SCALE Developer Document

Team SCALE

March 20, 2019

Contents

執筆のための注意点

本ドキュメントの目的は、読者が概要を把握するために必要な要点の記述と、詳細を調べる際に必要となる他のドキュメントや参考文献、ソースコードなどへのポインタを示すことである。

各自、自分の担当部分の執筆を行い、`\input` で指定されているファイル名で保存する。書き方は 5.1章のテンプレート(`phy_tb_sample.tex`)を参考に。節の構造は原則テンプレートに従うこととするが、必要であれば変更しても良い。物理過程以外の部分の節構造はそれぞれの執筆者が適切だと思うように設定する。各自の発表資料および SCALE_Week_2017春-メモ.docx の内容を適宜 copy & paste する。

参考文献の引用は `\citep` or `\citet` を使用すること (ref. `http://www.biwako.shiga-u.ac.jp/sensei/kumazawa/tex/natbib.html`). reference.bib の編集は、他の人の編集とコンフリクトする可能性があるので、他の人の編集内容を消してしまわないように注意すること。 scale/doc/description/reference.bib への反映はまとめて行うため、各自で行う必要はない。

ファイル名やモジュール名を記述するときは `\name{}` コマンドを使うこと。

締め切り: 6/2(金)

Chapter 1

開発体制

1.1 概略

1.1.1 体制

SCALE の開発グループは、Team SCALE と呼ばれる。開発のタスク等は Redmine (1.2節参照) によりチケット管理されており、原則、Redmine 上で議論や方針の通知が行われる。別途、月に一度程度開発者ミーティングを行い、現状の共有や Redmine 上ではやりにくい議論を行っている。

Team SCALE の中には、SCALE Core と User's Group (UG[1]) の2つのサブグループが組織されている。 SCALE Core グループは開発のマネージメントを行う。主に、チケットの管理 (優先順位の設定、担当者アサイン)、フィーチャーブランチのレビューアーアサインなどを行っている。 UG は、ユーザー視点から開発のドライブを行う。マニュアルの整備やユーザーからの質問等の管理、ユーザー利便性向上のための提案等を行っている。

1.1.2 リリースポリシー

開発は、develop ブランチに対して行われており、リリースに合わせて master ブランチが更新される (1.3節参照)。リリースには、メジャーリリース、マイナーリリース、バグフィックスリリースがある。バグフィックスリリースは致命的なバグが発見された場合に、最新版のリリースに対する修正版となる。メジャーリリースとマイナーリリースの明確な定義は決まってはいないが、これまでは、内部構造や設定項目が大きく変わったときにメジャーリリースとした。ある程度定義を決めておくべきであろう。

1.2 開発ツール

SCALE開発で使用しているツールやメーリングリストを紹介する。

1.2.1 Redmine

Redmineとは？

- Webベースのプロジェクト管理ソフトウェア

[1] User's Group または User's Guide はともに UG と称されている

- タスク管理・進捗管理(チケット)、情報共有(Wiki)が可能
- ソフトウェア開発の現場で10年前から使われているスタンダードなツール

SCALE開発における役割

- 開発の知見の蓄積、保存が大きな目的。今後新たに開発に参加した人も、過去の経過を見ることが出来る。
- 利用頻度が高いのはチケットの発行・閲覧・更新
- 発行・更新されたチケット・wikiの内容はメールで開発メンバーに周知される
- チケットの内容に関する議論についてもredmine上に残しておく
- 対応済みチケットは開発ミーティングでフォローアップ
- Ranchu上で運用されている (`https://ranchu.aics.riken.jp/redmine`)

1.2.2 GitLab

GitLabとは？

- WebベースのGitリポジトリ管理ソフトウェア (ClosedなGitHubのようなもの)
- マージリクエストの発行といったコラボレーションを促進するための機能
- コミットの内容やブランチの分岐の歴史をグラフィカルに確認可能

SCALE開発における役割

- 開発者(非管理者)的にはマージリクエストが主な用途
- 管理者的には権限設定や鍵の管理などユーザ管理が容易というメリットがある
- Ranchu上で運用されている (`http://ranchu.aics.riken.jp/gitlab/`)

1.2.3 Jenkins

Jenkinsとは？

- 継続的インテグレーション(CI)ツール　→　バグを早期に発見する
- ビルド・テストを自動化する
- CIツールのデファクトスタンダード(最近は競合も？)

SCALE開発における役割

- Gitリポジトリへのコミットを契機としてコンパイル・テストを行う
- コンパイル結果やテストケース実行結果の可視化画像をブラウザ上で確認可能
  - 居室のディスプレイで常時モニタリングしている
- Pearlscale上で運用されている (`http://pearlscale.aics27.riken.jp/jenkins/`)

1.2.4 Doxygen

Doxygenとは？

- ソースコード構造をドキュメント化するためのツール
- ソースコードそのものとソースコード中のコメントを解析しHTMLやLaTeX形式のドキュメントを生成する
- C, Java, Pythonなど様々な言語に対応。Fortranでも使える

SCALE開発における役割

- 開発時にはソースコード中にDoxygenに対応したコメントを付加する
- パッケージの中(scale-rm/doc)にDoxygen実行用のMakefileが用意されている
- リリース版で生成したHTML形式のドキュメントをSCALEホームページ(`http://scale.aics.riken.jp/doc/5.1.2/index.html`)で公開

1.2.5 メーリングリスト

- 情報基盤センターで運用しているMLサービス(GNU Mailman)を使用
- SCALE開発で使用しているのは以下の 3 つ
  - 開発者用(scale-dev@ml.riken.jp)
    - 開発者間の情報共有に用いる
    - 開発ミーティングのアナウンスなど
  - ユーザ用(scale-users@ml.riken.jp)
    - 一般ユーザとの情報交換に用いる
    - リリース時の周知など
  - 管理者用(scale@ml.riken.jp)
    - ユーザ用MLへの登録窓口

1.3 開発フロー

SCALEの開発には、チケット駆動開発ワークフローを採用している。これは多くのソフトウェアにおいて採用されている開発手法であり、同時並行での共同作業に向いている。加えて、ソースコードのリポジトリ管理にはgit-flowモデルを採用している。これはgitをベースとしたワークフローの手法の一つで、たくさんの変更が同時進行で行われていても、最終的に整然とした履歴を残すことが出来る優れた開発モデルである (`https://danielkummer.github.io/git-flow-cheatsheet/index.ja_JP.html`)。これを読む人はこれらのワークフローを理解し、積極的に共同開発に参加してもらいたい。自らの手元で死に絶え、日の目を見ない派生版ソースコードがなくなりますように。

1.3.1 役割

開発に関係する人は主に 3 種類に分類される。

- ユーザー：開発に対しては、要望や報告の形でコミットする人
- 開発者：コーディングなど、アプリケーションの実際の開発を行う人
- 管理者：開発者の中で、開発ブランチやリリースブランチの更新を行う権限を持った人

管理者はソースコードの変更依頼（マージリクエスト）に対して、コードレビューを行う人でもある。

1.3.2 チケット

チケットとは、単一の目的を持ったタスクにつけられた札である。開発はチケットを中心として進行する。チケットは完了されるために発行される。だからこそ、以下の点について心に留めておいてもらいたい。

- 完了条件のないチケットは作らない
- 壮大な課題を立ち上げていつまでも終わらないより、内容を細分化して個々のチケットに割り当て、順に片付けていくようにする
- やるほどゴールが遠ざかっていくような問題は、ある程度になったら、完了できるところまででタスクを分割し、チケットを分ける
- 完璧なタスクの完遂、完璧なソフトウェアなど存在しないので、凝りすぎず悩みすぎず、まず終わらせる

1.3.3 OODAループ

OODAループとは、チケット駆動開発の中での意思決定手順にあたる。以下の 4 つの頭文字をとってOODA(ウーダ)と呼ぶ。

- Observe: バグに気が付いたり、機能を増やしたいと思いつく
- Orient: 重要度の認識や、議論と検討を行う
- Decide: 担当者の決定、優先度の設定

- Act: 実際の作業を行う（コードを書く、実験をする、等）

これらのサイクルの各フェーズは、チケットのそれぞれの状態に対応している。すなわち、

- Observe=”New”
- Orient=”Discussion”
- Decide=”Assigned”
- Act→ソースコード変更（または実験結果報告）を経て”Complete”

である。以下でフローの詳細について説明する。

1.3.4 チケットの発行から解決まで

[**Observe=”New”**]

バグに気が付いたり、機能を増やしたいと思いついたり、テストすべき事柄を認識したユーザー・開発者は、これを報告する。すべての開発者、ユーザーが観察者である。開発者はredmineに自らチケットを発行する。ユーザーからの要望は、UG委員会が代わりにチケットを発行する。発行の際に留意する点は以下の通りである。

- 何をするためのチケットなのか、一目でわかる題名にする
- ステータス欄はNewに設定
- カテゴリ欄に該当するものがあれば選ぶ
- 優先度、対象バージョン、担当者、期日などの欄は入力しなくてよい

SCALE Core グループは発行されたチケットを振り分ける。

- 開発マイルストーンに従って対象バージョンを決定する
- まだ議論に乗せないものはNewのまま
- 今議論すべきものはDiscussionへ
- 議論の余地なく意思決定に進むもの(バグ修正など)はAssignへ
- 必要に応じて更に優先度を設定する
- 付随するチケット（ドキュメントの追加修正）がある場合は併せて発行する

[**Orient=”Discussion”**]

Discussionとなった事柄について、開発者間で議論する。議論の必要がある開発案件とは、いわばSCALEの開発の歴史における曲がり角である。絶対的な正解などなく、これまでの経験や開発ポリシーを踏まえて考えていく必要のある事柄に相当する。この議論の中で、SCALEらしさが育つと考えられる。議論は、その経過を将来にわたり検索できる形で残しておきたい。そのため、なるべくredmineに書き込むようにする。

[Decide=”Assigned”]

SCALE Core グループが決定者としての役を担い、意思決定を行う。まず、このタスクを実行するかしないかの決定を行い、実行するとなった場合にはチケットのステータスをAssignに変更し、さらに以下の事項を決める。

- 担当者
- 期日（担当者に決めてもらう場合も多い）
- 対象バージョン（変更の必要があれば）

[Act→”Complete”]

チケットの担当者は実際のソースコードの変更や実験の実行を行う。ソースコードの場合は次の1.3.5に説明する通り、作業内容をgitにプッシュし、マージリクエストを行う。コードレビューを経て、変更はマージされる。テスト評価等の場合、結果をredmineにチケットのコメントとして残しておき、内容を開発ミーティング等で報告する。ミーティングでの議論を踏まえて、作業を継続する完了するかをその場で判断する。継続の場合は完了まで追加のテストを進める。

タスクが完了したら、チケットのステータスはcompleteへ変更される。

1.3.5 ソースコードの変更からマージまで

チケットのタスクがソースコードの追加や修正である場合、以下に説明するフロー（git-flowモデルの手法）に従って進める。最終的に変更分は最新の開発ブランチに取り込まれる。フローでは、gitのコマンドを用いた作業、gitlab、redmine上での操作にそれぞれ(git)(gitlab)(redmine)のタグをつけてある。

ソースコードの入手

- (git) リポジトリをcloneして、checkoutする
- (git) developブランチから新たなfeatureまたはhotfixを分岐させる（トピックブランチ）

トピックブランチでの作業

- (git) 手元での変更分をcommit（コメントにredmineの該当チケット番号を付加してもよい）
- (git) 必要に応じて親リポジトリの変更をフォロー（=rebase）

リモートへのpush

- (git) 親リポジトリへトピックブランチをpush
- (gitlab) トピックブランチのマージリクエストを作成
  - コミット時にredmineのチケット番号を付加
  - Assign toは空欄で良い（指名してもよい）

レビュー

- (gitlab) SCALE Core グループがレビューアをアサイン
- (redmine) レビューアは該当するチケットにレビューのコメントを付ける
- (git) レビューを受けて、必要であればブランチに追加commitしてpush

マージ

- (gitlab) レビューアはOKになったら、developブランチへマージを実行。必要ならマージの前にrebaseする
- (redmine) マージが完了したら、redmineのチケットに ”commit:¡コミット番号¿ でマージ” 等のコメントを入れて、チケットのステータスをcompleteに変更する

Chapter 2

UG

2.1 目的・活動

UG活動は、SCALEのユーザーがモデルを使用したり、内容について理解する際のサポート活動の総称である。主な活動として、 1) SCALEの機能及び使い方に関する説明書であるユーザーズガイドの作成、 2) SCALEの各コンポーネントのインプリメンテーションを説明したModel descriptionの整備、 3) メーリングリストに投稿された質問対応などのユーザー対応、 4) 講習会等の実施、がある。これらの活動は全ての開発者全員で担当するが、 UG core委員会（UGC）は、これらの業務の実施計画を立て、適任者への担当の割り振りを行うといったマネジメント業務を行っている。本活動の目的は、ユーザーの利便性の充実にあるが、実際には、開発者自身の整理や記録、また、後続の開発者にもわかるような軌跡を文書として残すという面もあり重要な活動である。

2.2 SCALE Description

文書のソースディレクトリ

`scale/doc/descriptions`

文書の公開先

`https://scale.aics.riken.jp/doc/index.html`
`https://scale.aics.riken.jp/ja/doc/index.html`
SCALEのリリースとともに`scale_rm_description-X.X.X.pdf`というファイル名で公開する。 `X.X.X`はバージョン番号。

執筆者

原則、コードを書いた人が担当部分について書く。

進捗管理

原則、インプリと同じタイミングで執筆する。現在（17/5/31時点）は進捗管理を行っていないが、今後、定期的にupdateするシステムが必要である。

内容

- 基本的には、支配方程式を示すとともに、離散化後の形式も示し、文書とコードが 1 対 1 対応するように、コードに書いたことを全て文章にする。
- 論文に書いているものは論文を引用しても良いが、論文に書かれていない細かい情報、更新した情報も加える。
- 各パラメータについても、本論文と値が異なる場合には、根拠等を記載する。
- コードを読んだ人が、SCALE descriptionを読めば、詳細を理解出来ることが目安である。

2.3 SCALE User's Guide

文書のソースディレクトリ

`scale/doc/users-guide`

文書の公開先

`https://scale.aics.riken.jp/doc/index.html`
`https://scale.aics.riken.jp/ja/doc/index.html`
SCALEのリリースとともにバージョン名を入れて公開。
`scale_users_guide.vX.X.X.pdf`: SCALE-RM 日本語版。
`scale_users_guide_En.vX.X.X.pdf`: SCALE-RM 英語版。
SCALE-GMは準備中。
`X.X.X`はバージョン番号。

執筆者

開発者全員が対象だが、UGCによりアサインされる。

進捗管理

Redmineのチケットで管理。

執筆の流れ

1. 新たな機能を追加、既存の機能を変更など、ユーザーに見える部分で変更があった場合、コードを書いた人がRedmineのチケットを切る。具体的には、ネームリストの変更や追加があった場合には、該当することが多い。
2. UGCは各チケットに担当者をアサインする。必ずしもコードを書いた人とは限らない。これは、機能レビューを兼ねているためである。開発担当者が文書も書く場合には、マージの際にレビューを行う。
3. アサインされた人は、期日までに執筆を行い、マージリクエストを出す。
4. 日本語版と英語版の両方を更新する。

2.4 連絡先

第1.2.5節で紹介した通り、ユーザー用MLや管理者アドレスにおいて、SCALEに関する質問や要望を受け付けている。そこで受け付けた質問や要望は、図2.4.1のフローに沿って対応する。

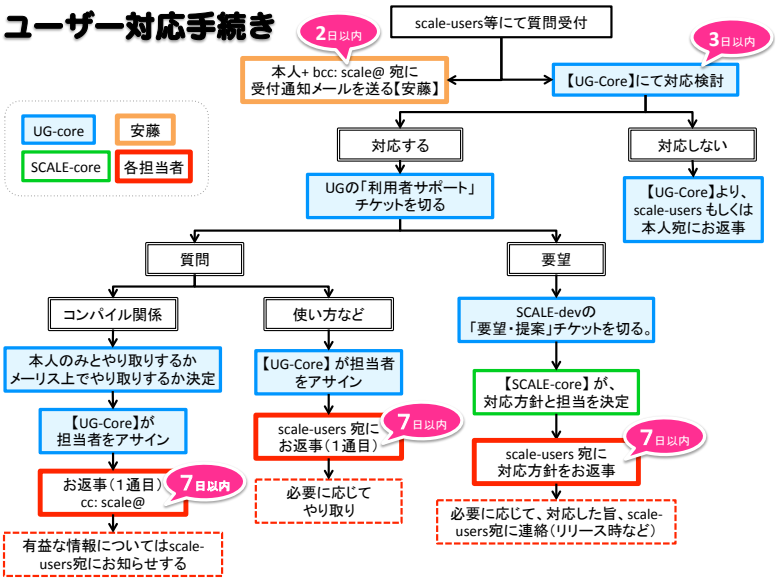

Figure 2.4.1: ユーザー対応フロー。

Chapter 3

全体構造

3.1 ライブラリとモデル

SCALE は Scalable Computing for Advanced Library and Environment であり、ライブラリ環境であると謳っている。アプリケーションであるモデルや解析ツール等は、ライブラリが提供するコンポーネントのなかで必要なものを適切に呼びだすことで目的の計算を行うという使い方を想定している。例えば SCALE-RM では、以下のような実行フローにより時間発展を計算している。

1. 各物理過程のコンポーネントを呼び出し、現在の予報変数をもとに各過程によるテンデンシーを得る
2. 力学過程コンポーネントを呼び出し、得られたテンデンシーをもとに次の時刻の予報変数を得る

しかしながら、現状ではSCALE-RM での利用を前提とした仕様になっており、他のアプリケーションからの利用はほとんど考慮されていない。また、各コンポーネントが互いに強く依存しており、単体のコンポーネントを簡単に呼び出すことができるようにはなっていない。つまり、表面的にはライブラリ的な構造にはなっているものの、実質的にはモデルである。今後、実質的なライブラリとするために構造の整理や修正が必要である。また、同様な機能が重複して複数のコンポーネント内に実装されているものがあるが、これらは切り出した上で、共通に利用するようにする必要がある。

3.2 ソースコード構造

ソースコードのトップディレクトリには以下のディレクトリが配置されている。

`dc_utils` 浮動小数点精度やテストのためのユーティリティー群。地球流体電脳倶楽部からもってきた

`scale-gm` SCALE-Global Model 全球二十面体準一様格子モデル

`scale-rm` SCALE-Regional Model 領域モデル

`scalelib` ライブラリ

`sysdep` システム依存情報を記述したファイル群

SCALE-LETKF 用のソースコードはそのままでは入っていない。必要な場合は、README.letkf を参照のこと。

Chapter 4

大気力学過程

4.1 SCALE-RM

力学過程では、連続の式、運動方程式、熱力の式により次の時刻における予報変数の値を求める。このとき、物理過程で計算されたテンデンシーを各時間発展方程式の右辺に加えて時間発展させる。

$$\frac{\partial \rho q_v}{\partial t} + \nabla \cdot (\rho q_v \mathbf{u}) = \left(\frac{\partial \rho q_v}{\partial t} \right)_{physics} \tag{4.1}$$

$$\frac{\partial \rho q_l}{\partial t} + \nabla \cdot (\rho q_l \mathbf{u}) = \left(\frac{\partial \rho q_l}{\partial t} \right)_{physics} \tag{4.2}$$

$$\frac{\partial \rho q_s}{\partial t} + \nabla \cdot (\rho q_s \mathbf{u}) = \left(\frac{\partial \rho q_s}{\partial t} \right)_{physics} \tag{4.3}$$

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = \left(\frac{\partial \rho}{\partial t} \right)_{physics} \tag{4.4}$$

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) = -\nabla p - \rho g \mathbf{e_z} + \left(\frac{\partial \rho \mathbf{u}}{\partial t} \right)_{physics} \tag{4.5}$$

$$\frac{\partial \rho \theta}{\partial t} + \nabla \cdot (\rho \theta \mathbf{u}) = \left(\frac{\partial \rho \theta}{\partial t} \right)_{physics} \tag{4.6}$$

$$p = p_{00} \left(\frac{\rho \theta R^*}{p_{00}} \right)^{\frac{c_p^*}{c_p^* - R^*}}. \tag{4.7}$$

ここで、

$$c_p^* \equiv q_d c_{pd} + q_v c_{pv} + q_l c_l + q_s c_s \tag{4.8}$$

$$R^* \equiv q_d R_d + q_v R_v. \tag{4.9}$$

詳しくは、SCALE Description の Chapter 2 や Nishizawa et al. (2015) を参照のこと。

力学過程の計算において考えるべき事を以下に挙げる。

- 空間微分の差分スキーム
- トレーサー変数の非負保証

- 時間積分の数値スキーム
- プロセス並列化のための袖領域のデータ交換

以下、それぞれの項目について説明する。

4.1.1 空間差分スキーム

SCALE-RM では空間差分スキームとして、一次、三次、五次の風上差分スキーム、二次、四次、六次の中央差分スキームが利用可能である。力学変数とトレーサー変数に対して、別々に差分スキームを選択する。選択するスキームに応じて、必要な袖領域の格子点数が異なる。

ソースコード

- `atmos-rm/dynamics/scale_atmos_dyn_fvm_flux_ud1.F90`
- `atmos-rm/dynamics/scale_atmos_dyn_fvm_flux_ud3.F90`
- `atmos-rm/dynamics/scale_atmos_dyn_fvm_flux_ud3Koren1993.F90`
- `atmos-rm/dynamics/scale_atmos_dyn_fvm_flux_ud5.F90`
- `atmos-rm/dynamics/scale_atmos_dyn_fvm_flux_cd2.F90`
- `atmos-rm/dynamics/scale_atmos_dyn_fvm_flux_cd4.F90`
- `atmos-rm/dynamics/scale_atmos_dyn_fvm_flux_cd6.F90`

4.1.2 非負保証

トレーサー変数は、物理的に負値をとり得ず、負の値とならないような取り扱いが必要である。 SCALE-RM では、非負保証のために、FCT (Flux-corrected transport) スキーム (Zalesak, 1979)、および Koren フィルター (Koren, 1993) の利用が可能である。ただし、Koren フィルターは三次風上スキームとともにしか使う事ができない。

ソースコード

- `atmos-rm/dynamics/scale_atmos_dyn_fvm_flux_ud3Koren1993.F90`
- `atmos-rm/dynamics/scale_atmos_dyn_common.F90`

  `ATMOS_DYN_fct`: FCT による anti-flux を計算する

4.1.3 時間積分スキーム

時間積分を計算するスキームとして、 Euler スキーム、 Heun's 三段三次 Runge-Kutta (RK) スキーム、 Wicher and Skamarock's 三段 RK スキーム (Wicker and Skamarock, 2002)、四段四次 RK スキームが利用可能である。ラージステップ、ショートステップ、トレーサーステップそれぞれにスキームを選択する。なお、CWC (Consistency with Continuity) (Gross et al., 2002) を満たすため、ショートステップで計算された質量フラックスを用いてトレーサー移流を計算している。

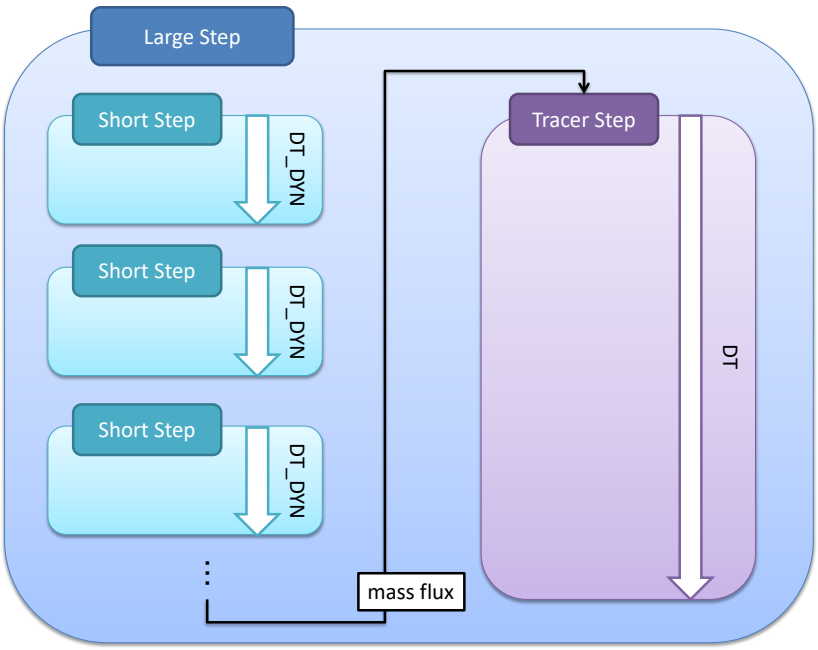

Figure 4.1.1: ラージステップ、ショートステップ、およびトレーサーステップのイメージ図。

ソースコード

- atmos-rm/dynamics/scale_atmos_dyn_tinteg_large.F90
- atmos-rm/dynamics/scale_atmos_dyn_tinteg_large_euler.F90
- atmos-rm/dynamics/scale_atmos_dyn_tinteg_large_rk3.F90
- atmos-rm/dynamics/scale_atmos_dyn_tinteg_short.F90
- atmos-rm/dynamics/scale_atmos_dyn_tinteg_short_rk3.F90
- atmos-rm/dynamics/scale_atmos_dyn_tinteg_short_rk4.F90
- atmos-rm/dynamics/scale_atmos_dyn_tinteg_tracer.F90
- atmos-rm/dynamics/scale_atmos_dyn_tinteg_tracer_euler.F90
- atmos-rm/dynamics/scale_atmos_dyn_tinteg_tracer_rk3.F90

4.1.4 袖領域データ交換

隣接する東西南北の計算ノードとの袖領域のデータ交換は、MPI を使って行う。ただし、MPI を陽に呼ぶのではなく scale_comm モジュールの comm_vars8 サブルーチンを用いる。

ソースコード

- atmos-rm/communication/scale_comm.F90

  COMM_vars8: 隣接計算ノードとの袖領域データの交換を開始する

  COMM_wait: 隣接計算ノードとの袖領域データ交換の完了を待つ

4.2 SCALE-GM

Chapter 5

大気物理過程

5.1 乱流

LES 用の SGS (Sug-Grid Scale) 乱流モデルとして、Smagorinsky-Lilly モデル、および、Deardorff モデルが実装されている。

RANS 用の境界層乱流モデルとして、MYNN level 2.5 モデルが実装されている。

Smagorinsky モデルは、RANSにおける水平の渦粘性にも用いられる。

5.1.1 Smagorinsky-Lilly model

解くべき物理法則

LES における SGS モデルは、移流項に起因する SGS 乱流の寄与を評価するものである。 Smagorinsky-Lilly モデル (Lilly, 1962; Smagorinsky, 1963) はSGS乱流の寄与を診断的に評価するものであり、 SGS モデルとしてはもっとも簡単なものである。

渦粘性フラックス τ_{ij} は、

$$\tau_{ij} = -2\nu_{\rm SGS}\left(S_{ij} - \frac{1}{3}S_{kk}\delta_{ij}\right) + \frac{2}{3}\delta_{ij}TKE \tag{5.1}$$

である。ここで、S_{ij} は応力テンソル、δ_{ij} はクロネッカーのデルタである。TKE は SGS の Turbulence Kinetic Energy (TKE) であり、渦粘性係数から診断的に計算される。渦粘性係数 $\nu_{\rm SGS}$ は

$$\nu_{\rm SGS} = (C_s f(a)\Delta)^2 |S| F_s. \tag{5.2}$$

C_s は Smagorinsky 定数とよばれ、デフォルトでは 0.13 である。成層の効果 F_s およびプラントル数の評価は Brown et al. (1994) にならっている。また、非等方格子のアスペクト比を考慮し、その効果を $f(a)$ として入れている (Scotti et al., 1993)。長さスケール Δ は、2-grid ノイズ削減のための超粘性を考慮して、2 格子幅としている (Nishizawa et al., 2015)。詳しくは、SCALE Description manual の Section 8.1 や Nishizawa et al. (2015) を参照のこと。

Smagorinsky model は、LES の SGS 乱流モデルとは別に、 RANS における水平渦粘性としても利用可能である。この目的のために利用するためには、`PARAM_ATMOS_PHY_TB_SMG` NAMELIST group の `ATMOS_PHY_TB_SMG_horizontal`

を .true. とする。この場合、水平の渦粘性のみが計算される。また、長さスケールが水平格子間隔のみを考慮して決定されるとともに、渦粘性フラックスのなかの TKE の項が無視される。

離散化

渦粘性を求めるためには、応力テンソルが必要になるが、その計算には3方向のフルレベルおよびハーフレベルの様々な点における値が必要となり、ナイーブに行うと、同じ計算をフルレベルやハーフレベルの様々な点で行う必要があり、ソースコードがが煩雑になる。したがって、まず、フルレベルのみにおける渦粘性係数を求め、ハーフレベルにおける渦粘性はその内挿によって求めている。

デフォルトでは陽的に解くが、鉛直解像度が高い場合、鉛直方向に陰的に解くことも可能である。その場合は、 PARAM_ATMOS_PHY_TB_SMG NAMELIST group の ATMOS_PHY_TB_SMG_implicit を .true. とする。陰解法には、3重対角行列の直接法を用いている。

ソースコード

Smagorinsky スキームによる計算を行っているサブルーチンを以下に示す。

- atmos-physics/turbulence/scale_atmos_phy_tb_smg.F90

  ATMOS_PHY_TB_SMB: 渦粘性によるテンデンシー計算

- atmos-physics/turbulence/scale_atmos_phy_tb_common.F90

  ATMOS_PHY_TB_calc_strain_tensor: ストレインテンソルの計算

  ATMOS_PHY_TB_diffusion_solver: 渦粘性陰解法ソルバー

5.1.2 Deardorff model

解くべき物理法則

Smagorinsky model は診断型であり力学変数から渦粘性の計算が可能であるが、一方、Deardorff model (Deardorff, 1980) は、別途 TKE e を予報し、それをもとに渦粘性係数 K_m を計算する。

$$K_m = 0.1le^{1/2}. \tag{5.3}$$

ここで、l は混合長で、不安定な場合は格子サイズである。安定の時は

$$l = 0.76e^{1/2}N^{-1}. \tag{5.4}$$

ただし、格子サイズを超えない。 N^2 はブラントバイサラ振動数である。

離散化

TKE は、質量を持たないトレーサーとして登録しており、移流計算は力学のなかで行われる。

渦粘性係数を求める格子位置や鉛直の陰的計算については、 Smagorinsky-Lilly model と同様である。

ソースコード

Deardorff スキームによる計算を行っているサブルーチンを以下に示す。

- `atmos-physics/turbulence/scale_atmos_phy_tb_d1980.F90`

  `ATMOS_PHY_TB_d1980`: 渦粘性によるテンデンシー計算

- `atmos-physics/turbulence/scale_atmos_phy_tb_common.F90`

  `ATMOS_PHY_TB_calc_strain_tensor`: ストレインテンソルの計算
  `ATMOS_PHY_TB_diffusion_solver`: 渦粘性陰解法ソルバー

5.1.3 MYNN model

解くべき物理法則

MYNN model (Mellor and Yamada, 1982; Nakanishi and Niino, 2004) は、RANS 用の境界層乱流モデルである。 SCALE では level 2.5 のみを実装している。

水平方向には一様であることを仮定し、鉛直方向の渦粘性を求める。level 2.5 では、TKE を予報しそれをもとに渦粘性係数を計算する。 sub-grid scale の部分凝結の効果も考慮されている。

詳しくは、SCALE description manual Section 8.2 を参照のこと。

離散化

TKE の扱いは、Deardorff model と同様である。鉛直混合は陰解法で計算される。

ソースコード

MYNN スキームによる計算を行っているサブルーチンを以下に示す。

- `atmos-physics/turbulence/scale_atmos_phy_tb_mynn.F90`

  `ATMOS_PHY_TB_mynn`: 渦粘性によるテンデンシー計算

- `atmos-physics/turbulence/scale_atmos_phy_tb_common.F90`

  `ATMOS_PHY_TB_diffusion_solver`: 渦粘性陰解法ソルバー

5.2 雲微物理

SCALEには以下の5つの雲微物理モデルが実装されている。
・ 1 モーメントバルク（水雲のみ）: KESSLER(Kessler, 1969) (雲水混合比（q_c)、雨水混合比（q_r）)
・ 1 モーメントバルク（氷含む）: TOMITA08(Tomita, 2008):NICAMの雲物理と同様 (q_c, q_r)、雲氷混合比（q_i)、雪水混合比（q_s)、霰水混合比（q_g）)
・ 2 モーメントバルク（氷含む）: SN14(Seiki and Nakajima, 2014)（$q_c, q_r, q_i, q_s, q_g, N_c, N_r, N_i, N_s, N_g$ (N**は数濃度を表す)）
・ 1 モーメントビン（水雲のみ、氷含むスイッチ可能）: SUZUKI10(Suzuki et al., 2010)（各ビンの質量濃度（gspc(m)：mは各ビンの質量、spcは粒子の

種類））
・超水滴法（水雲のみ）：SDM(Shima et al., 2009) （超水滴の属性（位置情報、サイズなど））
ここで括弧内は予報変数を表す。超水滴方以外は雲水や雲氷などの雲物理に関わる予報変数ををトレーサーとして扱い、各グリッドの平均量として雲関係の予報変数を定義するEulerian Cloud microphysical model(ECM)、超水滴法は雲粒を粒子的に扱うLagrangian Cloud microphysical model (LCM)である。

5.2.1 解いている雲微物理過程

雲微物理モデルが計算している雲微物理過程は（モデルによって扱っていないものもあるが）以下の5つである。
・雲粒、氷粒の発生（飽和調整、雲粒活性化）
・凝結成長（蒸発、昇華も含む）
・衝突併合
・融解、凍結
・分裂

雲の発生（飽和調節、雲粒活性化）

雲粒の発生は飽和調節と雲粒活性化のどちらかで表現される。(Kessler, 1969; Tomita, 2008)は飽和調節、その他は雲粒活性化の式を解くことで雲粒の発生を表現している。

飽和調節はある時刻にそのグリッドが過飽和であれば過飽和を解消するように水蒸気(q_v)を雲粒(q_c)または氷粒(q_i)として発生させる。また未飽和な時は雲粒・氷粒を水蒸気に変換する。
このとき過飽和とは、雲粒が発生する時は水に対する過飽和度、氷粒が発生する時は氷に対する過飽和度を用いる。

一方雲粒活性化は、以下の２種類の解き方をする。一つ目の解き方は、観測などの経験に基づく手法である。
ある時刻にそのグリッドの過飽和度が与えられた時に、雲粒数(N_c)、氷粒数(N_i)が経験的に

$$N_c = N_0 S_l^k \tag{5.5}$$

$$N_i = N_{0,i} exp(a_{in} + b_{in} S_i) \tag{5.6}$$

と定まる。ここでa_{in}, b_{in}, N_0, $N_{0,i}$は観測などから経験的に定まるパラメータ、S_l, S_iはそれぞれ水、氷に対する過飽和度である。この雲粒数よりも、そのグリッドに存在している雲粒数が少ない時は、N_cから不足している文を雲粒として発生させる。この方法は(Seiki and Nakajima, 2014; Suzuki et al., 2010)に採用されている。

また雲粒活性化を理論的に求める方法は、Kohler理論に基づく方法であり、(Suzuki et al., 2010)(オプションにより選択可能)とSDMはこの方法に基づいている。

凝結・蒸発・昇華成長過程

凝結・蒸発・昇華成長過程は以下の式に基づいて解く。

$$C_{spc}R_i\frac{dR_i}{dt} = \frac{(S-1) + \frac{a_{ces}}{R_i} + \frac{b_{ces}}{R_i^3}}{F_k + F_d} \tag{5.7}$$

$$F_k = \Big(\frac{L}{R_v T} - 1\Big)\frac{L\rho_w}{KT}$$

$$F_d = \frac{\rho_w R_v T}{De_s(T)}$$

ここで$S, a_{ces}, b_{ces}, C_{spc}, R_i, R_v, T, L, \rho_w, e_s, D, K$はそれぞれ、過飽和度（粒子が液体の時は水に対する過飽和度、個体の時は氷に対する過飽和度）、曲率の効果を表すパラメータ、溶質の効果を表すパラメータ、粒子の形状に関するパラメータ、雲粒子の半径、気体定数、温度、潜熱、水の密度、飽和水蒸気圧、分子の拡散係数、熱伝導度である。その他Ventilationの効果や、Limiterによって上記の式の補正が入ることはあるが、基本的にはこの式を解くことで凝結(Sが正の時)・蒸発(Sが負の時)・昇華成長を表現している。

衝突併合過程

衝突併合はStochastic Collision Equation(SCE)と呼ばれる以下の式を解くことで計算される。

$$\frac{\partial}{\partial t}f^{(\lambda)}(m) = \sum_{\nu}\sum_{\mu}\int_0^{\frac{m}{2}} f^{(\mu)}(m-m')K_{\nu\mu}(m', m-m')dm' - f^{(\xi)}(m)\sum_{\sigma}\int_0^{\infty} f^{(\sigma)}(m'')K_{\xi\sigma}(m, m'')dm''$$

ここでf, m, Kは雲粒数濃度（粒径分布関数）、雲粒の質量、衝突カーネル, μ, ν, σ, ξは粒子の種類（例えば雲粒、雨粒など）を表す。

融解

固体の雲粒が融解する際には例えば下記のような式を解くことで計算する。

$$\Big(\frac{dm}{dt}\Big) = \frac{2\pi}{L}K_a D(T - T_0) \tag{5.8}$$

ここでK_a, T_0は熱伝導度、基準温度（経験的なパラメータ）である。

凍結

凍結については以下の(Bigg, 1953)の式に従って計算する

$$F_i = m_i\frac{g_i^{(w)}}{\rho_w}N_{im,0}\big(0.1(T_0 - T)^{\gamma}\big) \tag{5.9}$$

ここで$F_i, g_i^{(w)}, N_{im,0}$はそれぞれ単位時間に凍結する質量、雲粒の質量粒径分布関数、経験的なパラメータである。

分裂

分裂の効果は(Seiki and Nakajima, 2014)のみが考慮しているが、詳細は割愛する。

5.2.2 雷モデル

SCALEには雷モデルを実装中である。実装したコンポーネントは
・電荷分離機構（着氷電荷分離(Takahashi, 1978)）
・電荷の移動
・電場の計算（電荷の空間分布からポアソン方程式を解く）
・放電（中和）機構(MacGorman et al., 2001) である。

5.3 エアロゾル微物理

エアロゾルモデルは(Kajino et al., 2013)の 3 モーメントバルク法が実装されており、説いている物理過程は
・新粒子生成（前駆体ガスからエアロゾル粒子）
・凝集
・衝突併合（エアロゾル同士の衝突）
である

5.4 放射

5.5 積雲パラメタリゼーション

積雲対流パラメタリゼーション（Convective Parameterization Scheme: CPS）とは，数値モデルが低解像度の場合にも対流雲の効果を表現するためのパラメタリゼーションである． Fig. **??**にコンセプトのイメージ図を示す．モデルの解像度が高く細かな積雲対流の上昇流や下降流を解像することが出来る場合は，力学コアと雲微物理スキームの組み合わせで積雲対流を直接解像することができる．しかし，水平格子間隔が10km以上あり積雲対流が格子の中に入ってしまうような場合には，そのカラムの気圧，温度，水蒸気量を入力することで大気の安定性をもとに積雲対流が発生するか否かを判断し，発生する場合，非断熱加熱や降水量を見積もる．これが積雲対流パラメタリゼーションである．これまでに様々な種類のCPSが考え出されているが，次の2つに大別することができる (Stensrud, 2007).

- Deep-layer control type： CAPEを消費するためにどれだけの対流を必要とするかを考えるコンセプト．以下のような例がある．(Kuo, 1965)は最も古いCPSの 1 つであり， (Arakawa and Schubert, 1974)の改良型は現在も多くの全球気候モデルで使用されている．
  - Kuo ((Kuo, 1965))
  - Arakawa-Schubert ((Arakawa and Schubert, 1974))
- Low-level control type (Mass-flux control type)：どんなパーセルがCINに打ち勝ってCAPEを消費できるかを考えるコンセプト．以下のような例がある．SCALE-RMにはKain-FritschのCPSが実装されている．

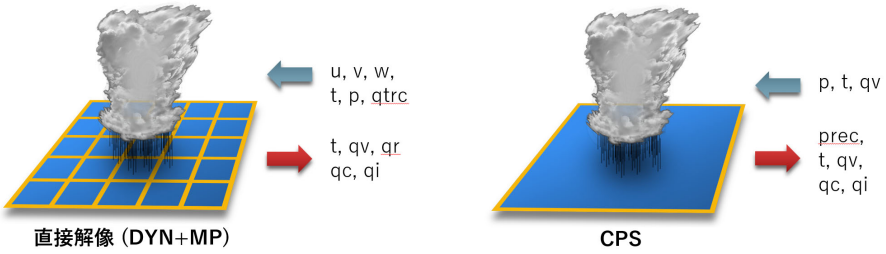

Figure 5.5.1: 積雲対流パラメタリゼーションのコンセプト：積雲対流を直接解像する場合(左)とパラメタリゼーションの場合(右)

- Tiedtke ((Tiedtke, 1989))
- KF ((Kain and Fritsch, 1990); (Kain, 2004))

5.5.1 Kain-Fritsch CPS

解くべき物理法則

CPSにおいては，明確に解く物理を表現することが難しい． CPSは単一の物理過程を解くのではなく，様々な物理過程を含んだ「積雲対流」そのものを解く．ここでは，物理法則を説明する代わりに，Kain-FritechのCPSで考えている積雲対流について説明する．

積雲対流は，ある空気塊を少し持ち上げたときに周りの空気に比べて浮力を持ち上昇流を発生させ，対流が降水を伴うことで下降流を発生させる．上昇流と下降流を足した結果として余ったMass Fluxは補償流として対流コアの外のFluxとなる．これらの対流圏内の深い混合が行われた結果として，積雲対流はもともとあったCAPEを90%消費すると考える．

離散化

基本的にはパーセル法に基づいて構築されており，全層を積算した結果として，消費した水蒸気量と降雨量＋凝結物が保存するように計算する．鉛直層毎にエントレインメントとデトレインメントを評価し，混合気体としての浮力を評価してMass-fluxを算出する．各パートは次の概略のように計算される．

1. 対流発生の判断
   - 持ち上げるパーセルの高度を診断する
   - パーセルのT, QvからLCLを診断する
   - 対流雲内の上昇流の見積もり
   - LCLにおけるT, Qv, Pから上昇流を見積もる
   - 上昇流から雲頂高度(CTL)を診断する
   - 同時に上昇流内で生成される雨滴や雲粒の量も診断する
   - 対流発生を判断する

- CTL > LCL：十分に高い→深い対流
- CTL > LCL：ただし高くない→浅い対流
- CTLがLCLを上回っていない→対流なし

2. 対流雲内の下降流の見積もり
   - 環境場と雲底高度から降水効率を見積もる
   - ETL(空気塊が浮力を失った高度)を開始高度として下降流を見積もる
   - パーセルの持ち上げ高度まで計算する
3. 補償流の見積もり
   - 対流のLifetimeを決定する
   - 正味のupward mass fluxだけ補償流が一面に存在すると仮定して補償流速度を見積もる
   - 補償流速度からΔtを決めてLifetime分だけ時間積分する
   - CAPEが元の10%以下になったことを確認する→まだCAPEが多ければ時間積分を継続
   - 全体の処理が保存しているかチェックする

ソースコード

Kain-Fritsch CPSによる計算を行っているサブルーチンを以下に示す。

- `atmos-physics/cumulus/scale_atmos_phy_cp.F90`

  `ATMOS_PHY_CP_setup`: CPタイプの選択（現状はKFしかない）

- `atmos-physics/cumulus/scale_atmos_phy_cp_kf.F90`

  `ATMOS_PHY_CP_main`: メイン関数 (テンデンシーの計算を含む)

  `ATMOS_PHY_CP_trigger`: トリガー判定ルーチン

  `ATMOS_PHY_CP_updraft`: 上昇流評価ルーチン

  `ATMOS_PHY_CP_downdraft`: 下降流評価ルーチン

  `ATMOS_PHY_CP_compensational`: 鉛直積算と補償流評価ルーチン

SCALEのKFではWRF型のトリガー関数と降水関数，およびJMA-NHM型のトリガー関数と降水関数を選択することができる．デフォルトはWRF型になっており，下記のようにrun.confに記述することでJMA-NHM型の設定を使用することができる．

```
 &PARAM_ATMOS_PHY_CP_KF
 PARAM_ATMOS_PHY_CP_kf_trigger   = 3,
 PARAM_ATMOS_PHY_CP_kf_dlcape    = 0.15,
 PARAM_ATMOS_PHY_CP_kf_dlifetime = 900.D0,
 PARAM_ATMOS_PHY_CP_kf_slifetime = 600.D0,
 PARAM_ATMOS_PHY_CP_kf_prec      = 2,
 PARAM_ATMOS_PHY_CP_kf_thres     = 2.0D-3,
 PARAM_ATMOS_PHY_CP_kf_w_time    = 8,
/
```

Chapter 6

下端モデル

6.1 地表面フラックス

地表面スキームでは、地表面における運動量収支と熱収支を考える。大気モデルの下端境界の情報を与えるものである。陸面では地表面に厚みがあるかどうかの選択を行う。海洋では粗度を計算手法の選択を行う。

解くべき物理法則

熱収支の式は以下の通りとする。

$$C_s \left(\frac{\partial T_s}{\partial t} \right) = (1 - \alpha_{SW})\, SW + (1 - \alpha_{LW})\, LW + SH + LH + GH \qquad (6.1)$$

ここでC_sは地表面熱容量、T_sは地表面温度、αは短波および長波のアルベド、SWは短波、LWは長波、SHは顕熱フラックス、LHは潜熱フラックス、GHは地中熱フラックスを示す。このうち、運動量、顕熱、および潜熱フラックスのバルク交換係数をどのように計算するかでスキームを選択する。陸面モデルでは、C_sが非零とするか否かによりスキームを選択する。海面モデルでは、海面粗度の扱い方によりスキームを選択する。

- 地表面熱容量C_sの扱い

  $C_s / = 0$: 厚みのある地表面スキームを仮定する（海洋はこちらのみ)。前ステップのT_sを用いてLW, SH, LH, GHを求め、$\frac{\partial T_s}{\partial t}$を直接計算する。

  $C_s = 0$: 厚みのない地表面スキームを仮定する。 SW, LW, SH, LH, GHのエネルギーバランスがとれるように、Tomita et al. (2008)による修正ニュートン法を用いて反復計算を行う。

- 海面粗度の扱い

  粗度をの扱いは固定か、診断的に判断するか（Miller 1992)、時間発展させるか（Moon et al. 2007）を選択する。 Moon et al. (2007)は台風などの強風時の海面粗度を観測値に合わせられるように設定されたもの。

- バルク交換係数の計算手法

Unoスキーム（Uno et al. 1995）: Louis (1979)の拡張版。バルク交換係数を診断的に求める。 Louis系は安定（夜間）時のフラックスが過大になる傾向がある

Beljaarsスキーム: Beljaars and Holtslag (1991), Beljaars (1994), Wilson (2001)の組み合わせている。バルク交換係数を反復計算で求める。より原理的なスキームといえる。

ソースコード

- `scalelib/src/coupler/scale_bulkflux.F90`

  `U95`: Unoスキームに従ってバルク交換係数を求める。

  `B91W01`: Beljaarsスキームに従ってバルク交換係数を求める。

- `scalelib/src/coupler/scale_roughness.F90`

  `CONST`: 海面粗度を変化させないようにする。

  `MILLER92`: Miller (1992)に従って海面粗度を計算する。

  `MOON07`: Moon et al. (2007), Fairall et al. (2003)に従って海面粗度を計算する。

- `scalelib/src/ocean/scale_ocean_sfc_slab.F90`

  `CONST`: 海洋のSSTを固定して、熱収支を計算する。

  `FILE`: 海洋のSSTを外部ファイルから与え、熱収支を計算する。

  `SLAB`: 海洋のSSTを熱収支を計算して時間変化させる。

- `scalelib/src/land/scale_land_sfc_const.F90`

  `CONST`: 陸面の表面温度を固定して、熱収支を計算する。

- `scalelib/src/land/scale_land_sfc_thin-slab.F90`

  `THIN-SLAB`: 地表面熱容量をゼロとして、ニュートン法による反復計算で熱収支および表面温度を計算する。

- `scalelib/src/land/scale_land_sfc_thick-slab.F90`

  `THICK-SLAB`: 地表面熱容量を与え、直接解法で熱収支および表面温度を計算する。

6.2 陸面

格子点の土地利用データにおいて、土地利用が陸で、かつ100%都市要素でない格子は全て陸面スキームを解く。土地利用の種類は、森林、裸地、草原、耕作地など、現在15種類ある。この陸面モデルについて説明する。

6.2.1 多層バケツモデル

解くべき物理法則

土壌温度、土壌水分量は鉛直1次元の拡散方程式を解く。

$$\frac{\partial T}{\partial t} = \kappa_T \frac{\partial^2 T}{\partial z} + F_T, \tag{6.2}$$

$$\frac{\partial W}{\partial t} = \kappa_W \frac{\partial^2 W}{\partial z} + F_W, \tag{6.3}$$

ここでTは土壌温度、Wは土壌水分量、κは拡散係数、Fは外部からの入力を示す。外部入力としては、地表面からの地中熱フラックス、地表面での蒸発、降水などを指す。

また海洋と異なり、多層のバケツモデルのうち、地面第1層の土壌水分量Wが蒸発（潜熱フラックス）の効率を決定する。すなわち、

$$E = \beta E_p, \tag{6.4}$$

ここで、

$$\beta = 1, (W \geq W_c) \tag{6.5}$$

$$\beta = \frac{W}{W_c}, (W < W_c) \tag{6.6}$$

であり、Eは実際の蒸発量、E_pはポテンシャル蒸発量、βは蒸発効率、W_cは臨界土壌水分量を示す。

離散化

地中内部の温度および含水量計算は、地表面熱収支におけるGH、LH、降水量を入力として時間発展を計算する。計算には陰解法を使用し、3重対角行列を解いている。

ソースコード

- `scalelib/src/land/scale_land_phy_slab.F90`

  `SLAB`: 多層バケツモデルについて、各層の土壌温度および土壌水分を計算する。

6.3 海洋

格子点の土地利用データにおいて、「陸面」が100%でない限り、海洋が混じっている。この海洋モデルについて説明する。

6.3.1 水たまりモデル

現在の海洋モデルは1層しかなく、単に深さのある水たまりといえる。熱容量が非常に大きく、一瞬で混合されるという想定。

解くべき物理法則

熱収支の式は以下の通りとする。

$$C_w \frac{\partial T_w}{\partial t} = WH \tag{6.7}$$

ここでC_wは海洋熱容量、T_wは海洋温度、WHは地表面スキームにおける残差として得られた海中熱フラックス。海洋の温度を外部ファイルから入力して更新する場合、海中熱フラックスは無視される。

ソースコード

- `scalelib/src/ocean/scale_ocean_phy_file.F90`

  `FILE`: 外部入力ファイルから海洋温度の変化を読み取り、海洋温度を更新する。

- `scalelib/src/ocean/scale_ocean_phy_slab.F90`

  `SLAB`: 地表面スキームから得られた残差の海中熱フラックスにより、海洋温度を時間発展させる。

6.4 都市キャノピー

ここでは、陸面過程のうち「都市」の扱いについて特化して記述する。大気モデルの中での都市モデルの役割は、基本的に陸面モデルと同じで、地表面からの熱・運動量エネルギーを計算することである。

6.4.1 Single urban canopy model: Kusaka et al.(2001)

解くべき物理法則

都市モデルは、陸面モデルと同じく、地表面からの熱フラックスと運動量フラックスを評価するものである。 (Kusaka et al., 2001)は、単層キャノピーモデルであり、都市キャノピー内部のエネルギーのやり取りをモデル化し、最終的にはキャノピーと大気の間での交換フラックス量を見積もる。平板モデルでは、地表面を平板で表すが、キャノピーモデルでは建物の凸凹を考慮し、建物間の放射の多重反射の効果を考慮する。

地表面（キャノピー面）での熱エネルギーバランスは、

$$(1-\alpha)S^{\downarrow} + \varepsilon L^{\downarrow} - \varepsilon\sigma {T_s}^4 - H - lE = G \tag{6.8}$$

で表される。ここで、α, ε, σ はアルベド、射出率、ステファンボルツマン定数である。 $S^{\downarrow}$、$L^{\downarrow}$ は下向きの短波と長波放射であり、入力データとして与えられる。 H と lE は顕熱と潜熱で、基本的にはバルク式で求める。Gは貯熱量である。予報変数は、H、lE、表面温度T_sである。

平板モデルであれば、Eq.(6.8)を直接解くことになるが、キャノピーモデルでは、屋根面-大気、壁面・道路面-キャノピー、キャノピー-大気の間のエネルギー方程式をそれぞれ立て、それらの総和（実際には重み付け平均）がEq.(6.8)になるようにモデル化されている。詳細については参考文献 (Kusaka et al., 2001)を参照いただきたい。

離散化

離散化が必要なところは、建物内部の温度の計算である。建物内部の温度は、Eq.(6.8) で残差として計算したGを入力として時間発展を計算する。計算には陰解法を使用し、3重対角行列を解いている。

ソースコード

Kusaka et al.(2001)の単層キャノピーモデルによる計算を行っているサブルーチンを以下に示す。

- `scalelib/src/urban/scale_urban_phy_slc.F90`

  `URBAN_PHY_SLC`: 都市グリットからの各フラックス、温度を計算。

Chapter 7

I/O

7.1 設定・ログ

SCALE の各種設定は、Fortran の NAMELIST を利用して与える。 NAMELIST ファイルは、実行時引数として与える。

ログは、エラーメッセージは標準出力に、各種情報メッセージはファイルに出力される。ログのファイル出力は、デフォルトでは MPI の rank0 のプロセスのみ行われるが、全プロセスからそれぞれ出力させることも可能である。全プロセスから出力させるためには、 `PARAM_IO` NAMELIST group の `IO_LOG_ALLNODE` を `.true.` とする。ファイル名は、 `PARAM_IO` NAMELIST group の `IO_LOG_BASENAME` に ".pe******" をつけたものである。 "******" は、MPIプロセスのランク番号を6桁で表したものである。

NAMELIST の一覧は、デフォルトでは上記のログファイルに出力されるが、別のファイルに出力させることも可能である。そのためには、`PARAM_IO` NAMELIST group の `IO_NML_FILENAME` にファイル名を与える。このファイルは、SCALE の設定ファイルとしてそのまま実行時引数に与えることができる。

7.1.1 ソースコード

- `io/scale_stdio.F90` 各種ファイル装置番号を管理する

7.2 History・Restart

解析のために出力する時系列データ(ヒストリーデータ)と、再計算のための瞬間値のチェックポイントデータ(リスタートデータ)がある。

現在の SCALE は、netCDF フォーマットを採用している。最下層のファイルを入れ替えると他のフォーマットでも出力することが可能になる(はず)。デフォルトは、プロセス毎に分割された netCDF4 形式のファイルが出力されている。ファイル内のデータは zip 圧縮されている。

コンパイル時に、`USE_NETCDF3=T`としてコンパイルすることで、 netCDF3 形式も利用可能である。コンパイルオプションについてっは、8.4 を参照のこと。 netCDF3 形式を利用する際は、同じファイルに異なる時間間隔のデータを出力できない、圧縮することができないという制約がある。

また、pnetCDF を利用することで、全プロセスで単一のファイルの入出力が可能である pnetCDF を利用するためには、 `ENABLE_PNETCDF=T` としてコンパイルするとともに、実行時に、`PARAM_IO` NAMELIST group の `IO_AGGREGATE` を `.true.` とする。 pnetCDF による単一ファイルは、netCDF3 形式であり、上記の制約がある。

7.2.1 ソースコード

- `atmos-rm/io/scale_history.F90`

  `HIST_in`: ヒストリー出力用のデータを与える

  `HIST_get`: ヒストリーファイルからデータを読み込む

- `atmos-rm/io/scale_fileio.F90` SCALE-RM のデータ入出力を管理する
- `gtool/gtool_history.f90` ヒストリー出力の制御
- `gtool/gtool_file.f90` ファイルフォーマットによらないアブストラクトレイアー
- `gtool/gtool_file_h.f90` I/O 関係の定数、構造体定義
- `gtool/gtool_file_f.c` Fortran - C 間の変換
- `gtool/gtool_netcdf.c` netCDF へのマッピング

7.3 ExtIn

external_inputモジュールは、configファイルに記述したデータを読み込み、必要なサブルーチンが時間内挿しながら参照するためのユーティリティモジュールである。ナッジングのための時系列外部データをモデル時刻に合わせながら準備する、などの用途に用いることができる。外部データはモデルの並列数・格子数と一致していなければならない。

- EXTIN_setup
  configファイルに書かれているEXTITEMネームリストを探して読む
  または、どこかのコンポーネントがEXTIN_registを直接コールする
- EXTIN_regist
  ファイルを読み込んで指定の変数があるか探す
  指定の変数の各時刻が現在時刻の期間内に含まれているかチェックする
  現在時刻の前と後ろの二つの値（1D,2D,3D）をバッファしておく
  便利オプションあり
  - 値にオフセット値を追加する、見つからない場合に定数を返す
  - 時刻無視してステップ番号指定してずっと利用
  - 年を無視して毎年使い回す
  - 月を無視して同年同月を毎日使い回す
  - 日を無視して同年同月同日の値を毎時使い回す

- モデル内での利用法
  あるサブルーチンからEXTIN_updateをコールして、指定した変数の値（1D,2D,3D）を受け取る

- EXTIN_update
  現在時刻が進んでいれば、後ろの値を前に移動して後ろの値を新たに読み込む
  時間方向に線形内挿した値を返す

7.3.1 ソースコード

- io/scale_external_input.F90

7.4 初期値・境界値

モデルの地形・土地利用や大気の初期状態・境界条件といった初期値・境界値の設定は、scale-pp、scale_init バイナリにて行われる。

7.4.1 理想実験

機能

- 理想実験の場合は、scale-ppは使わず、scale_initで全ての境界条件を設定する。

- あらかじめ用意されたテストケースに対応する設定は、ネイムリストを設定するだけで良い。

- ソースコードにない条件を設定したい場合には、mod_user.f90 により与えることが可能である。

ソースコードの参照先

- scale-rm/src/preprocess/mod_mktopo.F90（地形）

- scale-rm/src/preprocess/mod_mkinit.F90（初期値・境界値）

ドキュメント

- 今はない。いずれ、UGに追加の必要あり。

7.4.2 現実大気

機能

- scale-rm_initは大気条件のみ、地形はscale-rm_ppが担当。

- 元のフォーマットから、SCALEグリットの値を内挿により計算、NetCDF形式にて出力。

- 内挿方法には、いくつか選択肢が設けられている。PARAM_NESTのNEST_INTERP_LEVELにより設定。

対応済み地形・土地利用データ

- 地形
  - GTOPO30（全球、HPで配布）
  - DEM50M（日本域のみ）
- 土地利用
  - GLCCv2（全球、HPで配布）
  - LU100M（日本域のみ）

大気データの入力フォーマット

- GrADS形式
- SCALE history file
- WRF output file
- NICAM AMIP run（これは、単にNetCDFフォーマットに変更するのが望ましい）

ソースコードの参照先

- 地形
  `scale/scale-rm/src/preprocess/mod_cnvtopo.f90`
- 土地利用
  `scale/scale-rm/src/preprocess/mod_cnvlanduse.f90`
- 初期値・境界値
  `scale/scale-rm/src/preprocess/mod_realinput.f90`
  `scale/scale-rm/src/preprocess/mod_realinput_scale.f90`
  `mod_realinput_nicam.f90`
  `mod_realinput_wrfarw.f90`
  `mod_realinput_grads.f90`
- 内挿ルーチン
  `scalelib/src/atmos-rm/grid/scale_interpolation_nest.F90`

ドキュメント

- 地形と土地利用（UG 4章、5.3節）
- 大気データ（UG 5.4節）
- 内挿方法（記載無し）

Chapter 8

ツール群

8.1 CONeP

SCALE-RMには1-way Online Nesting システムが実装されている． Cost-effective Online Nesting Procedure (CONeP; (Yoshida et al., 2017))という並列計算時にも計算効率を維持しやすい機構が用いられている．

CONePでは，計算に使用するMPIプロセスを計算ドメイン数のグループに分割して各ドメインを同時的に計算する．プロセスの分割と割り当てイメージをFig. 8.1.1に示した．例えば，2つの計算ドメインを使用する場合，MPIプロセスを2つのグループに分割し，それぞれのMPIプロセスグループを各ドメインに割り当て，各プロセスは分割された計算ドメインを担当する．分割時の各グループのMPIプロセス数は，ドメインの問題サイズに依存する．最も単純なポリシーに従えば，どのドメインも1プロセス当たりの格子点数と単位時間の積分に必要なステップ数の積が揃うように設定する． Online Nestingの実行方法はUGの5章に詳しい説明がある．

実装方法

- MPIを用いて実装されており，MPIプロセスのグループ分割（`COMM_WORLD`の分割）は`MPI_COMM_SPLIT`を用いている．
- 実態としては，各ドメインが別々のモデル計算として動いており，必要なときに同期を取り，通信する形になっている．そのため，各ドメインの実験設定について特にOnline Nestingだけで発生する制約がない．
- ドメイン間通信にはISEND/IRECVを用いて，できる限り時間積分のバックグラウンドで通信出来るように実装している．
- 子ドメインにおける境界条件を作成するときに使用される内挿方法は，現状は簡単な線形内挿である．これはOnline NestingもOffline Nestingも共通である．

ソースコード

CONePを構成するソースコードを下記に挙げる．

- `scale-lib/src/atmos-rm/grid/scale_grid_nest.F90`： nestingのメインモジュール

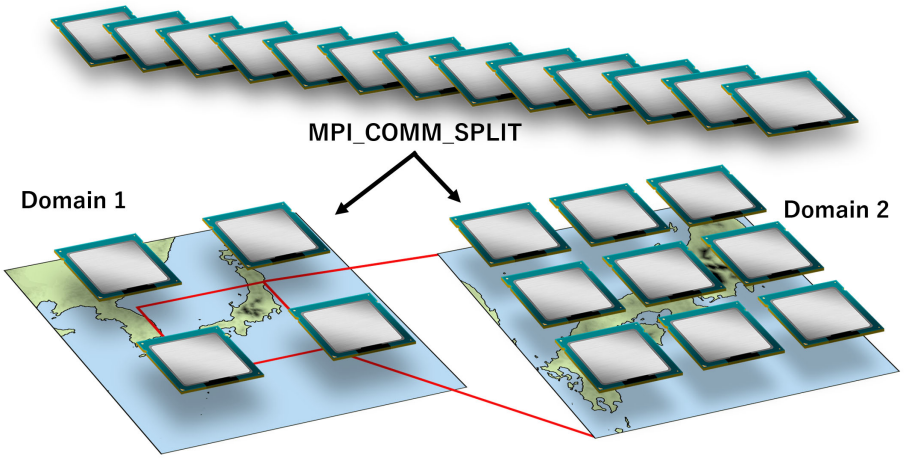

Figure 8.1.1: CONePのMPIプロセス分割とドメインへのプロセス割り当てイメージ

- `scale-lib/src/atmos-rm/grid/scale_interpolation_nest.F90` : nesting用の内挿ルーチン
- `scale-lib/src/atmos-rm/forcing/scale_atmos_sub_boundary.F90` : boundaryモジュール
- `scale-lib/src/common/scale_process.F90` : `COMM_WORLD`の管理とSPLIT関数
- `scale-lib/src/scale_rm.F90` : launcherプログラム

8.2 Bulk Job System

SCALE-RMにはバルクジョブ実行システムが実装されている．スーパーコンピュータ等でジョブを実行する場合，同時に投げられるジョブ数が制限されていることがある．そのような場合，ジョブ間で依存関係がない計算，例えばパラメータスウィープ実験，アンサンブル実験，Time Slice実験であればバルクジョブ実行システムを用いて一括実行することができる．

実装

CONePのために実装されたMPIプロセスのグループ分割を用いて実装されている． Fig. 8.2.1のイメージ図のようにMPIプロセスが分割される．バルクジョブ実行システムを用いる場合は，Online NestingのためにMPIプロセスが分割される前にジョブ数に応じて，もともとのMPIプロセスを分割しておく．そして，各ジョブの中で再びドメイン数に応じてMPIプロセスが分割される．使用方法は，UGの5.12節を参照して欲しい．

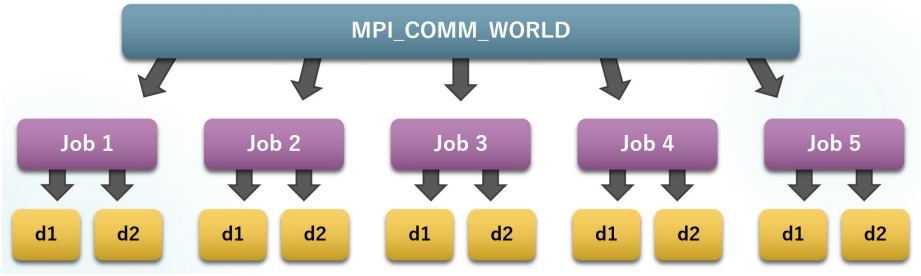

Figure 8.2.1: バルクジョブ実行システムにおけるMPIプロセス分割イメージ

ソースコード

Online Nestingシステムのフレームワークを使用して実現しているため，特別に追加されたファイルは無い．バルクシステムのメイン部分はlauncherになる．

バルクジョブ実行システムを構成するソースコードを下記に挙げる．

- `scale-lib/src/common/scale_process.F90`：`COMM_WORLD`の管理とSPLIT関数
- `scale-lib/src/scale_rm.F90`：launcherプログラム

8.3 net2g

net2gはSCALEの分散型netcdfファイルを単一のGrADS型バイナリデータに変換するツールである．当所は大規模計算結果をQuick Viewするため，任意の層，任意の変数に限って処理することで，実行時間を短縮し，ハンドリングしやすいデータサイズにすることを目的としたツールであった．現在は機能拡張がすすみ，Gradsで解析する際のポスト処理として使用するなど，より広範な目的で利用されている．

変換作業はFig. 8.3.1のイメージ図のように実行される．分散型netcdfファイルとSCALE-RMを実行したときのrun.conf，およびnet2g用のrun.confを入力として， Grads用のCTLファイル，およびバイナリデータを出力する．

- 鉛直層毎，時間ステップ毎など多様な出力ファイル形式が可能
- 鉛直層は任意の高度（m or Pa）に変換可能
- Grads用のCTLファイルも自動生成するようになっており，ランベルト図法に対応したCTLファイルを生成することも出来る．

実装

計算を自分で行わず，計算結果の解析だけをしたいというユーザーの要望に応えるため， SCALE内部のLibraryに依存しない形で実装されており，MPI並列からI/Oにまで独自関数によって構築されている． MPIを使わないでコンパイルすることも出来る．

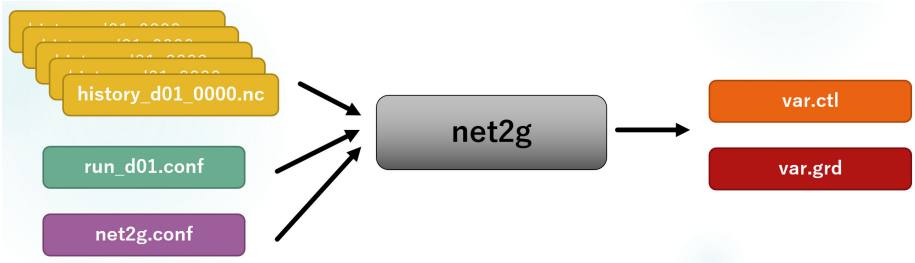

Figure 8.3.1: net2gの実行イメージ

ソースコード

net2gを構成するソースコード群を以下に示す。

- `scale-rm/util/netcdf2grads_h/src/mod_net2g_anal.F90`: 鉛直内挿等の計算
- `scale-rm/util/netcdf2grads_h/src/mod_net2g_calender.F90`: カレンダー処理
- `scale-rm/util/netcdf2grads_h/src/mod_net2g_comm.F90`: MPI通信
- `scale-rm/util/netcdf2grads_h/src/mod_net2g_error.F90`: エラー処理
- `scale-rm/util/netcdf2grads_h/src/mod_net2g_io.F90`: ファイルIOメインルーチン群
- `scale-rm/util/netcdf2grads_h/src/mod_net2g_netcdf.F90`: netcdf fortran関係のIOルーチン
- `scale-rm/util/netcdf2grads_h/src/mod_net2g_setup.F90`: プログラム開始時のセットアップ群
- `scale-rm/util/netcdf2grads_h/src/mod_net2g_vars.F90`: 変数コンテナ
- `scale-rm/util/netcdf2grads_h/src/prg_netcdf2grads_h.F90`: メインプログラム
- `scale-rm/util/netcdf2grads_h/inc/inc_net2g.h`: 定数や標準鉛直座標等の定義

8.4 コンパイルオプション

SCALEのビルドシステムは、configureやCMakeのようなツールは使っていない。汎用化は労力に見合わず、しかもクロスコンパイラを用いるようなスパコン環境では逆に障害となるケースも多いからである。基本的にGNU Makeの機能だけを使って、ビルドシステムを構築している。依存関係については、ソースコードから自動で収集するRubyスクリプトを準備しており、必

要に応じて更新する。いくつかのファイルは、Makeの実行時の環境変数によって使い分けるようになっている。

ビルドのための環境変数で最も重要なのは、`SCALE_SYS`である。この環境変数で指定された名前を元に、`/sysdep/Makedef.$SCALE_SYS`が利用される。このマシン別設定ファイルには、コンパイラやオプションについての設定群が記述されている。 FortranとCのコンパイルオプションには、fastとdebugの２つを用意し、後述する環境変数で切り替えるようにしている。

必ず依存しているライブラリはNetCDFのみである。NetCDFのライブラリパスは、原則として環境変数で指定されていることを期待している。スパコンなどで必ず誰もが同じライブラリを使うことが想定される場合は、マシン別設定ファイルに直接書かれていることもある。このどちらでも設定されていない場合、Makeの中でnc-config, nf-configコマンドを用いてライブラリを探そうとするが、万能ではないので明示的に環境変数で指定することを推奨する。

make時に設定できるSCALE環境変数は色々ある。bashだと`export XXX=T`という書き方で指定するが、makeコマンドに続けて`make XXX=T`という書き方でもよい。 INCLUDE, LIBS以外の変数は原則として、Tを設定していたら有効というルールに統一しており、環境変数が定義されているかいないかでの判定は用いない。

8.5 統計情報モニター

monitorモジュール実行時間を短縮し，ハンドリングしやすいデータサイズにすることを目的としたツールであった．現在は機能拡張がすすみ，Gradsで解析する際のポスト処理として使用するなど，より広範な目的で利用されている．

- MONIT_setup
  configファイルに書かれているMONITITEMネームリストを探して読む：リクエスト側

- MONIT_reg
  セットアップ時またはメインループ内でどこかのコンポーネントから呼ばれる：供給側
  リクエストされていれば登録

- MONIT_in, MONIT_put
  変数登録されていなければMONIT_reg
  変数配列を渡す
  STAT_total関数を使って領域平均値を作成

- MONIT_write
  PARAM_MONITORのMONITOR_STEP_INTERVALに指定されたステップ間隔でASCIIファイルに書き出す

- データの単位は？
  monitorモジュールはもらった変数を領域平均して指定のステップ間隔で書き出すのみ：MONIT_inやMONIT_regでユーザーが渡す変数の単位を明示的に与える

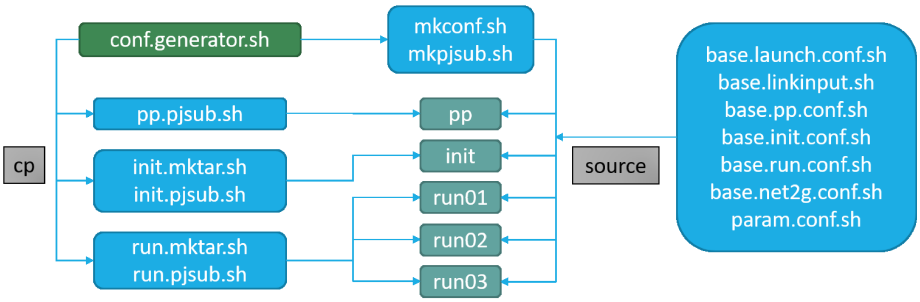

Figure 8.6.1: Configuration Generator の動作概念図。

- どの領域を平均しているの？
  PARAM_STATISTICSのSTATISTICS_use_globalcommの指定に依存する
  trueなら全領域を平均して、通信で集めて計算、1 ファイルだけ出す
  falseなら自分のプロセスの領域だけ計算、それぞれのプロセスが出す

8.5.1 ソースコード

- `io/scale_monitor.F90`

8.6 Conf Generator

Configuration Generatorは、もともとは京でプロダクトランを行う際に、設定の編集ミスを減らすために作られたものである。 SCALE-RMは現在、モデル実行の都合上、複数の設定ファイルに同じ記述、似た記述を何度も繰り返す必要がある。例えばバルクジョブ、リスタートジョブ、オンラインネスティングを含むジョブなどで、設定のコピペが発生しうる。そのため、共通部分を何度もコピペしなくても済むようにするため、ツールで設定ファイルを自動生成できるようにしている。

現在はSCALE-RM tutorialにおける実験設定自動生成ツールとしても組み込まれている。 Gitlabに登録してあるので、gitを使って簡単にダウンロードできる（`git clone git@ranchu.aics.riken.jp:scale/project.git`）。/project/config 以下、色々なプロジェクトに使ってきたconf.の数々がある。設定はプロダクトラン当時の設定のまま変更していないので、古いプロダクトランのものは最新版では使えない可能性が高い。自分で生成ツールを作るのであれば、最も新しいプロダクトラン設定のものを参考にするのが良い。

8.6.1 使い方

makeコマンドのみで実験設定が整う。ただし複雑な設定を施す場合や、基本以外の部分の設定を触る場合は、内部を直接弄る必要があるファイルのコピーや読込が複数あり、依存関係は一見するとわかりづらいので、動作を追うには図8.6.1を参照のこと。

Table 8.4.1: SCALE環境変数

名称	内容
SCALE_SYS	システム選択（必須）
SCALE_DISABLE_MPI	MPIを使わない（utilsのみ）
SCALE_DEBUG	デバッグ用コンパイルオプションでコンパイル
SCALE_QUICKDEBUG	クイックデバッグ用コンパイルオプション利用（高速化そのまま＋浮動小数点エラー検出）
SCALE_USE_MASSCHECK	質量保存チェック用の計算を追加（RM力学過程のみ）
SCALE_USE_SINGLEFP	単精度浮動小数点を使用（原則として全ソース）
SCALE_USE_FIXEDINDEX	格子サイズをコンパイル時に固定して最適化促進
SCALE_ENABLE_OPENMP	OpenMP機能を有効にする
SCALE_ENABLE_OPENACC	OpenACC機能を有効にする
SCALE_USE_AGRESSIVEOPT	副作用が出る可能のある強い最適化まで行う（京・FXのみ）
SCALE_DISABLE_INTELVEC	ベクトル化オプションの抑制（インテルコンパイラのみ）
SCALE_NETCDF_INCLUDE	NetCDFライブラリのincludeディレクトリパス
SCALE_NETCDF_LIBS	NetCDFライブラリのディレクトリパスとライブラリ指定
SCALE_ENABLE_PNETCDF	parallel NetCDFを利用する
SCALE_COMPAT_NETCDF3	NetCDF3互換の機能に限定する
SCALE_ENABLE_MATHLIB	数値計算ライブラリを利用する
SCALE_MATHLIB_LIBS	数値計算ライブラリのディレクトリパスとライブラリ指定
SCALE_ENABLE_PAPI	PAPIを利用する
SCALE_PAPI_INCLUDE	PAPIライブラリのincludeディレクトリパス
SCALE_PAPI_LIBS	PAPIライブラリのディレクトリパスとライブラリ指定
SCALE_DISABLE_LOCALBIN	テストケースディレクトリに特別版のバイナリが作られないようにする
SCALE_IGNORE_SRCDEP	コンパイル時にソースコードの依存関係確認を行わない
SCALE_ENABLE_SDM	超水滴モデルを利用する

Chapter 9

テスト

9.1 Test Case

テストとして、ライブラリの単体サブルーチンのバグ確認ための unit テストと、気象モデル計算の性能確認のための test case がある。

9.1.1 unit test

各サブルーチン毎に、入力を与え、出力が期待されたものであるかを確認する。確認には `dc_test` モジュールのチェックルーチンを利用する。

ソースコード

- `scalelib/test/unit/unit.f90` unit test メインプログラム
- `scalelib/test/unit/test_comm.f90` 通信モジュールテストルーチン
- `scalelib/test/unit/test_atmos_dyn.f90` 力学モジュールテストルーチン
- `scalelib/test/unit/test_atmos_phy_tb_smg.f90` SMG 乱流モジュールテストルーチン
- `dc_utils/dc_test.f90` データチェックルーチン集

9.1.2 test case

気象モデルとして、設定を与えて実行する。結果の妥当性は、別途描画等により確認する必要がある。

ソースコード

理想実験は `atmos-rm/test/case` 以下に、現実大気実験は `atmos-rm/test/case_real` 以下にある。表9.1.1 は、テストの一覧である。

Table 9.1.1: test case 一覧

理想実験	advection	パッシブトレーサー移流実験
	barocwave	順圧不安定実験
	boundarylayer	境界層実験
	boxaero	エアロゾルボックス実験
	cavity	キャビティー流実験
	coldbubble	密度流実験
	coupling	カップラー実験
	dns	DNS実験
	gravitywave	重力波実験
	grayzone	乱流グレーゾーン実験
	hydrostatic	静力学平衡実験
	khwave	K-H 不安定実験
	lambwave	ラム波実験
	mountanwave	山岳波実験
	orographicrain	山岳性降水実験
	rad-conv	放射対流平衡実験
	radiation	放射実験
	restart	リスタート実験
	seabreeze	海陸風実験
	shallowcloud	浅い雲実験
	squallline	スコールライン実験
	supercell	スーパセル実験
	turbulence	乱流実験
	twpice	TWC ICE 実験
	urban	都市キャノピー実験
	warmbubble	暖気塊実験
現実大気実験	`check_awajishima`	淡路島実験
	`check_japan`	日本実験
	`check_mass`	質量保存実験
	`restart_7.5km`	リスタート実験

9.2 CI

git の develop ブランチにコミットがあると、 9.1 節の各テストケースが自動的に実行され、結果の図が作成される。このように、ビルドやテストを継続的に実行していくことを継続的インテグレーション (Continuous integration; CI) と呼ぶ。 CI により問題の早期発見が期待される。また、問題が発覚した際、、図の履歴を見ることで、どのコミットの前後で変わったのか特定することも可能である。

unit test では、テストに失敗するとエラーになるため、テストの結果を自動で判別できる。 test case では、コンパイルや計算実行の失敗は自動判別ができるが、物理的な妥当性までは判断するようにしていない。そのため、作成される図を開発者が確認し、その妥当性を判断する必要がある。

SCALE では、CI を実現するために CI用ソフト Jenkins (1.2 参照) を利用している。結果の図は `http://jikin.aics27.riken.jp/jenkins_results/index.cgi` で見ることができる。また、AICS R504 に設置している SCALE health check monotor 上で常に表示されている。

References

Arakawa, A. and W. H. Schubert (1974), Interaction of a cumulus cloud ensemble with the large-scale environment. part i. *Journal of the Atmospheric Sciences*, 31, 674–701.

Bigg, E. K. (1953), The formation of atmospheric ice crystals by the freezing of droplets. *Quarterly Journal of the Royal Meteorological Society*, 79, 510–519.

Brown, A. R., S. H. Derbyshire, and P. J. Mason (1994), Large-eddy simulation of stable atmospheric boundary layers with a revised stochastic subgrid model. *Quarterly Journal of the Royal Meteorological Society*, 120, 1485–1512.

Deardorff, J. W. (1980), Stratocumulus-capped mixed layers derived from a three-dimensional model. *Boundary-Layer Meteorology*, 18, 495–527.

Gross, E. S., L. Bonaventura, and G. Rosatti (2002), Consistency with continuity in conservative advection schemes for free-surface models. *Int. J. Numer. Meth. Fl.*, 38, 307–327.

Kain, J. S. (2004), The kain-fritsch convective parameterization: an update. *J. Appl. Meteor.*, 43, 170–181.

Kain, J. S. and J. M. Fritsch (1990), A one-dimensional entraining/detraining plume model and its application in convective parameterization. *Journal of the Atmospheric Sciences*, 47, 2784–2802.

Kajino, M., R. Easter, and S. J. Ghan (2013), Modal bin hybrid model: A surface area consistent, triple-moment sectional method for use in process-oriented modeling of atmospheric aerosols. *Journal of Geophysical Research: Atmosphere*, 118, 10011?–10040.

Kessler, E. (1969), On the distribution and continuity of water substance in atmospheric circulation. *Meteorological Monograph*, 10, 1–84.

Koren, B. (1993), *A robust upwind discretization method for advection, diffusion and source terms*. Centrum voor Wiskunde en Informatica Amsterdam.

Kuo, H. L. (1965), On the formation and intensification of tropical cyclones through latent heat release by cumulus convection. *Journal of the Atmospheric Sciences*, 22, 40–63.

Kusaka, H., H. Kondo, Y. Kikegawa, and F. Kimura (2001), A simple single-layer urban canopy model for atmospheric models: comparison with multi-layer and slab models. *Boundary-Layer Meteorol.*, 101, 329–358.

Lilly, D. K. (1962), On the numerical simulation of buoyant convection. *Tellus*, 14, 148–171.

MacGorman, D., J. M. Straka, and C. L. Ziegler (2001), A lightning parameterization for numerical cloud models. *Journal of Applied Meteorology*, 40, 459?–478.

Mellor, G. L. and T. Yamada (1982), Development of a turbulence closure model for geophysical fluid problems. *Rev. Geophys. Space Phys.*, 20, 851–875.

Nakanishi, M. and H. Niino (2004), An improved mellor-yamada level-3 model with condensation physics: Its design and verification. *Bound-Lay. Meteorol.*, 112, 1–31.

Nishizawa, S., H. Yashiro, Y. Sato, Y. Miyamoto, and H. Tomita (2015), Influence of grid aspect ratio on planetary boundary layer turbulence in large-eddy simulations. *Geosci. Model Dev.*, 8(10), 3393–3419.

Scotti, A., C. Meneveau, and D. K. Lilly (1993), Generalized smagorinsky model for anisotropic grids. *Physics of Fluids A*, 5, 2306–2308.

Seiki, T. and T. Nakajima (2014), Aerosol effects of the condensation process on a convective cloud simulation. *Journal of the Atmospheric Sciences*, 71, 833–853.

Shima, S., K. Kusano, A. Kawano, T. Sugiyama, and S. Kawahara (2009), The super-droplet method for the numerical simulation of clouds and precipitation: a particle-based and probabilistic microphysics model coupled with a non-hydrostatic model. *Quarterly Journal of the Royal Meteorological Society*, 135(642), 1307–1320.

Smagorinsky, J. (1963), General circulation experiments with the primitive equations. *Mon. Weather Rev.*, 91, 99–164.

Stensrud, J., D. (2007), *Parameterization Schemes, Keys to Understarnding Numerical Weather Prediction Models*. The Edinburgh Building, Cambridge, CB2, 8RU, UK: Cambridge University Press.

Suzuki, K., T. Nakajima, T. Y. Nakajima, and A. P. Khain (2010), A study of microphysical mechanisms for correlation patterns between droplet radius and optical thickness of warm clouds with a spectral bin microphysics cloud model. *Journal of the Atmospheric Sciences*, 67(4), 1126–1141.

Takahashi, T. (1978), Riming electrification as a charge generation mechanism in thunderstorm. *Journal of the Atmospheric Sciences*, 35, 1536?–1548.

Tiedtke, M. (1989), A comprehensice mass flux scheme for cumulus parameterization in large-scale models. *Monthly Weather Review*, 117, 1779–1800.

Tomita, H. (2008), New microphysical schemes with five and six categories by diagnostic generation of cloud ice (¡special issue¿the international workshop on high-resolution and cloud modeling, 2006). *Journal of the Meteorological Society of Japan. Ser. II*, 86, 121–142.

Wicker, L. J. and W. C. Skamarock (2002), Time-splitting methods for elastic models using forward time schemes. *Monthly Weather Review*, 130(8), 2088–2097.

Yoshida, R., S. Nishizawa, H. Yashiro, S. A. Adachi, Y. Sato, T. Yamaura, and H. Tomita (2017), Conep: A cost-effective online nesting procedure for regional atmospheric models. *Parallel Computing*, 65, 21–31.

Zalesak, S. T. (1979), Fully multidimensional flux-corrected transport algorithms for fluids. *J. Comput. Phys.*, 31, 335–362.