JAEA-Data/Code
2007-002

JP0750056

JAEA-Data/Code

大気・海洋・陸域モデル結合のための モデルカップラー

Model Coupler for Coupling of Atmospheric, Oceanic, and Terrestrial Models

永井 晴康　小林 卓也　都築 克紀　金 庚玉

Haruyasu NAGAI, Takuya KOBAYASHI, Katsunori TSUDUKI
and Keyong-Ok KIM

原子力基礎工学研究部門
環境動態研究グループ

Research Group for Environmental Science
Nuclear Science and Engineering Directorate

February 2007

Japan Atomic Energy Agency　日本原子力研究開発機構

本レポートは日本原子力研究開発機構が不定期に発行する成果報告書です。
本レポートの入手並びに著作権利用に関するお問い合わせは、下記あてにお問い合わせ下さい。
なお、本レポートの全文は日本原子力研究開発機構ホームページ（http://www.jaea.go.jp/index.shtml）より発信されています。このほか財団法人原子力弘済会資料センター*では実費による複写頒布を行っております。

〒319-1195　茨城県那珂郡東海村白方白根2番地4
日本原子力研究開発機構　研究技術情報部　研究技術情報課
電話 029-282-6387, Fax 029-282-5920

*〒319-1195　茨城県那珂郡東海村白方白根2番地4　日本原子力研究開発機構内

This report is issued irregularly by Japan Atomic Energy Agency
Inquiries about availability and/or copyright of this report should be addressed to
Intellectual Resources Section, Intellectual Resources Department,
Japan Atomic Energy Agency
2-4 Shirakata Shirane, Tokai-mura, Naka-gun, Ibaraki-ken 319-1195 Japan
Tel +81-29-282-6387, Fax +81-29-282-5920

© Japan Atomic Energy Agency, 2007

JAEA-Data/Code 2007-002

大気・海洋・陸域モデル結合のためのモデルカップラー

日本原子力研究開発機構原子力基礎工学研究部門環境・放射線工学ユニット
永井　晴康、小林　卓也、都築　克紀、金　庚玉※

(2007年1月4日受理)

数値実験による様々な環境研究に資することのできる環境研究ツール「数値環境システムSPEEDI-MP」は、大気、陸域、海洋の物理モデル及び物質循環モデル等の数値実験ツール、数値実験ツールの入力として用いる気象データ、地図情報データ等のデータベースと、データ管理、可視化及びGUIを用いた制御機能からなる支援機能により構成される。数値実験ツール開発の一環として、多数のモデルを結合できるモデルカップラーを開発した。モデルカップラーは、複数のモデルを同時進行で平行計算させモデル間で計算結果を交換することにより、モデルを一体化したのと同等な結合状態を作り出すことができる。結合可能なモデルは、3次元構造格子を持つモデルで、ほとんどの環境モデルや流体力学モデルに汎用的に適用することができる。このモデルカップラーを用いて、大気、海洋、波浪、陸水、及び地表モデルの5モデルからなる水循環結合モデルシステムを開発した。水循環結合モデルシステムの適用試験として、サウジアラビアにおける2005年1月の洪水再現計算及び2005年8月のハリケーン・カトリーナによる高潮再現計算を実施した。

原子力科学研究所（駐在）：〒319-1195 茨城県那珂郡東海村白方白根2-4
※ 博士研究員

JAEA-Data/Code 2007-002

Model Coupler for Coupling of Atmospheric, Oceanic, and Terrestrial Models

Haruyasu NAGAI, Takuya KOBAYASHI, Katsunori TSUDUKI and Keyong-Ok KIM※

Division of Environment and Radiation Sciences
Nuclear Science and Engineering Directorate
Japan Atomic Energy Agency
Tokai-mura, Naka-gun, Ibaraki-ken

(Received January 4, 2007)

A numerical simulation system SPEEDI-MP, which is applicable for various environmental studies, consists of dynamical models and material transport models for the atmospheric, terrestrial, and oceanic environments, meteorological and geographical databases for model inputs, and system utilities for file management, visualization, analysis, etc., using graphical user interfaces (GUIs). As a numerical simulation tool, a model coupling program (model coupler) has been developed. It controls parallel calculations of several models and data exchanges among them to realize the dynamical coupling of the models. It is applicable for any models with three-dimensional structured grid system, which is used by most environmental and hydrodynamic models. A coupled model system for water circulation has been constructed with atmosphere, ocean, wave, hydrology, and land-surface models using the model coupler. Performance tests of the coupled model system for water circulation were also carried out for the flood event at Saudi Arabia in January 2005 and the storm surge case by the hurricane KATRINA in August 2005.

Keywords: SPEEDI-MP, Numerical Simulation, Environmental Studies, Model Coupler, Coupled Model System, Water Circulation

---

※ Post-Doctoral Fellow

目次

Contents

1. 緒言

近年、アジアでの原子力施設の増加、国内核燃料サイクル施設の稼動、原子力軍艦の寄港、核テロなど、公衆への被ばくの潜在的可能性は多様化・増加している。原子力事故時等の緊急時対策には、放射性物質の大気放出による早期の直接的な被曝防護と、大気から土壌や海洋への移行に伴う、中・長期的なモニタリングや食物摂取制限・立ち入り制限がある。早期対応に必要な情報を提供するシステムとしては、緊急時環境線量情報予測システムSPEEDI(System for Prediction of Environmental Emergency Dose Information) [1]及び世界版SPEEDI (WSPEEDI) [2]をこれまでに開発し、現在SPEEDIが「緊急時迅速放射能影響予測ネットワークシステム」[3]として文部科学省により運用されている。しかし、中・長期的な対策に必要な基礎情報を提供するためには土壌や海洋への拡散予測も必要であるが、そのような機能を持つ予測システムは存在しない。また、放射性物質は気圏、陸圏及び水圏を物理・化学的状態を変えて移行・循環し、公衆はその一部をいろいろな過程を経て摂取する。このような状況に包括的に対応するためには、放射性物質の大気・海洋・陸域での移行挙動を連続的に扱うことのできる新たな包括的数値シミュレーション技術が不可欠であり、大気、海洋、地表モデル群の同時計算とダイナミックリンクにより、水分、熱、運動量、汚染物質等の各圏間の相互作用を詳細に考慮できる局地規模の数値環境を開発する必要がある。

日本原子力研究開発機構（原子力機構）では、最新の数値シミュレーション技術を基盤とした大気、海洋、陸域環境モデルと環境中物質循環モデル、データ管理、可視化及びGUIを用いた制御機能から成る「数値環境システム SPEEDI-MP (SPEEDI Multi-model Package)」[4]の構築を進めている（図1)。SPEEDI-MPは、目的に応じて必要なソフトウエアを組み合わせることで、原子力事故時の早期対応から中・長期的な対策に必要な予測情報の提供が可能であるとともに、数値実験による様々な環境研究に資することのできる環境研究ツールである。数値実験ツールとしては、大気、陸域、海洋それぞれについての数値モデル群が登録され、システム上で実行可能となっている。また、これらのモデルの同時計算とダイナミックリンクにより、各圏間の相互作用を詳細に考慮できるモデル結合機能を提供するモデルカップラーを開発し、各種結合モデルシステムを構築している。このモデルカップラーによる結合モデルシステムの構築がSPEEDI-MPの数値実験ツールの特徴であり、原子力事故時の早期対応から中・長期的な対策に必要な予測情報の提供が可能であるとともに、数値実験による様々な環境研究に資することのできる環境研究ツールを実現する上で不可欠なソフトウエアである。本報告は、このモデルカップラーとこれにより構築された結合モデルシステムについて記述する。

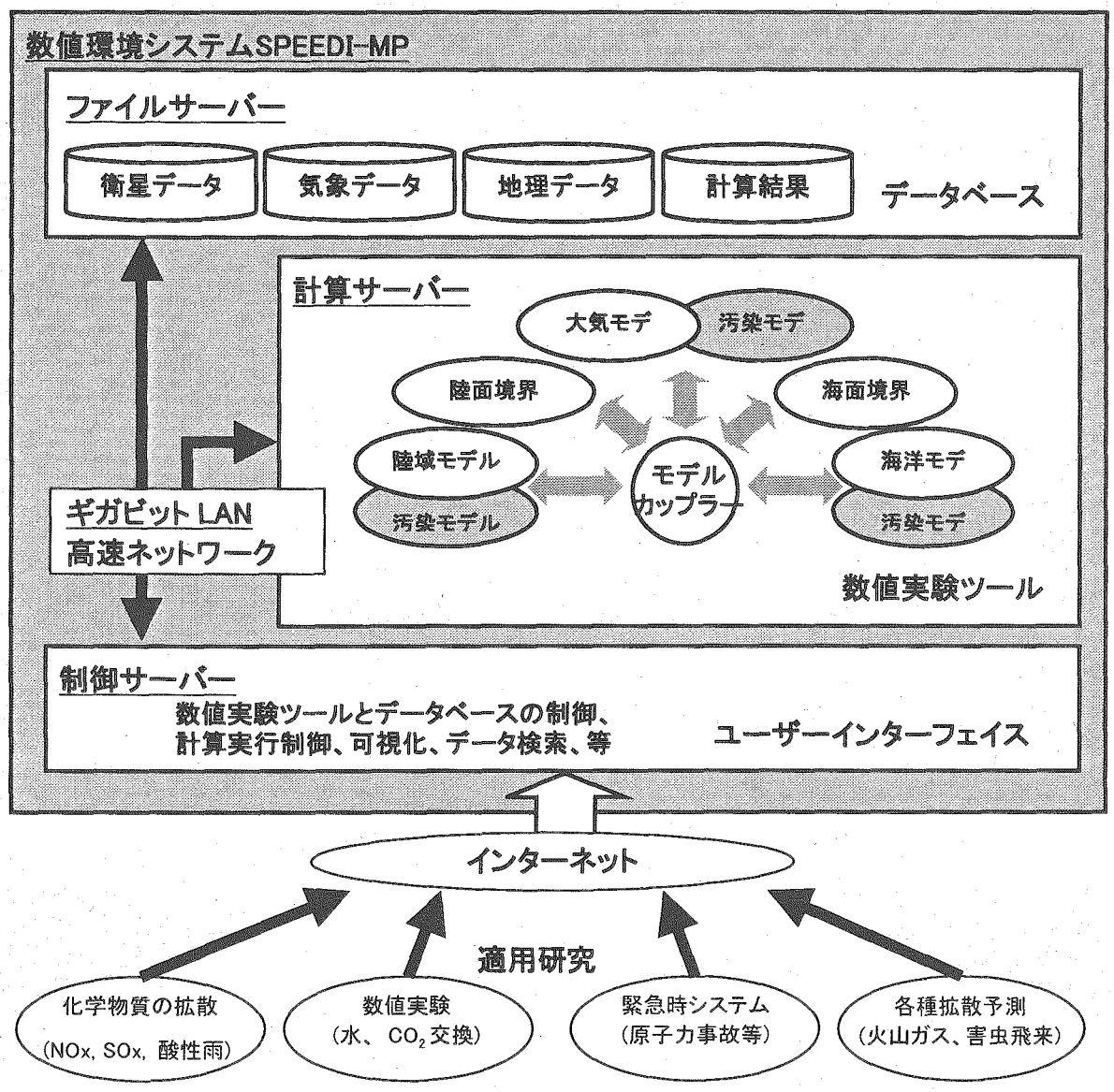


図 1　　数値環境システム SPEEDI-MP の構成

2. カップラーの概要

数値環境システム SPEEDI-MP の数値実験ツールにおける最終的な目標である大気、海洋、陸域を結合した包括的物質動態モデルを実現するために、多数のモデルを結合できるモデルカップラーの開発を行った。本カップラーは、複数のモデルを同時進行で平行計算させ、並列計算の通信ライブラリ MPI を用いてモデル間で計算結果を交換することにより、モデルを一体化したのと同等な結合状態を作り出すことができる（図 2)。複合系の連動計算において、従来は各系のモデル計算を逐次実行し計算結果をファイルに格納して受け渡すためモデル間相互作用のうち一方向のみ考慮していただけであるが、モデルカップラーを用いることで双方向の相互作用を考慮できるようになり、ファイル保存領域の制限を受けないため頻繁な計算結果の交換も可能である。

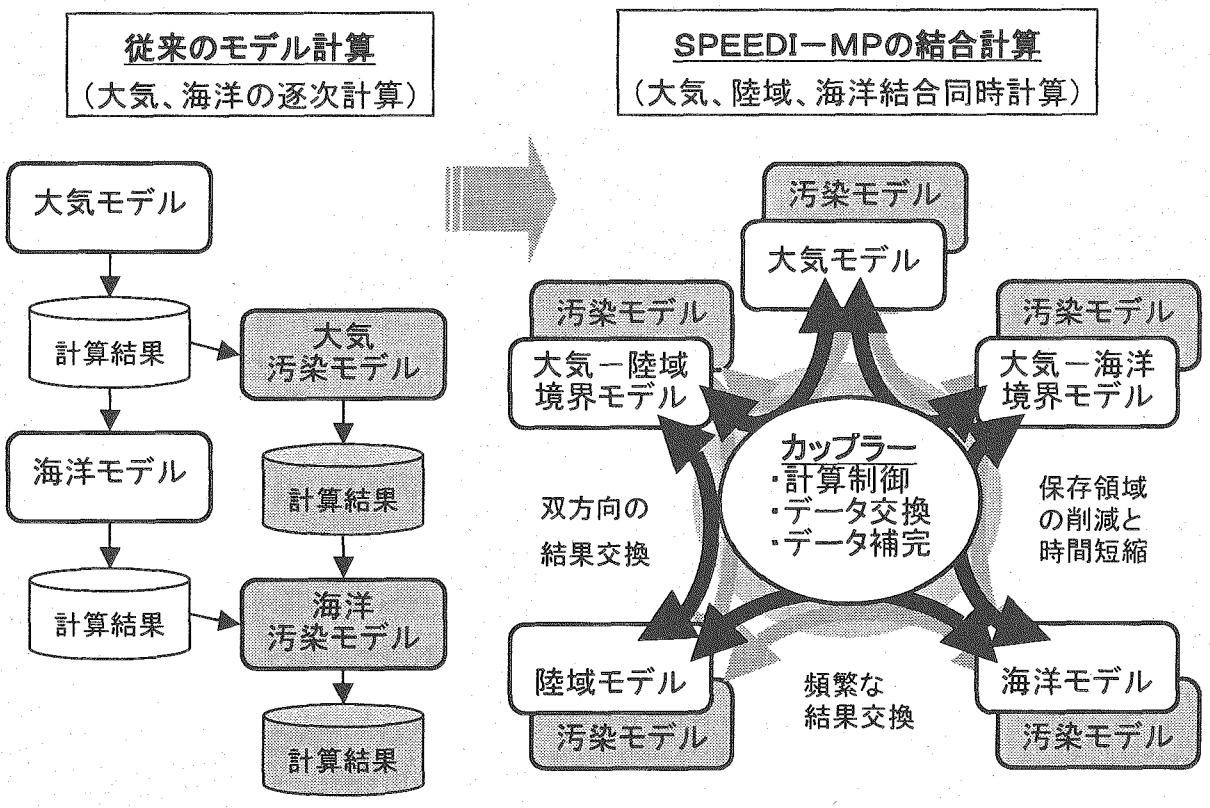


図 2　　数値環境システム SPEEDI-MP のカップラーによるモデル結合の概念

2．1　機能

カップラーの機能は、複数のモデルを並列に実行するための MPI 設定とモデル起動等の計算制御、モデル間でのデータ交換の中継とファイル入出力、及びデータの補間（時間、空間補間）である。必要な動作環境は、プラットフォームとして分散または分散共有メモリ型並列計算機で並列ライブラリ MPI が準備されていることと、結合する各モデルが構造格子を採用し非並列または MPI で並列化されていることである。カップラーの機能の特徴を列挙すると以下のようになる。

複数モデルのカップリング

結合するモデルの数は無制限である。設定ファイルの仕様上、デフォルトでは 8 つのモデルまでプログラムの修正なしにカップリングできる。それ以上の数のモデルをカップリングする際は、プログラムの定数を変更する必要がある。

3 次元構造格子のモデルを対象

結合するモデルは、3 次元構造格子を持つものとする。格子の間隔やモデル相互の格子の位置関係は任意である。

ヴォリュームデータの交換

データ交換では、1 次元、2 次元及び 3 次元配列までのデータを扱うことができる。

スタッガードグリッドに対応

モデル格子については、スタッガードグリッドに対応しており、一つのモデルが複数のグリッドを持つことを前提としている。各モデルのグリッドの数はモデル毎に任意に設定できる。

ネスティングに対応

各モデルは、ネスティングによる複数の領域を持つことができる。あるモデルの任意の領域のデータを、別のモデルの任意の領域と交換することが可能である。

任意の補間プログラムを実装可能

モデル間でデータ交換する際のデータ補間については、利用者が各自のコードを実装できる設計になっている。

パーティクル情報を交換可能

格子上のデータだけではなく、位置とあるスカラー量を持つ複数の粒子の情報を他モデルと交換可能である。

カップラーのタスク並列

カップラープログラム自体を複数のプロセッサで起動可能である。

データ入出力機能

モデル間で直接データを交換するだけでなく、送信モデルのみ起動して送信データを一旦ファイルに出力し、受信モデル起動時にそのファイルからデータを取得する（オフライン結合）ことも可能である。

可視化用データ出力

カップラーのユーティリティー機能として、カップラーからの出力を数値環境システムSPEEDI-MP の可視化機能で利用可能なデータフォーマットへ変換するプログラムを準備している。カップラーは、このデータ出力を行う際に異なる座標系のモデル出力を一つのモデルの座標系に統一する。これにより、結合した各モデルの計算結果を同一画面に重ね合わせて可視化することが可能となる。

2．2　処理

本節では、カップラーによるモデル結合の際の処理の流れを概説する。カップラーと各モデルの処理の流れを図3に示す。以下、処理の流れに従って各処理の概要を記述する。

（1）プロセスの初期化

プロセスの初期化処理には、カップラーの初期化とモデルの初期化がある。カップラーは、モデル設定ファイルとモデル毎のデータ設定ファイルの2種類（数は起動するモデルによる）の設定ファイルを読み込み、起動すべきモデルの情報を得た後にモデルを起動する。モデルを起動した後、カップラーから各モデルに対して設定ファイルの情報を送信する。カップラーによって起動された各モデルは、設定ファイルの内容をカップラーから受信し、他モデルの動作情報を得る。

（2）グリッド情報の設定

データ交換を行う前に、各モデルはグリッド情報と時間情報をカップラーに知らせる。図ではモデルの初期化の前にグリッド情報を設定するように記しているが、実際にはデータ交換の開始以前であればどこで行ってもかまわない。

（3）データ交換

各モデルは、時間積分の中で他モデルとのデータ交換を行うが、プログラム中任意の場所でデータ交換を行うことができる。各モデルは、データ送受信を行う前に、時刻と領域情報をカップラーに知らせる。送信については、時刻・領域ともに任意に設定でき、時間をさかのぼって送信を行うことも許される。受信については、時刻をさかのぼって同一領域の同一データを受信することはできない。交換／補間を行うデータは一度に一つだけなので、複数のデータを用いて調整する必要のある量（例えば風速成分 u、v、w について質量保存を満たすように調整する場合）については、受信後にモデル側で行う。

（4）カップリングの終了

各モデルが時間積分を終了しデータ交換の必要がなくなった時点で、その旨をカップラーに通知する。カップラーは、全てのモデルから終了通知を受けた時点で動作を停止する。

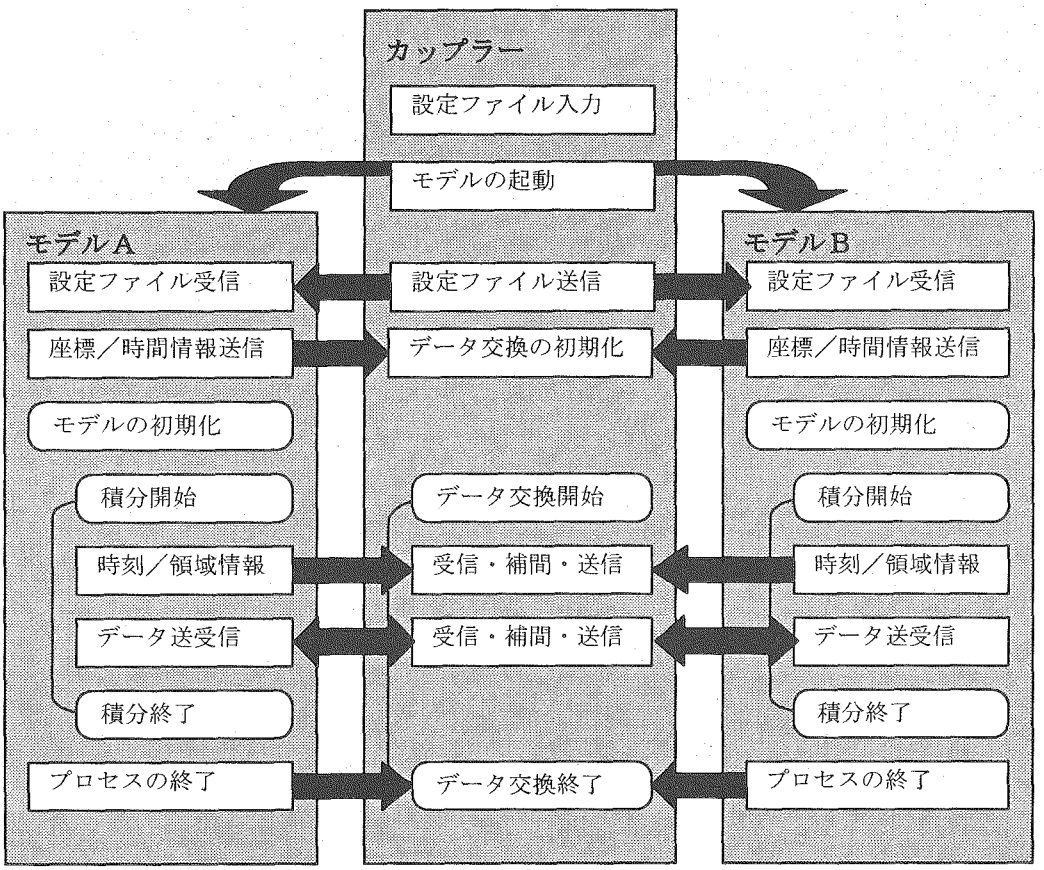


図3　カップラー結合計算の処理の流れ

3. 機能の詳細

3．1 起動と初期化

カップラーは、プロセスが起動されるとモデル設定ファイルを読み込む。起動されたカップラーのプロセス数からカップラーがタスク並列（マルチカップラーモード）か単一プロセス（シングルカップラーモード）かを判定し、設定ファイルの内容をチェックする。マルチカップラーモードの場合、起動されたカップラーのプロセス数とカップリングするモデルの数は同数で、かつ、モデルは全て実行する設定でなければならない。設定に問題があった場合、カップラーはエラーメッセージを出力し終了する。次いで、モデル設定ファイルの情報に基づき、データ設定ファイルからデータ送受信情報を取得する。データ送受信情報についてもエラーチェックを行い、問題がなかった場合はモデル設定ファイルの各モデルの動作環境情報に従ってモデルを起動する。その後、起動したモデルに対してモデル設定ファイル及びデータ設定ファイルの内容を送信する。起動されたモデルは、カップラーから設定ファイルの内容を受信し、自モデルの ID や他モデルの情報を取得する。結合を行う各モデルとカップラーは別プロセスとして動作するため、各モデルとカップラー間の MPI 通信は、図 4 に示すようにプロセス間コミュニケータを用いる。カップラーが各モデルを起動するため、カップラーが親プロセス、各モデルが子プロセスとなり、親子間通信を行う。カップラーはモデルの数だけの対子コミュニケータを持ち、モデルは一つの対親コミュニケータを持つことになる。モデルが並列化されている場合、モデル内の通信はローカル通信であり、カップラーとは無関係に動作する。

カップラーは、ネスティング機能やスタッガードグリッドを採用したモデルに対応可能となっている。このようなモデルでは、多数の計算領域や変数毎に異なるグリッドを持つため、結合した各モデルの任意の計算領域の任意の変数を交換するためには、カップラーが複雑なグリッド情報を管理する必要がある。グリッド情報の取得は、各モデルの初期化後、図 5 に示すフローに従って実行される。まず、各モデルの領域（ネストした領域）の数と格子系の数を受信し、グリッド情報を保持する変数を allocate する。次に、各モデルのグリッド情報を受信し、グリッド情報を保持する変数の各要素に代入する。マルチカップラーモードの場合は、結合するモデルは全て実行されるため、実行の有無の判定は必要ない。実行モデルから値を受信した場合は、その値をファイルに出力する。

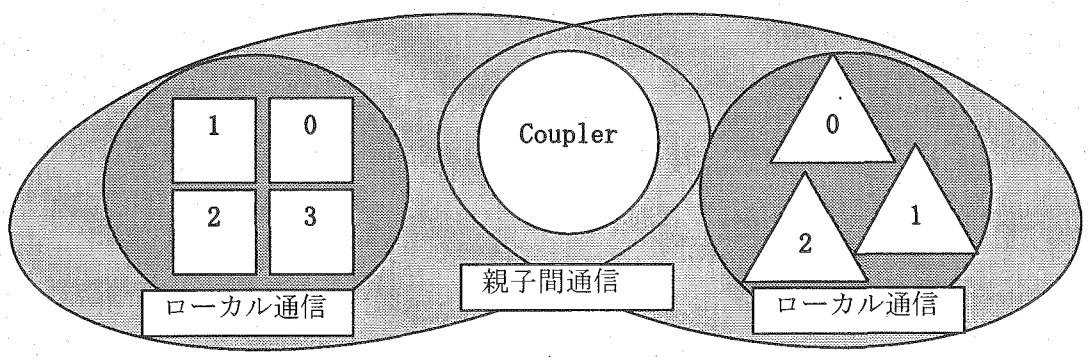


図 4　データ通信の概念図

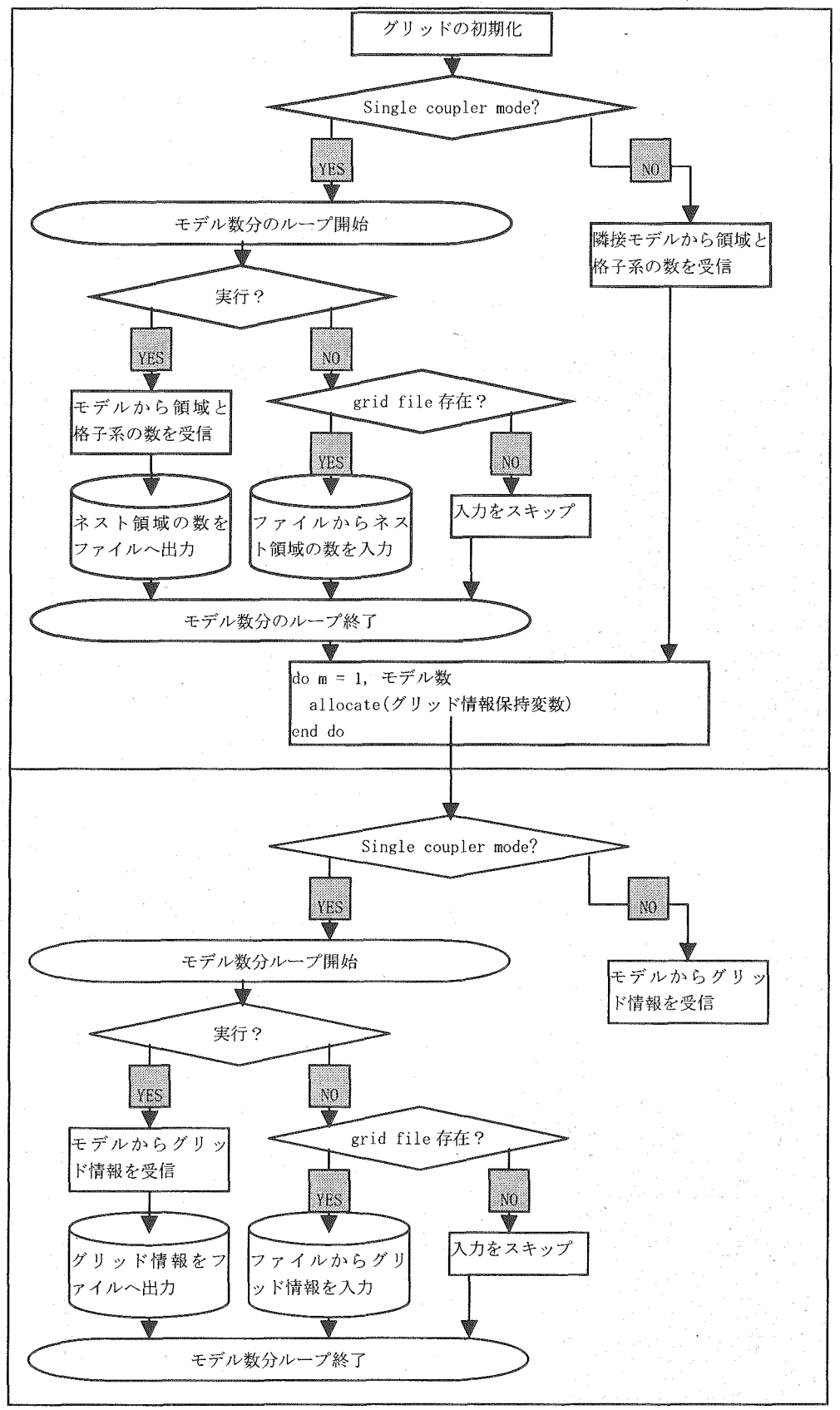


図5　グリッド初期化フロー

３．２　データ交換

本節では、モデル間のデータ送受信アルゴリズムについて記述する。カップラーを介したデータ送受信の処理の流れを、（１）送信の場合、（２）受信の場合、（３）データ交換の詳細について記述する。ここで、送信側をモデルA、受信側をモデルBとする。

（１）送信の場合

カップラーがモデルAからデータを受ける場合のアルゴリズムは、図6のようになる。カップラーは、送信モデルから送信要求を受信すると、そのデータの送受信情報からデータをファイルに出力すべきかどうかを判定する。その後、データを受信し、ファイルに出力すべき場合には出力する。次に、そのデータの送信待ち受けモデルを検索し、データ送信待ちのモデルがあった場合はデータ交換を行う。

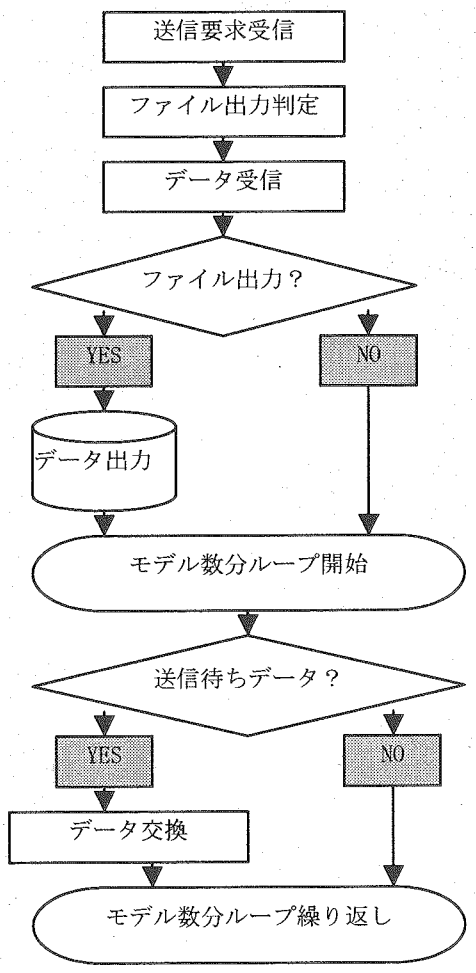


図6　データ送信の概念図

（2）受信の場合

カップラーがモデルBにデータを送信する場合のアルゴリズムを図7に示す。カップラーがデータ受信要求を受信すると、送信側モデルが実行されているかどうか確認し、送信データをファイルから読み込むかどうかを判定する。送信データが実行されておらずファイルから読み込む場合、データが存在しなければエラー終了する。データが存在する場合には、データ交換を行う。送信モデルが実行されている場合、メモリ上から送信データを検索し、データが存在していればデータ交換を行い、データが存在しなければ受信モデルを送信データ待ち受け状態とする。

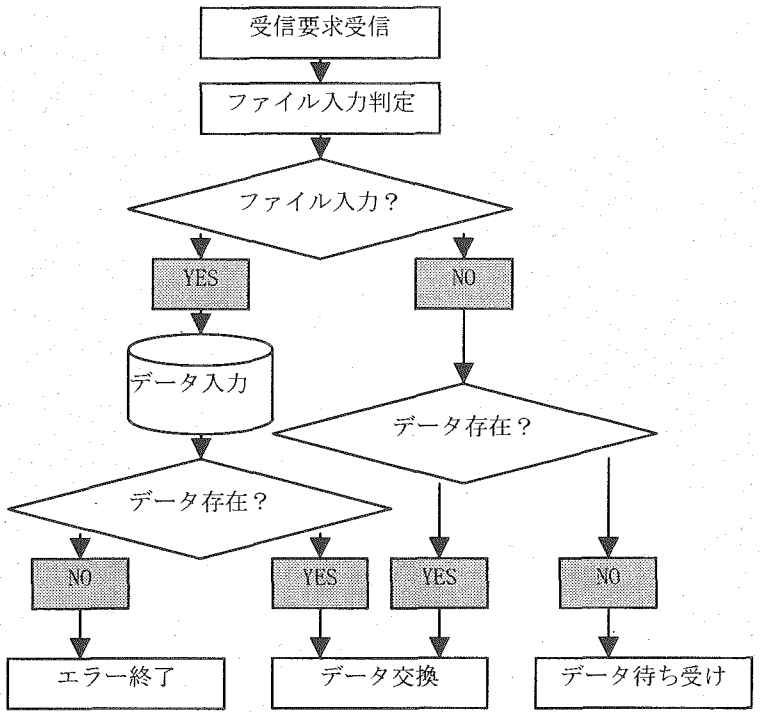


図7　データ受信の概念図

（3）データ交換の詳細

データ交換は、対象データが1次元か、2次元または3次元か、パーティクルデータかで動作が異なる。

1次元データの場合

送受信時刻から時間補間が必要かどうかを判定し、時間補間が必要な場合は補間処理を行った後、受信モデルへデータを送信する。時間補間が必要でない場合はそのままデータを送信する。

2次元または3次元データの場合

1次元データと同様に、時間補間が必要かどうかを判定し、時間補間が必要な場合は補間処理を行った後に空間補間を行う。時間補間が必要ない場合はそのまま空間補間を行う。空間補間

されたデータは、受信モデルへ送信される。

パーティクルデータの場合

パーティクルデータは、時間補間を行ってはならないため、送受信時刻が一致しない場合にはエラー終了する。一致した場合はそのデータを受信モデルへ送信する。

モデルからのデータ送信は、計算上の時刻、領域によらず自由に行え、また、各ステップのデータ数も自由に変えることができる。このため、送信データを保持するデータ構造は、この要求に対応するような形でなければならない。送信データを保持する構造型変数を図8に示す。

```
type data_type
  type(data_type),pointer :: next_ptr
  type(data_type),pointer :: before_ptr
  type(grid_type),pointer :: grid
  character(len=NAME_LEN) :: data_name
  integer                 :: tb_flag
  type(recv_flag_type)    :: recv_flag
  integer,pointer,dimension(:)     :: idata1d
  integer,pointer,dimension(:,:)   :: idata2d
  integer,pointer,dimension(:,:,:) :: idata3d
  real*4, pointer,dimension(:)     :: rdata1d
  real*4, pointer,dimension(:,:)   :: rdata2d
  real*4, pointer,dimension(:,:,:) :: rdata3d
  real*8, pointer,dimension(:)     :: ddata1d
  real*8, pointer,dimension(:,:)   :: ddata2d
  real*8, pointer,dimension(:,:,:) :: ddata3d
end type

type model_data_type
  integer :: num_of_data
  type(time_type) :: data_time
  type(model_data_type),pointer :: next_ptr
  type(model_data_type),pointer :: before_ptr
  type(data_type), pointer :: data
end type

type all_step_data_type
  integer :: num_of_step
  integer :: data_domain
  type(model_data_type), pointer :: md
end type

type all_domain_data_type
  integer :: num_of_domain
  character(len=NAME_LEN) :: model_name
  type(all_step_data_type), pointer :: ad(:)
end type

type(all_domain_data_type),allocatable,private :: ad(:) ! all model data
```

図8　送信データを保持する変数の定義

data_typeは、一つのデータを保持するための構造体である。メンバ変数gridはそのデータが位置するグリッド情報へのポインタ、data_nameはデータの名称、recv_flagは受信情報を保持す

る構造型変数、idata1d から ddata3d までは実際にデータを保持する変数で、データの型によって適切な変数が allocate される。この構造体は、data_type 型のポインタ next_ptr と before_ptr によってチェーン状に結合され、自由に伸び縮みすることで 1 ステップ分のデータを保持することができる。model_data_type は、1 ステップ分のデータを保持する構造体である。メンバ変数 num_of_data はデータの数、data_time はデータの時刻、data は data_type 型のポインタで、このポインタが指示する先を変えることで必要なデータにアクセスすることができる。この構造体は、model_data_type 型のポインタ next_ptr と before_ptr によってチェーン状に結合され、この構造型変数が伸び縮みすることによって必要なステップのデータを保持することができる。all_step_data_type は、全てのタイムステップのデータを保持する構造体である。メンバ変数 num_of_step は保持しているタイムステップの数、data_domain は保持しているデータの領域番号、md は model_data_type 型のポインタで、このポインタが指示する先を変えることでタイムステップを変更することができる。all_domain_data_type は、全ての領域のデータを保持する構造体であり、この構造体によって一つのモデルのデータが保持される。構造体のメンバ変数 num_of_domain は領域の数、model_name はモデルの名称である。最終的にデータを保持する変数は、all_domain_data_type 型の 1 次元配列 ad である。ad の配列の大きさはモデルの数に対応する。

メモリ上に保持された送信データは、不要になった時点で破棄される。不要かどうかは、そのデータを必要とする全てのモデルに受信されたかどうかで判断される。この情報を保持するのは、data_type 構造型のメンバ変数 recv_flag であり、recv_flag_type 型の変数である。カップラーは起動時にデータ設定ファイルを読み込み、各モデルの送信データがどのモデルの受信データとして必要とされているかを検索し、recv_flag に記録する。データ交換が行われると、recv_flag の該当する受信データのフラッグを立て、全てのフラッグが立つ、すなわち受信すべきデータが全て受信し終わったらそのデータを破棄する。データの破棄は、ある時間ステップで全てのフラッグが立った場合、それ以前の全てのデータについて行われる。

3．3　ファイル入出力

本カップラーは、特定のモデルのみを先行して実行し他のモデルを後から実行できるようになっている。そのため、送信または受信対象のモデルが実行されていない場合、データをファイルに出力またはファイルから入力する機能を持つ。この機能について、グリッドの初期化時とデータ送受信時に分けて記述する。

（1）グリッド初期化時のファイル入出力

モデルがカップラーに送信したグリッド情報は、カップラー内の変数に保持されると同時にファイルに出力される。ファイル名は、“[モデル名].grid.data”である。実行されているモデルすべてについてグリッド情報は出力される。また、実行されていないモデルについては、グリッド情報ファイルを検索し、ファイルが存在する場合にはデータを読み込みカップラー内の変数に保持する。モデルＡとモデルＢをそれぞれ独立に実行した場合のグリッド情報の流れを図９に示す。

図9左側では、モデルAのみ実行されており、カップラーはモデルAのグリッド情報をファイルに出力している。図9右側は、モデルAの実行後にモデルBのみ実行したときのグリッド情報の流れである。カップラーは、モデルAのグリッド情報をファイルから読み込み、モデルBからグリッド情報を受信した後ファイルに出力している。

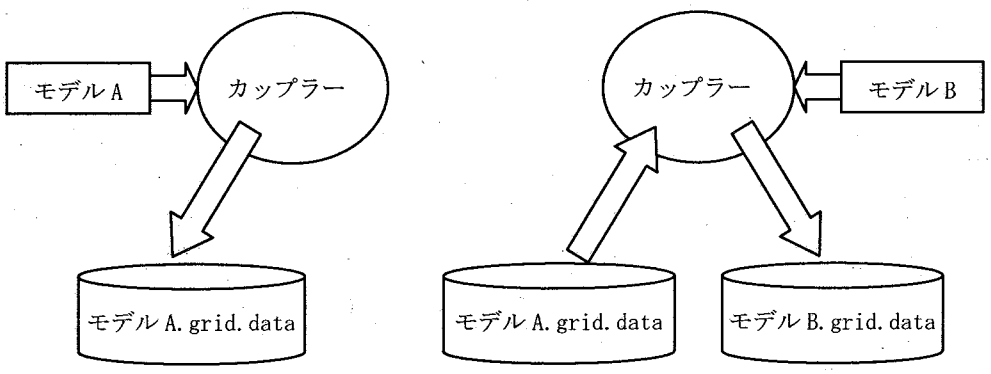


図9　グリッド情報の流れ

（2）データ交換時のファイル入出力

送信・受信両方のモデルが実行されている場合にはファイル入出力は行われず、データはカップラーを経由して直接送受信される。受信側モデルが実行されていない場合は、送信側モデルからの送信データはすべてファイルに出力される。ファイル名は“[送信モデル名][送信ドメイン番号].[年年年年月月日日時時分分秒秒].data”である。受信側モデルが実行され送信側モデルが実行されていない場合は、受信側モデル計算の時刻と領域に対応したファイルが読み込まれ、補間計算を経た後に受信側モデルに送信される。ファイルが出力される場所は、モデル設定ファイル中で各モデルの実行環境として記述されたディレクトリである。

3．4　データ補間

本節では、空間補間と時間補間について説明する。

（1）空間補間

モデル間でデータ交換を行う場合、モデルの解像度やグリッドポイントの位置は一致していないのが普通である。グリッドポイントの位置やグリッド間隔の異なるモデル間でデータを交換するには、図10に示すようにカップラーで補間計算が必要である。カップラーの補間用サブルーチンは、2次元データ、3次元データ用があり、これらのサブルーチンをデータにあわせてカップラーがコールすることにより補間計算が行われる。

2次元の補間用サブルーチンを図11に示す。引数nxs、nysは送信側データの配列の大きさ、xs、ysは送信側グリッドの位置、dtsは送信データである。また、nxr、nyrは受信側データの配列の大きさ、xr、yrは受信側グリッドの位置、dtrは受信データである。isr、jsrは、受信側のグリッドポイントが送信側のどのグリッドポイントに対応するかを表す変数である。例えば、受

信側のグリッド(3,3)が送信側グリッド(4,3)、(5,3)、(4,4)、(5,4)に囲まれていた場合、4点のうち左下グリッドを対応させisr(3,3) = 4、jsr(3,3) = 3となる。グリッドポイントの対応情報の取得については、汎用的なルーチンが組み込んであり、利用者が作成する必要はない。このアルゴリズムについては、付録Aで詳細に記述する。idxは、送受信するデータを識別するためのインデックスであり、送受信用サブルーチンの引数で与えられる値である。

データ補間サブルーチン jc_InterpolateData2D の中で受信側の1グリッドポイント毎に補間計算を行うようなプログラムが実装されている。受信データの1グリッドポイント毎に対応する送信データの値と位置の情報が配列に代入され、グリッドポイント毎の補間サブルーチンに渡される。グリッドポイント毎の補間サブルーチン jc_Interpolation2D のインターフェースは図12のようになる。引数snx、snyは送信側データ配列の大きさ、sx、syは送信側グリッドポイントの位置、sdtは送信側データ値である。rnx、rnyは受信側データ配列の大きさ、rx、ryは受信側グリッドポイントの位置、rdtは受信データ値である。idxはデータインデックスである。

3次元の補間用サブルーチンについては、3次元配列に変わるだけで引数の定義は2次元の場合と同様である。また、3次元のデータについてもグリッドポイント毎の計算用サブルーチンjc_Interpolation3Dが用意されている。

補間アルゴリズムはカップリングされるモデルの性質や用途によって様々であるため、補間サブルーチン jc_Interpolation2D 及び jc_Interpolation3D には、補間プログラムは実装されておらず、データ補間のためのインターフェースのみが提供されている。従って、利用者は用途に応じた補間アルゴリズムをサブルーチン内に実装しなくてはならない。あるいは、これらのサブルーチンを使用せずjc_InterpolateData2D、jc_InterpolateData3Dに新たなコードを実装してもかまわない。すなわち、データ補間用モジュールの内容については、利用者が自由に変更可能である。ただし、jc_InterpolateData2D、jc_InterpolateData3Dの引数の数と種類については呼び出し側との整合を保つ意味で変更してはならない。なお、後述の水循環結合モデルシステム用として、異なる地図投影法を持つ間のモデル間で水平2次元のデータ交換を行うことを目的とした比較的汎用性の高い補間プログラムが実装されている。この空間補間については、付録Bで詳細に記述する。

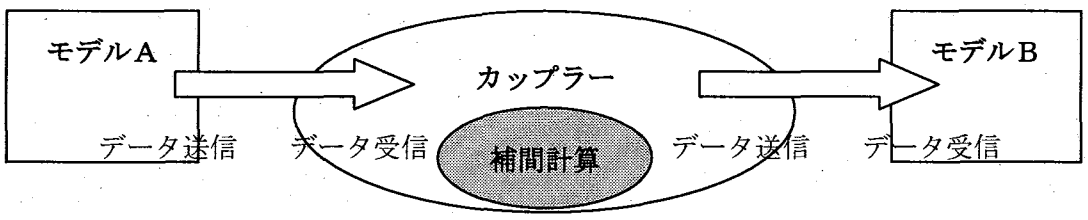


図10　データ処理の流れ

```
subroutine  jc_InterpolateData2D(nxs, nys, xs, ys, dts, nxr, nyr, xr, yr, dtr, &
                                 isr, jsr, idx, s_grd_num r_grd_num)
                                 integer,intent(IN) :: nxs, nys
                                 real,intent(IN)    :: xs(:,:), ys(:,:)
                                 real,intent(IN)    :: dts(:,:)
                                 integer,intent(IN) :: nxr, nyr
                                 real,intent(IN)    :: xr(:,:), yr(:,:)
                                 real,intent(IN)    :: dtr(:,:)
                                 integer,intent(IN) :: isr(:,:), jsr(:,:)
                                 integer,intent(IN) :: idx
```

図 11　2 次元データの補間サブルーチン

```
subroutine jc_Interpolation2D(snx, sny, sx, sy, sdt, rnx, rny, rx, ry, rdt,idx)
                              integer,intent(IN) :: snx, sny
                              real,intent(IN)    :: sx(snx,sny), sy(snx,sny)
                              real,intent(IN)    :: sdt(snx, sny)
                              integer,intent(IN) :: rnx, rny
                              real,intent(IN)    :: rx(rnx, rny), ry(rnx, rny)
                              real,intent(INOUT) :: rdt(rnx, rny)
                              integer,intent(IN) :: idx
```

図 12　グリッドポイント毎の補間サブルーチン

（2）時間補間

時間補間は、2 つの時刻の間を直線で補間するアルゴリズムを採用しており、利用者が特にプログラムを修正する必要はない。また、補外はできないようになっている。カップラーは送信・受信データの時刻から時間補間の計算が必要かどうかを判定する。時間補間が必要でない場合は、空間補間のみが実行される。時間補間が必要な場合は、補間に使われる送信データを検索し、データが存在した場合に補間計算を行う。このデータ検索の際、データ設定ファイルに指定された送信時間ステップ幅で検索を行う。モデルが実際に送信を行う時間ステップと設定ファイルで設定された時間ステップが整合しない場合（例えば、設定ファイルで毎ステップ送信設定になっているのにモデルでは 2 ステップおきに送信するようにした場合など）、カップラーは、実際にはデータが送られないステップのデータが送信されるまで送信待ち状態のままになってしまう。このエラーを回避するには、モデル内では毎ステップ送信・受信するようにして、設定ファイルで送信・受信ステップ幅を設定することが望ましい。既存モデルのプログラム改修が困難で、送受信ステップがモデルに依存せざるを得ない場合、設定ファイルとの間に矛盾が生じないように注意する必要がある。

4. 使用方法

本節では、モデルをカップラーに組み込み、結合計算を実行する手順を解説する。以下カップラーをインストールしたディレクトリを${C_HOME}と表記する。

4．1 プログラムの構成

本カップラーはfortran 90で記述されている。プログラムはメイン部分を除いて全て1ファイル1モジュールになっており、全体では18のモジュールから構成されている。各モジュールの名称、機能、モジュール相互の関係は表1のようになっている。また、利用者が用いるインターフェースモジュールjc_interfaceにpublicとして定義されている定数、関数、サブルーチンについて表2から表4にまとめる。定数の実体はjc_constantで定義されており、jc_interfaceはそれらをuseしている。jc_interfaceで利用者に公開されている定数を表2に示す。データを補間する際に対応するグリッドがない場合、定数 NO_GRID を代入するようになっている。この値はjc_constantで-999としているが、計算によっては-999が正常な値の範囲であり特別な意味を持たせることが不適切な場合も考えられる。この時は、jc_constant の該当する数値を適切な値に変更してプログラムを再コンパイルする必要がある。

表1　モジュールの名称・機能・相互関係

番号	名称	機能	使用するモジュール
1	jc_mpl	並列ライブラリ MPI 情報の管理・データ交換	2、5、6、7、8、16、17、18
2	jc_utils	ユーティリティーモジュール ログ出力・エラー処理	ほぼ全てのモジュール
3	jc_constant	カップラー定数の定義	ほぼ全てのモジュール
4	jc_file	ファイルの管理 入出力の管理	7、14、16
5	jc_data_ctl	データ設定ファイルの読込み データ送受信情報の管理	7、8、15、16、18
6	jc_recv_flag	受信完了フラッグ情報の管理	7、5、16
7	jc_data_grid	グリッドとデータ情報の変数の定義と保持	14、16、17
8	jc_process_ctl	プロセスコントロール 各プロセスの情報を管理	7、14、15、16、17、18
9	jc_interpolation_base	補間用のグリッド対応計算	10、11
10	jc_interpolation_interface	グリッド間の補間計算実行 利用者に公開	12
11	jc_time_interpolation	時間補間実行	16
12	jc_interpolation	データ補間の管理	14、16、17
13	jc_time_ctl	時間管理 モデルの時間を管理	5、7、8、14、16、18
14	jc_data_graph	作図用データ出力	15、16
15	jc_graph_lib	作図用データ読み出しのためのライブラリモジュール	利用者が使用
16	jc_data_exchange	各モデルからのデータ送受信 要求受付とデータの送受信	17
17	jc_main	メインモジュール	メインプログラム main.f90
18	jc_interface	モデル（利用者）が用いるルーチン群の定義	利用者が使用

表2　公開されている定数

定数名	値	意味
NO_GRID	-999	グリッドなし
NAME_LEN	30	モデルの名称の長さ
TOP_BOUNDARY	1	上面境界
BOTTOM_BOUNDARY	2	下面境界

表3　公開されているサブルーチン

名称	機能
jc_InitMyModel	カップラーの初期化
jc_InitGrid	グリッドの初期化（必ずコールする）
jc_SetGrid	グリッド情報の設定（すべてのグリッドについてコールする）
jc_SetTime	積分開始時刻、時間ステップ設定
jc_SetTimeDomain	現在時刻、領域設定（送受信前に必ずコールする）
jc_SendData1D	1次元データの送信
jc_RecvData1D	1次元データの受信
jc_SendData2D	2次元データの送信
jc_RecvData2D	2次元データの受信
jc_SendData3D	3次元データの送信
jc_RecvData3D	3次元データの受信
jc_SendParticle	パーティクル情報の送信
jc_RecvParticle	パーティクル情報の受信
jc_CouplingEnd	計算の終了をカップラーへ送信

表4　公開されている関数

名称	機能
jc_GetMyTotalProcNum	自モデルが用いているプロセッサ数の取得
jc_GetMyProcessID	自モデルのプロセスID番号の取得

４．２　設定ファイル

カップリングを行う際に作成すべき設定ファイルについて記述する。設定ファイルには、モデル設定ファイルとデータ設定ファイルがある。モデル設定ファイルは、1 つのファイルで、ファイル名は“c.conf”である。データ設定ファイルは、モデル設定ファイルで有効とされたモデルの数分必要であり、そのファイル名は“[モデル名].data.conf”である。これらのファイルは、カップラーが動作するディレクトリに存在しなければならない。設定ファイルは、Fortran 90 の Namelist 形式になっている。以下、各設定ファイルについて説明する。

（1）モデル設定ファイル

モデル設定ファイルの内容を図 13 に示す。設定ファイルのセッションは、カップラー本体の動作（作図用データ出力に関する動作）を記述する coupler セッションと、モデルの動作を記述する model1 から model8 までの計 9 セッションある。カップラーはこれらのセッションが存在するものとして設定ファイルを読むため、利用者はセッションおよびセッション内の要素を削除してはならない。実際にカップリングするモデルが 8 未満の場合、使用しないセッションにはダミーデータを記述し、model_flag = 0 としておく。以下、セッション中の各パラメータについて説明する。

coupler セッション

・graph_flag

カップラーが作図のためのデータ出力を行うかどうかのフラッグである。1 は作図を行い、0 は作図を行わないと言う意味である。後述するデータ設定ファイルにおいてデータ毎に作図を行うかどうかを設定するが、coupler セッションの graph_flag が 0 の場合、データ毎の設定は全て無視され作図用出力は行われない。

・graph_model

作図のためのデータ出力は、格子系を任意のモデルの任意のグリッドに統一している。この統一対象モデルを設定するのが graph_model である。graph_model に設定されるモデル名はモデルセッションで有効となっているモデル名のいずれかでなければならない。

・graph_domain

作図格子系のうち、領域を指定するのが graph_domain である。graph_domain は 1 以上の整数であり、対象モデルが用いる領域の数以下でなければならない。

・graph_grid

作図格子系のうち、グリッドを指定するのが graph_grid である。graph_grid は 1 以上の整数であり、対象モデルが用いるグリッド系の数以下でなければならない。

・boundary_2d

2 次元データを作図出力する場合に、対象グリッドの鉛直座標位置を指定するのが boundary_2d である。boundary_2d は 1 以上の整数であり、対象グリッドの鉛直グリッド数以下でなければならない。

model セッション

・model_flag
そのモデルをカップリング対象とするかどうかのフラッグである。1 はカップリングを行い、0 はカップリングを行わないと言う意味である。model_flag = 0 の場合、そのモデルは存在しないと見なされ、セッション中のその他の設定は全て無効となる。

・model_name
モデルの名称を記述する。名称の長さは 30 文字以内でなければならない。

・model_pe
各モデルに割り当てる PE の数である。

・run_flag
カップリング対象モデル（model_flag=1 とされたモデル）を実際に実行するかどうかのフラッグである。run_flag = 1 の場合、モデルは実際に起動され、データの送信受信は起動されたモデルに対して行われる。run_flag = 0 の場合、モデルは起動されず、当該モデルに対するデータ送信および当該モデルからのデータ受信はファイルに対して行われる。なおカップラーをマルチプロセスで実行する場合、model_flag=1 のセッションの run_flag は 1 でなければならない。

・exe_file
実行するプログラム名を記述する。起動プログラム名はモデルの実行モジュールそのものでもモデル実行も含めたシェルスクリプト名でもよい。

・exe_dir
実行するプログラム（スクリプト）のあるディレクトリを記述する。

・data_dir
データの記録場所を記述する。

・host_name
プログラム（モデル）を実行するマシン名を記述する。カップラーと同じマシンで起動する場合には LOCAL_HOST とする。

・user_name
プログラム（モデル）を起動するマシンのユーザーID を記述する。ローカルホストで起動する場合は LOCAL_USER とする。

・nqsq_name
プログラム（モデル）実行に用いるジョブのキュー名を記述する。ジョブに投入しない場合は TSS とする。

host_name 以下のセッションは、stampi を用いて異機種間カップリングを行う場合に有効となるパラメータである。

```
&coupler
  graph_flag   = 0
  graph_model  = "mm5"
  graph_domain = 1
  graph_grid   = 1
  boundary_2d  = 2
&end

&model1
  model_flag  = 1
  model_name = "mm5"
  model_pe    = 4
  run_flag     = 1
  send_write = 1
  exe_file = "run_mm5_couple.sh"
  exe_dir  = "/…/COUPLER/MM5/Run"
  data_dir = "/…/COUPLER/MM5/Run/OUTPUT"
  host_name = "LOCAL_HOST"
  user_name = "LOCAL_USER"
  nqsq_name = "TSS"
&end

&model2
  model_flag = 1
  model_name = "pom"
  model_pe = 1
  send_write = 0
  run_flag = 0
  exe_file = "go_pom.sh"
  exe_dir  = "/…/COUPLER/POM00/work"
  data_dir = "/…/COUPLER/POM00//area1/output"
  host_name = "LOCAL_HOST"
  user_name = "LOCAL_USER"
  nqsq_name = "TSS"
&end
```

図 13　モデル設定ファイルの例

（2）データ設定ファイル

図 14 にデータ設定ファイルの例を示す。データ設定ファイルはカップリング対象となるモデル毎に必要で、ファイル名は“[モデル名].data.conf”となる。データ設定ファイルには senddata と recvdata という 2 種類のセッションがある。どちらも送信／受信するデータの種類の数だけ記述する。なお、このファイルの読み込みは Namelist を用いていないため、セッション名は重複していても構わない。そのため、データ設定ファイルには senddata と recvdata の 2 種類のセッションしか存在しない。また、それぞれの数には特に制限はない。以下、セッション中の各パラメータについて説明する。

senddata セッション

・data_name
送信するデータ名を記述する。データ名は 30 文字以内でなければならない。

・send_flag
データを送信するかどうかのフラッグである。1 は送信し、0 は送信しないという意味で、0 の場合、プログラム中にデータ送信サブルーチンが記述されていても送信を行わない。

・graph_flag
データを作図出力するかどうかのフラッグである。1 は出力し、0 は出力しないという意味である。なお、“c.conf”ファイルの coupler セッションで graph_flag=0 である場合、senddata で graph_flag=1 となっていても出力は行われない。

・grid_index
このデータがモデル中のどのグリッドにあるかを示す。grid_index は 1 以上の整数で、モデルのグリッド数以下でなければならない。

・time_step
データを何ステップおきに送信するかを設定する。time_step はカンマで区切られた 5 つの整数があるが、これらはモデルの領域（ネストドメイン）に対応する。time_step には-1 または 0 または 1 以上の整数を記述する。-1 はプログラムの最初のステップのみデータを送信するという意味である。0 は送信を行わないという意味である。1 以上の整数は送信ステップ間隔を意味する。例えば、モデルの第 2 ドメインのデータのみを 10 ステップおきに送信する場合、time_step = 0, 10, 0, 0, 0 となる。

recvdata セッション

・data_name
受信するデータ名を記述する。データ名は 30 文字以内でなければならない。

・recv_flag
データを受信するかどうかのフラッグである。1 は受信し、0 は受信しないという意味で、0 の場合、プログラム中にデータ受信サブルーチンが記述されていても、受信を行わない。

・grid_index
このデータがモデル中のどのグリッドにあるかを示す。grid_index は 1 以上の整数で、モデルのグリッド数以下でなければならない。

・time_step
データを何ステップおきに受信するかを設定する。time_step はカンマで区切られた 5 つの整数があるが、これらはモデルの領域（ネストドメイン）に対応する。time_step には-1 または 0 または 1 以上の整数を記述する。-1 はプログラムの最初のステップのみデータを受信するという意味である。0 は受信を行わないという意味である。1 以上の整数は受信ステップ間隔を意味する。例えば、モデルの第 2 ドメインのデータのみを 10 ステップおきに受信する場合、time_step = 0, 10, 0, 0, 0 となる。なお、このパラメータは、各モデルの領域毎に計算がデ

ータ受信を必要としないモードへ切り替えることに利用することができる。これにより、通常双方向のデータ交換があるモデルは同時に実行される必要があるが、受信を行わないことによりそのモデルを切り離した計算が可能となる。この機能は、受信対象モデル毎、ドメイン毎に独立して設定可能である。

・send_model
受信データが受信するデータの送信元モデル名を記述する。モデル名は“c.conf”に記述された有効なモデル名でなければならない。

・send_data
受信データが受信するデータの送信元データ名を記述する。データ名は“[送信元モデル名].data.conf”ファイルに記述された有効な送信データ名に一致しなければならない。

・send_domain
各領域の受信データがどの領域の送信データを受信するかを設定する。例えば、受信モデル第1領域のデータが送信モデル第2領域のデータを、受信モデル第2領域のデータが送信モデル第3領域のデータを受信する場合、send_domain = 2, 3, 0, 0, 0となる。

```
&senddata
  data_name   = "pom u"
  send_flag   = 1
  graph_flag  = 0
  grid_index  = 1
  time_step   = 8,0,0,0,0
&end

&recvdata
  data_name   = "mm5_u_2d"
  recv_flag   = 1
  grid_index  = 1
  time_step   = 1,0,0,0,0
  send_model  = "mm5"
  send_data   = "mm5_u_2d"
  send_domain = 1,0,0,0,0
&end
```

図14　データ設定ファイルの例

4．3　サブルーチンの組込み

モデルがカップリングに用いるサブルーチンについて、呼び出す順序に従って説明する。

（1）モデルの初期設定

モデルがカップラーと通信するためには、モデルの初期化を行う必要がある。初期化用のサブルーチンはjc_InitMyModelである。サブルーチンjc_InitMyModelは図15のようになっている。引数my_nameは自モデルの名称であり、モデル設定ファイル“c.conf”に記したものと同じでなければならない。isCallInitは、MPI初期化サブルーチンMPI_Initをカップラー側がコールする

かどうかのフラッグである。モデルがすでにMPIで並列化されており、モデル側でMPI_Initをコールしている場合、この引数に“.FALSE.”を与える。引数isCallInitは省略可能であり、省略時には“.TRUE.”が与えられたものとして動作する。

```
subroutine jc_InitMyModel(my_name, isCallInit)
                    character(len=*),intent(IN) :: my_name
                    logical,optional,intent(IN) :: isCallInit
```

図15　サブルーチンjc_InitMyModel

（2）領域数／グリッド系の数設定

サブルーチンjc_InitGridは、モデルが用いる領域の数及び各領域のグリッド系の数情報をカップラーへ与える。jc_InitGrid図16のようになる。引数number_of_domainは、そのモデルが用いる領域の数（ネストしない場合は1）である。引数number_of_gridは、モデルの各領域が用いるグリッド系の数である。なお、カップラープログラム内の定数を変更しない限り、1≦number_of_domain≦5、1≦number_of_grid≦8でなければならない。

```
subroutine jc_InitGrid(number_of_domain, number_of_grid)
  integer,intent(IN) :: number_of_domain, number_of_grid
```

図16　サブルーチンjc_InitGrid

（3）グリッドポイント位置の設定

jc_InitGridで領域とグリッド系の数を設定した後、サブルーチンjc_SetGridで各グリッドのグリッドポイントの位置を設定する。サブルーチンjc_SetGridは図17のようになる。jc_SetGridの引数domainは、設定する領域の番号、引数grid_idは設定するグリッドの番号であり、いずれもjc_InitGridで与えたnumber_of_domain、number_of_grid以下でなければならない。nx、ny、nzは配列の大きさ（x、y、z方向のグリッドポイントの数）、x、y、zはグリッドポイントの位置（x座標、y座標、z座標）である。bottom_grid、top_gridは2次元データを交換するときに用いる座標を示す。例えば大気モデルで下層から3グリッド目の2次元データを交換するとしたらbottom_gridは3になる。なお、多くのモデルでは下層もしくは上層の値しか交換しない。この場合、交換しない側の座標は適当な値を入れておく。サブルーチンjc_SetGridはjc_InitGridで与えた全ての領域、全てのグリッド系について設定しなければならない。すなわち、number_of_domain×number_of_grid回コールする必要がある。また全ての領域・グリッドについて設定が終わるまでに他のカップリング関連のサブルーチンをコールしてはならない。

```
subroutine jc_SetGrid(domain, grid_id, nx, ny, nz, x, y, z, bottom_grid, top_grid)
  integer,intent(IN) :: domain, grid_id
  integer,intent(IN) :: nx, ny, nz
  real,intent(IN)    :: x(:,:,:), y(:,:,:), z(:,:,:)
  integer,intent(IN) :: bottom_grid, top_grid
```

図 17　サブルーチン jc_SetGrid

（4）積分開始時刻／時間ステップの設定

jc_SetGrid でグリッド位置を設定した後、サブルーチン jc_SetTime で積分開始時刻とタイムステップを設定する。サブルーチン jc_SetTime を図 18 に示す。引数 domain は設定する領域番号であり、1≦domain≦number_of_domain でなければならない。引数 s_time は積分開始時刻である。この引数は、14 文字の文字列で積分の開始時刻を年月日時分秒まで与える。すなわち“YYYYMMDDHHMMSS”である。引数 delta_t は積分の時間ステップを秒単位で与える。このサブルーチンは jc_InitGrid で与えた全ての領域について設定しなければならない。また、jc_SetTime は、jc_SetGrid で全ての領域・グリッド系について座標位置の設定が終わった後にコールしなければならない。

グリッドの位置情報設定および時刻設定は、MPI の初期化後データ交換が始まる（積分に入る）前であればプログラムの任意の位置で行うことができる。ただし、カップラーは設定ファイルに記述された順序でモデルからデータが送られてくるのを待っているため、後の方のモデルは前のモデルが情報を送り終わるまで待たなければならない。従って、グリッド初期化情報はモデル間で、できるだけ同じタイミングで送るようにするべきである。

```
subroutine jc_SetTime(domain, s_time, delta_t)
  integer,intent(IN) :: domain
  character(len=*),intent(IN) :: s_time
  integer,intent(IN) :: delta_t
```

図 18 サブルーチン jc_SetTime

以上が、積分開始前の初期設定の手順である。以下、データ交換の手順について説明する。

（5）現在時刻／領域設定

データの送受信を行う前に、モデル計算上の時刻と領域をカップラーに知らせる必要がある。そのためのサブルーチンが jc_SetTimeDomain である。jc_SetTimeDomain を図 19 に示す。引数 time は、サブルーチン jc_SetTime の引数と同様 14 文字の文字列で、“年月日時分秒”を表す。引数 domain は現在の領域番号である。このサブルーチンがコールされた後に送受信されるデータは全てこの時刻／領域のものであると解釈される。このサブルーチンは、jc_SetTime で積分開始時刻／時間ステップが設定された後ならどこで呼び出しても構わない。また、モデルの時刻は過去に

さかのぼることもできる。ただし、jc_SetTime で設定された開始時刻以前の時刻は設定できない。

```
subroutine jc_SetTimeDomain(time,domain)
  character(len=*),intent(IN) :: time
  integer,intent(IN) :: domain
```

図 19　サブルーチン jc_SetTimeDomain

（6）データ送信

jc_SetTimeDomain でモデル時刻と領域を設定した後にコールする。データ送信サブルーチンは 1 次元、2 次元、3 次元、パーティクルの 4 種類のデータそれぞれに対応するサブルーチンがある。以下、各々のサブルーチンに対して説明する。

1 次元データ送信

1 次元データを送信するサブルーチンを図 20 に示す。引数 dt は送信する 1 次元データであり、整数型、単精度実数型、倍精度実数型のいずれかである。引数 name はデータの名称である。名称は、“[モデル名] .data.conf” ファイルに記載された名前と一致しなければならない。引数 dt_size は省略可能な引数で、送信するデータの個数を表す。この引数が省略された場合、全てのデータが送信される。

```
subroutine jc_SendData1D(dt,name,dt_size)
                integer,intent(IN) :: dt(:)
                または
                real*4,intent(IN) :: dt(:)
                または
                real*8,intent(IN) :: dt(:)
                character(len=*),intent(IN) :: name
                integer,optional,intent(IN) :: dt_size
```

図 20 1 次元データ送信サブルーチン

2 次元データ送信

2 次元データを送信するサブルーチンを図 21 に示す。引数 dt は送信するデータであり、単精度実数か倍精度実数の 2 次元変数である。引数 name は送信データの名称である。tb_flag は送信するデータが上面境界か下面境界かのフラッグで、値は jc_constant で定義され、jc_interface で公開されている TOP_BOUNDARY もしくは BOTTOM_BOUNDARY である。

```
subroutine jc_SendData2D(dt, name, tb_flag)
                         real*(4 or 8),intent(IN) :: dt(:,:)
                         character(len=*),intent(IN) :: name
                         integer,intent(IN) :: tb_flag
```

図 21　2 次元データ送信サブルーチン

3 次元データ送信

3 次元データの送信サブルーチンを図 22 に示す。引数の定義は 2 次元の場合と同様である。

```
subroutine jc_SendData3D(dt, name)
                          real*(4 or 8),intent(IN) :: dt(:,:,:)
                          character(len=*),itnent(IN) :: name
```

図 22　3 次元データ送信サブルーチン

パーティクルデータ送信

パーティクルデータを送信するサブルーチンを図 23 に示す。引数 n はパーティクルの個数、s はパーティクル個々のスカラー値、x、y、z はパーティクルの位置である。name はデータの名称である。

```
subroutine jc_SendParticle(n,s,x,y,z,name)
                          integer,      intent(IN)  :: n ! # of particles
                          real*(4 or 8), intent(IN)  :: s(:),x(:),y(:),z(:)
                          character(len=*),intent(IN) :: name
```

図 23　パーティクル情報送信サブルーチン

（7）データ受信

データ受信サブルーチンも送信サブルーチンと同様、1 次元、2 次元、3 次元、パーティクルの 4 種類のサブルーチンがある。以下各々について説明する。

1 次元データ受信

1 次元データを受信するサブルーチンを図 24 に示す。引数 dt は受信する 1 次元データであり、整数型、単精度実数型、倍精度実数型のいずれかである。引数 name はデータの名称である。名称は、“[モデル名] .data.conf” ファイルに記載された名前と一致しなければならない。引数 dt_size は省略可能な引数で、受信するデータの個数を表す。この引数が省略された場合、配列の大きさのデータが受信される。

```
subroutine jc_RecvData1D(dt,name,dt_size)
                integer,intent(INOUT) :: dt(:)
                または
                real*4,intent(INOUT) :: dt(:)
                または
                real*8,intent(INOUT) :: dt(:)
                character(len=*),intent(IN) :: name
                integer,optional,intent(IN) :: dt_size
```

図 24　1 次元データ受信サブルーチン

2 次元データ受信

2 次元データを受信するサブルーチンを図 25 に示す。dt は、受信するデータで単精度実数または倍精度実数である。引数 name は受信するデータの名称である。tb_flag は受信するデータが上面境界か下面境界かのフラッグで、値は jc_constant で定義され jc_interface で公開されている TOP_BOUNDARY、もしくは BOTTOM_BOUNDARY である。intrpl_tag は補間計算を行う際のデータ識別のためのタグである。intrpl_tag は 1 以上の整数でなければならない。

```
subroutine jc_RecvData2D(dt, name,tb_flag, intrpl_tag)
                        real*(4 or 8),intent(INOUT) :: dt(:,:)
                        characterl(len=*),intent(IN) :: name
                        integer,intent(IN) :: tb_flag
                        integer,intent(IN) :: intrpl_tag
```

図 25　2 次元データ受信サブルーチン

3 次元データ受信

3 次元データの受信サブルーチンは図 26 のようになる。引数の定義は 2 次元の場合と同様である。

```
subroutine jc_RecvData3D(dt, name, intrpl_tag)
                        real*(4 or 8),intent(INOUT) :: dt(:,:,:)
                        character(len=*),intent(IN) :: name
                        integer,intent(IN) :: intrpl_tag
```

図 26　3 次元データ受信サブルーチン

パーティクルデータ受信

パーティクル情報を受信するサブルーチンを図 27 に示す。引数 n はパーティクルの個数、s はパーティクル個々のスカラー値、x、y、z はパーティクルの位置である。name はデータの名称である。パーティクルの数 n はデータを受信したときに決まる値であり、本来事前には知り得ない

ものである。一方、パーティクル情報を格納する変数 s、x、y、 z の配列の大きさは n 以上でなければならない。従って、事前に十分大きな配列を確保しておく必要がある。データ送信については特に制約はないが、データ受信については同一データ、すなわち名称、領域番号が同じデータを過去にさかのぼっては受信できない。これは保持データをカップラーが消去するアルゴリズムに起因するもので、プログラムはこのエラーをチェックできないため利用者の注意が必要である。

```
subroutine jc_RecvParticle(n,s,x,y,z,name)
                    integer,      intent(INOUT)  :: n ! # of particles
                    real*(4 or 8), intent(INOUT)  :: s(:),x(:),y(:),z(:)
                    character(len=*),intent(IN) :: name
```

図 27　パーティクルデータ受信サブルーチン

（8）計算終了の通知

計算終了時（あるいはデータ交換が必要なくなった時点）で、カップラーに対し計算が終了した旨を通知する必要がある。用いるサブルーチンは、jc_CouplingEnd である。サブルーチンを図 28 に示す。引数 isCallFinalize は省略可能な論理型の引数であり、jc_CouplingEnd が MPI の終了サブルーチン MPI_Finalize をコールするかどうかを与える。プログラムが並列化されており、他の部分で MPI_Finalize をコールしている場合、isCallFinalize には“.FALSE.”を与える。引数省略時には”.TRUE.”が与えられたものとして、MPI_Finalize をコールする。このサブルーチンを呼ばないとカップラーはデータ受信待ちになりプログラムが終了しないので注意が必要である。

```
subroutine jc_CouplingEnd(isCallFinalize)
                    logical,optional,intent(IN) :: isCallFinalize
```

図 28　計算終了通知サブルーチン

4．4　プログラム例

本節では、実際にカップリングを行うことを想定したプログラムの例を示す。想定しているモデルは名称を”test_model”とする。このモデルは num_of_domain 個のネスト領域を持っており、各領域のグリッドの種類は num_of_grid 個とする。送信するデータ名を”s_data”、受信するデータ名を”r_data”とし、これらは各々3 次元データであるとする。モデルは、一般に積分前の前処理、時間積分、積分後の後処理の 3 つの部分に大別される。これらをまとめた全体のカップリングルーチンの使用例を図 29 に示す。

```
character(len=14)  :: c_time  !カップリングに用いる文字列の宣言
integer            :: nx, ny, nz  ！グリッドポイント数（領域と座標系により異なる変数）
real ,dimension(:,:,:) :: x, y, z ！グリッドポイントの位置（領域と座標系により異なる変数）
real               :: delta_t  ！積分の時間ステップ（領域によって異なる変数）
real, dimension(:,:,:) :: sdt, rdt  ！送受信するデータを保持する変数

！カップリングのための前処理
call jc_InitMyModel(“test_model”)   ！自モデル名称の設定
call jc_InitGrid(num_of_domain, num_of_grid)  ！領域とグリッド数の設定

do n = 1, num_of_domain
  do s = 1, num_of_grid
    call jc_SetGrid(n, s, nx, ny, nz, x, y, z, 2, nz-1)  ！グリッド位置の設定
  end do
end do

do n = 1, num_of_domain
  c_time(1:14) = ???????  ！初期時刻を 14 文字の変数に格納
  delta_t = ??????        ！領域毎の時間ステップを変数に格納
  call jc_SetTime(c_time,delta_t)  ！積分開始時刻とタイムステップの設定
end do
！前処理終了

！積分開始
do t = 1, num_of_step  ！時間ループ
  do n = 1, num_of_domain  ！領域ループ

    c_time(1:14) = ?????  ！現在時刻を 14 文字の変数に格納
    tag          =?????   ！補間のためのタグ設定
    call jc_SetTimeDomain(c_time,n)  ！現在時刻と領域の設定
    call jc_SendData3D(sdt,” s_data”)  ！3 次元データ送信
    call jc_RecvData3D(rdt,” r_data”,tag)  ！3 次元データ受信

  end do
end do
！積分終了
call jc_CouplingEnd()  ！カップリング終了
```

図 29　カップリングプログラム例

前処理では、カップリングのための初期化ルーチンをコールする。初期化ルーチンは、jc_InitMyModel、jc_InitGrid、jc_SetGrid、jc_SetTime の 4 種類である。jc_InitMyModel と jc_InitGrid は 1 回ずつ、jc_SetGrid は num_of_domain×num_of_grid 回、jc_SetTime は num_of_domain 回コールする必要がある。また、これらのサブルーチンは上に挙げた順序でコールする必要がある。特に、jc_SetGrid と jc_SetTime は必要な回数分コールし終わった後に、次のサブルーチンをコールしなければならない。

時間積分では、サブルーチン jc_SetTimeDomain でモデル時刻と領域を設定した後、1 次元、2 次元、3 次元、パーティクルの各々の送受信サブルーチンをコールし、データの送受信を行う。

送受信間隔を毎時間ステップ以上にしたい場合も多い。例えば、積分を60秒ごとに行い、データの送信は2時間おきに行いたい場合である。何ステップ毎にデータ送受信を行うかはデータ毎に設定ファイルで設定できるため、プログラム上は特にIF文などで場合分けする必要はない。また、送受信を行うかどうかも各データで領域毎に設定できるため、この場合もプログラム上特に場合分けする必要はない。カップラーが設定ファイルに記述された情報に従って時間補間他の計算を行うため、これらのコントロールは全て設定ファイルに任せるべきである。

後処理では、カップリング終了ルーチン jc_CouplingEnd をコールする。jc_CouplingEnd をコールしないと、カップラーはいつまでもモデルからの送受信要求を待ち続けるため、必ずコールしなければならない。

4．5　コンパイルと実行

はじめにカップラー本体のコンパイルについて述べる。“${C_HOME}/src”の下には“Makefile”やMakeのための設定ファイル、メインモジュールファイル、JC用設定ファイルが置かれている。ディレクトリ“${C_HOME}/src/c”には、カップラーのソースコードとmakefileが置かれている。“compile.conf”はMake用設定ファイルであり、このファイルでコンパイルのためのオプションやディレクトリを設定する。“${C_HOME}/src”から“make”コマンドを実行すると各ディレクトリに移動し、そこでディレクトリ内のmakefileを用いてmakeを改めて実行する。各ディレクトリで作成されたオブジェクトファイルは“${C_HOME}/src”にコピーされる。最後にオブジェクトをリンク、実行モジュールを生成し、コピーされたオブジェクトファイルを消去する。

次に、モデルのコンパイルについて述べる。モデルをカップリングする際に必要となるファイルは、“${C_HOME}/src/c”配下にある jc_mpl.f90、jc_constant.f90、jc_utils.f90、jc_process_ctl.f90、jc_interface.f90の5つである。これらのファイルをモデルのソースディレクトリにコピーしてコンパイル・リンクするか、カップラー作成時にコンパイルされたオブジェクトモジュールを用いてリンクする。

実行は、環境に合わせたMPIの実行コマンドを用い、カップラーの実行モジュールを起動する。例えば、“mpirun　np 1 jc.exe”のようになる。

カップラーのプロセッサ数を1とした場合、カップラーはシングルプロセッサモードで起動される。シングルプロセッサモードの場合、model_flag = 1かつrun_flag = 0の状態（すなはち、カップリングは行うが実際には実行せず、ファイルを対象にデータ送受信を行う）ことが許される。一方、カップラーのプロセッサ数を2以上にした場合、カップラーはマルチプロセッサモードで起動されたと判断する。この場合、model_flag = 1のモデルは全てrun_flag = 1でなければならず、かつカップラーのプロセッサ数とカップリングを行う（つまりmodel_flag = 1の）モデルの数は同じでなければならない。

4．6　ログ出力

カップラーの動作状態をモニターするために、カップラーは送受信状態を標準出力（通常は画面）に出力している。同時に、より詳細な動作状態をログファイルに出力している。ログファイ

ルはカップラーと各モデルで出力しており、これによってカップリングの詳細な動作を知ることができる。カップラーが出力するログファイル名はシングルタスクの場合“c.log”、マルチタスクの場合“c0.log”、“c1.log”、・・・となる。モデルが出力するログファイル名は“[モデル名].coupling.log”となる。出力場所は各プログラムが動作するディレクトリである。

一般に詳細なログの参照が必要になるのは、何らかの原因でエラーが発生しカップラーやモデルが異常終了したときである。このとき、カップラーやモデルがエラーをトラップできてログファイルのクローズ処理を行った場合にはログファイルが作成されるが、エラートラップできずファイルクローズしないまま異常終了した場合、システムによってはログファイルが作成されないことがある。この場合、動作状態を詳細にモニターするには、“compile.conf”でコンパイルオプションにDEBUGを定義し、カップラープログラムを再コンパイルする。これにより、ログファイルに出力されるのと同じ内容が標準エラー出力にも出力されるようになる。

4．7　可視化出力

カップラーのユーティリティー機能として、カップラーからの出力を数値環境システムSPEEDI-MPの可視化機能で利用可能なデータフォーマットへ変換するプログラムを準備している。システムの統一データフォーマットとしては、異なるサーバープラットフォームでの互換性があるNetCDF (Network Common Data Form) [5]を採用している。カップラーは、このデータ出力を行う際に異なる座標系のモデル出力を一つのモデルの座標系に統一する。これにより、結合した各モデルの計算結果を同一画面に重ね合わせて可視化することが可能となる。

可視化データ変換プログラムは、図30に示すように、カップラーが作図用出力として出力したデータを読み込み、可視化システム用データフォーマットに変換して出力する。個々のモデルがカップラーに送信したデータは、データ定義ファイルの定義に従って座標系を統一した後、カップラーから出力される。可視化データ出力プログラムは、カップラー用データ定義ファイルと、可視化データ出力用定義ファイルを読み込み定義ファイルの内容に従って、可視化データを出力する。可視化データ出力プログラムは、座標情報定義時刻、可視化開始時刻、可視化終了時刻、データ間隔の4つの引数を用いる。また、可視化データ出力定義ファイルのファイル名は、環境変数で定義される。このため、可視化データ出力プログラムは、これらの情報を記述したシェルスクリプトから起動されるようになっている。

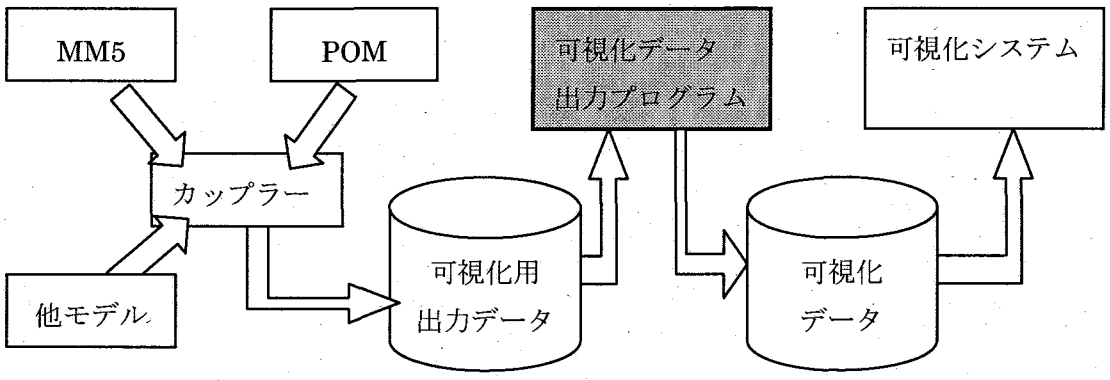


図 30　可視化出力及びフォーマット変換処理の流れ

（1）変換プログラムのコンパイル

可視化データ変換プログラムのコンパイル方法について記述する。

必要なライブラリとファイル

可視化データ変換プログラムは、数値環境システムの NetCDF データ入出力用 Access ライブラリを用いて可視化データを出力するため、Access ライブラリ“libAccess2005.a”が対象マシンにインストールされている必要がある。また、Access ライブラリが netcdf ライブラリを用いるため、netcdf のライブラリ“libnetcdf.a”が対象マシンにインストールされている必要がある。カップラーが出力したデータを扱うため、可視化データ出力プログラムはカップラープログラムのモジュールファイル jc_mpl.f90, jc_utils.f90, jc_constant.f90, jc_recv_flag.f90, jc_time_ctl.f90, jc_file.f90, jc_data_ctl.f90, jc_process_ctl.f90, jc_data_grid.f90, jc_interpolation_base.f90, jc_interpolation_interface.f90, jc_interpolation.f90, jc_data_graph.f90, jc_graph_lib.f90 が対象マシンに存在する必要がある。

Makefile の修正とコンパイル

Makefile の内容を図 31 に示す。Makefile の中で環境に合わせて変更する必要がある項目は以下のようになる。

JC_HOME：カップラーのソースディレクトリを指定
MPIDIR：MPI のルートディレクトリを指定
MPI*DIR：MPI のライブラリやインクルードファイルのディレクトリを指定
NCDIR：netcdf のディレクトリを指定
NCINC：netcdf のインクルードファイルのディレクトリを指定
LDFLAGS：Access ライブラリや netcdf ライブラリの情報を記述
FC：mpi の fortran コンパイラを指定
OPTION：コンパイルオプションを指定

Make ファイルを適切に修正し make コマンドを実行すると、実行ファイル“ncconv.exe”が生成される。

```
SHELL           =       /bin/csh

JC_HOME = ${HOME}/COUPLER/jc/src/c
MY_HOME = ${PWD}

JC_SRC =jc_mpl.f90¥
        jc_utils.f90¥
        jc_constant.f90¥
        jc_recv_flag.f90¥
        jc_time_ctl.f90¥
        jc_file.f90¥
        jc_data_ctl.f90¥
        jc_process_ctl.f90¥
        jc_data_grid.f90¥
        jc_interpolation_base.f90¥
        jc_interpolation_interface.f90¥
        jc_interpolation.f90¥
        jc_data_graph.f90¥
        jc_graph_lib.f90

MPIDIR = /opt/FJSVmpi2
MPILIBDIR = ${MPIDIR}/lib
MPIINCDIR = ${MPIDIR}/include
MPIBINDIR = ${MPIDIR}/bin
MPILINK    = -ljmpif -ljmpi

NCDIR = /.../netcdf-3.6.0
NCINC = ${NCDIR}/include

LDFLAGS = -L../access -lAccess2005 -L${NCDIR}/lib -lnetcdf

EXEFILE         = ncconv.exe

FC = mpifrt
OPTION =   -Cpp -Am -Mc -KV8PLUS -DMPI2 -DSTAMPI -DSKIP_INTERPOLATION -I${MPIINCDIR}
-I${NCINC}

ncconv  :
        cd ${JC_HOME} ; cp ${JC_SRC} ${MY_HOME} ; cd ${MY_HOME}
        ${FC} -o ${EXEFILE} ${JC_SRC} nc_graph_lib.f90 make_nc.f90 ${OPTION} ${LDFLAGS}
        rm ${JC_SRC} *o

clean           :
        rm ${JC_SRC} *o *exe
```

図 31　可視化データ出力プログラムの Makefile

（2）設定ファイル

可視化データ変換プログラムが必要とする設定ファイルは、カップラー用の設定ファイル“c.conf”と“[モデル名].data.conf”および Access ライブラリが用いる設定ファイル（環境変数でファイル名を定義）である。カップラー設定ファイル“c.conf”及びモデル毎のデータ設定ファイル“[モデル名].data.conf”ファイルについては、プログラム実行時にカップリング計算の設定をコピーするため、特に変更する必要はない。可視化データ出力プログラムは、これらの

設定ファイルから、どのモデルが実行され、どの名称のデータが可視化用に出力されているかを知ることができる。

Accessライブラリが用いる設定ファイルのファイル名は、環境変数で指定するようになっているが、モデル毎に“graph_[モデル名].nmlist”がプログラム実行時に作成される。このファイルには、グリッドや描画対象データの属性が記述される。このファイルは、モデル毎に用意されたテンプレートファイル“[モデル名].template”を基に、プログラム実行シェルスクリプト“nc.sh”により生成される。利用者は、計算条件に応じて“nc.sh”のパラメータ（後述）を修正する必要がある。また、各モデルの可視化用データ出力を変更した場合には、テンプレートファイルのデータ項目を変更に合わせて修正する必要がある。

（3）実行方法

可視化データ変換プログラムは、シェルスクリプト“nc.sh”により実行される。実行シェルスクリプト“nc.sh”の内容を図32に示す。実行時に修正を必要とする“nc.sh”内の設定の名称と意味を下に記す。

モデルグリッド及び座標変換パラメータ

OUTMODEL：可視化出力を作成するモデル名
V_RATIO：3次元可視化AVSにおける鉛直格子の拡大率
DX：X方向格子間隔
DY：Y方向格子間隔
X_size：X格子サイズ
Y_size：Y格子サイズ
Z_size：Z格子サイズ
project：可視化出力の地図投影法（1:Mercator, 2:Polarstereo, 3:Lambert）
lonc：地図投影の中心経度（度）
latc：地図投影の中心緯度（度）
tlat1：地図投影で正距離となる緯度1
tlat2：地図投影で正距離となる緯度2（Lambert図法のみ）

Z格子サイズ以外は、可視化出力の基準として設定したモデルのパラメータ

時刻設定

TSCAL：MM5の積分開始時刻（YYYYMODDHHMMSS）
TSOUT：可視化データ作成開始時刻（YYYYMODDHHMMSS）
TEOUT：可視化データ作成終了時刻（YYYYMODDHHMMSS）
TOINT：可視化データ作成時間間隔（秒単位）

```
#!/bin/csh

### Set model parameters
setenv OUTMODEL pom
setenv V_RATIO 2.0                    # Vertical scale ratio to mm5
### Set grid parameters
setenv DX 15000.0                     # X grid interval (m)
setenv DY 15000.0                     # Y grid interval (m)
setenv X_size    100                  # X grid size
setenv Y_size    130                  # Y grid size
if( $OUTMODEL == "mm5" ) then
        setenv Z_size    23          # mm5 Z grid size
else if( $OUTMODEL == "pom" ) then
        setenv Z_size    11          # pom Z grid size
else if( $OUTMODEL == "solveg" ) then
        setenv Z_size     6          # solveg Z grid size
endif
setenv project   3                    # 1:Mercator, 2:Polarstereo, 3:Lambert
setenv lonc      40.0                 # Longitude of projection center
setenv latc      22.0                 # Latitude   of projection center
setenv tlat1     60.0                 # True latitude 1
setenv tlat2     30.0                 # True latitude 2
### Set time parameters
set TSCAL = 20050120000000          # Calculation start time
set TSOUT = 20050120000000          # Graph output start time
set TEOUT = 20050120030000          # Graph output end time
set TOINT = 3600                    # Graph output time intarval

#####################################################################

### nc convergin start
setenv INFOFILE ./graph_$OUTMODEL.nmlist
setenv FILEHEAD ./out/$OUTMODEL-
set G_DIR = `pwd`
set C_DIR = $HOME/COUPLER/jc/src

cd $C_DIR
/bin/cp c.conf $G_DIR
/bin/cp *data.conf $G_DIR
cd $G_DIR
/bin/rm $FILEHEAD*.nc

nmlist.sh

ncconv.exe $TSCAL $TSOUT $TEOUT $TOINT
```

図 32　可視化データ出力プログラムの実行シェルスクリプト“nc. sh”

5. 水循環結合モデルシステム

カップラーによる結合モデル開発として、図33に示すような物質動態の媒体となる水循環についての結合モデル開発を完了した。本結合モデルシステムは、大気モデル、海洋モデル、波浪モデル、水文モデル、及び陸面モデルの5モデルを結合したものである。大気モデルには、ペンシルバニア州立大学と米国大気研究センター（NCAR）が開発した非静力大気力学モデル（MM5）[6]、海洋モデルには、プリンストン大学が開発した海洋モデル（POM）[7]、波浪モデルには、米国海洋大気局（NOAA）の第3世代海洋波浪推算モデル（WW3）[8]、陸面及び水文モデルには、日本原子力研究開発機構で開発した多層陸面モデル（SOLVEG）[9), 10)]及び3次元水文モデル（RIVERS）[11)]をそれぞれ用いている。カップラーを用いた結合手法は、このような複雑な結合モデルシステムを構築する上で非常に有効な特徴を有する。各モデルの修正は、データ交換を行うモジュールを追加するだけで、計算コードの構成は元のまま維持される。これにより、定期的にアップグレードされるMM5のようなコミュニティーモデルを用いる場合のバージョンアップの適用や、新たなモデルの導入など、結合モデルシステムを常に最新の状態に保つための労力が大幅に低減される。また、モデルカップラーの補間機能により、モデル毎に異なる時間ステップや格子間隔を用いるなど、柔軟な適用が可能である。

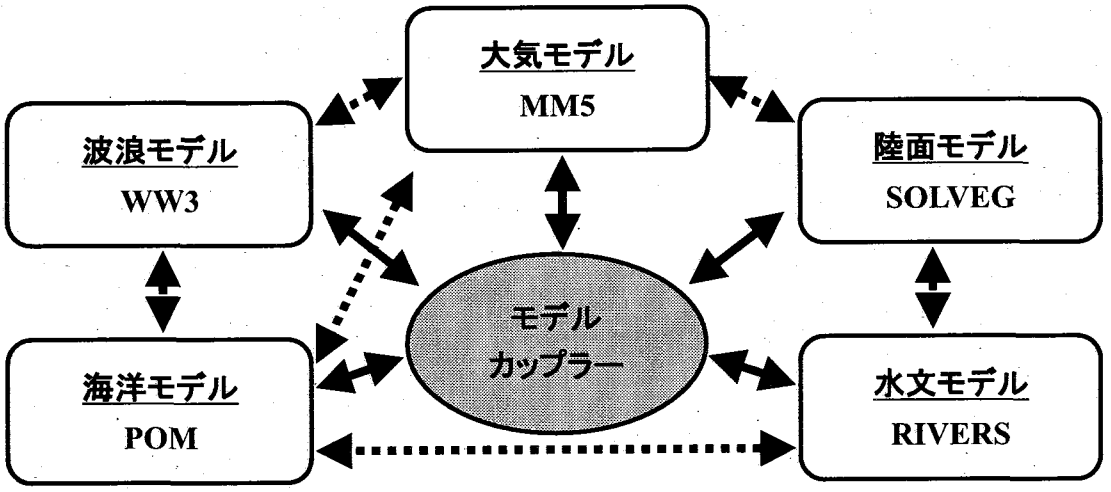


図33　大気・海洋・陸域結合水循環モデルの相互作用（破線矢印）とデータ交換（実線矢印）

5．1　結合の概要

モデル間でのデータの遷移についてカップラーを省略した形で図34に一例を示す。データ交換は送受信の対となるモデルのうち、時間ステップの長いモデルの毎ステップに行われる。例では、各モデルの時間ステップを、MM5：30秒、POM：900秒、WW3：900秒、SOLVEG：5秒、RIVERS：5秒とすると、MM5－SOLVEGは30秒、MM5－POMは900秒毎にデータ交換することになる。

MM5、SOLVEG及びWW3はMPIによって並列化されているが、その他の2モデルはシングルプロセッサで動作する。システム実行時は、まずカップラーが起動される。その後、設定ファイルに従ってモデルが起動され、モデルからの送受信命令に従ってデータが中継される。通常モデル間

のデータは双方向で交換されるが、MM5 と SOLVEG については設定ファイルにより送受信の ON/OFF の切り替えが可能である。あるモデルから別のモデルへ一方通行で情報が渡される場合には、送り側のモデルだけを先に計算することが可能である。各モデルが送受信するデータを表 5 から表 9 にまとめて記述する。また、各モデルに組み込む結合モジュールの詳細は、付録 C に記述する。

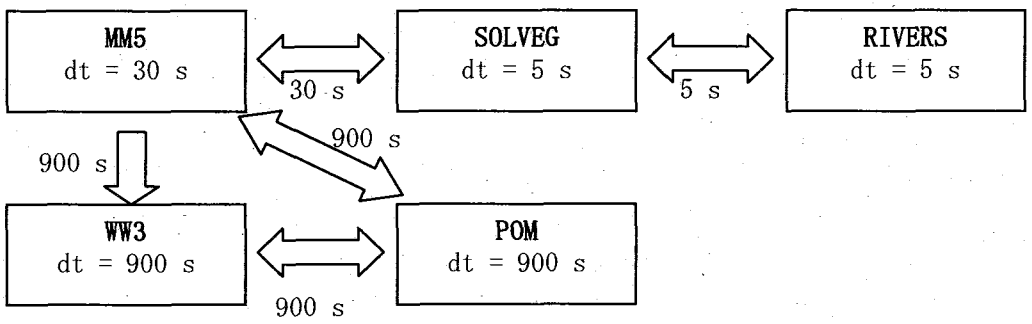


図 34　水循環モデルのデータ交換：図中の時間はデータ交換間隔

モデルカップラーを用いた上記 5 モデルの結合試験計算を以下の条件で実施し、モデル間のデータ交換が正しく行われ結合計算が正常に実行されることを確認した。計算実行には、原子力機構システム計算科学センターの大型計算機 Altix3700Bx2 を用いた。

- MM5：2 領域の 2way ネスティング計算
  領域 1：格子数 100×100×23、分解能 45km、時間ステップ 90s
  領域 2：格子数 100×130×23、分解能 15km、時間ステップ 30s
- SOLVEG：MM5 の領域 2 と同じ領域
  水平格子数、100×130、大気 10 層（10m）、土壌 6 層（1m）、植生 5 層、時間ステップ 5s
- RIVERS：SOLVEG（MM5 領域 2）と同じ領域
  水平格子数 100×130、鉛直：表層＋土壌 4 層＋河川、時間ステップ 5s
- POM：海洋に合わせて領域設定
  格子数 56×81×11、分解能 22km、時間ステップ 900s
- WW3：POM と同じ領域
  格子数 56×81、分解能 22km、時間ステップ 900s

各モデル間のデータ交換の時間間隔は、図 34 に示した例と同じである。

MM5、SOLVEG 及び WW3 については、MPI による並列計算が可能であり、並列数を変更して試験計算を行った。その結果 MM5 を 16CPU、SOLVEG を 40CPU、WW3、POM 及び RIVERS はそれぞれ 1CPU、モデルカップラーが MPI 制御用を含めて 2CPU 使用し、合計 61CPU 使用して結合計算を実行したケースの計算時間が最も短く、24 時間の積分期間に対して経過時間 43 分、総 CPU 時間 2554 分であった。全モデルを 1CPU（総 CPU 数 7）で実行したケースでは、経過時間 473 分、総 CPU 時間 2821 分であり、並列化による経過時間短縮が効率よく達成できているだけでなく、総 CPU 時間も短縮されている。これは、後者の場合では各モデルに割り振られた CPU の負荷の差が大きく、データ交換の際に通信待ちが多く発生していたためである。このような多数の異なるモデルの結合計算

では、CPUの負荷が均等になるように各モデルの並列数を設定することが重要である。なお、MM5単独計算を16CPUで実行した場合の経過時間は33分であり、結合計算の1/4程度はデータ通信あるいは通信待ちの時間であった。この通信時間を短縮するために、カップラーによるデータ交換を並列化するなどの改良が今後の課題である。

表5　MM5が送受信するデータ

変数名		データの名称	相手モデル	通信頻度
送信	XLAT	latitude	SOLVEG	第一ステップのみ
	XLONG	longitude	同上	同上
	HT	ground_height	同上	同上
	RHO1	air_density	同上	同上
	TMN	sb_temp	同上	同上
	TSS	sst	同上	同上
	ISLTYP	soil_type	同上	同上
	IVEGTYPE	vet_type	同上	同上
	SMCA	soil_moist	同上	同上
	PSB, PP3D	surface_press	同上	30秒（毎ステップ）
	GSW, ALB	solar_rad	同上	同上
	GLW	longwave_rad	同上	同上
	RAINP	precipitation	同上	同上
	UA10	u_wind	POM, SOLVEG, WW3	同上
	VA10	v_wind	同上	同上
	aMAG1	aMAG1	WW3	900秒
	TA10	temprerature	POM, SOLVEG	30秒（毎ステップ）
	QA10	specific_humid	SOLVEG	同上
	PSLV	pslv	POM	900秒
	Z3D	3d_height	Graph	第一ステップのみ
	U3D	u3d	同上	任意
	V3D	v3d	同上	同上
	W3D	w3d	同上	同上
	T3D	t3d	同上	同上
受信	RECV_SH	heat_flux	SOLVEG	第二ステップ以降
	RECV_QF	vapor_flux	同上	同上
	RECV_TF	surface_temp	同上	同上
	RECV_AL	albedo	同上	同上
	RECV_SST	pom_sst	POM	900秒（POM毎ステップ）

表6　SOLVEGが送受信するデータ

変数名		データの名称	相手モデル	通信頻度
送信	SEND_SH	heat_flux	MM5	第二ステップ以降
	SEND_QF	vapor_flux	同上	同上
	SEND_TS	surface_temp	同上	同上
	SEND_AL	albedo	同上	同上
	HW, DZS	send_w1	RIVERS	毎ステップ
	RES	send_w0	同上	同上
	WF	send_qd1	同上	同上
	ET, DZS	send_sroot	同上	同上
	PRT	send_qra	同上	同上
	EB	send_qu5	同上	同上
	ZS	z_depth	Graph	第一ステップのみ
	RES	w_sfc	同上	任意
	HW	w_soil	同上	同上
	TS	t_soil	同上	同上
受信	FLAT	latitude	MM5	第一ステップのみ
	FLON	longitude	同上	同上
	GHGT	ground_height	同上	同上
	ROU	air_density	同上	同上
	TBOTM	sb_temp	同上	同上
	TWATR	sst	同上	同上
	STY	soil_type	同上	同上
	UTY	veg_type	同上	同上
	HWI	soil_moist	同上	同上
	PHPI	surface_press	同上	30秒（MM5毎ステップ）
	RSOLI	solar_rad	同上	同上
	RINFI	longwave_rad	同上	同上
	RRI	precipitation	同上	同上
	UI	u_wind	同上	同上
	VI	v_wind	同上	同上
	TI	temperature	同上	同上
	QI	specifig_humid	同上	同上
	ARF	river_ar	RIVERS	第一ステップ
	HWB	river_w2	同上	毎ステップ
	ER (I,J,0)	river_dw0	同上	同上
	ER (I,J,K)	river_dw1	同上	同上
	WFR	river_qu2	同上	同上

表 7　POM が送受信するデータ

変数名		データの名称	相手モデル	通信頻度
送信	T	pom_sst	MM5	900 秒（毎ステップ）
	UA	ua	WW3	同上
	VA	va	同上	同上
	oMAG1	oMAG1	同上	同上
	EL	el	同上	同上
	ZZ	zz	Graph	第一ステップのみ
	H	depth	同上	同上
	T	temperature	同上	任意
受信	mm5_u	mm5_u	MM5	900 秒（毎ステップ）
	mm5_v	mm5_v	同上	同上
	mm5_t	mm5_t	同上	同上
	pslv	pslv	同上	同上
	TAUBRX	taubrx	WW3	同上
	TAUBRY	taubry	同上	同上
	wMAG1	wMAG1	同上	同上

表 8　RIVERS が送受信するデータ

変数名		データの名称	相手モデル	通信頻度
送信	ARF	river_ar	SOLVEG	第一ステップのみ
	W2	river_w2	同上	毎ステップ
	DW0	river_dw0	同上	同上
	DW1	river_dw1	同上	同上
	QU2	river_qu2	同上	同上
受信	W1	recv_w1	SOLVEG	毎ステップ
	W0	recv_w0	同上	同上
	QD1	recv_qd1	同上	同上
	QROOT	recv_sroot	同上	同上
	QRA	recv_qra	同上	同上
	QU5	recv_qu5	同上	同上

表 9　WW3 が送受信するデータ

変数名		データの名称	相手モデル	通信頻度
送信	TAUBRXFLD	tbrx	POM	毎ステップ
	TAUBRYFLD	tbry	同上	同上
	wMAG1	wMAG1	同上	同上
受信	WXRECV	ua10	MM5	毎ステップ
	WYRECV	va10	同上	同上
	aMAG1	aMAG1	同上	同上
	CXRECV	ua	POM	同上
	CYRECV	va	同上	同上
	oMAG1	oMAG1	同上	同上
	WLEVRECV	el	同上	同上

5．2　適用計算例

水循環結合モデルシステムの適用研究について記述する。これまでに水循環結合モデルシステムの適用試験として、サウジアラビアにおける 2005 年 1 月の洪水再現計算及び 2005 年 8 月のハリケーン・カトリーナによる高潮再現計算を実施した。サウジアラビアの洪水再現計算は、文部科学省「人・自然・地球共生プロジェクト」(2002～2006 年) の課題「広域水循環予測及び対策技術の高度化」[12]に参加し、水循環結合モデルシステム適用の一環として実施したものである。これらの適用結果については、数値環境システム SPEEDI-MP の報告書[4]に詳細が記述してあり、ここでは概略のみ記述する。

サウジアラビアにおける洪水は、2005 年 1 月 22 日の豪雨に起因して Madinah 周辺地域で発生した。この洪水を再現するために、気象モデル MM5、陸面モデル SOLVEG、及び水文モデル RIVERS の 3 モデルをカップラーにより結合した陸域のみの結合モデルを用いた。計算期間が短く海洋場変動の影響は無視できるため、海洋計算は結合せずに MM5 に期間を通して一定の海面境界条件を与えた。MM5 の計算領域を 3 段のネスティング領域とし、領域 3 の局地計算を SOLVEG 及び RIVERS と結合した。入力データとして NCEP/NCAR 再解析データ (NOAA-CIRES Climate Diagnostics Center, Boulder, Colorado, USA, from the Web site at http://www.cdc.noaa.gov/) を用いて、2005 年 1 月 20 日 00UTC から 7 日間の計算を行った。MM5 の降水計算は 1 月 22 日の豪雨をほぼ再現し、RIVERS は地表にたまった雨水が流出し谷間に集中する様子をシミュレートすることができ、洪水の氾濫域を再現することに成功した。

ハリケーン・カトリーナによる高潮の再現計算においては、気象モデル MM5、波浪モデル WW3、及び海洋モデル POM の 3 モデルをカップラーにより結合した海洋のみの結合モデルを用いた。大気・海洋・波浪結合モデルは、海面水位観測値を良好に再現した。波浪モデルを用いない計算では、ハリケーン通過前に水位上昇を過大評価し、通過後に過小評価となっていた。これは、ハリケーンの強風による海水の吹き寄せ効果 (通過前は水位を低くする方向に働いている) の計算に砕波ストレスの寄与を考慮していないためである。この結果より、波浪に起因する大気―海洋間の運動量交換を考慮するために、波浪モデルを結合することの重要性が確認できた。

6. まとめ

本研究開発では、数値実験による様々な環境研究に資することのできる環境研究ツール「数値環境システム SPEEDI-MP」の数値実験ツールとして、モデル結合プログラム（モデルカップラー）を開発した。本カップラーは、複数のモデルを同時進行で平行計算させ、並列計算の通信ライブラリ MPI を用いてモデル間で計算結果を交換することにより、モデルを一体化したのと同等な結合状態を作り出すことができる。このモデルカップラーを用いて、大気モデル MM5、海洋モデル POM、波浪モデル WW3、陸水モデル RIVERS、及び地表モデル SOLVEG の 5 モデル結合システムを開発し、統合的な水循環シミュレーション研究の基盤を確立した。

数値環境システム SPEEDI-MP の数値実験ツールのうちカップラーによる結合モデル開発として、物質動態の媒体となる水循環についての結合モデル開発を完了した。この水循環結合モデルシステムの適用試験として、サウジアラビアにおける 2005 年 1 月の洪水再現計算及び 2005 年 8 月のハリケーン・カトリーナによる高潮再現計算を実施した。その結果、結合モデルでのみ実現可能な複合系の相互作用や現象を再現するなど良好な結果が得られ、水循環研究における本システムの有効性が示された。今後、数値シミュレーションによる様々な環境研究への適用を図る計画である。

謝辞

水循環結合モデルシステムの適用研究は、システム計算科学センターの大型計算機 Altix3700Bx2 を利用した成果です。

参考文献

1) K. Imai, M. Chino, H. Ishikawa, M. Kai, K. Asai, T. Homma, A. Hidaka, Y. Nakamura, T. Iijima, and S. Moriuchi : "SPEEDI: A Computer Code System for the Real-Time Prediction of Radiation Dose to the Public due to an Accidental Release", JAERI-1297 (1985).

2) M. Chino, H. Ishikawa, H. Yamazawa, H. Nagai, and S. Moriuchi : "WSPEEDI (Worldwide Version of SPEEDI): Emergency Response System to Predict Radiological Impacts on the Japanese People due to a Nuclear Accident in Abroad", JAERI-1334 (1995).

3) 文部科学省："緊急時迅速放射能影響予測ネットワークシステム", パンフレット(2003).

4) 永井晴康, 茅野政道, 寺田宏明, 原山卓也, 小林卓也, 都築克紀, 金庚玉, 古野朗子："数値環境システム SPEEDI-MP", JAEA-Research 2006-057 (2006).

5) Unidata : NetCDF (Network Common Data Form) , (online) available from 〈http://www.unidata.ucar.edu/software/netcdf/〉 (accessed 2006-06-21).

6) G. A. Grell, J. Dudhia, and D. R. Stauffer : "A description of the Fifth-generation Penn State/NCAR Mesoscale Model (MM5)", NCAR Tech. Note, NCAR/TN-398+STR, 122pp. (1994).

7) G. L. Mellor : "Users Guide for a Three-dimensional, Primitive Equation, Numerical Ocean Model", Princeton University, 40 pp. (1998).

8) H. L. Tolman : "User Manual and System Documentation of WAVWWATCH-III Version 2.22", Tech. Note 222, NOAA/NWS/NCEP/OMB, 133 pp. (2002).

9) H. Nagai : "Atmosphere-soil-vegetation Model Including CO_2 Exchange Processes: SOLVEG2", JAERI-Data/Code 2004-014 (2004).

10) H. Nagai : "Incorporation of CO_2 exchange processes into a multilayer atmosphere-soil-vegetation model", J. Appl. Meteor., 44, 1574-1592 (2005).

11) K. Tsuduki, and T. Matsunaga : "Importance of hydrological parameters in contaminant transport modeling in a terrestrial environment", Proceedings of International Symposium on Environmental Modeling and Radioecology, Rokkasho (2006).

12) 文部科学省：新世紀重点研究創生プラン Research Revolution 2002 (RR2002)人・自然・地球共生プロジェクト, (online) available from 〈http://kyousei.aesto.or.jp/〉 (accessed 2006-06-21).

付録A：空間補間のグリッド対応アルゴリズム

データの空間補間に関わる最も重要な問題は、異なるグリッド間でのグリッドポイントの対応、すなわちあるモデルのあるグリッドポイントが他モデルのどのグリッドにある（あるいはどのグリッドによって囲まれているか）をいかに計算し、いかにその情報を保持するかである。

（1）グリッド情報の保持

本カップラーは、複数のモデルでデータ交換を行い、個々のモデルは複数のネストされたドメインを持ち、個々のドメインは複数の格子系を持つことを前提としている。格子系については、スタッガリングを考えるとスカラー点、ｘベクトル、ｙベクトル・・・ｘｙｚベクトルの最大８通りがある。従って、仮に２つのモデルで各モデルがネストしていないという最も簡単な場合においても、データ交換に用いるグリッドの組み合わせは8×8＝64通りある。これら全ての組み合わせについて、グリッド位置の対応を計算し、結果を保持しておくことは計算時間およびメモリリソースの双方で問題が生じる可能性が高い。従って本カップラーでは、以下の方針を採用した。

- 受信モデル、送信モデルについて（モデル番号、ドメイン番号、格子系番号）の３つの変数および交換するデータが２次元か３次元かのフラッグを設定する。
- 新しい組み合わせのデータ交換が発生した場合にのみ、対応するグリッド位置を計算し結果を保持する。
- 同じ組み合わせがあった場合、対応するグリッド位置情報は今までの計算結果を用いる。

このためには、グリッド位置情報を保持する領域を動的に増やす仕組みが必要である。これを実現しているのは、モジュール jc_interpolation で定義されている構造体 c_grid_type と構造型変数 start_ptr 及び now_ptr である。図A1に構造体 c_grid_type を示す。構造体のメンバ変数の定義は、以下のとおりである。

is_reset：	グリッド位置情報が再設定されたかどうかのフラッグ
s_model：	送信モデル番号
s_domain：	送信モデル領域番号
s_stag：	送信モデル格子番号
d_model：	受信モデル番号
d_domain：	受信モデル領域番号
d_stag：	受信モデル格子番号
flag_2D3D：	２次元データか３次元データかのフラッグ
di_2D, dj_2D：	対応グリッド情報２次元
di_3D, dj_3D, dk_3D：	対応グリッド情報３次元
next_ptr：	次の情報へのポインタ

メンバ変数のうち、s_model、s_domain、s_stag、d_model、d_domain、d_stag、flag_2D3D が

組み合わせの検証に使われる変数である。これらの情報は、送信、受信用サブルーチンjc_SendData2D、jc_RecvData2D、jc_SendData3D、jc_RecvData3Dを通してデータ補間モジュールjc_interpolationに渡される。データ補間モジュールの関数isNewGridCombinationで、今までに同じ組み合わせがあるかどうかのチェックが行われ、同じ組み合わせがない場合、サブルーチンInitNewGrid2DまたはInitNewGrid3Dで新たな構造型変数の生成とデータのアロケーションを行い、サブルーチンCalCorGrid_2DまたはCalCorGrid_3Dで対応グリッドの計算を行う。同じ組み合わせがある場合、その変数で保持している対応グリッドを用る。

next_ptrはc_grid_type型のポインタである。構造体の中にその構造体の型を指すポインタ変数を定義し、チェーン状にデータを構築してゆく手法は動的にデータの大きさを増減するために用いられるC言語などでは一般的な手法である。従って、ここで詳細については解説しない。

```
type c_grid_type
  logical   :: is_reset
  integer :: s_model, s_domain, s_stag
  integer :: d_model, d_domain, d_stag
  integer :: flag_2D3D
  integer,pointer :: di_2D(:,:), dj_2D(:,:)
  integer,pointer :: di_3D(:,:,:), dj_3D(:,:,:), dk_3D(:,:,:)
  type (c_grid_type), pointer :: next_ptr
end type

type(c_grid_type), private, pointer :: start_ptr
type(c_grid_type), private, pointer :: now_ptr
```

図A1　データ交換情報を保持する構造体

（2）対応グリッドの計算法

モデルA（受信側モデル）のあるグリッドポイントの値をモデルB（送信側モデル）の計算結果から補間する場合、モデルBのどのグリッドポイントの値を使うかを最初に求める必要がある。ここではこの計算を対応グリッドの計算と呼ぶことにする。2次元の例を図A2に示す。太線をモデルAのグリッド、細線をモデルBのグリッドとする。モデルAのグリッドサイズをni、nj、グリッド位置をi、jで、モデルBのグリッドサイズをNI、NJ、グリッド位置をI、Jで示す。また、グリッドポイントの位置はp(i,j)=(x(i,j)、y(i,j))、P(I,J)=(X(I,J)、Y(I,J))として表す。例えば、(i,j)=(1,1)の点の値を4点補間で求めるには(I,J)=(2,2)、(2,3)、(3,2)、(3,3)の値を必要とする。(i,j)=(1,1)に対して(I,J)=(2,2)を求めることが対応グリッドの計算である。対応グリッドをdi_2D、dj_2Dで表す(2次元の場合)。上記の例の場合、di_2D(1,1)=2、dj_2D(1,1)=2となる。

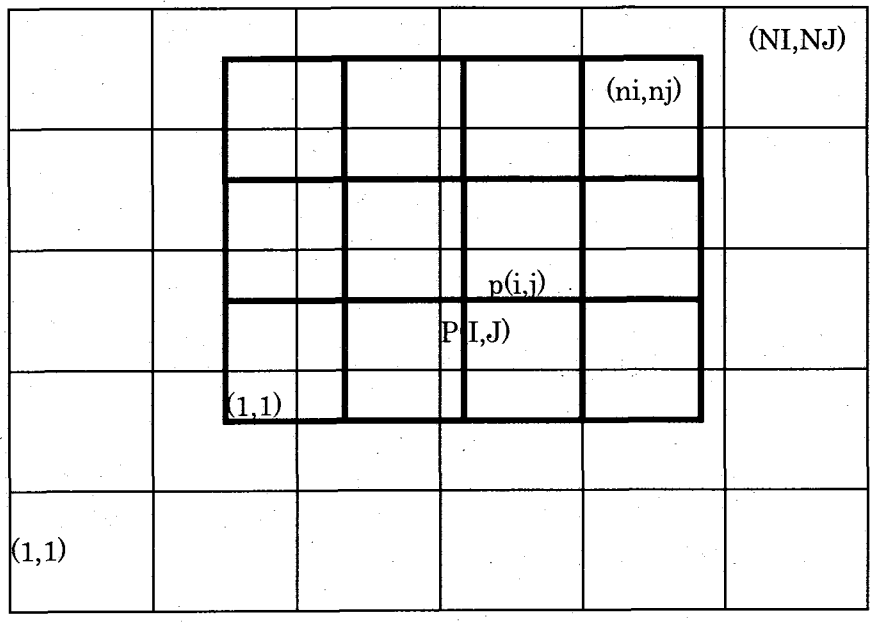


図 A2　グリッドの対応関係（太線：受信モデルグリッド、細線：送信モデルグリッド）

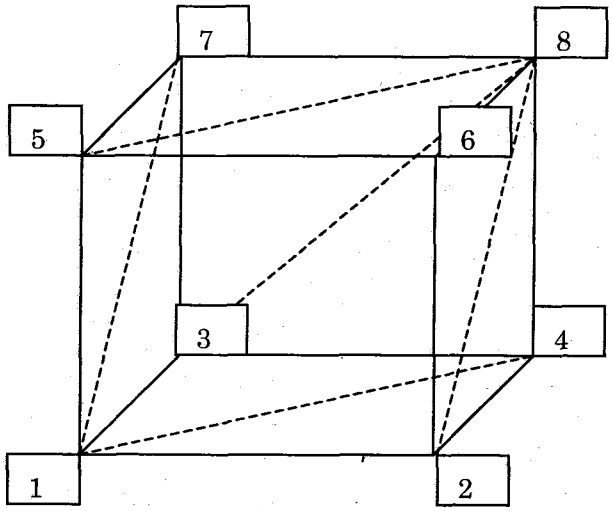
```
do j = 1 to nj
  do i = 1 to ni
    do J=1 to NJ
      do I=1 to NI
        if (p(i,j)⊂四角形P(I,J),P(I+1,J), P(I,J+1), P(I+1,J+)) then
          di_2D(i,j) = I ; dj_2D(i,j) = J
          Goto 探査終わり
        end if
      end do
    end do
    ラベル：探査終わり
  end do
end do
```

図 A3　対応グリッド位置計算アルゴリズム

対応グリッドの計算を管理するモジュールは jc_interpolation_base である。このモジュール内のサブルーチン CalCorGrid_2D と CalCorGrid_3D が対応グリッド計算のためのメインルーチンであり、jc_interpolation 内のサブルーチン InterpolateData2D と InterpolateData3D からコールされる。対応グリッド位置の計算アルゴリズムを 2 次元の場合について図 A3 に示す。受信側グリッドの一つ一つの点 p(i,j)に対して、送信側グリッドのループ J=1 to NJ、I=1 to NI を回し、p(i,j)を囲む 4 角形を探査するというアルゴリズムを用いている。実際のプログラムでは、j=1 to nj、i=1 to ni のループの中で p(i,j)についての探査用サブルーチンをコールするようになっており、Goto 文は使用されていない。また、内側のループで I=1 to NI、J=1 to NJ というループを回しグリッドの端から探査を開始したのでは効率が悪いため、一つ前の対応ポイント I=di_2D(i-1,j)、j=dj_2D(i-1,j)から探査を始め、同心円状に探査範囲を広げてゆくというスキ

ームを採用している。

次に、p(i,j)⊂四角形(P(I,J)、P(I+1,J)、P(I,J+1)、P(I+1,J+1))の判定について説明する。ある点pが四角形（2次元の場合）または立方体（3次元の場合）に含まれるかどうかは、pから任意の方向にのばした半直線が四角形の辺または立方体の面と交差する回数が偶数か奇数かで判断できる。交差する回数が奇数の場合、p は領域に含まれることになる。この判定を行っているサブルーチンは、isMySquare_2D と isMyArea_2D 及び isMyCube_3D と isMyVolume_3D である。まず、isMySquare_2D と isMyCube_3D では、p(i,j)が頂点と一致するかどうかを判定し、ついで辺上にあるかどうかを判定する。isMyCube_3D では、面上にあるかどうかの判定も行う。isMyArea_2D と isMyVolume_3D は、点 p(i,j)が領域内かどうかの判定を行う。ここでは、p(i,j)が頂点や辺、面上にないことが保証されているため、p(i,j)は境界上にはないことを前提として計算を行っている。また、判定のための半直線は任意の方向ベクトルを持つことができるが、頂点や辺を横切っている場合、奇偶の判定が困難なため、頂点や辺を横切るかどうかをまず判別し、横切っていた場合には方向ベクトルを変えて計算する。プログラム中での6面体の頂点の定義は図A4のようになっている。



図A4　頂点の定義と面の分割

各面を構成する4頂点は、同一平面上にあるとは限らない。従って、p(i,j)が面上にあるかどうかの判定、半直線が面を横切るかどうかの判定は、4頂点からなる4角形を2つに分割してできる2つの3角形に対して行われる。この分割は任意であるため、本プログラムでは図A4の点線で示すように分割している。この分割は、jc_interpolation_base のサブルーチン InitInterpolationBase で定義されている。

付録B：水循環結合モデルシステム用空間補間プログラム

座標系の異なるモデル間でデータを交換する場合データの補間が必要となるが、補間アルゴリズムはカップリングされるモデルの性質や用途によって様々であるため、カップラー本体には補間プログラムは実装されておらず、データ補間のためのインターフェースのみが提供されている。従って、利用者は個々の必要に応じて補間プログラムを実装することが求められているが、水循環結合モデルシステム用の空間補間については具体的なアルゴリズムを実装している。ここでは、この空間補間方法について説明する。

（1）対応グリッドと補間係数の計算

対応グリッドと補間係数を定義した構造体（interp_type_MPW）及びinterp_type_MPW型の変数は、モジュール jc_interpolation_interface のインターフェース部に定義されている。構造体interp_type_MPWを図B1に示す。isComputedは、対応グリッドと補間係数が計算されたかどうかのフラッグで、計算された場合“.true.”となる。整数の2次元配列itiとitjは対応グリッドを保持する変数であり、受信モデルの配列の大きさをni×njとするとiti(ni,nj)、itj(ni,nj)となる。実数型の3次元配列dintpは補間係数を保持する変数であり大きさはdintp(ni,nj,4)となる。

```
type interp_type
     logical                :: isComputed = .false.
     integer, pointer :: iti(:,:),itj(:,:)
     real     , pointer :: dintp(:,:,:)
end type
```

図B1　補間情報を保持する構造体

送信側のグリッド数をnxs×nys、座標位置を（xs(nxs, nys), ys(nxs,nys)）とし、受信側のグリッド数をnxr×nyr、座標位置を（xr(nxr,nyr), yr(nxr,nyr)）とする。このとき対応グリッド(isr(nxr,nyr), jsr(nxr,nyr))は、図B2に示す様に計算される。すなわち、受信側のグリッドポイントに最も近い送信側グリッドポイントを求め、受信側グリッドポイントより送信側グリッドポイントが大きい値の場合一つ前の座標を採用するというアルゴリズムである。送信側のループ範囲は、最外側のグリッドを含まないため、受信側のグリッドポイントが送信側の座標系の外側にある場合、最も外側にある送信側グリッドポイントが選択されることになる。

```
do j2 = 1, nyr
  do i2 = 1, nxr
    dist = 大きな数
    do j = 2, nyr-1
      do i = 2, nxr-1

        dist1 = 二点(xs(i,j),ys(i,j)),(xr(i2,j2),yr(i2,j2))の距離
        if (dist1<dist) thne
          dist = dist1
          ii = i
          jj = j
        end if
      end do
    end do
    if (xs(ii,jj)>xr(i2,j2)) ii = ii-1
    if (ys(ii,jj)>yr(i2,j2)) jj = jj-1
    isr(i2,j2) = ii
    jsr(i2,j2) = jj
  end do
end do
```

図 B2　対応グリッドの計算方法

ある受信側グリッドポイント(i, j)のデータは、それを囲む 4 点の送信側グリッドポイントの値から計算される。具体的な計算式は以下のとおりである。

$$\begin{aligned} dtr(i,j) = {} & dts(isr(i,j), jsr(i,j))*f1+dts(isr(i,j)+1, jsr(i,j))*f2 \\ & +dts(isr(i,j), jsr(i,j)+1)*f3+dts(isr(i,j)+1, jsr(i,j)+1)*f4 \end{aligned} \quad (B1)$$

ここで、f1、f2、f3、f4 は、補間係数である。これらの係数は、受信側グリッドポイントとそれを囲む 4 つの送信側グリッドポイントの距離を dist1、dist2、dist3、dist4 とすると、以下に示すように計算される。

```
d1 = dist2*dist3*dist4
d2 = dist1*dist3*dist4
d3 = dist1*dist2*dist4
d4 = dist1*dist2*dist3
f1 = d1/(d1+d2+d3+d4)
f2 = d2/(d1+d2+d3+d4)
f3 = d3/(d1+d2+d3+d4)
f4 = d4/(d1+d2+d3+d4)                                    (B2)
```

受信側グリッドが送信側座標系の外側にある場合、最も近い送信側グリッドが対応グリッドとなるため、上の式に基づいて最も近い 4 つのグリッドポイントを用いて補間計算が行われる。グリッドポイントと距離の定義を図 B3 に示す。

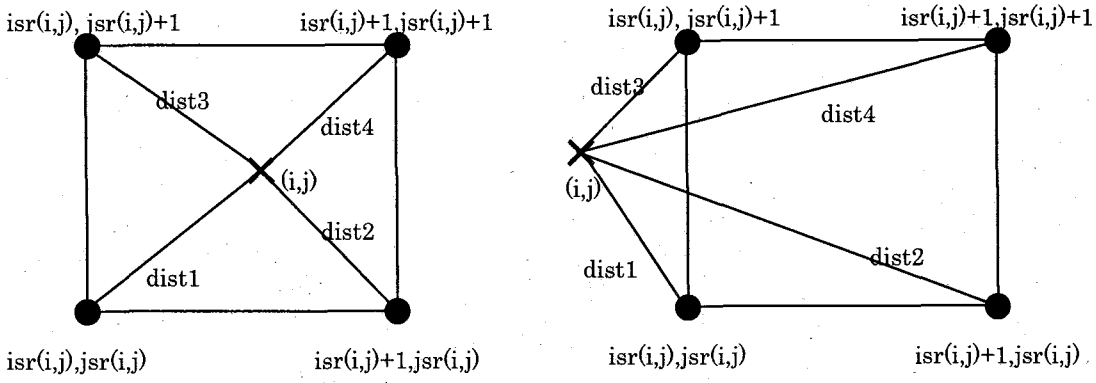


図 B3　グリッドポイントと距離の関係

（4）補間計算の使用法

プログラムファイル jc_interpolation_interface.f90 内のサブルーチン jc_InterpolateData2DReal 等の select case(idx)文が、補間計算を行っている箇所である。idx の値は、(送信モデル ID) × 100 + (受信モデル ID) で与えられ、モデル ID を MM5=1、POM=2、WW3=5 とすると、idx=102 及び idx=201 が MM5 と POM 間、idx=105 及び idx=501 が MM5 と WW3 間、idx=205 及び idx=502 が POM と WW3 間の補間となる。MM5、POM、WW3 以外のモデルについても、case(idx)ブロックを追加することにより、本補間機能を利用することができる。select case 文の該当部分を図 B4 に示す。各通信方向で、対応グリッドと補間係数の計算が行われていない場合、即ち“intp_mpw(x)%isComputed=.false.”のときは、対応グリッドと補間係数の計算サブルーチン jc_Init_intp_MPW がコールされる。サブルーチン jc_Init_intp_MPW では図 B2 及び式(B2)で示したアルゴリズムに基づいて iti 及び itj と dintp が計算される。次いでサブルーチン jc_Interpolation_MM5_POM_WW3 がコールされる。サブルーチン jc_Interpolation_MM5_POM_WW3 では、式(B1)で示したアルゴリズムに基づいて補間計算が行われる。この際に、POM の陸域を示す値 LAND_MASK=-9999.0 が与えられたデータが含まれる場合は、そのデータを除いて補間係数が再計算され、残りのデータのみ用いて補間計算が行われる。全てのデータに LAND_MASK=-9999.0 が与えられている場合には、補間値は LAND_MASK=-9999.0 となる。

```
case(102)
   if(.not.intp_mpw(5)%isComputed) then
      call jc_Init_intp_MPW(nxs,nys,xs,ys,nxr,nyr,xr,yr,intp_mpw(5))
   end if
   call jc_Interpolation_MM5_POM_WW3(nxs,nys,dts,nxr,nyr,dtr,intp_mpw(5))
   return
 case(201)
   if (.not.intp_mpw(6)%isComputed) then
      call jc_Init_intp_MPW(nxs,nys,xs,ys,nxr,nyr,xr,yr,intp_mpw(6))
   end if
   call jc_Interpolation_MM5_POM_WW3(nxs,nys,dts,nxr,nyr,dtr,intp_mpw(6))
   return

case(105)
   if(.not.intp_mpw(1)%isComputed) then
      call jc_Init_intp_MPW(nxs,nys,xs,ys,nxr,nyr,xr,yr,intp_mpw(1))
   end if
   call jc_Interpolation_MM5_POM_WW3(nxs,nys,dts,nxr,nyr,dtr,intp_mpw(1))
   return
 case(501)
   if (.not.intp_mpw(2)%isComputed) then
      call jc_Init_intp_MPW(nxs,nys,xs,ys,nxr,nyr,xr,yr,intp_mpw(2))
   end if
   call jc_Interpolation_MM5_POM_WW3(nxs,nys,dts,nxr,nyr,dtr,intp_mpw(2))
   return

 case(205)
   if (.not.intp_mpw(3)%isComputed) then
    call jc_Init_intp_MPW(nxs,nys,xs,ys,nxr,nyr,xr,yr,intp_mpw(3))
   end if
   call jc_Interpolation_MM5_POM_WW3(nxs,nys,dts,nxr,nyr,dtr,intp_mpw(3))
   return
 case(502)
   if (.not.intp_mpw(4)%isComputed) then
    call jc_Init_intp_MPW(nxs,nys,xs,ys,nxr,nyr,xr,yr,intp_mpw(4))
   end if
   call jc_Interpolation_MM5_POM_WW3(nxs,nys,dts,nxr,nyr,dtr,intp_mpw(4))
   return
```

図 B4　MM5-POM-WW3 間の補間プログラム

付録C：水循環結合モデルシステムの結合モジュール

水循環結合モデルシステムにおける各モデルのカップリングに関わる処理内容を説明する。モデル毎にカップリング用のファイルとして“jc_[モデル名].f90”もしくはそれに類するファイルを持つ。このファイル中には、カップラーのインターフェースサブルーチンをコールして初期化やデータ交換を行う複数のサブルーチンが記述されている。以下、モデル毎にファイルの詳細とカップリング方法について説明する。

（1）MM5

jc_mm5.f90及びcouple_pom_ww3.F

カップリング用ルーチンは、jc_mm5.f90 と couple_pom_ww3.F 内に含まれる。各ルーチンの名称・引数・機能を表C1に示す。なお、引数のうち配列をもつものについては表記の煩雑化を避けるため、大きさを省略している。これは以下のモデルについても同様である。

jc_mm5.f90及びcouple_pom_ww3.Fを用いるファイル

jc_mm5.f90及びcouple_pom_ww3.Fのサブルーチンをコールしているファイルは“$(MM5_HOME)/Run/mm5.F”、“$(MM5_HOME)/physics/pbl_sfc/mrfpbl/mrfpbl.F”、” $(MM5_HOME)/dynamics/nonhydro/solve.F”の3つであり、表C2に各ファイルでコールしているサブルーチンをまとめた。また、図C1に時間積分ループとカップリング関連サブルーチンのコールフローを示す。時間積分ループは、メインプログラム“mm5.F”の10 continure～goto 10の間である。サブルーチンNSTLEV1は、ネスト第一ドメインの計算でこのサブルーチンが数値計算サブルーチンSOLVEをコールしている。SOLVEの中で、データの送受信を行うサブルーチンjc_exchange*がコールされている。また、サブルーチンMRFPBLの中で、フィードバックサブルーチンjc_fbackがコールされている。

表C1　jc_mm5.f90及びcouple_pom_ww3.Fのサブルーチン

名称	引数	機能
jc_start	integer:: num_of_domain	カップラーの初期化：引数num_of_domainはネスト領域数。
jc_set_grid	integer:: inest, ni, nj	グリッド情報の設定：inestは領域番号、ni, njは領域グリッド数。領域数(num_of_domain)分コール。
jc_set_initial_time	integer:: inest, delta_t charactare(len=*):: cdate	積分時刻情報の設定：inestは領域番号、delta_tは積分時間ステップ、cdateは14文字の文字列で表した積分開始時刻。領域数（num_of_domain）分コール。
jc_coupling_end	なし	カップリングの終了
jc_fback	integer:: inest, j, ist, ien real:: xland(:,:), hfx(:,:), qfx(:,:), tgb(:,:), alb(:,:)	他モデル（POMとSOLVEG）から受信したデータのフィードバック
jc_recv	integer:: inest, ny, nx real:: recv_data(:,:) character(len=*):: data_name	2次元データの受信：引数data_nameはデータ名称でmm5.data.confに記述された名称と対応させる。
jc_exchange1_PS	character(len=*):: cdatenew integer:: inest, ny, nx, ktau, ntrad real:: gsw(:,:), alb(:,:)	SOLVEG及びPOMからのデータ受信：引数cdatenewは19文字で表した現在時刻、inestは領域番号、ny, nxは南北、東西グリッド数。
jc_exchange2_PSW	character(len=*):: cdatenew integer:: inest, ny, nx real:: dt, xtime, ptop, xlat(:,:), xlong(:,:), ht(:,:), rh01(:,:), tmn(:,:),tss(:,:), mavail(:,:), psb(:,:), pp3d(:,:,:), glw(:,:), rainp(:,:), ua10(:,:), va10(:,:), ta10(:,:), qa10(:,:)	SOLVEG、POM、WW3へのデータ送信：引数cdatenew_jcは19文字で表した現在時刻、inestは領域番号、ny, nxは南北、東西グリッド数。
jc_exchange3_PW	character(len=19):: cdatenew integer:: inest, ny, nx	POM、WW3へのデータ送信：引数cdatenewは19文字で表した現在時刻、inestは領域番号、ny, nxは南北、東西グリッド数。
SEAPRSNH	integer:: imx, jmx, kx real:: t(:,:,:), ter(:,:), ps(:,:), psfc(:,:), pp(:,:,:), sigma(:), ptop, pslv(:,:)	POMへ送信する海面気圧計算
Jc_graph_out	character(len=*) :: cdatenew integer:: inest, ny, nx real:: z3d(:,:,:), u3d(:,:,:), v3d(:,:,:),w3d(:,:,:),t3d(:,:,:)	Graph出力用のデータ送信：現状では、3次元風速U,V,Wと気温を送信。必要に応じて送信データを追加する。

表 C2　MM5 でカップリングサブルーチンをコールしているファイル

MM5 のファイル名	コールしているサブルーチン名
MM5.F	jc_start, jc_set_grid, jc_set_initial_time, jc_coupling_end
mrfpbl.F	jc_fback
solve.F	jc_exchange1_PS, jc_exchange2_PSW, jc_exchange3_PW, jc_graph_out

```
main(mm5.F)
  call jc_start
  do i = 1, maxnes
    call jc_set_grid
  end do
  call i = 1, maxnes
    call jc_set_initial_time
  end do
  10 continue
    call NSTLEV1(nestlev1.F)
          call SOLVE(solve.F)
               call jc_exchange1_PS
                  if (NSTAMPI>0) recv (heat_flux～albedo)
                  if (mod(NSTAMPI,30)==0) recv (pom_sst)
               call MRFPBL(mrfpbl.F)
                    call jc_fback
               end subroutine MRFPBL
               call jc_exchange2_PSW
                    if (NSTAMPI==0) send (latitude～soil_moist)
                    NSTAMPI++
                    send (surface_press～specific_humid)
               call jc_exchange3_PW
                    send (aMAG1, pslv)
          end subroutine SOLVE
    end subroutine NSTLEV1
  goto 10
  call jc_coupling_end
end main
```

図 C1　MM5 の時間積分ループとカップリング関連ルーチンの呼び出し

（2）POM

jc_pom.f90

カップリング用ルーチンは、jc_pom.f90 内に含まれている。各ルーチンの名称・引数・機能を表 C3 に示す。

表 C3　jc_pom.f90 のサブルーチン

名称	引数	機能
jc_start_init	integer:: ni, nj, delta_t real:: lat(:,:), lon(:,:) character(len=*):: cdate	カップラーの初期化：引数 ni, nj は東西、南北のグリッド数、lat, lon はグリッドポイントの緯度・経度、cdate は積分開始時刻、delta_t は積分時間ステップ。
jc_set_current_time	character(len=14):: cdate integer:: delta_t	現在時刻と領域番号の設定：引数 cdate は時間ステップ増加前の時刻を表す 14 文字の文字列、delta_t は積分時間間隔。カップラーのサブルーチン jc_IncTime で現在時刻を計算し、カップラーに送信。
jc_exchange1_M	integer:: im, jm, kb real:: t(:,:,:), fsm(:,:)	MM5 への SST 送信：引数 im, jm, kb は配列の大きさ。t は温度、fsm は海陸のマスク値。
jc_exchange1_W	integer:: im, jm real:: ua(:,:), va(:,:), el(:,:)	WW3 へのデータ送信：引数 im, jm は配列の大きさ。ua, va は海面流速、el は海面の高さ。
jc_exchange2_M	integer:: im, jm real:: mm5_u(:,:), mm5_v(:,:), mm5_t(:,:), mm5_pslv(:,:)	MM5 からのデータ受信：引数 im, jm は配列の大きさ。mm5_u, mm5_v は最下層の風速、mm5_t は最下層の気温、mm5_pslv は海面気圧。
jc_exchange2_W	integer:: im, jm real:: ww3_tbrx(:,:), ww3_tbry(:,:)	WW3 からのデータ受信：引数 im, jm は配列の大きさ。ww3_tbrx, ww3_tbry は WW3 の TAU。
jc_fback	integer:: im, jm, kb real:: mm5_u(:,:), mm5_v(:,:), mm5_t(:,:), mm5_pslv(:,:), ww3_tbrx(:,:), ww3_tbry(:,:), wusurf(:,:), wvsurf(:,:), wtsurf(:,:),dum(:,:), dvm(:,:), gamma, tb(:,:,:)	受信データを用いた wusurf, wvsurf, wtsurf の計算
jc_coupling_end	なし	カップリングの終了
Jc_graph_out	integer:: im, jm, kb, iint real:: t(:,:,:), fsm(:,:), h(:,:), zz(:)	Graph 出力用のデータ送信：現状では、格子設定用データ以外では、3 次元の温度データのみ送信。必要に応じて送信データを追加。

jc_pom.f90 を用いるファイル

jc_pom.f90 のサブルーチンをコールしているファイルは、"$(POM_HOME)/src/main.f" である。図 C2 に POM の時間積分ループとカップリング関連サブルーチンのコールフローを示す。時間積分ループはメインプログラム "main.f" の do 9000～end do の間である。このループの中で、データ送受信を行うサブルーチン jc_exchange*とフィードバックを行うサブルーチン jc_fback がコールされている。

```
main(main.f)
  call jc_start
  call jc_set_current_time

  do 9000 IINT = 1, IEND
    call jc_exchange1_M
         send (pom_sst)
    call jc_exchange2_M
         recv (mm5_u, mm5_v, mm5_t, pslv)
    call jc_exchange1_W
         send (ua, va, oMAG1, el)
    call jc_exchange2_W
         recv (tbrx, tbry, wMAG1)
    call jc_fback

    call jc_set_current_time
  end do
  call jc_coupling_end
end main
```

図 C2　POM の時間積分ループとカップリング関連ルーチンの呼び出し

（3）SOLVEG

jc_solveg_mm5.f90

MM5 とのカップリング用ルーチンは、jc_solveg_mm5.f90 内に含まれる。各ルーチンの名称・引数・機能を表 C4 に示す。

表 C4　jc_solveg_mm5.f90 のサブルーチン

名称	引数	機能
jc_start	なし	カップラーの初期化
jc_set_grid	なし	グリッドの設定
jc_set_initial_time	character(len=14):: start_time integer:: delta_t	積分時間の設定：引数 start_time は積分開始時刻を表す 14 文字の文字列、delta_t は時間ステップを表す整数。
jc_set_current_time	character(len=14):: sdate integer:: delt, tstep	現在時刻の設定：引数 sdate は積分開始時刻を表す 14 文字の文字列、delt は時間ステップ幅、tstep は現在のステップ数。
jc_mksend_M	real*8:: fsh(:,:), flq(:,:), flw(:,:), alb(:,:)	MM5 へ送信するデータ作成：引数 fsh, flq, flw, alb は倍精度実数の送信データ。
jc_exchange1_M	integer:: istampi1 real*8:: flat(:,:), flon(:,:), ghgt(:,:), rou(:,:), tbotm(:,:), twatr(:,:), sty(:,:), uty(:,:), hwi(:,:), ui(:,:), vi(:,:), ti(:,:), qi(:,:), co2i(:,:)	MM5 からの初期データ受信：引数 istampi はデータ交換のフラッグ。他の引数は受信するデータを表す倍精度実数。
jc_exchange2_M	integer:: istampi1, itime real*8:: timer(:), phpi(:,:,:), rsoli(:,:,:), rinfi(:,:,:), rri(:,:,:), ui(:,:,:), vi(:,:,:), ti(:,:,:), qi(:,:,:), co2i(:,:,:)	MM5 とのデータ送受信:引数 istampi はデータ交換のフラッグ、itime と timer は時間のフラッグ。他の引数は受信するデータを表す倍精度実数。
jc_coupling_end	なし	カップリングの終了

jc_solveg_rivers.f90

RIVERS とのカップリング用ファイル jc_solveg_rivers.f90 内に記述されているサブルーチンについて、名称・引数・機能を表 C5 に示す。

表 C5　jc_solveg_rivers.f90 のサブルーチン

名称	引数	機能
jc_start_r	なし	RIVERS とのカップリング開始
jc_exchange1_R	integer:: iflgfeed1 real*8:: timet, rouw, dzs(:), arf(:,:), hwb(:,:), er(:,:,:), wfr(:,:)	RIVERS からのデータ受信：引数 iflgfeed1 はデータ交換のフラッグ。arf, hwb, er, wfr は受信するデータを表す倍精度実数。
jc_exchange2_R	real*8:: timet, rouw, dzs(:), hw(:,:,:), res(:,:), wf(:,:,:), et(:,:,:), prt(:,:,:), eb(:,:,:)	RIVERS へのデータ送信
Jc_graph_out	real*8:: timet, rouw, dzs(:), ts(:,:,:), hw(:,:,:), res(:,:)	Graph 出力用のデータ送信：現状では、格子設定用データ以外では、2 次元の地表水と 3 次元の土壌温度及び水分量データのみ送信。必要に応じて送信データを追加。

jc_solveg_mm5.f90 及び jc_solveg_rivers.f90 を用いるファイル

jc_solveg_mm5.f90 及び jc_solveg_rivers.f90 のサブルーチンをコールしているファイルは、“$(SOLVEG_HOME)/SRC” ディレクトリの main.f、ppread.f、ptint.f、solveg.f の 4 つであり、表 C6 に各ファイルでコールしているサブルーチンをまとめた。また、図 C3 に SOLVEG の時間積分ループとカップリング関連サブルーチンのコールフローを示す。時間積分ループは、メインプログラム “main.f” の do 1001～end do の間である。時間積分ループの前に、サブルーチン PREAD で MM5 から初期データを受信するサブルーチン jc_exchange1_M がコールされる。時間積分ループの中でサブルーチン TIMEINT がコールされ、jc_exchange2_M によって MM5 とのデータ交換が行われる。またサブルーチン SOLVEG の中で、jc_exchange1_R と jc_exchange2_R によって RIVERS とのデータ交換が行われる。

表 C6　SOLVEG でカップリングサブルーチンをコールしているファイル

SOLVEG のファイル名	コールしているサブルーチン名
main.f	jc_start, jc_start_r, jc_set_grid, jc_set_initial_time, jc_set_current_time, jc_mksend_M, jc_coupling_end
ppread.f	jc_exchange1_M
ptint.f	jc_exchange2_M
solveg.f	jc_exchange1_R, jc_exchange2_R, jc_graph_out

```
main(main.f)
  call jc_start
  call jc_start_r
  call jc_set_grid
  call jc_set_initial_time
  call jc_set_current_time
  call PREAD(ppread.f)
            call jc_exchange1_M
                 recv (latitude～specific_humid)
  end subroutine PREAD

  DO 1001 L = 1, NTIME
    call jc_set_current_time
    call TIMEINT(ptint.f)
              call jc_exchange2_M
                   if (TIMER>0) send (heat_flux～albedo)
                   recv (sureface_press～specific_humid)
    end subroutine TIMEINT

    call SOLVEG
               call jc_exchange1_R
                   if (NSTEP==0) recv (river_ar)
                   recv (river_w2～river_qu2)
               call jc_exchange2_R
                   NSTEP++
                   send (send_w1～send_qu5)
    end subroutine SOLVEG
  END DO
  call jc_coupling_end
end main
```

図 C3　SOLVEG の時間積分ループとカップリング関連ルーチンの呼び出し

（4）RIVERS

jc_rivers.f90

カップリング用ルーチンは、jc_rivers.f90 内に含まれる。各ルーチンの名称・引数・機能を表 C7 に示す。

表 C7　jc_rivers.f90 のサブルーチン

名称	引数	機能
jc_start	なし	カップラーの初期化
jc_set_grid	なし	グリッド情報の設定
jc_set_initial_time	なし	積分時刻情報の設定
jc_set_current_time	character(len=*):: cdate integer:: delta_t	現在時刻の設定：引数 cdate は時間増加前の現在時刻を表す 14 文字の文字列、delta_t は時間ステップ。
jc_coupling_end	なし	カップリングの終了
jc_exchange1_S	real*8:: timet, arf(:,:), w2(:,:), dw0(:,:), dw1(:,:), qu2(:,:)	SOLVEG へのデータ送信
jc_exchange2_S	real*8:: timet, w1(:,:), w0(:,:), qd1(:,:), sroot(:,:), qra(:,:), qu5(:,:)	SOLVEG からのデータ受信

jc_rivers.f90 を用いるファイル

jc_rivers.f90 のサブルーチンをコールしているファイルは、“$(RIVERS_HOME)/SRC” ディレクトリの main.f90 と cal_main.f90 の 2 つである。main.f90 では jc_start、jc_set_grid、jc_set_initial_time、jc_coupling_end を、cal_main.f90 では jc_set_current_time、jc_exchange1_S、jc_exchange2_S をコールしている。

図 C4 に RIVERS の時間積分ループとカップリング関連サブルーチンのコールフローを示す。時間積分ループはサブルーチン cal_main (cal_main.f90) の 100 continue～go to 100 の間である。このループの中で SOLVEG とのデータ交換を行うサブルーチン jc_exchange1_S と jc_exchange2_S がコールされる。

```
main(river.f90)
  call jc_start
  call jc_set_grid
  call jc_set_initial_time
  call calculate_m(calculate_m.f90)
       call calmain(cal_main.f90)
          100 continue
              call jc_set_current_time
              call jc_exchange1_S
                   if (NSTEP==0) send (river_ar)
                   send (river_u2～river_qu2)
              call jc_exchange2_S
                   NSTEP++
                   recv (recv_w1～recv_qu5)
          goto 100
       end subroutine calmain
  end subroutine calculate_m
  call jc_coupling_end
end main
```

図 C4　RIVERS の時間積分ループとカップリング関連ルーチンの呼び出し

（5）WW3

w3couple.ftn

カップリング用のサブルーチンを格納したファイルは“$(WW3_HOME)/ftn”ディレクトリのw3couple.ftnである。ファイルw3couple.ftn内に記述されているサブルーチンについて、名称・引数・機能を表C8に示す。

表C8　w3couple.ftnのサブルーチン

名称	引数	機能
jc_start	integer:: ni, nj, delta_t character(len=*):: cdate	カップラーの初期化：引数ni, njは東西、南北のグリッド数。cdateは積分開始時刻、delta_tは積分時間ステップ。データ補間に用いる格子点緯度経度情報を基盤50（実行ディレクトリのlonlat.inp：ww3_g.shによりPOMの入力データから作成）から入力。
jc_set_current_time	character(len=14):: sdate integer:: istep, delta_t	現在時刻と領域番号の設定：引数sdateは積分開始時刻を表す14文字の文字列、istepは現在のステップ数、delta_tは積分時間間隔。カップラーのサブルーチンjc_IncTimeで現在時刻を計算し、カップラーに送信。
jc_send_data_2d	real:: dt(ni,nj) character(len=*):: name integer:: ni, nj	2次元データの送信：引数nameはデータの名称でww3.data.confに記述された名称と対応させる。
jc_recv_data_2d	real:: dt(ni,nj) character(len=*):: name integer:: ni, nj	2次元データの受信：引数nameはデータの名称でww3.data.confに記述された名称と対応させる。
jc_coupling_end	なし	カップリングの終了：必ずコールする。

w3couple.ftnを用いるファイル

w3couple.ftnのサブルーチンをコールしているファイルは、“$(WW3_HOME)/ftn/ww3_shel.ftn”と“$(WW3_HOME)/ftn/w3wavemd.ftn”の2つのファイルである。ww3_shel.ftnはメインプログラムであり、ここでjc_start及びjc_coupling_endの各サブルーチンをコールしている。w3wavemd.ftnは時間積分を行うサブルーチンW3WAVEを格納したファイルである。W3WAVE中にデータ交換を行う部分があり、ここでjc_ww3_set_current_time、jc_send_data_2d、jc_recv_data_2dをコールしている。また、カップラーにわたす時間ステップを取得するためにプログラムファイルw3iogrmd.ftnの一部を以下のように修正した。

修正前：IF (INXOUT .NE. 'GRID') THEN

```
修正後：IF ( INXOUT .EQ. 'GRID' ) THEN
              READ (NDSM,END=801,ERR=802,IOSTAT=IERR)                    &
              DTCFL, DTCFLI, DTMAX, DTMIN, DMIN, CFLTM
        ELSE
```

図 C5 に WW3 の時間積分ループとカップリング関連サブルーチンのコールフローを示す。時間積分ループはサブルーチン W3WAVE (w3wavemd.ftn) の do IT = ～ end do の間である。このループの中で、MM5、POM とデータ送受信を行っている。

```
main(ww3_shel.ftn)
  call jc_start
  call W3WAVE (w3wavemd.ftn)
    do IT = IT0, NT
      call jc_set_current_time
      call jc_send_data_2d
        send (tbrx, tbry, wMAG1)
      call jc_recv_data_2d
        recv (ua10, va10, aMAG1, ua, va, oMAG1, el)

    end do
  end subroutine W3WAVE
  call jc_coupling_end
end main
```

図 C5　WW3 の時間積分ループとカップリング関連ルーチンの呼び出し

格子生成プリプロセッサ

水平計算格子及び水深等の格子情報を生成するプリプロセッサ ww3_grid の実行について記述する。本結合システムにおいては、WW3 の入力は MM5 及び POM の出力を用いることを前提とし、格子生成以外のプリプロセッサを用いた入力作成は行わないものとする。格子生成については、他のモデルと独立して設定可能であるが、ここでは POM と同じ水平格子及び入力データを用いることで簡便化を図った。

格子生成の実行は、設定ファイル“$(WW3_HOME)/RUN2/ww3_grid.inp”の格子定義（図 C6）を POM の格子に合わせて設定し、同じディレクトリのシェルスクリプト ww3_g.sh を実行することにより行う。この際、POM の入力データ depth_xy.dat が既に作成され所定のディレクトリ“$(POM_HOME)/area1/input”に格納されている必要がある。シェルスクリプトの実行により、水深入力ファイル bottom.inp 及び緯度経度情報ファイル lonlat.inp がまず作成され、ww3_grid により WW3 計算の入力ファイル mod_def.ww3（バイナリファイル）が作成される。緯度経度情報ファイルは、カップリング初期化サブルーチン jc_start 内での緯度経度入力に用いられる。

```
$ Define grid ---------------------------------------------------- $
$ Four records containing :
$   1 NX, NY. As the outer grid lines are always defined as land
$     points, the minimum size is 3x3.
$   2 Grid increments SX, SY (degr.or m) and scaling (division) factor.
$     If NX*SX = 360., latitudinal closure is applied.
$   3 Coordinates of (1,1) (degr.) and scaling (division) factor.
$   4 Limiting bottom depth (m) to discriminate between land and sea
$     points, minimum water depth (m) as allowed in model, unit number
$     of file with bottom depths, scale factor for bottom depths (mult.),
$     IDLA, IDFM, format for formatted read, FROM and filename.
$       IDLA : Layout indicator :
$                    1    : Read line-by-line bottom to top.
$                    2    : Like 1, single read statement.
$                    3    : Read line-by-line top to bottom.
$                    4    : Like 3, single read statement.
$       IDFM : format indicator :
$                    1    : Free format.
$                    2    : Fixed format with above format descriptor.
$                    3    : Unformatted.
$       FROM : file type parameter
$                 'UNIT' : open file by unit number only.
$                 'NAME' : open file by name and assign to unit.
$
$ Example for longitude-latitude grid (switch !/LLG), for Cartesian
$ grid the unit is meters (NOT km).
$
     56      81
   22240.   23840.   1.
    33.0     12.5    1.
   -0.1 2.50   9   -1. 2 1 '(....)' 'NAME' 'bottom.inp'
$
$ If the above unit number equals 10, the bottom data is read from
$ this file and follows below (no intermediate comment lines allowed).
$
   0   0   0   0   0
   0   0   0   0   0
   0   0   0   0   0
   0   0   0   0   0
   0   0   0   0   0
$
```

図 C6　WW3 の格子設定ファイル ww3_grid.inp の格子定義部分

This is a blank page.

国際単位系（SI）

表１．SI 基本単位

基本量	SI 基本単位	
	名称	記号
長さ	メートル	m
質量	キログラム	kg
時間	秒	s
電流	アンペア	A
熱力学温度	ケルビン	K
物質量	モル	mol
光度	カンデラ	cd

表２．基本単位を用いて表されるSI組立単位の例

組立量	SI 基本単位	
	名称	記号
面積	平方メートル	m^2
体積	立法メートル	m^3
速さ，速度	メートル毎秒	m/s
加速度	メートル毎秒毎秒	m/s^2
波数	毎メートル	m^{-1}
密度（質量密度）	キログラム毎立法メートル	kg/m^3
質量体積（比体積）	立法メートル毎キログラム	m^3/kg
電流密度	アンペア毎平方メートル	A/m^2
磁界の強さ	アンペア毎メートル	A/m
（物質量の）濃度	モル毎立方メートル	mol/m^3
輝度	カンデラ毎平方メートル	cd/m^2
屈折率	（数の）1	1

表５．SI 接頭語

乗数	接頭語	記号	乗数	接頭語	記号
10^{24}	ヨタ	Y	10^{-1}	デシ	d
10^{21}	ゼタ	Z	10^{-2}	センチ	c
10^{18}	エクサ	E	10^{-3}	ミリ	m
10^{15}	ペタ	P	10^{-6}	マイクロ	μ
10^{12}	テラ	T	10^{-9}	ナノ	n
10^{9}	ギガ	G	10^{-12}	ピコ	p
10^{6}	メガ	M	10^{-15}	フェムト	f
10^{3}	キロ	k	10^{-18}	アト	a
10^{2}	ヘクト	h	10^{-21}	ゼプト	z
10^{1}	デカ	da	10^{-24}	ヨクト	y

表３．固有の名称とその独自の記号で表されるSI組立単位

組立量	SI 組立単位			
	名称	記号	他のSI単位による表し方	SI基本単位による表し方
平面角	ラジアン(a)	rad		$m \cdot m^{-1}=1$(b)
立体角	ステラジアン(a)	sr(c)		$m^2 \cdot m^{-2}=1$(b)
周波数	ヘルツ	Hz		s^{-1}
力	ニュートン	N		$m \cdot kg \cdot s^{-2}$
圧力，応力	パスカル	Pa	N/m^2	$m^{-1} \cdot kg \cdot s^{-2}$
エネルギー，仕事，熱量	ジュール	J	N・m	$m^2 \cdot kg \cdot s^{-2}$
工率，放射束	ワット	W	J/s	$m^2 \cdot kg \cdot s^{-3}$
電荷，電気量	クーロン	C		s・A
電位差（電圧），起電力	ボルト	V	W/A	$m^2 \cdot kg \cdot s^{-3} \cdot A^{-1}$
静電容量	ファラド	F	C/V	$m^{-2} \cdot kg^{-1} \cdot s^4 \cdot A^2$
電気抵抗	オーム	Ω	V/A	$m^2 \cdot kg \cdot s^{-3} \cdot A^{-2}$
コンダクタンス	ジーメンス	S	A/V	$m^{-2} \cdot kg^{-1} \cdot s^3 \cdot A^2$
磁束	ウエーバ	Wb	V・s	$m^2 \cdot kg \cdot s^{-2} \cdot A^{-1}$
磁束密度	テスラ	T	Wb/m^2	$kg \cdot s^{-2} \cdot A^{-1}$
インダクタンス	ヘンリー	H	Wb/A	$m^2 \cdot kg \cdot s^{-2} \cdot A^{-2}$
セルシウス温度	セルシウス度(d)	℃		K
光束	ルーメン	lm	cd・sr(c)	$m^2 \cdot m^{-2} \cdot cd=cd$
照度	ルクス	lx	lm/m^2	$m^2 \cdot m^{-4} \cdot cd=m^{-2} \cdot cd$
（放射性核種の）放射能	ベクレル	Bq		s^{-1}
吸収線量，質量エネルギー分与，カーマ	グレイ	Gy	J/kg	$m^2 \cdot s^{-2}$
線量当量，周辺線量当量，方向性線量当量，個人線量当量，組織線量当	シーベルト	Sv	J/kg	$m^2 \cdot s^{-2}$

(a) ラジアン及びステラジアンの使用は、同じ次元であっても異なった性質をもった量を区別するときの組立単位の表し方として利点がある。組立単位を形作るときのいくつかの用例は表４に示されている。

(b) 実際には、使用する時には記号rad及びsrが用いられるが、習慣として組立単位としての記号“１”は明示されない。

(c) 測光学では、ステラジアンの名称と記号srを単位の表し方の中にそのまま維持している。

(d) この単位は、例としてミリセルシウス度m℃のようにSI接頭語を伴って用いても良い。

表４．単位の中に固有の名称とその独自の記号を含むSI組立単位の例

組立量	SI 組立単位		
	名称	記号	SI 基本単位による表し方
粘度	パスカル秒	Pa・s	$m^{-1} \cdot kg \cdot s^{-1}$
力のモーメント	ニュートンメートル	N・m	$m^2 \cdot kg \cdot s^{-2}$
表面張力	ニュートン毎メートル	N/m	$kg \cdot s^{-2}$
角速度	ラジアン毎秒	rad/s	$m \cdot m^{-1} \cdot s^{-1}=s^{-1}$
角加速度	ラジアン毎平方秒	rad/s^2	$m \cdot m^{-1} \cdot s^{-2}=s^{-2}$
熱流密度，放射照度	ワット毎平方メートル	W/m^2	$kg \cdot s^{-3}$
熱容量，エントロピー	ジュール毎ケルビン	J/K	$m^2 \cdot kg \cdot s^{-2} \cdot K^{-1}$
質量熱容量（比熱容量），質量エントロピー	ジュール毎キログラム毎ケルビン	J/(kg・K)	$m^2 \cdot s^{-2} \cdot K^{-1}$
質量エネルギー（比エネルギー）	ジュール毎キログラム	J/kg	$m^2 \cdot s^{-2} \cdot K^{-1}$
熱伝導率	ワット毎メートル毎ケルビン	W/(m・K)	$m \cdot kg \cdot s^{-3} \cdot K^{-1}$
体積エネルギー	ジュール毎立方メートル	J/m^3	$m^{-1} \cdot kg \cdot s^{-2}$
電界の強さ	ボルト毎メートル	V/m	$m \cdot kg \cdot s^{-3} \cdot A^{-1}$
体積電荷	クーロン毎立方メートル	C/m^3	$m^{-3} \cdot s \cdot A$
電気変位	クーロン毎平方メートル	C/m^2	$m^{-2} \cdot s \cdot A$
誘電率	ファラド毎メートル	F/m	$m^{-3} \cdot kg^{-1} \cdot s^4 \cdot A^2$
透磁率	ヘンリー毎メートル	H/m	$m \cdot kg \cdot s^{-2} \cdot A^{-2}$
モルエネルギー	ジュール毎モル	J/mol	$m^2 \cdot kg \cdot s^{-2} \cdot mol^{-1}$
モルエントロピー，モル熱容量	ジュール毎モル毎ケルビン	J/(mol・K)	$m^2 \cdot kg \cdot s^{-2} \cdot K^{-1} \cdot mol^{-1}$
照射線量（X線及びγ線）	クーロン毎キログラム	C/kg	$kg^{-1} \cdot s \cdot A$
吸収線量率	グレイ毎秒	Gy/s	$m^2 \cdot s^{-3}$
放射強度	ワット毎ステラジアン	W/sr	$m^4 \cdot m^{-2} \cdot kg \cdot s^{-3}=m^2 \cdot kg \cdot s^{-3}$
放射輝度	ワット毎平方メートル毎ステラジアン	$W/(m^2 \cdot sr)$	$m^2 \cdot m^{-2} \cdot kg \cdot s^{-3}=kg \cdot s^{-3}$

表６．国際単位系と併用されるが国際単位系に属さない単位

名称	記号	SI 単位による値
分	min	1 min=60s
時	h	1h =60 min=3600 s
日	d	1 d=24 h=86400 s
度	°	1° =(π/180) rad
分	’	1’ =(1/60)° =(π/10800) rad
秒	”	1” =(1/60)’ =(π/648000) rad
リットル	l、L	1l=1 $dm^3=10^{-3}m^3$
トン	t	$1t=10^3$ kg
ネーパ	Np	1Np=1
ベル	B	1B=(1/2)ln10(Np)

表７．国際単位系と併用されこれに属さない単位で SI単位で表される数値が実験的に得られるもの

名称	記号	SI 単位であらわされる数値
電子ボルト	eV	$1eV=1.60217733(49) \times 10^{-19}J$
統一原子質量単位	u	$1u=1.6605402(10) \times 10^{-27}kg$
天文単位	ua	$1ua=1.49597870691(30) \times 10^{11}m$

表８．国際単位系に属さないが国際単位系と 併用されるその他の単位

名称	記号	SI 単位であらわされる数値
海里		1海里=1852m
ノット		1ノット=1海里毎時=(1852/3600)m/s
アール	a	1 a=1 $dam^2=10^2m^2$
ヘクタール	ha	1 ha=1 $hm^2=10^4m^2$
バール	bar	1 bar=0.1MPa=100kPa=1000hPa=10^5Pa
オングストローム	Å	1 Å=0.1nm=10^{-10}m
バーン	b	1 b=100$fm^2=10^{-28}m^2$

表９．固有の名称を含むCGS組立単位

名称	記号	SI 単位であらわされる数値
エルグ	erg	1 erg=10^{-7} J
ダイン	dyn	1 dyn=10^{-5}N
ポアズ	P	1 P=1 dyn・s/cm^2=0.1Pa・s
ストークス	St	1 St =1$cm^2/s=10^{-4}m^2/s$
ガウス	G	1 G ^10^{-4}T
エルステッド	Oe	1 Oe ^(1000/4π)A/m
マクスウェル	Mx	1 Mx ^10^{-8}Wb
スチルブ	sb	1 sb =1$cd/cm^2=10^4cd/m^2$
ホト	ph	1 ph=10^4lx
ガル	Gal	1 Gal =1$cm/s^2=10^{-2}m/s^2$

表10．国際単位に属さないその他の単位の例

名称	記号	SI 単位であらわされる数値
キュリー	Ci	1 Ci=3.7×10^{10}Bq
レントゲン	R	1 R = 2.58×10^{-4}C/kg
ラド	rad	1 rad=1cGy=10^{-2}Gy
レム	rem	1 rem=1 cSv=10^{-2}Sv
X線単位		1X unit=1.002×10^{-4}nm
ガンマ	γ	1γ=1 nT=10^{-9}T
ジャンスキー	Jy	1 Jy=$10^{-26}W \cdot m^{-2} \cdot Hz^{-1}$
フェルミ		1 fermi=1 fm=10^{-15}m
メートル系カラット		1 metric carat = 200 mg = 2×10^{-4}kg
トル	Torr	1 Torr = (101 325/760) Pa
標準大気圧	atm	1 atm = 101 325 Pa
カロリー	cal	
ミクロン	μ	1 μ =1μm=10^{-6}m

（第７版，1998年改訂）