモデル、局地モデルと同じ仕様で実行される。

[4] これ以外にも、有用なテストが作成できれば、随時テストハーネスに組み込んでいる。

行われていない場合などはレビューアから開発者へ差し戻しが行われることもある。

レビューアに理解してもらえるように丁寧に説明することで、そのチケットが開発の記録として非常に価値の高いものとなり、現在だけでなく、将来の開発者への説明責任を果たすことになる。

(7) 管理者によるトランクへのマージ

レビューアによる承認が得られれば、管理者によるトランクへのマージが行われる。開発者は自由にトランクを改変することはできない。トランクへのマージを管理者に限定することで、開発の流れの一つとして、レビューアによる承認を受けることが必須となり、ソースコードの品質維持の担保となる。トランクへのマージの際、ビットレベルで結果が異なる場合には、その旨のメモが Wiki に追記され、履歴を辿りやすいように工夫している。

なお、複数の開発ブランチの差分が、競合（コンフリクト）を起こす場合もある。軽微なコンフリクトは、マージを行う管理者により修正されるが、複雑な場合には、新しいトランクから再度ブランチを作成してもらうよう、管理者から開発者へ依頼する場合もある。

(8) チケットのクローズ

前述の通り、テストハーネスは毎日自動実行されており、その結果はメールで報知される。まれに、自動実行の環境と開発者の環境の違い等により、意図せず結果が一致しない場合があるため、念の為、トランクへのマージが行われたら翌日のテストハーネスの結果を確認し、結果が意図通りなら、チケットをクローズして開発を終了する。なお、後に改変に関連する不具合が発生した場合、特に不具合の修正が小さい場合はチケットを再開して対応することもある。修正が大きい場合は、関連チケットとして、新しくチケットを作成することが多い。

なお、これらのフローに対応して、Redmine のステータスには「新規」「割り当て済み」、「進行中」、「レビュー」「コミット済み」、「終了」を定義しており、これらを活用して、各チケットの状態をわかりやすくしている。

以上、asuca や物理過程ライブラリにおける開発のワークフローの典型例を紹介した。開発管理ツールを活用することで、開発のワークフローが定型化され、レビューとそれに必要な説明を丁寧に心がけることで、ミスを減らすことにもつながるような仕組みとなっていることがお分かりいただけるだろう。

実際の asuca の開発においては、チケット上の議論だけでなく、2 週間に 1 度程度開発者が集まって、ミーティングを行い、主な開発の進捗について議論、共有を行っている。特に、改変の内容について複数の方向性が考えられ、開発者間で意見がわかれるような場合には、ミーティングの形式を取るほうが効率的に議論を行うことができる。こうしたミーティングにおける進捗の共有や改変内容の議論にもチケットをそのまま活用することにより、準備の省力化をはかることができている。

2.6.4 モデルの評価実験を行う際の開発管理と開発管理関連システムの利用

(1) 評価実験のプロセスとその管理

これまで述べてきた開発管理、および関連システムの活用は、モデルのソースコードを修正する際の作業の流れに関するものであった。一方、モデルに対して変更を加える際には、その変更によってもたらされるモデル予測の変化を確認するための評価実験が行われる。以下では、その評価実験のプロセスやその管理について解説する。

まず、asuca の開発においては、実験の管理や記録を一元化するために、ある程度まとまった期間や事例に対して評価実験を行う場合には、開発者各自がそれぞれ実行するのではなく、その実験の必要性、設定を開発者間で議論した上で、共用アカウントを用いて実験を行っている。共用アカウントによる実験に統一することで、実験の必要性を開発者個人の判断ではなく開発者間で議論する機会が必須となる。実験の必要性の議論を通じて、開発状況の進捗の共有や開発者間で現在進行中の開発内容についての理解を深めることに寄与するとともに、必要な実験だけを計画的に行うことで、計算機資源の有効活用にも役立っている [5]。

実験を行うことを決めたら、NAPEX を活用して実験環境を構築し、実験環境を構築した人とは別の人による実験環境の確認（レビュー）を行う。実験設定に誤りがあると、計算機資源が無駄になるのはもとより、その後の開発にも大きな影響を及ぼしかねないためである。

レビューが終わったら実験の実行を開始し、実験が終了したら検証プログラムを実行して統計検証を行う。また、事例調査も併せて行う。

asuca の開発においては、これらの一連のプロセスの進捗状況の管理や記録に Redmine を用いている。特に、2016 年度に進められた MSM 向け asuca の開発では、実験の数が最終的に大小合わせて 200 以上にのぼったが [6]、Redmine を実験の管理に活用することで、そ

[5] 加えて、共用アカウント上の共通の実験環境を引き継いで差分を加えていく事で、実験の間で意図せず仕様が異なってしまうことを防止することができる。

[6] 2016 年度に行った MSM 向け asuca の開発ではメソ解析システムに変更がないため、解析・予報サイクル実験をする必要がなく、必要な計算機資源が比較的小さかった。また、実験期間として連続する期間を設定する必要がなかったため、様々な現象（台風、梅雨前線、高気圧、冬型、南岸低気圧など）を含む事例を夏冬からそれぞれ 20 事例程度を選び出して、それらに対してインパクト実験（モデルの変更に対して個別事例の予測結果や統計的な性質の変化を確認する実験）

のような多数の進捗管理、検証結果の記録、実験結果の参照などを統一的に、かつ効率よく行うことができた。

(2) 評価実験の管理における Redmine の活用

以下では、これまでに述べた評価実験の管理における Redmine の活用法について、実際の開発に即して具体的に紹介する。

開発者間の議論により実験を行うことを決めたら、その実験に対応するチケットを作成し、実験の目的、変更内容などを記す。理論的背景や変更意図については、asuca や物理過程ライブラリで作成済みのチケットへ関連付けを行うことで参照する。実験を行う担当者が決まったら、担当者は NAPEX を用いて実験環境の構築を行い、実験の実行環境などをチケットにまとめる。レビューアによってその環境が妥当であるかの確認を受けた後、担当者は実験の実行を開始する。実験が終了したら、統計検証やモデルの予測結果への影響がどのようなプロセスによってもたらされたものなのか確認・分析し、チケットに記載する。

この一連の流れに対応して、「新規作成」「割り当て済み」「進行中」「実験環境レビュー」「実験環境承認済み」「実験中」「実験終了」「統計検証済み」「終了」というステータスを定義しており、実験の進捗とともにステータスを変更して、各実験の進捗を把握しやすいようにしている。

チケットに記録した内容は、メールや RSS で開発者間に共有されるため、情報共有及び議論のきっかけとしても有用である。実験環境は NAPEX 上でリポジトリ管理されているため、誰でも容易に再現することができる。また、これら実験のチケットは Redmine 上でまとめて表示することができ、各実験が現在どのようなステータスにあるか、これまでにどのような実験を行ったかを一覧で確認することができる。もちろん、行った変更の中には、物理過程間の相互作用などにより思わぬ結果が得られたものや、不採用になった変更も多いが、そのような情報も含めてチケットに残すことも非常に重要である。なぜなら、改善につながった変更はドキュメントや報告書という形で残りやすい一方、採用に至らなかった変更はドキュメント化されにくいが、そこで得られた知見はモデルの解釈やその後の開発の方向性を決める上で重要であるためである。

実際に MSM としての asuca・物理過程ライブラリの開発においては、多くの変更及びそれを評価するための実験を行った。これらの実験の中には採用に至らなかった変更が数多くあるものの、チケットに記録した内容はそのまま簡易なドキュメントとして残るため、開発者の記録用としてだけでなく、他の開発者への解説資料としても役立っている。

2.6.5 まとめと今後の課題

以上、メソモデルの開発において行われている開発管理についてその概要を紹介した。数値予報モデルという特殊なソフトウェアを対象とするものの、ソフトウェア開発管理としては、現在ではごく一般的となった手法を取り入れており、継続的なモデル開発を行っていく上で不可欠となる、ソースコードの品質の維持、向上に大いに役立っている。また、実験を行う際にも Redmine を利用することで、改善につながらなかった変更のように、ドキュメントに残りにくい重要な知見を記録する仕組みが出来上がっている。今後は、asuca-Var(幾田 2014) の開発が本格化し、解析サイクルを含めた実験の重要度が増すことになり、新たな工夫が必要となるが、これまでの経験を活かして、効率的に開発を進めていきたい。

参考文献

原旅人, 2012: 物理過程ライブラリの開発. 数値予報課報告・別冊第 58 号, 気象庁予報部, 205–208.

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.

原旅人, 幾田泰酵, 伊藤亨洋, 松林健吾, 2015: asuca が導入された局地数値予報システム. 平成 27 年度数値予報研修テキスト, 気象庁予報部, 1–23.

幾田泰酵, 2014: asuca 変分法データ同化システム. 数値予報課報告・別冊第 60 号, 気象庁予報部, 91–97.

石田純一, 藤田匡, 2014: asuca の開発理念. 数値予報課報告・別冊第 60 号, 気象庁予報部, 19–28.

石田純一, 河野耕平, 松林健吾, 2014: 理想実験を通じたドライモデルとしての評価. 数値予報課報告・別冊第 60 号, 気象庁予報部, 62–87.

気象庁予報部, 2014: 次世代非静力学モデル asuca. 数値予報課報告・別冊第 60 号, 気象庁予報部, 151pp.

Saito, K., T. Fujita, Y. Yamada, J. Ishida, Y. Kumagai, K. Aranami, S. Ohmori, R. Nagasawa, S. Kumagai, C. Muroi, T. Kato, H. Eito, and Y. Yamazaki, 2006: The Operational JMA Nonhydrostatic Mesoscale Model. *Mon. Wea. Rev.*, **134**, 1266–1298.

を実施した。インパクト実験の限られた事例数であっても、性能評価試験（夏冬それぞれ 2 週間、計 224 事例）、業務化試験（夏冬それぞれ 1ヶ月、計 448 事例）と同じ期間による実験に近い形でモデルの特性が評価できることがわかっており、このような手法によって一つの実験に必要な計算機資源を小さくすることが可能となり、多くのインパクト実験を実行することができた。

2.7 活用例 (3)–観測データ処理開発 [1]

2.7.1 はじめに

数値予報における観測データ処理では、先ず (a) 国内外から観測データを収集し、(b) 数値予報システムで利用しやすい形式に変換する。次に、(c) 誤差の大きなデータを除去したり系統的誤差を補正したりして観測データの品質を管理する。そして、(d) データ同化システムを用いて観測データを同化して数値予報の初期値を作成する。前述の通り数値予報課では、開発管理にプロジェクト管理システム Redmine を、バージョン管理システムでは主に Subversion を利用している。(a) 及び (b) は数値予報課プログラム班が担当し、(c) 及び (d) は同課数値予報班観測データ処理グループが担当している。観測データ処理については、それぞれで開発管理方針を定めており、かつ、相互に密に連携を取ることで補っている。ここでは、(c) 及び (d) に係る開発管理について述べる。以下、「観測データ処理」は特段の断りが無い限り (c) 及び (d) のことを意味する。

2.7.2 Subversion と Redmine の活用

数値予報で利用されている観測データは、地上観測や高層観測などに代表される従来型観測データから、静止気象衛星や地球観測衛星などによる衛星観測データなど多岐に渡る。また、データ同化システムも全球解析・メソ解析・局地解析・毎時大気解析など複数存在する。このような状況の中、観測データ処理に係る開発者は、品質管理用の実行プログラム及びデータ同化システムの実行プログラムに組み込まれている観測演算子を管理する必要がある。前者については、全ての観測データの品質管理を一つの実行プログラムで行うのは条件分岐が多くなり開発の効率が悪いため、ある程度の観測種別で分類して管理している。後者については、データ同化システム毎に管理している。

ここでは、観測データのうち、衛星観測輝度温度データのデータ処理に関する開発管理について説明する。観測データ処理に関する実行プログラムはひとつの Subversion リポジトリにまとめられており、品質管理に関する実行プログラムとデータ同化システム [2] の実行プログラムのソースコード、定数等が登録されている。前者は、観測データ処理グループで開発を管理している。一方、後者については、データ同化システム本体は、数値予報班の他のグループが開発を担当し、実行プログラムのソースコード、定数等を別のリポジトリで管理しているため、お互いに齟齬が生じないように注意する必要がある。観測データ処理では、現業運用されているデータ同化システム本体のソースコード、定数等に変更があった場合、それを自分たちで管理しているリポジトリのトランクに取り込むことで他のグループの開発に追随し、統一性を確保している。

Redmine には観測データ処理に関するプロジェクトの下に、衛星観測データに関する開発を管理する「衛星 QC」のサブプロジェクトを設けている。主に活用している機能は開発項目の進捗を管理するチケット機能であるが、情報共有のために Wiki 機能も活用している。チケットと実行プログラムをリンクさせるために、前述の Subversion リポジトリが Redmine に登録されている。

2.7.3 実例 – SSMIS (183 GHz) の利用開始

数値予報課では、米国の軍事気象衛星 DMSP に搭載されたマイクロ波放射計 (SSMIS) の 183 GHz 帯の輝度温度データを全球解析で新規に利用するための開発に取り組んでいる。この開発を、Subversion と Redmine の利用の具体例として示す。

この開発では、第 1.2.4 項に記載されている開発プロセスに則り、基礎開発及び性能評価試験を実施し、解析や予報の精度に与える影響について調査をした。本原稿の執筆段階では業務化試験を実施しており、これを経て現業化する予定である。Redmine サブプロジェクト「衛星 QC」に、基礎開発及び性能評価試験の評価に関する進捗を記録するためのチケット「SSMIS 輝度温度データ (183GHz) の利用」や「SSMIS F-17 UPP データの利用に向けた開発」を作成した。また、実験システムを構築する際には、別のチケット「SSMIS UPP 利用に向けた Napex 実験環境の確認」を作成し、動作確認の結果を記録すると共に、他の開発者と相互に確認を行うことで、ミスの混入を防止した。業務化試験は、他の輝度温度データ関係の開発課題と組み合わせて実施したため、新しいチケット「【業務化試験】2016T3：実験環境の構築」や「【業務化試験】SSMIS-UPP 追加」を作成した。Subversion リポジトリに業務化試験用の雛形ソースコードを用意し、各自が雛形ソースコードに加えた変更点及びマージ版の動作確認方法等を Redmine 上で共有・確認するなど、実験環境の構築にあたっては、各開発課題の仕様が適切にソースコードに反映されるように留意しながら作業を行った。

グループ内で利用方法等について意見交換を行った場合には、その記録をチケットに保存し共有している。また、Redmine は、これまでの開発で作成されたツール類の共有にも活用されている。今回の開発でも、183 GHz 帯以外の SSMIS の輝度温度データの利用に向けた開発 (江河・計盛 2009) の際に作成された観測データの可視化のためのツール類等を活用している。

2.7.4 最後に

観測データの品質管理では、それぞれの観測データに合わせて複雑な処理が行われているため、観測データの種別に応じて実行プログラムを分けて開発している。今後も、開発管理のためのシステムやツールを活用した開発体制の構築を進め、解析予報精度の改善に取り組みつつ、より堅固な数値予報システムとすることを目指す。

参考文献

江河拓夢, 計盛正博, 2009: マイクロ波放射計 SSMIS の利用. 平成 21 年度数値予報研修テキスト, 気象庁予報部, 54–56.

[1] 本田 有機、計盛 正博、村上 康隆

[2] 全球解析及びメソ解析のみ登録されている。局地解析は、開発体制により現時点では別の Subversion で管理している。

2.8 活用例 (4)–共通基盤 [1]

2.8.1 概要

開発管理サーバでは、モデル開発に関連する基盤技術の開発とユーザ支援を目的に共通基盤 Redmine を運用している。共通基盤 Redmine では共通基盤ライブラリ、気象庁自主開発のジョブスケジューラ (ROSE) などの自主開発ミドルウェア、第 3 章で解説する NAPEX や第 4.2.2 項で解説する NuSDaS といった数値予報システム開発全体の基盤となる各種ツールの開発管理を行っている。共通基盤 Redmine の利用者としては主に数値予報課プログラム班と数値予報課数値予報班基盤整備グループのメンバーが登録されているが、本節では基盤整備グループが担当している開発管理の内容を中心に説明する。

基盤整備グループでは、NAPEX 及び共通基盤ライブラリの担当者が日常的な開発業務で開発管理サーバを利用しており、担当者が開発の要所でそれぞれ相互にレビューを行うことを開発プロセスの基本として採用している。本節では基盤整備グループでの開発管理の事例として第 3.3 節で述べる NAPEX モデル [2] の管理について概説するとともに、共通基盤 Redmine は開発管理サーバ全体の開発基盤としての役割も担っていることから、全体の管理に関する話についても併せて解説する。

2.8.2 NAPEX モデルの管理事例

基盤整備グループでは実験用の数値予報システムである NAPEX モデルの維持・管理を行っており、この管理作業に開発管理サーバを利用している。NAPEX モデルの管理作業では JDF（第 3.2.3 項 (2)）などを除き、最新数値予報ルーチンとほぼ同等の実験用の数値予報システムを構築することが重要となる。NAPEX モデルの構築においては数値予報システムの入力として利用される観測データや初期値・境界値などのデータである引継ぎデータの登録、実験本体の設定、雛形実験やコントロールデータの作成といった複数の作業が必要である（第 3.3 節）。

基盤整備グループではこれら複数の作業の管理に子チケットによる管理を導入している。Redmine ではチケット同士に関連付けができ、この機能によってチケット間に親子関係を導入することができる。NAPEX モデルの管理では、作業全体の進捗管理を親チケットで行い、個別の作業の管理は親チケットに結びつけられる子チケットを用いて管理するという方法を用いている。図 2.8.1 に、子チケットを利用して NAPEX モデル更新の進捗管理を行った例を示す。この例では全球 NAPEX モデルの更新作業が行われた。図の下部に並んだ各項目が、分割された各作業に関する子チケットである。子チケットの進捗状況は親チケットから簡単に参照することができる。さらに、親チケットの進捗状況（図の上段）は子チケットの進捗に連動しており、子チケットの進捗に応じて自動的に全体の進捗が一目で把握できるようになっている。

子チケットを用いた進捗管理では、子チケットを五月雨式に作成しない点に留意すべきである。進捗が進むたびに子チケットを追加すると、全体の進捗状況が把握できなくなる。親チケットを作成した時点で、必要となる子チケットを予め作成しておくことが望ましい。これには親チケットを作成した時点で必要となる作業を明らかにしておく必要があり、子チケットとして分割するべき作業の洗い出しが必要となる。この作業によって必要な作業の見通しを作業者自身の手で自然と把握することができるようになっている。NAPEX モデルの管理では、子チケットの内容は相互に関連しているものの独立性があり、しかも定型的な作業が多いため事前に見通しが立てやすく、子チケットによる管理に適している。

分割した子チケットに沿って作業を進めていくわけだが、チケットで行う作業の中には基盤整備グループ以外の開発者へ影響が大きい作業も存在する。たとえば、NAPEX モデル本体の更新作業や引継ぎデータの登録作業は、登録結果が他のユーザに利用される前提で作業を行う。こうしたものを不適切な設定で公開してしまうと、利用した多くのユーザの実験結果が誤ったものとなり、実験のために消費された時間と労力が無駄になるといった状況になりかねない。そこで重要な作業のチケットでは相互レビューを必ず行い、設定に問題ないかどうかをレビューアが確認することにしている。指針決定後に実施された NAPEX モデル更新では、全ての更新作業で相互レビューが行われた。相互レビューを行ったことで、登録期間の誤りや定数ファ

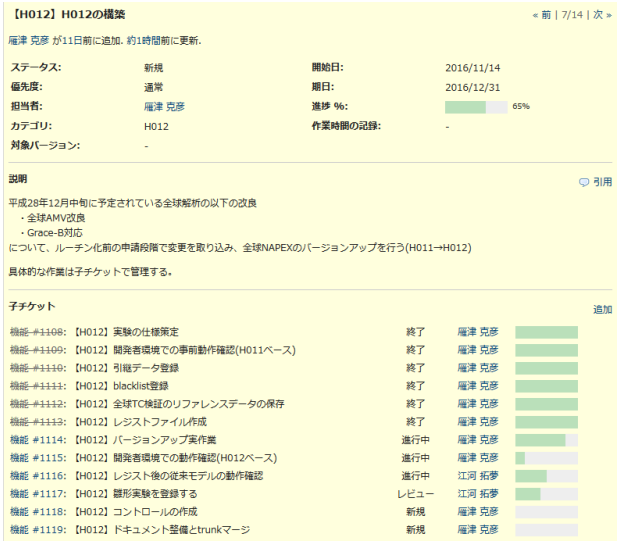

図 2.8.1 子チケットによる進捗管理

[1] 雁津 克彦
[2] NAPEX 環境で動作させる数値予報システムのこと。

イル・ソースコードの登録漏れといったミスが事前に回避された実績があり、相互レビューの導入は信頼性向上に大きな貢献をしていると考えている。

2.8.3 共通プラットフォームとしての共通基盤 Redmine

共通基盤 Redmine は数値予報システム開発の全体の基盤を担っているため、開発管理サーバ全体の情報や共通ツールなどの情報を広く提供する共通プラットフォームとしての役割も果たしている。

開発管理サーバの運用開始当初から継続しているプロジェクトとして「開発管理サーバ（ユーザサポート）」プロジェクトがある。このプロジェクトではフォーラムに「質問掲示板」を設けており、開発管理サーバの利用方法に関する疑義や提案事項がある場合に、管理者への質問や議論を行うことができるようになっている。フォーラムにはこの他にも「メンテナンス情報」のページを用意しており、開発管理サーバのメンテナンスに関する事前告知や新規プラグインの導入に関する情報を掲載し、開発管理サーバを利用する上でのユーザサポートの場を提供している。

2015 年には開発管理調整グループ会合で開発情報の蓄積を目的としたプロジェクトの新設が議論され、新たに「開発情報」プロジェクトが開設された。このプロジェクトでは、ユーザが作成した各種ユーティリティツールの共有を行うことができるようになっており、個別の数値予報システム向けに限らない汎用的なツールを作成した場合に、気軽にツールを公開できる環境を提供している。また、現在利用している第 9 世代スーパーコンピュータシステム (西尾 2011) での開発に関する情報の提供が行われており、一例として OS のイメージ更新 [3] などで開発環境に変更が生じた場合には、変更内容の情報を参照することができるようになっている。

加えて 2016 年には、2018 年に予定されている第 10 世代スーパーコンピュータシステムへの移行を見据えて、新システムへの移植を支援する「NAPS10」プロジェクトが導入されることとなった。

このように、共通基盤 Redmine は共通基盤に関する開発管理にとどまらず、開発管理サーバを利用する全ユーザに対しての共通した情報提供を行う役割も担っている。

参考文献

西尾利一, 2011: 計算機（スーパーコンピュータシステム）. 平成 23 年度数値予報研修テキスト, 気象庁予報部, 68–70.

[3] OS アップデートに相当する作業のこと。スーパーコンピュータシステムでは複数あるノードで更新が必要となるため、イメージファイルを各ノードに適用する形で更新作業を行う。

2.9 活用例 (5)–アプリケーション開発 [1]

2.9.1 はじめに

本節では数値予報の応用処理としてガイダンスやFAX 図など、アプリケーション (松下 2012) の開発を行っている数値予報課アプリケーション班（以下、AP 班）での開発管理について説明する。AP 班の担当するガイダンスや FAX 図を作成するジョブは、数値予報モデルに比べると一つ一つの規模は小さいものの数は多く、数十個にも及ぶ。そのため複数のジョブを担当者一人で管理・開発する体制を執ってきたため、以下のような問題点があった。

- ドキュメントが個人の Wiki ページや Word 文書などで作成されており、引き継ぎ時に新規担当者が自環境に移設する手間が発生するほか、担当者以外がドキュメントを確認・把握することが困難。
- 各ガイダンス間でソースコードの記述形式や開発管理体制などが統一されておらず、新規担当者への引き継ぎコストが大きい。
- ルーチン変更までの手順がルール化されておらず、担当者の裁量に依るため、バグの混入やミスを誘発しやすい。

本節で説明する統一的な開発管理や、ルーチン変更におけるガイドラインの導入により、この状況は改善されてきている。

2.9.2 Redmine と Subversion による開発管理

AP 班では 2010 年から Redmine と Subversion を利用し、前項の問題解決に努めてきた。まずドキュメントを Redmine の Wiki に集約することで、閲覧性を向上し、新規担当者への引き継ぎも円滑に行えるようになった。また各開発項目はルーチン変更に限らず、Redmine のチケット機能を利用して管理することとした。そしてチケット番号をドキュメントや Subversion のリポジトリと関連付けることで、変更点やその意図を把握するのに大いに役立てている。

2.9.3 ガイダンスにおける共通モジュール

第 2.9.1 項で述べた通り、AP 班では開発業務が個人単位で行われていたため、ソースコードの記述形式がほとんど統一されていなかった。こういった状況は、引き継ぎコストを大きくするだけでなく、開発効率の低下やバグの混入、ミスを誘発する。そこで 2012 年から各ガイダンスで共通して利用される機能（地点テーブルの読み込みや GPV の内挿、係数更新の手続など）を共通モジュールとして整備し、各ガイダンスに導入した。この共通モジュールを利用してガイダンスを開発することで、新規にソースコードを書く必要が無くなり、動作実績のあるソースコードを利用できるなど、開発効率・動作の信頼性が大幅に高まった。

2.9.4 ルーチン変更におけるガイドライン

AP 班のジョブは、ユーザが直接利用するという特性上、軽微なルーチン変更（アメダス移設や空港の観測時間変更によるテーブル変更や配信要素の変更など）が多く、かつ従来はその手順が個人依存で、ケアレスミスを誘発しやすい状況であった。そこでルーチン変更時のミスやバグの混入防止のため、2015 年に Redmine と Subversion を利用したルーチン変更ミス防止ガイドラインを設けた。ガイドラインの概要は以下のとおりである。

- Redmine を利用した開発管理を必須とし、開発の目的や経緯、スケジュールなどを記入する。
- チケットのウォッチャーに副担当を入れ、進捗を随時記載することで、情報の共有を行う。
- ソースコード類は Subversion 管理を必須とし、数値予報ルーチンからの変更点について Subversion の差分機能を使って示す。
- ルーチン変更に合わせ、作成したプロダクトの品質を確認するためのモニタとドキュメントの作成・更新を行う。
- ある一定期間以上の実行試験を行い、挙動やログに問題点がないかを確認する。
- ルーチン変更申請作業者は変更に先立ち、副担当に変更点について説明、また上記項目が適切に行われているか確認を受ける。

このミス防止ガイドラインの利用により、担当者に依存していたルーチン変更までの手順が統一・明文化され、副担当による確認作業がより確実かつ効率的に行えるようになった。また、このガイドラインとそれに付随するチェックリストにより、見逃されがちなミスが実際に検出されるなど、より確実な変更作業が行えるようになった。Redmine と Subversion はこれらの確認作業を円滑に進め、その記録の保存に必須のツールとなっている。

2.9.5 AP 班における開発管理の今後について

AP 班で開発管理に Redmine と Subversion を利用するようになって 6 年以上経ったが、ルーチン変更におけるガイドラインの作成や円滑な情報共有等のように、利用していく中でよりよい使い方が見出されることがある。また、描画や検証についてもツールの共通化を進め、他の開発者でも利用しやすいよう汎用性を高めることで、開発コスト削減となったケースもある。今後も開発効率の向上やミス防止を目指して開発管理ツールの使い方を模索していきたい。

参考文献

松下泰広, 2012: アプリケーション. 平成 24 年度数値予報研修テキスト, 気象庁予報部, 42–53.

[1] 後藤 尚親

2.10 活用例 (6)–システム管理 [1]

2.10.1 概要

数値予報課プログラム班（以下、P 班）における開発管理及び、数値予報ルーチンの管理への応用について述べる。

P 班では、数値予報システムの前処理･後処理として、観測データのデコード (佐藤 2012; 西尾 1995)・FAX 図等のプロダクト作成を担当しているほか、NAPEX でも利用されている ROSE (Routine Operation and Scheduling Environment) の開発等を行っている。ここでは、P 班の開発管理の代表例として第 2.10.2 項で ROSE プロジェクトを取り上げる。

また、P 班では数値予報ルーチンの管理を行っている [2]。数値予報課の各班・グループ、他課室の開発成果は、P 班にて「ルーチン変更」（後述）を実施することにより、数値予報ルーチンの改善として反映される。「ルーチン変更」を実施するにあたり、その作業内容の記録と共有を Redmine を活用して実施しているので第 2.10.3 項で紹介する。

2.10.2 ROSE の開発管理

(1) ROSE について

ROSE とは、P 班が主体となって開発しているジョブフロー制御ソフトウェア（ジョブスケジューラ）である。2008 年に開発が始まり、2009 年 6 月には予報支援資料作成 BCP (Business Continuity Plan) 対応装置 [3]（以下、BCP サーバ）での運用を開始した。BCP サーバでは、現在でも毎日 5 回、米国環境予測センター (NCEP: National Centers for Environmental Prediction) からのデータの取得とガイダンス・FAX 図を作成するジョブグループ [4]（以下、JG）が実行されている。

第 9 世代スーパーコンピュータシステム [5] においては、NAPEX に代表される開発業務において利用され、2018 年 6 月運用開始予定の第 10 世代スーパーコンピュータシステムでは、数値予報ルーチン及び各課ルーチンの実行にも ROSE を導入する計画となっている。なお現在は、スーパーコンピュータシステム導入ベンダー作成のジョブスケジューラ「数値予報ルーチン業務運用支援ソフトウェア (JNOS) 」にて運用している。

一般にジョブスケジューラとは、複数のジョブの起動や終了を制御したり、ジョブの実行・終了状態の監視や報告などを行うソフトウェアである。ROSE も同様であり、主な機能を挙げると次の通りである。

- JG を構成するジョブを依存関係に従って実行する
- 一連の JG を依存関係に従って実行する
- 指定した時刻に JG・ジョブの実行を開始する
- 実行開始時刻を指定しない JG を連続して実行する（開発用途向け）
- JG・ジョブの強制的な実行や実行中止 (a)
- JG・ジョブに与える環境変数の変更 (b)
- (a), (b) の操作を GUI で視覚的に行う
- ジョブ異常終了時にユーザへ通知する

ただし、ROSE 単体ではジョブを実行することはできず、バッチジョブのキューイングシステムとして LoadLeveler[6] 等を利用している。

また ROSE は、以下の構成要素から成る。

- ROSE DB[7]：全ての情報を管理
- ROSE Server：データベースの制御
- ROSE Agent：ジョブの投入制御
- ROSE GUI (CGI) ：監視・操作

Server, Agent, GUI (CGI) は、データベースを介して相互に情報をやりとりし、連携して動作する。

(2) 開発体制

ROSE は、Server, Agent, GUI (CGI) それぞれ独立して開発作業が進められるようになっているため、各構成要素に対して担当者が割り当てられている。ROSE の開発は、開発当初から P 班が行っていたが、第 10 世代スーパーコンピュータシステムでルーチン利用することも踏まえ、情報通信課システム運用室においても、ROSE の改善要望の集約や GUI (CGI) のソースコードの改修を行っている。

(3) 開発の進め方

ROSE の機能の改善・追加要望やバグ報告がユーザから上がり、それらの開発課題に対応していくことが基本的な開発の流れである。

開発課題は、まず Redmine のチケットに登録する。この時、Server, Agent, GUI (CGI) のどの課題なのか明確にするため、担当者を設定するとともに、チケットのカテゴリも設定する。

担当者はそれぞれ独立して開発作業を進めるが、開発方針で調整が必要な場合等は、適宜担当者間で調整する。決定事項はチケットに記述し、確実に記録が残るようにする。なお、独立した開発作業には、当然な

---

[1] 河野 正和

[2] 数値予報ルーチンの運用は、情報通信課システム運用室現業にて実施される。

[3] スーパーコンピュータシステムの大規模障害等で数値予報ルーチンプロダクトを提供できない場合に備えて、予報作業で必要となる最低限のプロダクトを作成するバックアップシステムを検証する装置のこと。

[4] ある一連の処理を行う複数のジョブの集合のこと。

[5] 第 7 世代以前は「数値解析予報システム (NAPS: Numerical Analysis and Prediction System)」と呼ばれていたが、第 8 世代からは、静止気象衛星資料処理局電子計算機システム (DPC: Data Processing Center System) と統合一体化し、「スーパーコンピュータシステム」と呼んでいる。

[6] `http://www.ibm.com/support/knowledgecenter/SSFJTW/loadl_welcome.html`

[7] DBMS (DataBase Management System) として PostgreSQL を利用している。

がら単体試験[8]も含まれている。

そして、それぞれの開発成果は、年1, 2回程度のアップデートのタイミングにあわせて結合試験[9]・総合試験[10]を行い、問題がないことを確認した上で、ROSEの更新を実施する。

上述の開発作業において、ソースコード等はSubversionによりバージョン管理を行う。各担当者の開発はブランチで行い、単体試験が終わったものはトランクにマージする。そして、結合試験・総合試験が終了してROSEを更新するタイミングでタグを作成し、ユーザにはタグから作成したROSEの環境を利用してもらうことにしている。

P班では、ROSE開発初期から、P班独自のサーバに整備したRedmineとSubversionを利用して開発を行ってきた経緯があるが、「(2) 開発体制」でも述べた通り、今後の体制強化も視野に入れて、Redmineプロジェクトと Subversionリポジトリを P班独自のサーバから開発管理サーバ上に移し、開発作業の連携や情報共有が行い易くなるようにした。

2.10.3 数値予報ルーチンの管理への応用

(1) 背景

数値予報ルーチンのジョブ及びJGは、以下から構成されている。

- JCLから生成されるスクリプト
- PBFから生成される実行プログラム
- 定数ファイル
- ジョブが生成する可変データ[11]
- JDFで定義され、ジョブスケジューラに登録されるJG・ジョブの依存関係

JCL, PBF, JDFについては、第3.2.3項(2)を参照されたい。

スーパーコンピュータシステムで数値予報ルーチンを運用するにあたって最も重要なことは、毎日決まった時刻にプロダクトを提供できるように、安定してジョブが実行されることである。数値予報ルーチンの安定運用とは、ハードウェア障害などの外的要因を除いた場合、現在動作しているスクリプトや実行プログラム等を全く変えることなくジョブを実行し続けることである。しかし実際には、開発成果を取り込み、数値予報の精度向上を図る必要があるため、スクリプトや実行プログラム等を変更する作業を実施することになる。この作業を、「ルーチン変更」と呼んでいる。ルーチン変更には人が介在するためにどうしてもミスが発生する可能性があり、ジョブが異常終了するなど、安定運用を乱す結果となる場合がある。このため、P班ではルーチン変更に起因する障害を無くすための取り組みを継続してきた。

(2) データベースとSubversionの利用

第3.2.3項(2)で述べている通り、第7世代数値解析予報システムでは、自由度が高くなる利点を得た半面、ミスを誘発しやすい環境でもあった。さらに、数値予報ルーチンもより複雑になり、管理するジョブも増加する一方であった。このため、次のようなミスが多かった。

JGの実行に必要な引継ぎデータがない

各JGは、JG内のジョブの実行に必要な引継ぎデータを最初のジョブで用意する。このジョブを「START」と呼んでいる。STARTのシェルスクリプトは、当該ルーチン変更を担当するP班員が作成していたが、データセットの記述漏れ等により引継ぎデータがなく、ジョブが実行できないことがあった。

定数ファイルがない

ルーチン変更申請を忘れたために定数ファイルがなく、ジョブが実行できないことがあった。

可変データが作成されていない

ジョブの依存関係を間違えたために、入力とするデータセットが後続のジョブで作成されていて、ジョブが実行できないことがあった。

ジョブの実行に必要なノードが足りない

スーパーコンピュータのノード数は有限なので、同時に実行できるジョブ数は制限される。しかし、JGのスケジューリングが悪いと、必要なノード数が不足してジョブの実行ができず、その結果、数値予報ルーチンの遅延を招くおそれがあった。

ルーチン変更時の引継ぎデータの準備不足

ルーチン変更の内容によっては、ルーチン変更時に引継ぎデータを準備しておく必要がある。しかし、ルーチン変更作業に手間取ってJG開始までに準備が終わらない、またはそもそも用意しないといけないことを忘れていたなどの理由で必要な引継ぎデータが不足し、ジョブの実行ができないことがあった。

このようなミスを減らすため、ルーチン変更作業の運用面からは、ルーチン変更1件につき、作業者1名、確認者1名を割り当てて複数名での作業とするなどの対応をしてきた。しかし、人による申請内容のチェックには限界があるため、システマティックに解決する手段が求められた。このため、第7世代数値解析予報システム運用中にJCLを開発し、ジョブの入出力が分かり易くなるよう改善した。さらに第8世代スーパーコンピュータシステム運用開始に向けてPBFとJDFを開発し、JCL等の内容を全てデータベースに格納し

[8] 各構成要素内の1つのプログラム、1つの機能のテスト。
[9] 各構成要素を任意に結合させたテスト。
[10] ROSEとして要求を満たしているか総合的に確認するテスト。
[11] JG実行毎に作成されるプロダクト等のこと。JG内や他のJGのジョブの入力データとしても利用される。

て連携させた。これにより、JG やジョブ間の矛盾等はなくなり、START ジョブも自動生成が可能となった。

また、JCL, PBF, JDF やソースコード等のファイルは Subversion に登録し、ルーチン変更情報と結びつけてバージョン管理することで、万が一問題が発生した場合でも、簡単に以前のバージョンに戻すことが可能になった。

このデータベースと Subversion を用いて管理するシステムを、数値予報ルーチン管理システム (RENS: Routine Environment for Numerical weather prediction System) と呼んでいる。JCL や JDF などの情報を有機的に連携して管理する RENS を開発したことは、これまでの取り組みの中でも大きな成果であった。

(3) Redmine 導入のきっかけ

数値予報ルーチンを RENS で管理するようになって、ルーチン変更に起因するミスは大幅に減少した。しかし、Redmine 導入以前は、ルーチン変更に係る作業内容は、作業者と確認者の間でメールにより共有するのみであった。このため、そのルーチン変更に関わらない他の P 班員は、どのような試験が実施されたかを知る機会は基本的に無く、ルーチン変更内容に対して十分な試験が行われたか分からない状態であった。

そのような中で第 9 世代スーパーコンピュータシステムへの更新作業が始まり、数値予報課内のルーチン移行の進捗管理のために、ROSE の開発で利用してきた Redmine を導入し、チケットのステータスで進捗を把握するようにした。また進捗を管理すると共に、移行作業においてどのようなことを実施したか、どのような問題が起きているかをチケットに記載することで、移行に係る情報共有が容易になり、それらを記録としてきちんと書き残すことができた。

この Redmine の特徴をルーチン変更作業にも取り入れれば、ルーチン変更の作業内容の共有と、記録として残していくことが容易にかつ同時にできると考えて、第 9 世代スーパーコンピュータシステムのルーチン変更作業に Redmine を導入することになった。

(4) Redmine の利用方法と導入の効果

ルーチン変更作業に Redmine を導入するにあたり、以下のように利用することにした。

- ルーチン変更 1 件に対してチケット 1 件を自動発行する
- チケットのデフォルトのウォッチャーは、作業者と確認者の 2 名とする
- 作業者は、試験内容についてチケットの注記に詳細に記述する
- 確認者は、確認内容についてチケットの注記に記述する
- 試験終了、確認終了などのフェイズに併せてチケットのステータスを更新する
- ステータスを更新することにより発信されるメールを契機に、作業者・確認者は次の作業を始める

担当者間でメールで行っていたことが、ウェブベースに変わっただけと思われるかもしれないが、両者には大きな違いがある。それは、作業内容を記録としてきちんと残していくことが可能になったとともに、他の P 班員が能動的に試験内容を知る機会が生まれたことである。これにより、試験内容がさらに多くの人の目に触れることになるだけではなく、それぞれのルーチン変更に対して、担当者の技量の差などから生まれるミスが軽減され、知見を共有することで数値予報ルーチン全体としての品質の向上につながったのである。

また、ルーチン変更に起因する障害の際には、従来はルーチン変更担当者でないと実施内容が不明なために対応が難しい面もあったが、Redmine を導入してからは、他の P 班員に情報が共有されることにより、担当者以外による対応が容易になった。

2.10.4 おわりに

P 班の開発管理についての取り組みと、開発以外の分野への応用例を紹介した。Redmine などのツールを上手く利用することで業務の品質は向上し、良い結果をもたらしたことは明白である。しかし、それぞれのコミュニティに合ったやり方でツールを導入しなければ、開発効率を落とすだけではなく、業務の品質も落とすことになるので、その点には注意が必要と考える。

参考文献

西尾利一, 1995: 観測データとデコード・ソート. 平成 7 年度数値予報研修テキスト, 気象庁予報部, 14–16.

佐藤芳昭, 2012: デコード. 平成 24 年度数値予報研修テキスト, 気象庁予報部, 14–15.

第 3 章　数値解析予報実験システム (NAPEX)

3.1 数値解析予報実験システム (NAPEX) の必要性とその歴史 [1]

数値予報システムは非常に多くのプログラム（ジョブ）で構成され、多くの種類の入力データを必要とする。そのため、プログラムの相互依存性を考慮した実行（ジョブスケジューリング）、入出力データのハンドリングが複雑になっており、開発に必要な実験システムの構築は容易ではない。また、複数のモデル開発者によってさまざまな開発実験が行われるようになっている中で、基準となる実験設定（用いる観測データ、モデル、実験期間など）を明確にして開発者間で統一することは、実験間の比較や評価をしやすくするために必要かつ重要なことである。

現在の数値予報モデルの開発手順では、予報モデルや同化スキームの変更をまずは単体で評価し、期待する結果が得られたことを確認して、データ同化サイクル実験を行うのが一般的である。単体での評価では良好な結果が得られても、データ同化サイクルを通じて予期しない悪影響が顕在化することがあるため、その変更の現業化の可否の判断に当たってはデータ同化サイクル実験が不可欠である。しかし、大野木・入口 (2003) によれば、1990 年代には開発段階でデータ同化サイクルを簡便に実行できるシステムが存在しておらず、データ同化サイクル実験を行うには、それぞれのモデル開発者が数値予報ルーチンに組み込まれた環境を自分の開発環境にコピーし、それを手動で修正することによって実験を行っていた。その修正箇所は、基準実験のための環境構築だけでも、入力データの入力元や出力データの出力先を変更したり、数値予報ルーチン専用の記述を取り除いたりするなど、多岐にわたった。そのため、実験環境を誤りなく構築し、正しい実験を行うことは非常に手間のかかる作業であった。また、モデル開発者によって基準となる実験設定が異なっている場合が多く、モデル開発者間でお互いの実験結果を比較することにも困難を伴った。

そのような中で、1998 年から 2000 年に欧州中期予報センター (ECMWF) に派遣されて再解析システムの開発に携わった数値予報課の職員が ECMWF におけるデータ同化実験システムや予報実験システムの充実ぶりを見て、気象庁のモデル開発における同様のシステムの必要性を強く認識し (大野木 2000)、帰国後の 2001 年に第 7 世代スーパーコンピュータシステム（2001 年 3 月〜2006 年 2 月運用）が導入され OS が UNIX に切り替えられたのを機に構築したのが、数値解析予報実験システム (NAPEX: Numerical Analysis and Prediction EXperiment system) である (大野木・入口 2003)。NAPEX は数値予報ルーチンで運用されている解析サイクル、モデル予測をモデル開発者の開発環境でも簡便に実行できるようにしたシステムであり、このシステムの登場によって、モデル開発者が実験システムの構築に煩わされることなく、そのシステムを用いた実験の評価に集中できるようになった。次節で説明するように、NAPEX はその後、さまざまな改良を経て [2]、今日でも開発基盤として最も重要なシステムの一つとして活用されている。

2001 年に構築された初版の NAPEX (大野木・入口 2003) では、標準の実験環境が管理者によって準備されており、ユーザは指定の設定ファイルを編集して、環境構築のためのシェルスクリプトを実行すれば、その設定に従って標準の実験環境のユーザの開発環境へのコピー、および必要な修正が自動的に行われるようになっていた。また、バージョン管理を厳格に行っており、各モデル開発者の実験がどのバージョンの標準環境を用いたかを明確にして、実験間の比較がしやすいようにしている。これらの点は、15 年経過した現在の NAPEX でも大きく変わるところはない。一方、その当時の NAPEX では、標準の実験環境を管理者が構築する際に数値予報ルーチン特有の処理を取り除いたり、使用する計算機資源を節約するために、数値予報ルーチンで使用されているスクリプトに多数の修正を加えて利用していた。NAPEX のバージョン更新のたびに同様の作業を繰り返す必要があり、その管理コストは大きかった。

2006 年 3 月に運用が開始された第 8 世代スーパーコンピュータシステム（2012 年 6 月運用終了）への数値予報ルーチンの移行の際に、数値予報ルーチンの実行スクリプトの生成に JCL、Makefile の生成に PBF と呼ばれるファイルの記述法および関連ツールが導入された（JCL や PBF については第 3.2 節を参照）。それをきっかけに、NAPEX も数値予報ルーチンと同様に JCL と PBF を用いたシステムに移行し、かつ、数値予報ルーチンとして登録された JCL や PBF ファイルには極力、手を加えないこととする大きな変更がモデル開発者によって行われた。これは、前世代のスーパーコンピュータシステムと比べて開発に利用できる計算機資源が相対的に増加し、使用する計算機資源を節約するために高い管理コストを投じて数値予報ルーチンで使われているファイルを修正する意義が小さくなったことが背景にある。この変更により、数値予報ルーチンと NAPEX の間の差異が小さくなり、管理コストも小さくなった。また、検証システムなど、数値予報ルーチンにはない NAPEX 独自のジョブを簡単にプラ

[1] 原 旅人

[2] 2006 年、2012 年のスーパーコンピュータシステム更新のタイミングで NAPEX の大きな改良が行われている。

グインできるようになった。

その後、2008 年から 2010 年まで英国気象局に派遣された著者が、英国気象局においては実験設定がデータベースで管理され、それを他のユーザが参照したり、コピーして同じ実験を行うシステムになっていること、また、そのようなシステムを構築するなどの開発支援を専門に行う部署があることを報告し (原・高谷 2013)、帰国後、気象庁のモデル開発においても同様のことが必要であると主張した。従来は、標準の実験環境は管理されているものの、それに対するユーザによる変更履歴の管理機能が NAPEX 本体には実装されていなかったため、どのような変更が加えられたのかはすぐには判然とせず、後日、同じ実験を行おうとしても再現できない場合すらあった。また、第 1.1 節で述べた通り、それまでの NAPEX の構築はシステム面での専門家ではないモデル開発者が行ってきたことから、先に述べたような、実験設定のデータベース管理とそれを簡単に利用できる機能、ユーザによる実験設定の変更履歴の管理機能、数値予報ルーチンシステムからの差異を小さくすることによるさらなる構築・管理コストの縮小など、システムの専門家が構築すればできたと考えられることが不十分になっている面もあった。

そのような状況を受けて、第 9 世代スーパーコンピュータシステム向けの NAPEX の開発には数値予報ルーチンシステムの専門家である数値予報課プログラム班（P 班）が参画し、数値予報ルーチンシステムで用いている技術を活用しながら、数値予報ルーチンの環境からの差異がより小さく、また、実験の履歴の記録や再現性の確保を兼ね備えた現在の NAPEX が構築された。数値予報ルーチン向けに開発されたツールを開発で利用するための拡張などが開発元の P 班で積極的に行われるようになり、モデル開発者とシステムの専門家がお互いの得意分野を活かしながら、数値予報モデルの効率的な開発に寄与している。数値予報課外でも、NAPEX は気候情報課が実施した長期再解析 JRA-25、JRA-55 のデータ同化実験・実行システムのベースとして導入され、その業務に大きく貢献している。

本章では、これまでに述べた NAPEX の歴史を踏まえ、現在用いられている NAPEX の設計思想および実装、管理および利用の実際について報告する。

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.

大野木和敏, 2000: 開発環境の整備. 数値予報課報告・別冊第 47 号, 平成 12 年度数値予報研修テキスト合併号, 気象庁予報部, 134–137.

大野木和敏, 入口武史, 2003: 数値解析予報実験システム NAPEX. 気象庁測候時報, **70**, 171–187.

3.2 現在の NAPEX の設計思想と実装 [1]

3.2.1 はじめに

NAPEX は 2001 年の運用開始 (大野木・入口 2003) から 15 年が経過し、その間に複数の管理者及びユーザの利用を通じて頻繁に改良が続けられ、現在では開発に不可欠な共通の実験基盤として広範囲に利用されるようになっている。

近年の NAPEX 開発史で最も大きな転機は、第 9 世代スーパーコンピュータシステム (西尾 2011) への移植作業時に訪れた。この移植時には第 8 世代スーパーコンピュータシステム (中山 2005) での NAPEX の利用を通じて、以下の項目が課題として挙げられていた。

管理コストが大きい

当時の NAPEX は実際に数値計算を行うためのプログラム群である数値予報システム本体のほかに、制御系とよばれる数値予報システムの実行制御を行う管理ツールが存在した。制御系は数値予報システムの大幅な変更が発生すると改修が必要となっており、特に初期値・境界値・観測データといった、数値予報システムの入力として利用される引継ぎデータの設定や、データ保存に関する部分では、毎度慎重な設定作業が要求され管理者負担が大きかった。これに対して数値予報ルーチン管理の現場では、こうした引継ぎデータ・保存データ処理の自動化が開発・実装されており、数値予報ルーチンの手法を用いることで制御系のメンテナンスを大幅に簡素化できる可能性が指摘されていた。

また、全球数値予報システム、週間・台風アンサンブル予報システム（週間・台風 EPS)、メソ数値予報システムで、NAPEX 環境が異なっており、個別管理を行う必要があったことから、統一的な NAPEX 環境によって管理コストを低減することが求められていた。

実験が再現できない

当時の NAPEX では制御系のバージョン管理はされていたものの、数値予報システム本体はバージョン管理されておらず、各ユーザが実行した実験の設定や実行プログラム、定数ファイルなどの管理はユーザに一任されていた。こうした背景からユーザが過去に行った実験を再現する必要に迫られた際、管理が不十分で当時の実験設定を紛失してしまった場合、実験が再現できないという状況だった [2]。

こうした指摘を受け、現在の第 9 世代スーパーコンピュータシステム向けの NAPEX 構築では「数値予報ルーチンとの互換性確保」と「実験再利用性の確保」の二つの設計思想を中心に据え、NAPEX 環境の実装を行った。本節では現在の NAPEX の基礎となっているこれらの設計思想について紹介した後、実装方法について解説する。解説には数値予報ルーチンに関する知識が不可欠であることから、必要に応じて数値予報ルーチンに関する事項を説明する。

NAPEX は第 8 世代スーパーコンピュータシステム向け、第 9 世代スーパーコンピュータシステム向けなど、スーパーコンピュータシステムの更新により数値予報ルーチンの運用や開発環境が変更される際に仕様が変更されている。これらを明示的に区別する正式な名称は存在しないが、説明の便宜上 NAPEX8, NAPEX9 のように、末尾に対象となるスーパーコンピュータシステムの世代数を付加することで以降の説明を簡略化したい。

3.2.2 基本思想

第 3.2.1 項で示した NAPEX8 での課題を踏まえ、現在運用している NAPEX9 では以下の二つを基本的な設計思想と位置付けて実装を行った。

数値予報ルーチンとの互換性確保

NAPEX9 では数値予報ルーチンで標準的に利用されている JCL (Job Control Language), PBF (Program Build File-format), JDF (Job Definition File) といった各種数値予報ルーチン用のツールを活用して環境を構築した。これによって NAPEX で実行する数値予報システムを、ほぼそのままルーチンで利用する、あるいはその逆を容易に実行できる。また、現在の数値予報ルーチンで利用されている JCL, PBF, JDF は、必要となるファイルの依存関係を解決することができ [3]、例えば JCL であればジョブを実行する際に必要な引継ぎデータの一覧を、PBF であれば実行プログラムのコンパイルに必要なソースファイルの一覧を、自動的に取得できるようになっている。NAPEX8 でも JCL や PBF の利用自体はされていたが、NAPEX9 ではこうした自動的に環境を構築できる機能も導入し、実験構築時の設定作業や実験環境の維持管理に掛かるコストを大幅に軽減できるようになった。

実験再利用性の確保

NAPEX9 では個人が設定した実験設定の全ての情報をデータベースとバージョン管理システムに記録し、任意の過去の実験を再利用できることを第二の基本思想とした。これにより NAPEX8 で問題となっていた過去実験の再現が保障されるとともに、自分が実施した実験に更なる修正を加えて新たな実験とすることも簡単にできるようになり、実験の再利用性が飛躍的に向上した。さらに、自分以外の実験も再利用可能とする

[1] 雁津 克彦

[2] なお、週間・台風 EPS の NAPEX は全球数値予報システムやメソ数値予報システムの NAPEX より後に実装され、ユーザ実験にバージョン管理を導入したため、こうした状況が生じにくかった。

[3] 数値予報ルーチンの場合はファイルから直接依存関係を解決するのではなく、ファイルの内容を一度 RENS（第 2.10.3 項）に格納してから解決している。

ことで、他の開発者の開発成果を引用して自分の開発成果とマージして新たな実験を行うといった、グループとしての開発に不可欠なプロセスを簡潔に行うことが可能となった。また、実験の設定だけでなく実験結果のデータを引き継いで自分の実験を行うことも簡単にできるようになった。例えば、他の開発者が実験した全球数値予報システムの結果を境界値として引き継ぎ、メソ数値予報システムの実験を実行することができる。このように、実験の再利用性の確保によって開発者間で開発成果を相互利用することが可能になり、開発効率の向上が図られた[4]。

これらの基本思想は、単に設計に関する思想というだけでなく、運用面においても重視されることとなった。例えば、NAPEX で構築済みの実験をユーザが後から自前で修正したり、実行期間を延長して実行したりすることができるが、NAPEX を利用しない修正では設定が記録されないため、基本思想の「実験再利用性の確保」を大きく損なってしまう。結果として後に実験の再現が必要になった際に、他のユーザや開発者自身の時間を浪費することに繋がりかねない。このため、NAPEX で設定した実験を独自に修正することは運用ルールで禁止している。

このように、単にシステム設計面にとどまらず、実験時の運用ルールでも上記基本思想が重視されており、各ユーザはルールを遵守して実験を構築することが求められる。

3.2.3 NAPEX 環境の実装方法

NAPEX9 では数値予報ルーチンとの互換性に重きをおいて実装することになったことから、それまで NAPEX 環境を整備していた数値予報課数値予報班に代わって、実際に数値予報ルーチン環境の整備を行っている数値予報課プログラム班が主体となって実装が進められた。新たに加えられた主な機能、すなわち NAPEX8 環境までとの差異は主に以下の四つの項目に集約することができる。

- 全ての実験の設定をデータベースで管理する
- ジョブの管理や実行プログラムの管理に数値予報ルーチン管理の言語を用いる
- ファイル管理にバージョン管理システムを利用する
- 実験の実行に独自開発のジョブスケジューラを用いる

以下、順を追って見ていく。

[4] 過去実験の再現には実験の設定だけでなく利用した引継ぎデータの存在が不可欠である。このため再利用性を完全に確保するには、引継ぎデータも保持しておくことが必要であるが、NAPEX9 では容量の都合などで入力した引継ぎデータの保持は行っておらず、古いデータの場合は消去されてしまうことも多くあることから、今後の課題である。

(1) 実験設定をデータベースで管理する

NAPEX9 の最大の特徴は実験設定を RDBMS (Relational DataBase Management System) を用いて管理している点であり、これにより過去の実験を容易に再現できるようになったのは非常に革新的であった。具体的には PostgreSQL を用いて実験に必要な情報をデータベース上に保持することで、実験の再現を可能にしている。実験に必要な情報とは、実験を実行する期間、データの保存先、実行するサーバの指定、利用する引継ぎデータ、保存するデータの種類など、数値予報システム本体以外の各種設定を指す。

実装上の基盤となるのは、各個人の実行する実験に対して一意の「実験番号」を与えて区別する点である。ユーザが実験を行う際は新規に実験番号を取得して実験を構築していくわけだが、実験構築時の作業では設定内容の修正が頻繁に行われることが想定される。そこで個々の修正内容を記録するために実験番号とは別に「枝番」を用意し、修正が行われるたびに枝番が更新される実装としている。ユーザが設定した内容は実験番号と枝番に紐付けられ、データベース上に格納する仕組みとなっている。再利用時はこの仕組みを逆に利用して、過去に実験を行った際の実験番号と枝番をキーに、データベースに対して SQL コマンドを発行することで当時の実験設定を検索する。

SQL によるデータベースの検索や格納は、NAPEX 環境を利用するためのツール（Ruby スクリプト）を通じて行う。ユーザが直接 SQL コマンドを実行する必要はない。例えば「新規の実験番号を取得する」際は napex_init.rb というツールを、「実験の修正内容の記録」では napex_setup.rb というツールを利用することで、必要な情報がデータベースに登録されるようになっている。別の実験を再現する場合も、再現したい実験番号と枝番を手元に用意し、NAPEX 環境の標準的なツールを利用することで、対象の実験設定を自分の環境で再現することができる。

過去に設定された全ての実験は、CGI を利用してイントラサーバ上に一覧が公開されている。ユーザはウェブブラウザによる閲覧で利用したい実験の実験番号と枝番を調べることができる。また、多くの開発者が共通して利用することを前提に構築された実験は「雛形実験」（第 3.3.3 項も参照）として、開発管理サーバ上の Wiki のリンクなどを通じて実験番号と枝番を広く周知する運用を行っている。

NAPEX で必要となる実験の設定内容は多岐に渡ることから、データベースには設定保持のためのテーブルを複数用意している。利用されているテーブルの一例を表 3.2.1 に挙げる。テーブルアクセスの応答速度は、テーブルの設計方法、とりわけインデックスの作成方法で大きく変化する。NAPEX のデータベース設計では頻繁に利用される実験番号と枝番を中心に、各テーブルの主要なカラムに対しインデックスを作成す

表 3.2.1 NAPEX9 で実験情報を保存するために利用しているテーブルの例

テーブル名	格納されている主要な情報（抜粋）
exp_control	基本設定（実験名、ユーザ名、実行ディレクトリ等）
exp	実験設定（実行期間、保存先ディレクトリ等）
prec_exp	引継ぎ設定
set_exp	詳細設定（実行ホスト、スケジュール等）
jcl	利用する JCL ファイルの情報
pbf	利用する PBF ファイルの情報
jdf	利用する JDF ファイルの情報
const	利用する定数ファイルの情報
var	保存するデータの設定

ることで高速化を実現している。

(2) ジョブの管理や実行プログラムの管理に数値予報ルーチン管理の言語を用いる

NAPEX9 では数値予報ルーチンのツールを利用した管理が徹底されるようになった。数値予報ルーチンではジョブの管理や実行プログラムの管理に、以下に挙げる独自開発の言語を利用している。

JCL (Job Control Language)

JCL は気象庁が開発したジョブを記述するための言語である [5]。

Unix 系マシンでのジョブの記述は通常シェルスクリプトが利用される。気象庁でも 2001 年の第 7 世代数値解析予報システムへの更新時に OS が Unix 系 OS (HI-UX/MPP) に変更となり、ジョブ制御にシェルスクリプトが採用されるようになった (平 2001)。しかし、シェルスクリプトは高機能で記述の自由度が高いためループや条件分岐を手軽に利用でき、複雑な動作をするジョブを簡単に書けてしまう。これにより機械的なジョブ内容のチェックが困難で、数値予報ルーチンの管理上弊害があった。また、データハンドリングのような数値予報ルーチンで定型的に利用する処理に複数行の記述が必要であり、ミスを誘発しやすい問題もあった。そこで第 7 世代数値解析予報システム運用中に、数値予報ルーチン用のジョブを記述するため独自開発したのが JCL である。

[5] 本来 JCL とはメインフレームに代表される企業向け汎用計算機で、バッチ処理を記述するスクリプト言語を指す。気象庁でも第 6 世代数値解析予報システムまでは、ジョブ制御言語として本来の JCL が採用されていた。第 7 世代数値解析予報システムで一時シェルスクリプトが利用されたものの、本文中のような問題が発生したため、本来の JCL を参考に気象庁で独自開発したものが JPP/JCL と呼ばれる言語である。JPP/JCL はその後改名され、現在は JCL が正式名称となっており、本文中の JCL はこちらを指す。

JCL はシェルスクリプトと比較して簡素な記述が特徴であり、数値予報ルーチン用途の定型的な処理を簡潔に記述することができる。また、構文構造がシンプルであり、構文解析器を相対的に低コストで実装することができる。現在では jcl2sh.rb という Ruby スクリプトで JCL からシェルスクリプトへの変換を行い、変換後のシェルスクリプトでジョブ制御を行うという運用を行っている。

さらに、JCL は構文解析の時点でジョブが必要とする実行プログラムと、実行時に必要な引継ぎデータ、定数ファイルの一覧を得ることができる。このことは、ジョブに必要な各種ファイルの用意を自動化できることを意味し、実行時に必要な引継ぎデータを一括して準備するシェルスクリプト（全てのジョブに先駆けて実行されることから START と呼ぶ）を自動で生成できるようになっている [6]。

PBF (Program Build File-format)

PBF は実行プログラムをソースコードから生成するルールを記述するため、気象庁が開発した言語である。

実行プログラムを一定の規則に基づいてソースコードから生成する手法としては、Makefile を用いる方法が一般的である。実際に第 7 世代数値解析予報システムでは、数値予報ルーチンの実行プログラムの生成に Makefile を利用していた (平 2001)。一方で、数値予報ルーチンでは実行プログラムのディレクトリ構成や利用するコンパイラなどに運用上一定のルールを定めている。このことからシェルスクリプトの事例と同様、ユーザが複雑な生成規則を利用できることは管理上望ましくない。そこで複雑な生成規則を抑制するとともに、プログラム生成規則に関する各要素を簡単に取り出せるようにすることを目的に、第 8 世代スーパーコンピュータシステムの運用に向けて開発されたのが PBF である。

JCL からシェルスクリプトを生成できるように、PBF からは Makefile を生成することができ、生成された Makefile を用いて数値予報ルーチンの実行プログラムがコンパイルされる。また、PBF は Makefile と同様にコンパイル時に必要なソースコード間の依存関係を解決することが可能であり、PBF があれば実行プログラムを生成するために必要なソースコード一覧を自動的に作成することができる。

この他、数値予報ルーチンでは実行プログラムをスクリプト言語で書くこともできる。スクリプト言語で実行プログラムを作成する場合はコンパイル不要であるが、数値予報ルーチンで利用する実行プログラムを一律に管理するという観点から、利用するスクリプトファイルを PBF に記載することになっている。このた

[6] 数値予報ルーチンの場合、正確には直接 JCL から START を生成するのではなく、一度 JCL の内容を RENS に登録した後、db2auto.rb で START 用の JCL を出力し、jcl2sh.rb で START を作成するという手順を踏む。

めスクリプト言語で記載された実行プログラムについても、PBF を通じて必要なファイルが把握できるようになっている。

JDF (Job Definition File)

JCL がジョブの動作を記述する言語であったのに対し、JDF は複数のジョブが集まったジョブグループ (JG: Job Group) の動作やジョブの環境を記述するために気象庁が開発した言語である。

ジョブと JG の関係の一例として全球サイクル解析の例を挙げる。全球サイクル解析では以下のような処理を行っている。

1. 解析に必要な入力データを集める
2. 前回の解析結果から予報を実行して第一推定値を作成する
3. 観測データの品質管理を行う
4. 品質管理済みデータと第一推定値を用いて 4 次元変分法によるデータ同化を行う
5. 結果を保存する

個々の処理に相当するのがジョブ、これらの処理全体が JG に相当する [7]。ジョブが必要とする環境はそれぞれ異なっており、4 次元変分法の実行や予報の実行では複数のノードを利用した大規模なプログラムを十数分間実行する必要があるが、観測データの品質管理ではノード数の小さな数分程度のプログラムを並列に多数実行することになる。こうしたノード数や実行時間などのジョブに関する資源割り当てや、各ジョブの実行順序などの依存関係を記述するのが JDF である。

さらに、JG やジョブの依存関係を制御してジョブを実行することができるジョブスケジューラ ROSE（第 2.10.2 項）を数値予報課プログラム班が主体となって開発を進めており、JDF はジョブスケジューリングそのものを記述できる言語として、利用目的が拡大している。

これらのツールはいずれも自主開発の言語で構文解析器が整備されており、設定ファイルから依存するファイルを自動で解決できる特徴がある。こうした機能を NAPEX に導入することで、多くの処理が自動化され管理コストの削減に繋がった。

(3) ファイル管理にバージョン管理システムを利用する

実験の各種設定はデータベース上に保存されるが、JCL, PBF, JDF といった各種ファイルの内容をデータベース上に直接保持すると、レコード数が膨大となり問合せ時のレスポンス低下によって実験構築のボトルネックとなることが懸念される。そこで NAPEX ではデータベース上に直接ファイルの内容を登録することは行わず、バージョン管理システム Subversion のリポジトリにコミットする実装としている [8]。まず、実験番号とユーザ ID の情報を元に Subversion のリポジトリに新規のブランチを作成し、このブランチに必要なファイルをコミットする。次に、ブランチのパスとコミット時に与えられたリビジョン番号を実験番号・枝番と紐付ける形でデータベースに登録する。これによってデータベースとリポジトリが関連付けられることになる。コミットするファイルは、前述の JCL, PBF, JDF に加え、PBF が利用するソースコードとスクリプトファイル、プログラム実行時に必要となる定数ファイルも対象となる。

定数ファイルの多くはバイナリ形式であり、ファイルサイズが大きいという特徴がある。Subversion はテキストファイルについては差分情報が保存されるが、バイナリファイルは差分ではなくファイル全体が直接保存されるため、サイズが大きなファイルを複数コミットすると、リポジトリの実体サイズが急増する危険性がある。幸い、数値予報システムで利用する定数ファイルの多くは、他の実験で利用した定数ファイルをそのまま利用することが多い。そこで NAPEX では、他の実験を引用して実験を行う場合、明示的に変更するファイル以外はリポジトリにコミットせず、過去のデータベースのレコードをそのまま流用する仕様としている。これにより、直接リポジトリにコミットする機会は僅かとなり、実体サイズが急増する危険性は低くなっている。

(4) 実験の実行に独自開発のジョブスケジューラを用いる

JDF の項目で述べたが、数値予報課では独自開発のジョブスケジューラ「ROSE」の整備が進められており（第 2.10.2 項）、NAPEX9 では実験用のジョブスケジューラとして採用している。

NAPEX8 までのジョブスケジューリングは、数値予報システムによって手法が異なっており、全球数値予報システムや週間 EPS では欧州中期予報センターが開発した SMS (Supervisor Monitor Scheduler) をジョブスケジューラとして採用し、メソ数値予報システムではシェルスクリプトでジョブスケジューリングを行っていた。しかし、数値予報ルーチンと NAPEX で JG を別途管理する必要があったこと、SMS の開発が既に停止していること、各システムで統一的なスケジューリングが利用できる方が複数システムを利用するユーザにとって望ましいことなどを踏まえ、NAPEX9 からは自主開発の ROSE を統一的に採用することとなった。ROSE への移行によって、数値予報ルーチンで利用されている設定をある程度流用して JG の設定を行うことができるようになり、この点でも NAPEX 管理にかかるコストを抑えることができるようになっている。

現在、ROSE はバッチジョブのキューイングで

[7] 説明のために簡略化しているが、実際はこれより複雑なジョブ構成となっている。例えば観測データの品質管理は、一つのジョブではなく複数のジョブに分割して実行している。

[8] 数値予報ルーチンでも同様である。

LoadLeveler を始めとする複数のキューイングシステムに対応しており、移植性がかなり高くなっている。これによって、多様な機器で数値予報システムを実行できると期待されている。現在の ROSE は BCP サーバ（第 2.10.2 項）及び NAPEX で利用されているが、2018 年度に運用開始が予定されている第 10 世代スーパーコンピュータシステムでは数値予報ルーチンの運用にも利用される予定となっており、今後移植される NAPEX10 でも引き続き ROSE が利用される予定である。

3.2.4 まとめ

現在の NAPEX9 では中心となる基本思想を最初に設定した上で、必要となる各種技術を組み合わせて実装を行った。これによって NAPEX8 で課題とされていた点の多くが改善され、より利便性の高い実験システムとなっている。加えて数値予報ルーチンとの互換性が向上することによって、NAPEX 自体の管理にかかるコストを低減するとともに、数値予報ルーチンとの相互利用も以前より容易になっている。

参考文献

中山博義, 2005: 新数値解析予報システムについて. 平成 17 年度数値予報研修テキスト, 気象庁予報部, 10–11.

西尾利一, 2011: 計算機（スーパーコンピュータシステム）. 平成 23 年度数値予報研修テキスト, 気象庁予報部, 68–70.

大野木和敏, 入口武史, 2003: 数値解析予報実験システム NAPEX. 気象庁測候時報, **70**, 171–187.

平隆介, 2001: NAPS の数値予報運用スケジュールと実行システム (3.4). 気象庁測候時報, **68**, 53–56.

3.3 NAPEX の管理 [1]

3.3.1 はじめに

NAPEX は数値予報システムの開発で広汎に利用されていることもあり、適切な管理が重要となっている。数値予報課では NAPEX の管理を目的として 2002 年度から NAPEX 管理グループを組織し、NAPEX の管理に係る作業の調整を行っている (大野木・入口 2003)。現在の参加メンバーは、NAPEX のデータベースや実験設定ツールといった NAPEX 環境の開発を担当しているプログラム班の担当者、NAPEX 環境で動作させる数値予報システム（NAPEX モデルと呼ぶ）の管理を担当している数値予報班基盤整備グループの担当者、数値予報班の各グループ（全球・台風グループ、メソモデルグループ、観測データ処理グループ）の担当者及びアプリケーション班の担当者となっている。

NAPEX の管理は NAPEX 環境の管理と NAPEX モデルの管理に大別される。NAPEX 環境はスーパーコンピュータシステムの更新に連動して開発するため、新しいスーパーコンピュータシステムへの移植が終わると基本的に不具合修正を除き大規模な開発は行われない。それに対し、NAPEX モデルは数値予報システムの更新が計画的に実施されているため、定期的にメンテナンスを行うことが必要となっており、スーパーコンピュータシステム更新時を除き、NAPEX 管理作業の大半は NAPEX モデルの管理が占めることになる。このため本節では NAPEX モデルの管理を中心に解説する。

3.3.2 NAPEX モデルの管理

NAPEX8 までは NAPEX モデルの管理に大きなコストがかかっており、現業的に実行される数値予報システムである数値予報ルーチンの変更申請とは別に、NAPEX モデルの変更にも申請が必要であった。NAPEX9 では NAPEX モデルと数値予報ルーチンとの互換性が向上したこともあり（第 3.2 節）、NAPEX モデルの変更申請を廃止し、数値予報ルーチンの変更申請の状況を鑑みながら数値予報班内での協議を踏まえ、NAPEX 管理グループの担当者が計画的に更新作業を行っている。2017 年 1 月時点での NAPEX モデル一覧を表 3.3.1 に示す。

数値予報システムに変更が生じた際は、変更内容を対象となる NAPEX モデルに反映させる作業が必要となる。作業は大まかに、引継ぎデータの確認と、NAPEX モデル本体の更新の二つに分かれる。

(1) 引継ぎデータの確認

NAPEX9 では JCL（第 3.2.3 項 (2) 参照）の機能を用いて実験開始時に必要な引継ぎデータを自動で取得することができるため、多くの場合は特に設定を行う必要がない。ただし、数値予報システムで新規に利用開始されるデータについては、自動取得の対象となるように登録作業を行う必要があり、こうした場合は NAPEX 管理グループの担当者が登録作業を実施している。

NAPEX 特有の事情に由来する作業もある。図 3.3.1 は数値予報ルーチンと NAPEX モデルでの最新データの引継ぎを示した概念図である。数値予報ルーチンの場合、実行時にどのデータが最新のデータであるかは、ほぼ自明である。しかし、NAPEX モデルの場合は過去の実験を行う必要があるため、どの時刻のデータが当時の最新データだったかを判別しなければならない。この判別には NAPEX データベースに登録された実行時刻の情報が利用されるため、必要に応じて NAPEX 管理グループの担当者が時刻の前後判別に関する設定を調整する必要がある。

表 3.3.1 2017 年 1 月時点の NAPEX モデル一覧。サポートが終了したものは除く。

NAPEX モデル	内容
Dc	観測データデコード処理
Da	全球サイクル解析
Df	全球サイクル解析からの予報
Ea	全球速報解析
Ef	全球速報解析からの予報
Vn	全球検証スコア作成
Ma	メソ解析
Mf	メソ予報
La	局地解析
Lf	局地予報

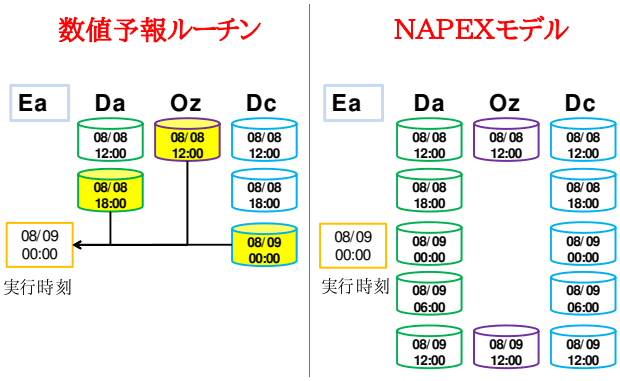

図 3.3.1 数値予報ルーチンと NAPEX モデルにおける最新データの引継ぎ概念図。Ea の実行に Da, Oz（オゾン解析）, Dc の最新データが必要であると仮定する。数値予報ルーチンはリアルタイムで実行するため、Da, Oz, Dc の最新データが黄色で示したデータであると容易にわかる。しかし、NAPEX モデルは過去の実験を行うため、当時の最新データがどの時刻のデータなのかを判別しなければならない。

[1] 雁津 克彦

```
% if NAPEX
  dd out nusdas=17 data=jpp:/jgdir/Fcst/fcst_phy3m_llpp.nus &
                   nusdef=jpp:/const/Gnaps/<%=prefix1%>nusdef/Comm/<%=prefix2%>phy3m_llpp.def
  dd out nusdas=18 data=jpp:/jgdir/Fcst/fcst_phy2m_ll_A.nus &
                   nusdef=jpp:/const/Gnaps/<%=prefix1%>nusdef/Comm/<%=prefix2%>phy2m_ll.def
  dd out nusdas=19 data=jpp:/jgdir/Fcst/fcst_physub_ll_A.nus &
                   nusdef=jpp:/const/Gnaps/<%=prefix1%>nusdef/Comm/<%=prefix2%>physub_ll.def
% end
```

図 3.3.2 Ef での JCL による NAPEX 専用処理の記載例。`if NAPEX` から `end` までの処理は、NAPEX で実行する場合だけ利用される。この例では出力するファイルを追加している。

(2) NAPEX モデル本体の更新

NAPEX モデルは基本的に数値予報ルーチンと同じものが登録される。基本的に、と述べたのは実際には異なる部分が存在するためである。

数値予報ルーチンを実行する場合は、障害などで観測データや初期値のデータが存在しない場合でも、システムが停止することなく動作することが求められる。こうした場合、データの出力自体は行われるが本来とは異なる予測特性を持つデータが出力される。しかし、実験では新たに開発した数値予報システムが現在の数値予報システムと比較して統計的に改善されているか、あるいは予測特性にどういった変化が生じているかを評価することが重要で、予測特性が異なるデータの混入は望ましくない。また、実行に異常が生じた場合は、異常時の状態が上書きされないように実験を停止して、数値予報システムの安定化のために、異常の原因について調査することも重要である。したがって、数値予報ルーチンとは逆に実験が継続しないことが望ましい。

実行時に異常が検知された場合、数値予報ルーチンでは関係者に警告メールを送り、後続処理を行うなど障害対応のための構成が不可欠であるが、こうした処理も実験では不要である。業務用のデータ配信処理が含まれる場合も、誤って実行されないよう停止しておく必要がある。この他、開発ではモデルを改善するための調査として数値予報ルーチンでは出力されない情報を追加で出力したり、本来とは異なる調査用のモジュールを実行する場合がある。NAPEX モデルでは、こうした実験用の設定も追加しておく必要がある。

このように、数値予報ルーチンと NAPEX モデルで異なる処理を行うために、NAPEX9 では数値予報ルーチン用の設定と NAPEX モデル用の設定を自動的に切り替えられるよう配慮している。参考として Ef での JCL の記述例を図 3.3.2 に挙げる。この例では NAPEX モデルの場合だけ特定のファイルを出力する処理を記述しているが、JCL に if NAPEX という文を挿入することで一連の処理が NAPEX モデルの場合だけ利用されるようになっている。こうした機能を活用することで、数値予報ルーチンと NAPEX モデルとで異なるシェルスクリプトが生成され、適切に実験専用の処理を実行できるようになっている。また、数値予報ルーチンには存在しない実験専用の定数ファイルや実行プログラムを追加する場合には、登録したファイルが NAPEX 専用のファイルであることをデータベース上のフラグの有無で判別できるようになっている。

数値予報システムの更新では新規に機能が追加されることがあるが、数値予報ルーチンの処理だけ記述され、NAPEX モデル用の処理が含まれない場合がある。こうした際は、NAPEX モデル用の処理と自動判別のための設定を NAPEX 管理グループの担当者が手作業で導入するなどして、ユーザが誤って数値予報ルーチン用の処理を実行しないよう、NAPEX モデルの管理を行っている。

3.3.3 コントロールデータと雛形実験設定

前項でも述べたが、実験では新たに開発した数値予報システムが現在の数値予報システムと比較して統計的に改善されているか、あるいは予測特性にどういった変化が生じているかを評価することが重要である。評価時の比較対象としては一見すると現業的に実行されている数値予報ルーチンの結果が利用できると考えられる。しかし多くの場合、数値予報ルーチンの結果は比較対象として適切ではない。

統計的に改善しているかを確認するためには、比較対象の予測特性が均質である必要がある。ところが、数値予報ルーチンは開発成果の反映や不具合修正などで随時変更が行われるため、程度に差はあれ変更の前後で予測特性に変化が生じることになる [2]。こうした理由により数値予報ルーチンの予測特性は一定とならず、比較対象として適切ではない。

そこで NAPEX では同じ数値予報システムを用いて過去の一定期間を計算し、予測特性が均質なデータを作成している。このデータのことをコントロールデータと呼ぶ。開発者が NAPEX で実験した結果を評価する際はコントロールデータとの比較を行う。このため開発者は、コントロールデータが作成された期間と同じ期間で実験を行うことが求められるが、過去の実験を行う場合、特に解析実験を行う場合は観測データの利用で煩雑な設定が必要となる場合がある。利用開始から間もない観測データの場合には過去のデータが存

[2] この他、障害で初期値や観測値などのデータが存在しない場合も、業務継続のための代替処理が行われ、通常とは異なる予測特性を持つ結果が作成されるが、頻度としては稀。

在しない、又は存在しても通常と異なる場所に保存されていることがある。こうした場合、過去のデータを適切に利用するための設定を実験に組み込んでおく必要があるが、設定すべき項目が多くミスも生じやすい。そこで、過去の観測データを適切に利用できる実験をNAPEX 管理グループの担当者が予め用意し、開発者が設定を行わなくて済むよう簡素化を図っている。このように、過去の観測データに関する設定を施し、広く開発者に利用されることを想定した実験のことを雛形実験と呼ぶ。前述のコントロールデータは、この雛形実験をベースに実行することで作成されている [3]。

3.3.4 まとめ

NAPEX モデルの管理においては、最新の数値予報システムを用いて実験できるよう NAPEX モデルを整備するとともに、開発者が実験を行う上で標準的に利用するコントロールデータと雛形実験の提供を適宜行っている。

NAPEX モデルの管理作業とユーザの開発の流れを時系列でまとめると次のようになる。

1. 開発者が作成・変更した数値予報システムを、数値予報ルーチンの担当者が数値予報ルーチンに反映する
2. NAPEX 管理グループの担当者が、反映後の数値予報ルーチンを NAPEX 環境用に修正して NAPEX モデルとして登録する [4]
3. NAPEX 管理グループの担当者が、NAPEX モデルを過去の実験期間でも実行できるように修正した雛形実験を作成する
4. NAPEX 管理グループの担当者が、雛形実験を実行してコントロールデータを作成する
5. 開発者は雛形実験を利用して自分の実験を構築・実行し、実験結果とコントロールデータの比較を行う
6. 数値予報課内で新たな数値予報システムを数値予報ルーチンに導入することが合意されれば、開発者は 1. を行うための申請を行う

このように、現在では数値予報モデル開発において、NAPEX モデルの管理業務が開発サイクルの一部に組み込まれており、効率的な開発を行うためにも適切な NAPEX モデルの管理が重要となっている。

[3] Da, Df, Ea, Ef などでは、雛形実験で保存するデータ種別をコントロールデータより少なく設定している。これは大量のデータを出力する設定で全ユーザが実験を行うと、ストレージを逼迫するおそれがあるためである。こうした事情から、コントロールデータを作成する実験と雛形実験の設定には異なる箇所が存在する。

[4] 開発スケジュールによっては、数値予報ルーチンに反映する前の段階で NAPEX モデルを構築する場合もある。

参考文献

大野木和敏, 入口武史, 2003: 数値解析予報実験システム NAPEX. 気象庁測候時報, **70**, 171–187.

3.4 NAPEX による実験の実行手順 [1]

3.4.1 はじめに

本章でこれまで述べられてきたように、開発者は NAPEX を利用することで、単発実験やサイクル実験を容易に実行、再現することが可能となっている。

本節では、実際に NAPEX を利用するに当たっての方法などについて述べる。まず第 3.4.2 項で実験をそのまま再現して実行するための手順について述べ、第 3.4.3 項でさらにそこから変更を加えた実験を設定する方法について示す。第 3.4.4 項では、設定した実験を実行するためのジョブスケジューラ ROSE（詳細は第 2.10.2 項参照）について、ユーザの視点から説明する。第 3.4.5 項に本節のまとめを記述する。

3.4.2 コントロール実験の再現

数値予報課で数値実験を行う場合、通常は基準となる実験（コントロール実験）と何らかの変更を加えた実験（テスト実験）の差を比較、評価する。テスト実験を行う場合、そもそも基準となるコントロール実験を再現することができなければ、コントロール実験との正しい比較を行うことはできないだろう。

第 3.2.3 項で述べられているように、NAPEX では各実験に対して実験番号と枝番が一意に設定されるが、それはコントロール実験についても同様である。コントロール実験の実験番号と枝番についてはイントラネット内のウェブページにまとめられており、開発者は基準としたいコントロール実験の実験番号と枝番を容易に取得することができる。

あるコントロール実験をそのまま再現して実行する手順は以下の通りである [2]。

手順 1. 参照するコントロール実験の実験番号と枝番のほか、自分のユーザ ID、実験登録時の作業ディレクトリ、実行プログラムや定数ファイル（地形データ、パラメータファイルなど、実験実行時刻で内容の変わらない入力ファイル）の格納ディレクトリなどを設定ファイルに記述し、基本初期設定用のコマンドを実行することで、実験を初期化する。

手順 2. 実験の期間、実行ディレクトリ、出力ファイルの保存ディレクトリなどを設定ファイルに記述し、実験設定用コマンドを実行することで、内容をリポジトリ [3] に登録する。

手順 3. 実行プログラムや定数ファイル、ジョブを実行するシェルスクリプトを、それぞれコンパイル、コピーまたは作成するための設定用コマンドを実行することで、実験をスーパーコンピュータ上にセットアップする。

手順 4. 実験をジョブスケジューラ ROSE に登録するためのコマンドを実行する。

手順 5. ROSE に登録した実験を開始するためのコマンドを実行する。

指定した実験番号や各種ディレクトリの設定等が適切であれば、実行結果は実行時間等のログを除いて元のコントロール実験と一致するはずである。

3.4.3 テスト実験の実行

初めて NAPEX に触れて、動作を試してみるような場合を除いて、前項の様にコントロール実験をそのまま実行するようなことはほとんどないだろう。実際に開発者が変更を加えたテスト実験を行うためには、基本的に前項の手順 2. で、変更についての記述を設定ファイルに追加すればよい。

以下では、実際に変更を加える方法について概説する。

(1) 実行プログラムや定数ファイルの差し替え

入出力ファイルの名称が変わらないような実行プログラムの変更、あるいは定数ファイルの差し替えの場合には、元のファイルの配置のパスと、PBF[4] や定数ファイルの格納ディレクトリをそれぞれ設定ファイルで指定すればよい。

手順 2. の実験設定用コマンドを実行すると、実験の枝番の数字が 1 つ増える。また、登録した実行プログラムのソースコードと PBF、定数ファイルがリポジトリに登録される。手順 3. で配置されるファイルはリポジトリに登録されたものから取得されるため、登録時に格納していたファイルが後に削除されても問題なく設定を再現することができる。

変更を繰り返す場合には、再度手順 2. を実行する。これにより、実験の枝番の数字がさらに 1 つ増える。手順 3. から手順 5. までは、いずれも適切な実験番号と枝番を指定する必要がある。

(2) 実行プログラムや定数ファイルの追加など

入出力ファイルの変更を伴うような実行プログラムの修正を行う場合には、バッチ処理を記述する JCL[4] ファイルの変更を行う必要がある。JCL ファイルで、変更した実行プログラム名や入出力ファイルの記述を修正する。手順 2. の設定ファイルには追加する PBF と定数ファイルの記述を加え、さらに修正した JCL ファイルの格納ディレクトリのほか、実行されるジョブの種類や時刻について指定する。さらに保存データの追加を行う場合には、データのパスのほか、長期保存するか（データバンクストレージに圧縮して保存する）、短期保存するか（NFS (Network File System) に基本的に非圧縮で保存する）の設定などをそれぞれ記述する。

[1] 江河 拓夢

[2] このほか、NAPEX を初めて実行する場合のみ、リモートサーバへのアクセス設定や、スケジューラの環境設定が必要となる。

[3] Subversion が用いられているが、ユーザは Subversion のコマンドを直接実行することはない。

[4] 詳細は第 3.2.3 項参照。

(3) ジョブの追加など

ジョブの追加のほか、スーパーコンピュータの使用ノード数の変更、実行時のジョブフローの依存関係の変更などを行う場合には、それらを記述するための JDF[4] ファイルの変更を行う。手順 2. の設定ファイルには、変更した JDF ファイルの格納ディレクトリのほか、ジョブグループ [5] の実行間隔などを設定する。

(4) 新規の引継ぎデータの登録

新規の観測データなど、実験で使用する外部データ（引継ぎデータ）を変更する場合には、手順 2. に先駆けて、データの格納ディレクトリなどをデータベースに登録する必要がある。登録用の設定ファイルには、データを格納しているディレクトリのほか、ファイル名、データの保存期間、圧縮の有無などを記述する。引継ぎデータ登録用のコマンドを実行すると、ここで登録したファイルに一意な番号が割り当てられる。この番号を、手順 2. のリポジトリ登録時に使用する。

(5) 任意の実験の引用

ここまでコントロール実験から変更を行うための方法について述べてきたが、あるテスト実験から変更を行うことも同様に可能である。設定された各種のファイルは全てリポジトリに登録されているので、テスト実験のために変更・追加したファイルを全て取得することが可能となっている。過去に実行された実験を再現することが容易であり、過去の NAPEX と比べて開発者の利便性が格段に向上している（第 3.2.3 項参照）。

3.4.4 ROSE の利用について

実験が開始された後は、ROSE によりジョブの投入、制御が行われる。ROSE の GUI (graphical user interface) には当初は Adobe AIR が用いられていた (平原 2012) が、保守性の観点から現在はウェブベースに移行している。モデル開発者は、GUI だけで様々な操作を行うことができる。

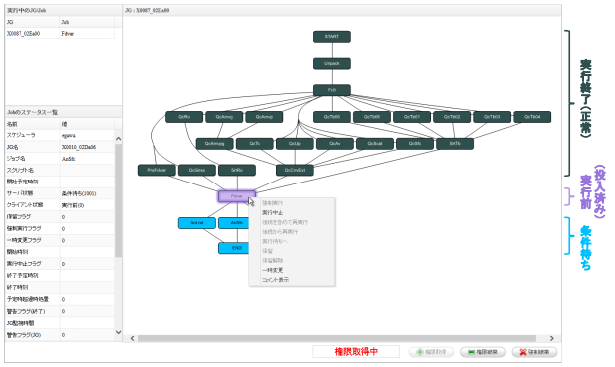

図 3.4.1 ROSE のウェブベースでの GUI の表示例（枠内）。枠内左側にジョブのステータスの一覧表、右側にジョブグループの進行状況のフローチャートなどが表示されている。枠外右側に進行状況の凡例を示す。

[5] 特定の時刻に行われるジョブの集合。

図 3.4.1 に GUI でのジョブグループの表示例を示す。右側のフローチャートの部品はジョブとその依存関係を表しており、ジョブの状態が色別で表示される。ユーザは視覚的にジョブの進行状況を確認することができる。図の中央付近に表示されているポップアップは、ジョブをクリックすることで表示される。ジョブの実行状況によっても異なるが、ここでジョブの実行の中止、再実行、保留などの制御を行ったり、実験設定の一時変更を行うことができる。図の左側には、ジョブのステータス一覧が表示されている。これにより、ジョブ投入時の環境変数などの設定や、ジョブの終了ステータスを確認することができる。図の下部には、赤い文字で権限取得中と表示されているが、この権限はスケジューラに対して排他処理を行うために設定されている。様々なジョブに対する制御や設定の変更を行う場合には権限を取得してから作業を開始する。

なお、GUI と同様な操作を CUI (character user interface) で行うことも可能である。通常の操作は GUI ベースで実行できるので、使用するかは任意となっている。例えば第 3.4.2 項の手順 5. は GUI と CUI のどちらでも行うことができるが、このケースでは CUI で実行する方が容易である。

3.4.5 まとめ

本節では、まず実験をそのまま再現するための手順について述べ、さらにそこから変更を加えた実験を設定する方法について示した。これらの大まかな流れは過去の NAPEX での設定と大きく変わらないが、使用するファイルをリポジトリに登録するようになったのは、実験の再現性の確保の点から大きく進歩したといえる。過去の NAPEX では、実行プログラムをただ特定の場所に配置するだけであったため、ソースコードが不明であったり、そもそも実行プログラムが削除される場合もあった。現 NAPEX では他の開発者が実験で使用した実行プログラムのソースコードをリポジトリから取得し、さらに変更を加えることも容易である。

ジョブスケジューラ ROSE の利用についてはユーザの視点から説明した。自主開発の ROSE を用いることで、数値予報ルーチンと実験の親和性が上がっているほか、スケジューラの機能に関する要望も挙げやすくなり、開発者の利便性を向上させやすくなったといえる。

NAPEX は数値予報モデルの開発において必要不可欠なツールになっている。今後もより良いものにするために改良を続けていくことが必要である。

参考文献

平原洋一, 2012: 気象庁における数値解析予報実験システムとバージョン管理. 地球流体データ解析・数値計算ワークショップ, https://www.gfd-dennou.org/library/davis-workshop/2012-12-12/hirahara_20121212.pdf.

3.5 NAPEX の利用の広がり [1]

3.5.1 はじめに

NAPEX は数値予報課が主体となって開発したが、近年では数値予報課の枠を超えて他課室でも利用されている。本節では数値予報課以外での利用実績について簡単に紹介するとともに、第 10 世代スーパーコンピュータシステムに向けた取組についても概説する。

3.5.2 気候情報課での利用

全球数値予報システム及び全球アンサンブル予報システム（全球 EPS）の開発は、数値予報課と気候情報課が開発項目を分担・連携する体制で行われている (経田 2016)。このような背景から、数値予報課だけでなく気候情報課においても実験システムとして NAPEX が採用されている。両者が共通の開発基盤を利用することで、数値予報課と気候情報課の間で開発成果を速やかに共有できるようになっている。気候情報課での NAPEX の利用は数値予報課で実施しているような全球 EPS のコントロールメンバー（摂動なしメンバー）予報実験やアンサンブル予報実験の実施のみならず、再予報、1 年積分共通評価ツール、さらには長期再解析の実験システムとしても活用されている。

再予報とは、過去の多数の事例について調査対象となる同一のモデルで予測計算を実行することである。一般にモデルによる予測では、モデルの不完全性などにより積分が進むとともに誤差が時間発展し、現実的な場から離れていく。このため、予測期間が長い場合には、予測精度を事前に把握するとともにモデルの誤差を補正して予測結果を解釈する必要が生じる。再予報により過去の多数の事例の予測計算を行い、検証を通じて予測誤差特性に関する資料を得る (高谷 2012)。気候情報課では 1 か月及び季節アンサンブル予報システムの更新の際、それぞれ必要な再予報を実施し、予測値の特性の把握やバイアス補正、ガイダンスの作成などに役立てている（たとえば金浜ほか 2017; 高谷・石川 2015)。実行にあたっては、特定の期間の特定の計算開始対象日（たとえば金浜ほか 2017 では、期間として 1981~2012 年の 32 年間、対象日として毎月 10 日、20 日、最終日の 3 日× 12 か月＝年間 36 日）を初期値とした予報を実行する必要がある。全球 EPS では、一年のうちの特定の初期日（たとえば 1 月 10 日）の計算を 1 年刻みで全期間（32 年間）実行する設定を NAPEX で作成し、これを全ての初期日（1 月 10 日、1 月 20 日、…）で実行することにより実現している。NAPEX により、共通の実験設定を容易に複数の実験に対応させられることから、実験設定の労力軽減につながっている。

1 年積分共通評価ツールは、数値予報モデルの内容（力学過程や物理過程）を変更した際に影響を調査するためのツールで、特定期間で積分を実施し、予測特性に問題がないかを調べるために用いられる。これまでは個々の開発者が自身で環境を構築する必要があったが、NAPEX で統一的に実施できるよう開発を行った。これにより、開発者が同様の実験を行う際に一から環境構築をする必要がなくなると共に、意図的に変更を加えた部分以外は共通の設定の実験を容易に行えるようになった。設定の違う部分が明確になることから、別の開発者の実験結果との比較も容易に行えるようになった。

長期再解析とは、数十年以上という長期のスパンで、最新のデータ同化システムと、当時の観測データを用いて均質な精度を持つ解析値を作成することである。現在準備が進められている次期長期再解析では実際に再解析プロダクトを作成する「本計算」を NAPEX を用いて実施することが予定されている。その際、利用可能な観測データ（特に衛星観測データ）は時代と共に変遷していく（たとえば古林ほか 2015）ことから、過去の期間の解析を行うためには、期間毎に異なる観測データを同一のデータ同化システムで利用していくことが要求される。NAPEX は本来府県天気予報・週間天気予報用途での開発を主目的に開発されてきたことがあり、このような複雑な状況での利用は想定されていなかったが、 対応できるよう気候情報課で検討が進められているところである。

以上のように NAPEX は、府県天気予報・週間天気予報用途だけでなく、 よりタイムスパンの長い数値予報システムの開発においても、活用が進められている。

3.5.3 気象研究所での利用

近年の数値予報システムの高度化により、その精度改善には最新の基礎研究の成果を速やかに現業システム設定での実験に適用できることが望ましく、気象研究所でも現業数値予報システムの改善・改良に資する研究開発が重要となっている。このような背景の下、重点研究課題「全球大気データ同化の高度化に関する研究（平成 23 年度〜27 年度)」の開始を控えていた 2009 年度（平成 21 年度）後半から 2010 年度（平成 22 年度）にかけ、全球大気データ同化の研究を本格的に開始するためのシステムとして、第 8 世代スーパーコンピュータシステム向け NAPEX (NAPEX8) をベースに、第 6 世代気象研究所スーパーコンピュータシステム (MRI-SC6) 向けに NAPEX の移植作業が実施された [2] (MRI-NAPEX6) 。

当時の NAPEX8 環境では既に JCL や PBF といった数値予報ルーチン管理のツールが利用されていたこともあり、これら関連ツールも同時に気象研究所へ移植が行われ、数値予報課と気象研究所との間で共通的な基盤で実験する環境が整備された (大野木ほか 2011)。そ

[1] 雁津 克彦、佐藤 芳昭（地球環境・海洋部 気候情報課)、石橋 俊之（気象研究所 台風研究部）

[2] 幸い MRI-SC6 と第 8 世代スーパーコンピュータシステムは同一系統の計算機であり、NAPEX8 と MRI-NAPEX6 は NAPEX モデルで同じ実行モジュールを利用することができた。ただし MRI-SC6 のスペックの問題で、現業水平解像度での実験ができず、解像度の変更が必要など運用に苦労を伴う点があった。

の後、気象庁本庁では 2012 年 6 月に第 9 世代スーパーコンピュータシステムの運用が開始され、第 9 世代スーパーコンピュータシステム向け NAPEX (NAPEX9) 環境では ROSE やデータベースの利用、引継ぎデータの自動解決や JDF によるノード数・ジョブ依存関係の管理等が実装された (第 3.2 節) 。一方、気象研究所では 2015 年 3 月に第 7 世代気象研究所スーパーコンピュータシステム (MRI-SC7) の運用が開始された。気象研究所中期研究計画（平成 26 年度～平成 30 年度）では、重点研究 (A3)「台風の進路予報・強度解析の精度向上に資する研究」において、引き続き全球数値予報システムを用いた研究開発が実施されており、気象研究所台風研究部の担当者により MRI-NAPEX6 の新システムへの移植・構築が行われた (MRI-NAPEX7)。

MRI-NAPEX7 のうち、NAPEX モデルは NAPEX9 で利用されているもの（第 3.3.2 項参照）を利用して、計算機環境依存への対応が行われた [3]。このため MRI-NAPEX7 の NAPEX モデルは NAPEX9 と同等である。一方、NAPEX 環境は MRI-NAPEX6 の NAPEX 環境を新計算機に対応する形で利用し、NAPEX9 環境で利用開始された ROSE やデータベースによる実験管理機能の移植は行わず、ジョブスケジューラはシェルスクリプトで構築された。これは、気象研究所には数値予報課プログラム班や基盤整備グループのような部署がなく、研究開発と並行して NAPEX 環境の維持を行う必要があり、ROSE やデータベースを導入した場合、ROSE やデータベース自体の維持管理も必要となるため、少人数で利用するにあたって総合的に NAPEX のメンテナンスが容易となるようにこのような構成としている。

こうした構成は現行の気象研究所での研究目的を達成するためには十分な設計であり、不要な設定等を行う必要がないことから、最適化されたシステムとも言うことができる。試験的には NAPEX9 環境が気象研究所スーパーコンピュータシステム上で動作することは確認されており、移植自体は可能である。しかし、現行の MRI-NAPEX7 がユーザニーズを十分満たしていることも注意する必要があり、不要な機能の移植はかえってユーザ利便性を損なうおそれもある。一方で、気象研究所で独自の NAPEX 環境を運用するために、本庁のルーチン変更に追随するコストが高くなるといった弊害もある。他にも、第 3.2 節で紹介した NAPEX9 環境で可能になった各種利点が反映されておらず、引継ぎデータの自動解決や、JDF によるノード数・ジョブ依存関係の管理といった、維持コストを大幅に軽減できるメリットも享受できない。このため、維持コストの低い利用形態を想定しつつ共通的な NAPEX 環境の構築や維持管理体制を作ってゆくことが今後の課題と考えられる。

[3] 現業水平解像度で実験ができないという MRI-SC6 の問題は、MRI-SC7 への移行でスペックが強化され解消したが、今度は MRI-SC7 と第 9 世代スーパーコンピュータシステムが異なる系統の計算機となり、実行モジュール移植時に計算機に依存する箇所の修正が必要となっている。

なお、気象研究所では新たに NAPEX モデルとしてメソ数値予報システムの導入を検討しており、現在移植作業を進めているところである。

3.5.4 次期スーパーコンピュータシステムでの利用

気象庁では第 10 世代スーパーコンピュータシステムを 2018 年から運用する計画で、2017 年 1 月現在各種移行作業に取りかかっている。NAPEX についても第 10 世代スーパーコンピュータシステム向けに仕様検討を進めているところである。次期 NAPEX でも現行の NAPEX と同様に実験再利用性と数値予報ルーチンとの互換性を重視し、

- 全ての実験の設定をデータベースで管理する
- ジョブの管理や実行プログラムの管理に数値予報ルーチン管理の言語を用いる
- ファイル管理にバージョン管理システムを利用する
- 実験の実行に独自開発のジョブスケジューラ「ROSE」を用いる

といった現在の基本的な実装を踏襲する予定である。また、NAPEX9 は利用開始から 4 年が経過し、いくつか課題も指摘されている。主な課題として

- 実験構築にかかる時間の短縮化 [4]
- 実験環境構築時の設定ミス防止強化
- 引継ぎデータ利用の利便性強化

といったものが挙げられる。これらを踏まえ、2016 年度中に数値予報課内の NAPEX 管理グループ会合を通じ、次期 NAPEX の仕様を決定する予定である。

参考文献

金浜貴史, 関口亮平, 足立恭将, 宮岡健吾, 久保勇太郎, 新保明彦, 西村明希生, 2017: 再予報と検証. 平成 28 年度季節予報研修テキスト, 気象庁地球環境・海洋部, 9–47.

古林慎哉, 太田行哉, 原田やよい, 海老田綾貴, 守谷昌己, 小野田浩克, 大野木和敏, 釜堀弘隆, 小林ちあき, 遠藤洋和, 宮岡健吾, 高橋清利, 2015: 気象庁 55 年長期再解析 (JRA-55) の概要. 平成 26 年度季節予報研修テキスト, 気象庁地球環境・海洋部, 66–115.

経田正幸, 2016: 全球アンサンブル予報システムの開発. 数値予報課報告・別冊第 62 号, 気象庁予報部, 52–57.

大野木和敏, 江河拓夢, 小林ちあき, 石橋俊之, 田浦俊太郎, 石元裕史, 釜堀弘隆, 上清直隆, 2011: 全球大気データ同化実験システム NAPEX の気象研究所への移植 ～MRI-NAPEX の構築～. 気象庁測候時報, **78**, 19–29.

高谷祐平, 2012: 新用語解説　再予報・ハインドキャスト. 天気, **59**, 493–495.

高谷祐平, 石川一郎, 2015: 再予報による新システムの評価. 平成 27 年度季節予報研修テキスト, 気象庁地球環境・海洋部, 42–95.

[4] ユーザの設定によっては、コンパイルに 1 時間近くかかる場合があり、NAPEX 環境の改良によって冗長なコンパイルを省略することで時間の短縮が見込まれている。

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

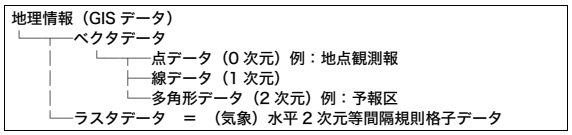

図 4.1.1　地理情報科学におけるデータモデルの分類。

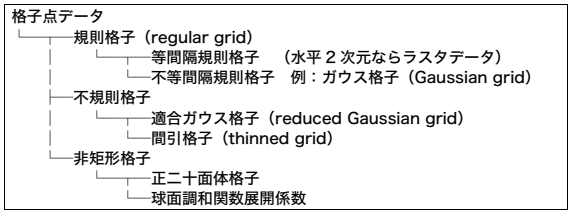

図 4.1.2　気象データにおける格子点データの分類。

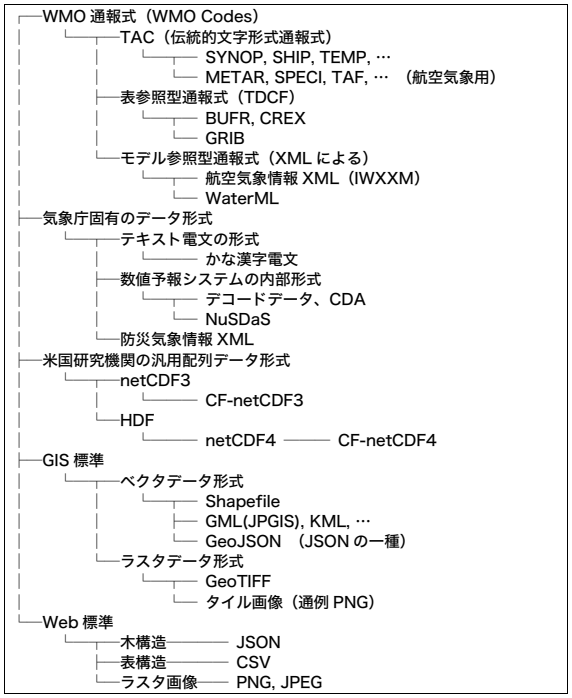

図 4.1.3　多種多様なデータ形式の分類。

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

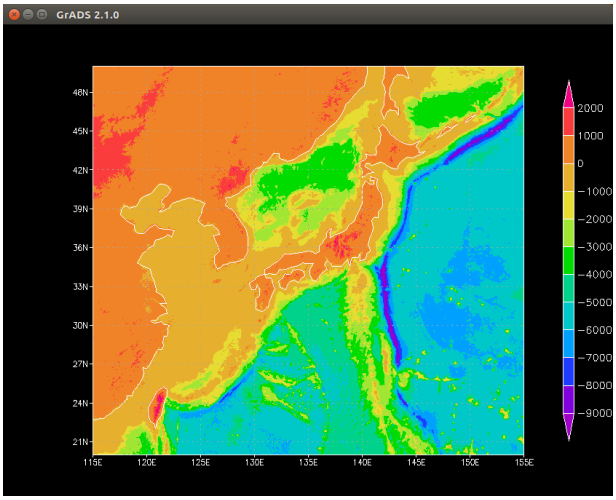

図 4.4.1 GrADS による X Window での表示例。1 分格子の標高データ (ETOPO1) を入力として、日本周辺を緯度経度直交座標で描画している。

```
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。

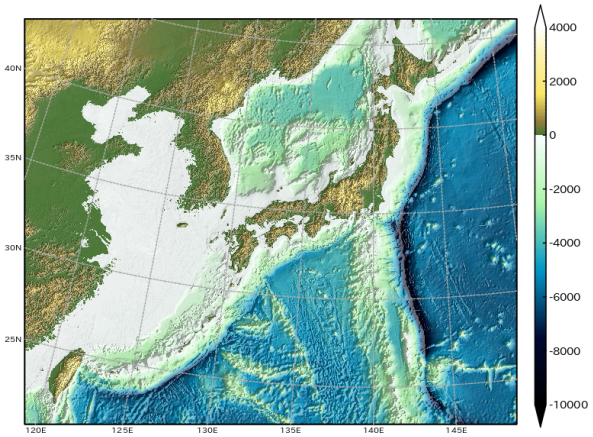

図 4.4.3 GrADS による GSL を駆使した描画の例。入力データは図 4.4.1 と同様であり、標高の描画に加え、標高から計算した地形の勾配と向きを陰影で重ねている。投影法はランベルト正角円錐図法を用いている。

[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。

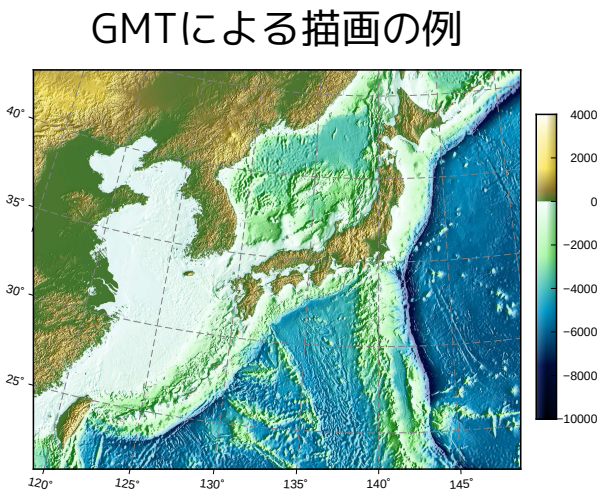

図 4.5.1 GMT による描画の例。入力データは図 4.4.1 と同様であり、ランベルト正角円錐図法で、標高（カラー）とその勾配と向き（陰影）を描画している。図上部の日本語は GMT で出力したものである。

[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。

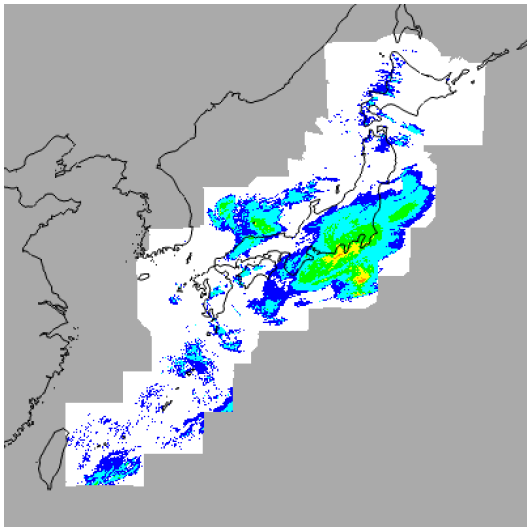

図 4.6.1 TAG の表示例。解析雨量を用いて 2016/08/20 00:00(UTC) の前 3 時間積算雨量を表示。

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

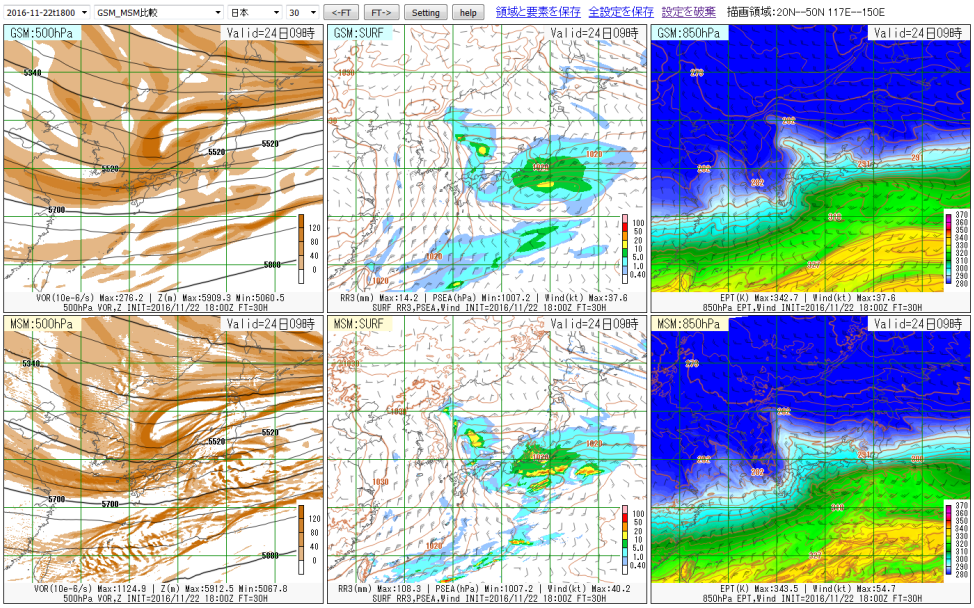

図 4.6.3　ガイダンスモデルモニタ。ユーザは画面上の 6 画面に任意の要素を表示させることができる。マウス操作により領域や時刻を自由に変更でき、表示状態の保存も可能である。この例では上段に GSM を、下段に MSM を表示させている。

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。

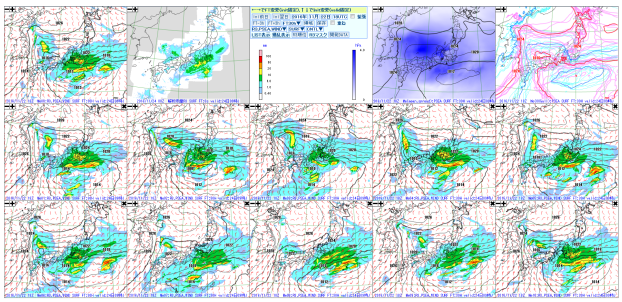

図 4.6.4　MEPS インタラクティブツールと呼ばれる MEPS の結果確認ツール。上段右から二番目の青みがかった画像は気圧のアンサンブル平均とスプレッドを表示しているが、各メンバー右上の×ボタンをクリックすることで対象メンバーを統計対象から外すことができ、統計結果をその場で動的に確認することができる。

(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

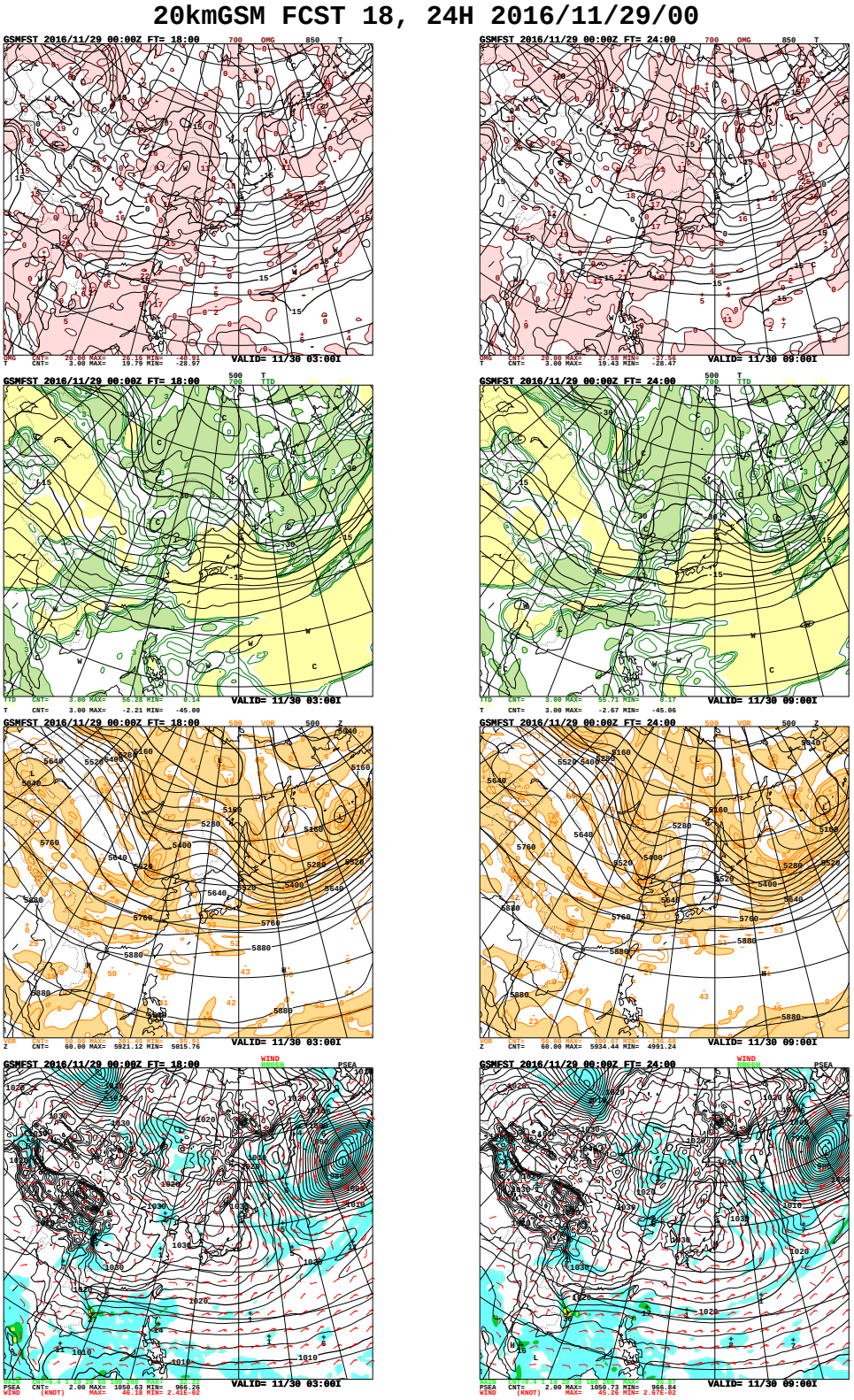

図 4.7.2　PANDAH で描画された GSM 予測のモニタ図の例。最上段： 850 hPa の気温（等値線。極大値、極小値をそれぞれ W, C で表示。）と 700 hPa の鉛直 p 速度（負の部分を桃色に着色。極大値、極小値をそれぞれ+, −で表示。）、2 段目： 500 hPa の気温（等値線。極大値、極小値をそれぞれ W, C で表示。）と 700 hPa の湿数（3 K 以下を緑で、15 K 以上を黄色で着色）、3 段目： 500 hPa のジオポテンシャル高度（等値線。極大値、極小値をそれぞれ H, L で表示。）と渦度（正の部分を橙色に着色。極大値、極小値をそれぞれ+, −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。

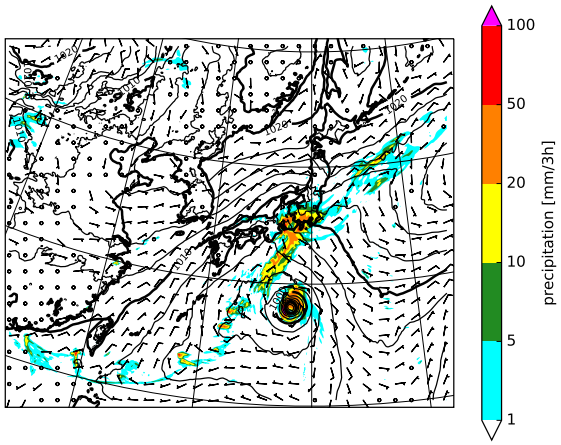

図 4.7.3 Python で描画した NuSDaS データの例。ある時刻の MSM 予測値の前 3 時間降水量（塗り分け、単位：mm/3h）、海面更正気圧（等値線、単位： hPa）、地上風（矢羽、短い矢羽は 5 m/s に対応）。

[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")
```

付録 A　電子計算室報告、同別冊、数値予報課報告・別冊 発行履歴

発行年月	発行号	表題
2017 年（平成 29 年） 3 月	数値予報課報告・別冊第 63 号	数値予報モデル開発のための基盤整備および開発管理
2016 年（平成 28 年） 3 月	数値予報課報告・別冊第 62 号	確率的な気象予測のためのアンサンブル予報の課題と展望
2015 年（平成 27 年） 3 月	数値予報課報告・別冊第 61 号	観測データ利用の現状と課題
2014 年（平成 26 年） 3 月	数値予報課報告・別冊第 60 号	次世代非静力学モデル asuca
2013 年（平成 25 年） 3 月	数値予報課報告・別冊第 59 号	物理過程の改善に向けて (II)
2012 年（平成 24 年） 3 月	数値予報課報告・別冊第 58 号	物理過程の改善に向けて (I)
2011 年（平成 23 年） 3 月	数値予報課報告・別冊第 57 号	データ同化の改善に向けて
2010 年（平成 22 年） 3 月	数値予報課報告・別冊第 56 号	非静力学メソ 4 次元変分法
2009 年（平成 21 年） 3 月	数値予報課報告・別冊第 55 号	全球モデルの課題と展望
2008 年（平成 20 年） 3 月	数値予報課報告・別冊第 54 号	気象庁非静力学モデル II ──現業利用の開始とその後の発展──
2007 年（平成 19 年） 3 月	数値予報課報告・別冊第 53 号	数値予報と衛星データ ──同化の現状と課題──
2006 年（平成 18 年） 3 月	数値予報課報告・別冊第 52 号	アンサンブル技術の短期・中期予報への利用 ──激しい気象現象の予測向上を目指して──
2005 年（平成 17 年） 3 月	数値予報課報告・別冊第 51 号	全球モデル開発プロジェクト (II)
2004 年（平成 16 年） 3 月	数値予報課報告・別冊第 50 号	全球モデル開発プロジェクト (I)
2003 年（平成 15 年） 3 月	数値予報課報告・別冊第 49 号	気象庁非静力学モデル
2002 年（平成 14 年） 3 月	数値予報課報告・別冊第 48 号	変分法データ同化システムの現業化
2000 年（平成 12 年）10 月	数値予報課報告・別冊第 47 号	新しい数値解析予報システム（数値予報解説資料 (33) 平成 12 年度数値予報研修テキスト合併）
2000 年（平成 12 年） 3 月	数値予報課報告・別冊第 46 号	全球モデル開発の現状と展望 ──気象業務の基幹モデルとして──
1999 年（平成 11 年） 3 月	数値予報課報告・別冊第 45 号	数値予報のための衛星データ同化
1998 年（平成 10 年） 3 月	数値予報課報告・別冊第 44 号	メソ数値予報の実用化に向けて
1997 年（平成　9 年） 3 月	数値予報課報告・別冊第 43 号	データ同化の現状と展望
1996 年（平成　8 年） 3 月	数値予報課報告・別冊第 42 号	一ヶ月予報に向けた全球モデルの開発 ──バイアスの小さな予報モデルを目指して──
1994 年（平成　6 年） 9 月	数値予報課報告・別冊第 41 号	数値予報の実際（数値予報解説資料 (27) 平成 6 年度数値予報研修テキスト合併）
1994 年（平成　6 年） 3 月	数値予報課報告・別冊第 40 号	気候監視のための海洋データ同化システム ──大気海洋結合モデルによる季節予報に向けて──
1993 年（平成　5 年） 3 月	数値予報課報告・別冊第 39 号	数値予報とリモートセンシング
1992 年（平成　4 年） 3 月	数値予報課報告・別冊第 38 号	力学的 1ヶ月予報の課題と展望
1991 年（平成　3 年） 3 月	数値予報課報告・別冊第 37 号	狭領域モデルの課題と展望
1990 年（平成　2 年） 3 月	数値予報課報告・別冊第 36 号	気象データと客観解析
1989 年（平成　元年） 3 月	数値予報課報告・別冊第 35 号	力学的長期予報をめざして
1988 年（昭和 63 年） 3 月	数値予報課報告・別冊第 34 号	数値予報モデルの物理過程
1987 年（昭和 62 年） 3 月	数値予報課報告・別冊第 33 号	低緯度の数値予報
1986 年（昭和 61 年） 3 月	数値予報課報告・別冊第 32 号	メソスケール現象と数値予報
1985 年（昭和 60 年） 3 月	電子計算室報告・別冊第 31 号	延長予報に関する最近の話題
1984 年（昭和 59 年） 3 月	電子計算室報告・別冊第 30 号	ノーマル・モード・イニシャリゼーション

発行年月	発行号	表題
1983 年（昭和 58 年） 3 月	電子計算室報告・別冊第 29 号	北半球およびファインメッシュ予報モデル（8L NHM および 10L FLM）と解析システム
1982 年（昭和 57 年） 3 月	電子計算室報告・別冊第 28 号	スペクトル法による数値予報（その原理と実際）
1981 年（昭和 56 年） 3 月	電子計算室報告・別冊第 27 号	数値予報モデルの時間差分スキームと物理過程
1980 年（昭和 55 年） 3 月	電子計算室報告・別冊第 26 号	気象衛星資料と数値予報
1979 年（昭和 54 年） 3 月	電子計算室報告・別冊第 25 号	4 層北半球プリミティブ・モデルの改良について
1978 年（昭和 53 年） 3 月	電子計算室報告・別冊第 24 号	数値予報による延長予報
1977 年（昭和 52 年） 3 月	電子計算室報告・別冊第 23 号	数値予報と天気予報
1976 年（昭和 51 年） 3 月	電子計算室報告・別冊第 22 号	客観解析
1975 年（昭和 50 年） 3 月	電子計算室報告・別冊第 21 号	4 層北半球プリミティブ・モデルについて
1974 年（昭和 49 年） 3 月	電子計算室報告・別冊第 20 号	数値予報特別研修のまとめ
1973 年（昭和 48 年）10 月	電子計算室報告・別冊第 19 号	プリミティブ・モデルについて（数値予報解説資料 (6) 合併）
1973 年（昭和 48 年） 3 月	電子計算室報告・別冊第 18 号	プリミティブ・モデルをめぐって
1972 年（昭和 47 年）10 月	電子計算室報告別冊第 17 号	新しく予報を担当される方のための電計資料の見方（数値予報解説資料 (5) 合併）
1971 年（昭和 46 年）10 月	電子計算室報告別冊第 16 号	じょう乱の構造について（数値予報解説資料 (4) 合併）
1971 年（昭和 46 年） 9 月	電子計算室報告別冊第 15 号	中間規模じょう乱をめぐって
1970 年（昭和 45 年）11 月	電子計算室報告別冊第 14 号	北半球 3 層非地衡風バランス・モデル（数値予報解説資料 (3) 合併）
1969 年（昭和 44 年）10 月	電子計算室報告別冊第 13 号	北半球 3 層非地衡風バランス・モデル（数値予報解説資料 (2) 合併）
1969 年（昭和 44 年） 9 月	電子計算室報告別冊第 12 号	数値予報のはじめ（数値予報解説資料 (1) 合併）
1968 年（昭和 43 年）10 月	電子計算室報告別冊第 11 号	予報技術改善の方向
1968 年（昭和 43 年） 3 月	電子計算室報告別冊第 10 号	数値予報
1966 年（昭和 41 年）10 月	電子計算室報告別冊第 9 号	北半球 4 層傾圧予報について
1965 年（昭和 40 年）11 月	電子計算室報告別冊第 8 号	IUGG 大気科学委員会第一回活動概要報告
1964 年（昭和 39 年） 2 月	電子計算室報告 VIII	
1963 年（昭和 38 年） 6 月	電子計算室報告別冊第 7 号	バロクリニツク大気の性質
1963 年（昭和 38 年） 6 月	電子計算室報告別冊第 6 号	アジア地区のバロクリニツク予報
1962 年（昭和 37 年） 7 月	電子計算室報告別冊第 5 号	北半球バロトロピツク予報
1962 年（昭和 37 年） 6 月	電子計算室報告別冊第 4 号	気象庁電子計算室におけるルーチン傾圧モデルの概要
1961 年（昭和 36 年） 5 月	電子計算室報告 別冊 No.3	500MB 面渦度及びその予報図の利用法
1961 年（昭和 36 年） 3 月	電子計算室報告 VI VII	
1960 年（昭和 35 年） 8 月	電子計算室報告 別冊 No.2	機械でつくる天気図について
1960 年（昭和 35 年） 7 月	電子計算室報告 V	
1960 年（昭和 35 年） 5 月	電子計算室報告 別冊 No.1	渦度分布図の利用法並びに高層天気図の予報えの応用について、1 パラメーターモデルによる上昇速度とその利用法について
1960 年（昭和 35 年） 4 月	電子計算室報告 IV	
1960 年（昭和 35 年） 1 月	電子計算室報告 III	
1959 年（昭和 34 年）10 月	電子計算室報告 II	
1959 年（昭和 34 年） 7 月	電子計算室コータリーレポート	

数値予報モデル開発のための基盤整備および開発管理

数値予報課報告・別冊第 63 号

平成 29 年 3 月 17 日発行

編　集　気象庁予報部 数値予報課

〒 100–8122 東京都千代田区大手町 1–3–4

発　行　気象庁予報部

Copyright © 気象庁予報部 2017　Printed in Japan
著作権法で定める範囲を超えて、無断で転載または複写することを禁止します。

リサイクル適性: A