JAERI-Data/Code
2003-021

NEW METHOD FOR MODEL COUPLING USING STAMPI: APPLICATION TO THE COUPLING OF ATMOSPHERE MODEL(MM5) AND LAND-SURFACE MODEL(SOLVEG)

December 2003

Haruyasu NAGAI

日本原子力研究所
Japan Atomic Energy Research Institute

本レポートは，日本原子力研究所が不定期に公刊している研究報告書です。
入手の問合わせは，日本原子力研究所研究情報部研究情報課（〒319-1195 茨城県那珂郡東海村）あて，お申し越しください。なお，このほかに財団法人原子力弘済会資料センター（〒319-1195 茨城県那珂郡東海村日本原子力研究所内）で複写による実費頒布をおこなっております。

This report is issued irregularly.
Inquiries about availability of the reports should be addressed to Research Information Division, Department of Intellectual Resources, Japan Atomic Energy Research Institute, Tokai-mura, Naka-gun, Ibarakiken 319-1195, Japan.

編集兼発行　日本原子力研究所

New Method for Model Coupling Using Stampi: Application to the Coupling of Atmosphere Model (MM5) and Land-surface Model (SOLVEG)

Haruyasu NAGAI

Department of Environmental Sciences
Tokai Research Establishment
Japan Atomic Energy Research Institute
Tokai-mura, Naka-gun, Ibaraki-ken

(Received November 7, 2003)

A new method to couple atmosphere and land-surface models using the message passing interface (MPI) was proposed to develop an atmosphere-land model for studies on heat, water, and material exchanges around the land surface. A non-hydrostatic atmospheric dynamic model of Pennsylvania State University and National Center for Atmospheric Research (PUS/NCAR-MM5) and a detailed land surface model (SOLVEG) including the surface-layer atmosphere, soil, and vegetation developed at Japan Atomic Energy Research Institute (JAERI) are used as the atmosphere and land-surface models, respectively. Concerning the MPI, a message passing library named Stampi developed at JAERI that can be used between different parallel computers is used. The models are coupled by exchanging calculation results by using MPI on their independent parallel calculations. The modifications for this model coupling are easy, simply adding some modules for data exchanges to each model code without changing each model's original structure. Moreover, this coupling method is flexible and allows the use of independent time step and grid interval for each model.

Keywords: Model Coupling, MPI, Atmosphere Model, Land-surface Model, Parallel Calculation, MM5, SOLVEG, Stampi

Stampiを用いた新しいモデル結合手法:

大気モデル(MM5)と陸面モデル(SOLVEG)の結合への適用

日本原子力研究所東海研究所環境科学研究部
永井 晴康

(2003年11月7日受理)

地表付近の熱、水及び物質交換の研究に適用する目的で、大気—陸面結合モデルの開発を行っており、並列計算通信ライブラリMPI(Message Passing Interface)を用いたモデル結合手法を考案した。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が開発したメソスケール非静力大気モデル(PSU/NCAR-MM5)を、陸面モデルには原研で開発した接地層大気、土壌及び植生を含む詳細なモデル(SOLVEG)を用いた。また、MPIとしては、原研で開発した異機種並列計算機間通信ライブラリ(Stampi)を用いた。各モデルの独立した平行計算において、それぞれの計算結果をMPIにより交換し合うことにより結合を行う。結合のためのモデル修正は、各モデルの構成を変更することなく、データ交換を行うモジュールを計算コードに追加するだけである。さらに、本結合手法は、それぞれのモデルの時間ステップや格子間隔を任意に設定するなど柔軟な適用も可能である。

東海研究所:〒319−1195 茨城県那珂郡東海村白方白根2−4

Contents

目　次

This is a blank page.

1. Introduction

A coupled atmosphere and land-surface model has been developed to study heat, water, and material circulations around the land surface. A non-hydrostatic atmospheric dynamic model of Pennsylvania State University and National Center for Atmospheric Research (PSU/NCAR-MM5)[1] and a detailed land surface model (SOLVEG)[2)3)] including the surface-layer atmosphere, soil, and vegetation developed at Japan Atomic Energy Research Institute (JAERI) are used as the atmosphere and land-surface models, respectively.

In this coupling, a new method is applied to combine the models. The method uses the function called MPMD (Multiple Program Multiple Data) of the parallel programming library MPI (Message Passing Interface), which allows parallel computers to run a parallel program with different executables. The atmosphere model MM5 and land-surface model SOLVEG are coupled by exchanging calculation results by using MPI on their independent parallel calculations. Concerning the MPI, a message passing library named Stampi[4] developed at JAERI is used. Stampi is a library to enhance the function of MPI, allowing MPI to be used between different parallel computers. The modifications of models for this coupling are easy, simply adding some modules for data exchanges to each model code without changing each model's original structure. This feature is helpful to make a model code coupling with a community model like MM5, which is updated regularly. Moreover, this coupling method is flexible and allows the use of independent time step and grid interval for each model.

In this paper, the coupling method and test calculations to validate the method are described.

2. Overview of Models

The non-hydrostatic mesoscale atmosphere model MM5[1] is a community model having many users all over the world, and is used for many purposes, even for the official weather forecast by some countries. It has many useful functions such as nesting calculations, four-dimensional data assimilation, and many options of parameterizations for cloud micro-physics, cumulus cloud, planetary boundary layer (PBL), radiation, and land-surface scheme.

The land-surface model SOLVEG[2)3)] consists of one-dimensional multi-layer sub-models for the surface-layer atmosphere, soil, and vegetation, and a radiation scheme for the transmission of solar and long-wave radiation in the canopy. It simulates diurnal variation and seasonal changes of variables in the surface-layer atmosphere, soil, and vegetation canopy, and exchanges of energy and water among these systems, by using meteorological data of the surface-layer atmosphere as top boundary conditions. For coupling with the atmosphere model, one-dimensional model variables are expanded to three-dimensional ones whose horizontal coordinates coincide with those of atmosphere model. However, no interactions in

horizontal directions are considered in the model.

3. Coupling Method

To couple an atmosphere model and a land-surface model, it is general to make a single model code by incorporating the land-surface model into the atmosphere model. However, a totally different way of the model coupling is proposed in this study. MM5 and SOLVEG calculations are carried out as independent tasks for different processors, and their coupling is achieved by exchanging outputs of models as MPI communication between processors as schematically shown in Fig. 1. This type of parallel calculation is called MPMD. MM5 calculation starts at first. Then, it invokes SOLVEG calculation in its initialization processes. After the PBL calculation in the first time step of MM5 time integration, MM5 processor sends SOLVEG the initial values and the first boundary conditions for SOLVEG: air pressure, radiation, precipitation, wind speed, temperature, humidity, etc, which are mostly calculated in PBL calculation. With these inputs, SOLVEG calculation proceeds for the same time interval as MM5 and sends its results to MM5: skin temperature, surface heat and vapor fluxes, and albedo. MM5 receives these values before PBL calculation in the next time step and uses them as the land-surface boundary conditions for the PBL process. After PBL calculation, MM5 sends SOLVEG the top boundary condition for SOLVEG for the next time step, and the same cycle of processes are carried out repeatedly until the end of calculation. The time step of SOLVEG calculation is usually smaller than that of MM5, and several time steps are carried out for SOLVEG calculation during a single time step of MM5 calculation.

This coupling method is flexible and allows us to use the nesting functions of MM5. Figure 2 shows the flow of coupling calculation and data exchanges for two-domain, two-way nesting calculation. MM5 nesting calculations for outer and inner domains (dom. 1 and dom. 2 in Fig. 2, respectively) are carried out successively in one time step of the outer domain, and three time step for the inner domain in this procedure. On the other hand, SOLVEG calculations for large and local domains (dom.1 and dom. 2 in Fig. 2, respectively) are processed as independent tasks by different processors. Data exchanges between corresponding domains of MM5 and SOLVEG take place independently. This method is also applicable to more complex nesting domains.

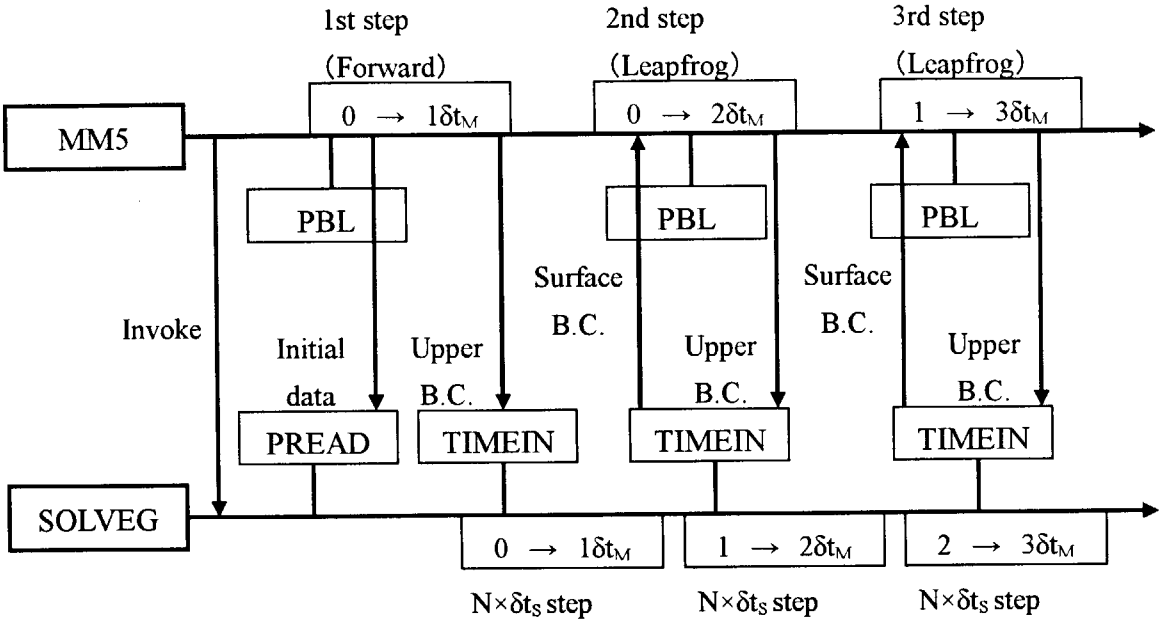

Fig. 1 Data exchanges between coupled models. PREAD and TIMEINT represent the SOLVEG routines for the initial and boundary conditions, respectively. The parameters δt_M and δt_S are time increments of MM5 and SOLVEG calculations, respectively.

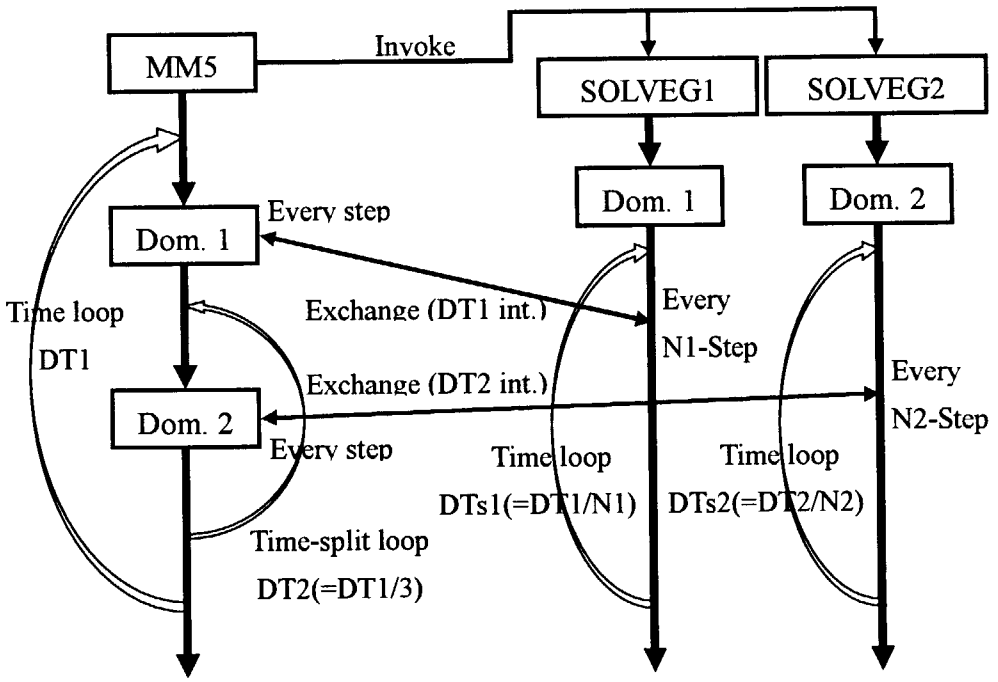

Fig. 2 Data exchanges for nesting calculations. Parameters DT1/2 and DTs1/2 represent time increments of MM5 and SOLVEG for Domain-1/2, respectively.

This coupling method also has flexibility in use of parallel computers, by adopting the Stampi[4] as the MPI library. While the MPI usually establishes communications among processors in a single parallel computer, JAERI has developed the advanced version of MPI, Stampi, to achieve the communication between different parallel machines. The utilization of the Stampi for the MPI library in the coupling programs allows us to execute two models on different computers. By executing two models on different computers, the load balance of calculations for two models can be optimized. MM5 usually has higher computational cost than SOLVEG, and its program code is vectorized as well as parallelized. Therefore, the combination of computers with different spec, for example, a vector parallel computer for MM5 and scalar parallel computer for SOLVEG, is effectively used to execute this coupling model.

The modification of each model code for this coupling is simple and easy, just adding some data exchange routines and put some sentences in the original model code that call the routines. Each model code can keep its original structure. Therefore, we can use compile options, namelist, shell-script, and input dataset of each model in their original form. Details of the data exchange routines and modification of each model are described in the following section. This coupling method also has flexibility to use different resolution of grid for each model by interpolating one model's data to the other model's grid.

4. Program Code

In this section, data exchange routines added to MM5 and SOLVEG and modifications of each model are described in detail. Sample program codes of these routines are presented in appendix. These sample codes are used for the coupling of MM5 on Fujitsu vector-parallel computer VPP5000 and SOLVEG on Fujitsu scalar-parallel computer AP3000.

4.1 MM5

The module (stampi_solveg.F) to exchange data with SOLVEG consists of seven subroutines coded by FORTRAN90. Since two subroutines are called in other subroutines, five subroutines are called from original MM5 routines. Positions to put sentences calling these subroutines are shown in Figs. 3 and 4.

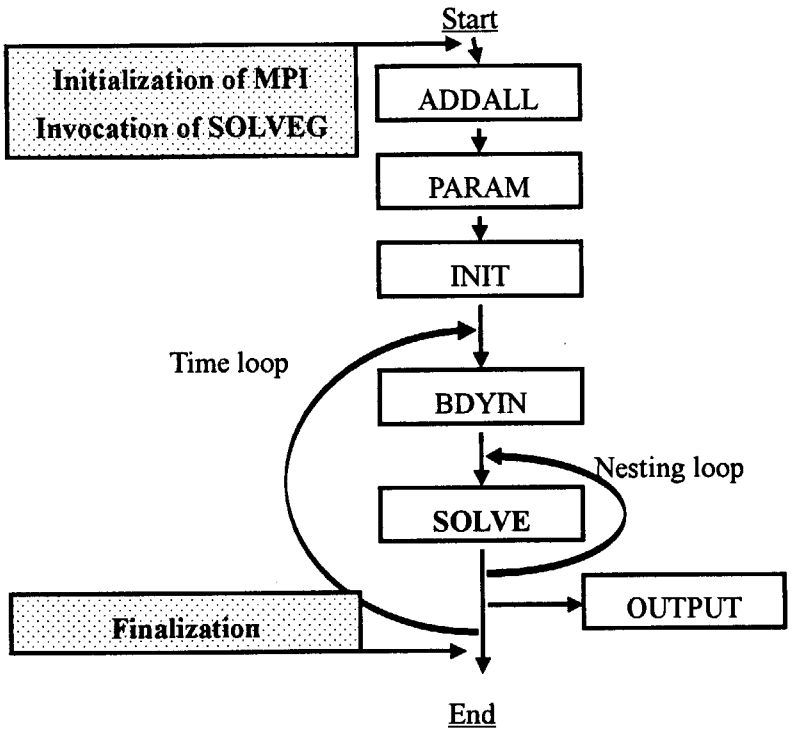

Fig. 3 Calculation flow of MM5 main routine and positions to put coupling routines.

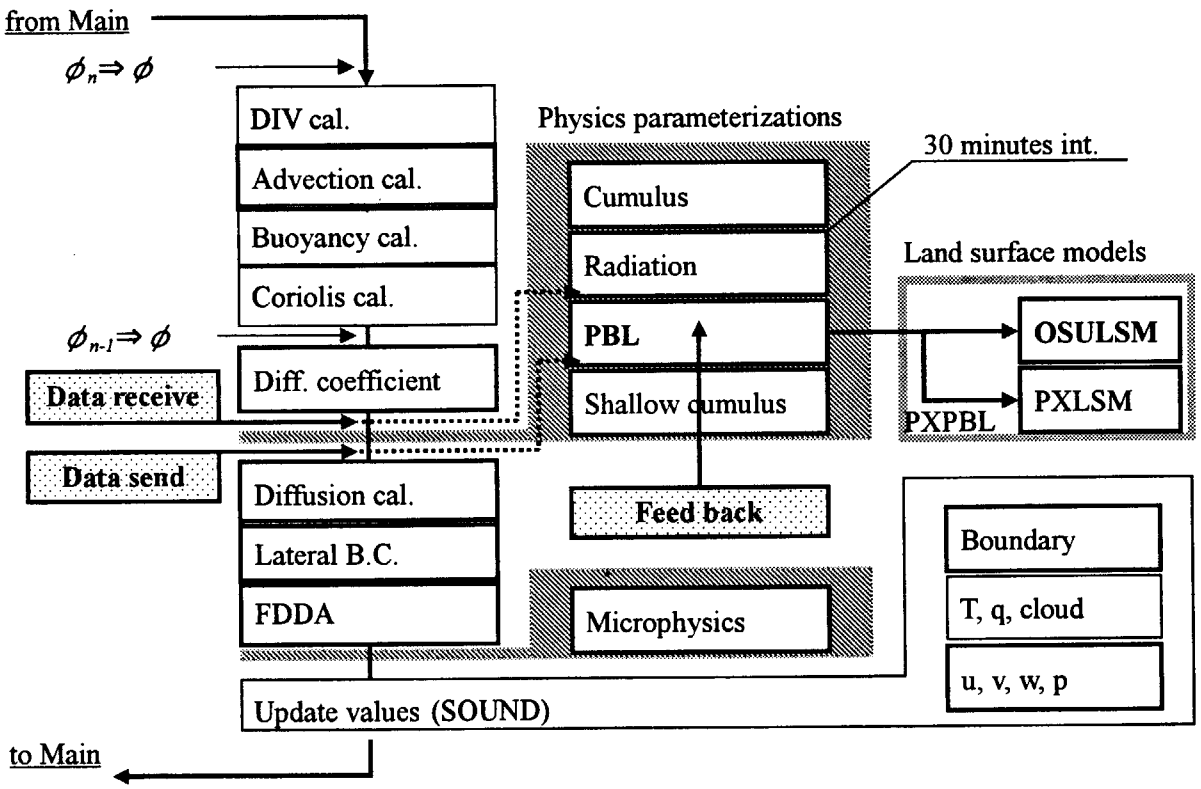

Fig. 4 Calculation flow of MM5 dynamics calculation routine "SOLVE" and positions to put coupling routines.

1) Initialization routine: STAMPI_START

This routine is called at the beginning of the MM5 main routine (Fig. 3) and initializes the environment of MPI communication, and calls the "process invocation routine". The following MPI variables and those defined coupling conditions are initialized.

<MPI variables>

RANK	integer	MPI rank
SIZE	integer	number of processors
SERVER	integer	node number of server processor
CLIENT(NNEST)	integer	node numbers of client processors

<Variables for coupling conditions>

NSTAMPI(NNEST)	integer	time step number
IFLGFEED(NNEST)	integer	feed back on/off flag (1/0)
IFLGOFF(NNEST)	integer	offline data output on/off flag (1/0)
ICPLNEST(NNEST)	integer	coupling on/off flag (1/0)

where NNEST is the number of nesting domains. These variables are defined as common variables and used by all the coupling routines of MM5. The MPI variables are also used in the coupling routines of SOLVEG.

2) Finalization routine: STAMPI_END

This routine is call at the end of the MM5 main routine (Fig. 3) and finalizes the MPI communication after calling the "communication finalization routine".

3) Process invocation routine: INVOKE_CLIENT (N)

This routine invokes SOLVEG execution for each nesting domain by calling MPI routine "MPI_COMM_SPAWN". The Stampi allows us to invoke processors on different computers by giving the machine's host name and user account. Other variables for this routine are the same as those for usual MPI.

<Input variable>

N	integer	domain number

4) Communication finalization routine: SERVER_FINALIZE (N)

This routine checks the finalization of SOLVEG process for each domain.

<Input variable>

N	integer	domain number

5) Feed back routine: STAMPI_FBACK (INEST, J, IST, IEN, HFX, QFX, TGB, ALB)

This routine is called in the MM5 PBL routine (Fig. 4) and feeds back data received from

SOLVEG to the surface boundary values of the PBL calculation of MM5. The data feed back processes in this routine are carried out if the feed back flag is on (IFLGFEED = 1). Data from SOLVEG are passed through a common sentence from the data receive routine. This routine is called just after the original procedure for land surface calculation in the PBL routine.

<Input variables>

INEST	integer	domain number
J	integer	index number of longitudinal grid
IST	integer	start index number of latitudinal grid
IEN	integer	end index number of latitudinal grid
HFX(MIX, MJX)	real	sensible heat flux (W m^{-2})
QFX(MIX, MJX)	real	vapor flux (kg m^{-2} s^{-1})
TGB(MIX, MJX)	real	surface skin temperature (K)
ALB(MIX, MJX)	real	albedo (0.0 to 1.0)

<Variables via common sentence>

RECV_SH(MIX, MJX)	real	sensible heat flux (W m^{-2})
RECV_QF(MIX, MJX)	real	vapor flux (kg m^{-2} s^{-1})
RECV_TS(MIX, MJX)	real	surface skin temperature (K)
RECV_AL(MIX, MJX)	real	albedo (0.0 to 1.0)

where MIX and MJX are the maximum numbers of latitudinal and longitudinal grids, respectively. These parameters are defined by including MM5 parameter file "parame.incl". Data feed back is achieved by overwriting MM5 variables; HFX, GFX, TGB, and ALB, by SOLVEG calculations; RECV_SH, RECV_QF, RECV_TS, and RECV_AL, respectively.

6) Data receive routine: STAMPI_EXCHANGE1 (INEST, NY, NX, KTAU, NTRAD, GSW, ALB)

This routine is called just before the PBL calculation in the MM5 dynamics calculation routine "SOLVE" (Fig. 4) and received data from SOLVEG by calling MPI routine "MPI_RECV". Received data from SOLVEG are sensible heat flux, vapor flux, surface skin temperature, and albed. As described before, these variables are passed to the feed back routine through the common sentence. In this routine, one of the data sent to SOLVEG, the global solar radiation is prepared and passed to the data send routine through a common sentence.

<Input variables>

INEST	integer	domain number
NY	integer	number of latitudinal grid
NX	integer	number of longitudinal grid
KTAU	integer	calculation time step number

NTRAD	integer	time interval for radiation calculation (min)
GSW(MIX, MJX)	real	global solar radiation (W m^{-2})
ALB(MIX, MJX)	real	albedo (0.0 to 1.0)

<Variable via common sentence>

SEND_RS(MIX, MJX, NNEST)	integer	global solar radiation (W m^{-2})

7) Data send routine: STAMPI_EXCHANGE2 (INEST, NY, NX, DT, XTIME, PTOP,
XLAT, XLONG, RHO1, TMN,
TSS, MAVAIL, PSB, PP3D, GLW,
RAINP, UA10, VA10, TA10, QA10)

This routine is called just after the PBL calculation in the MM5 dynamics calculation routine "SOLVE" (Fig. 4) and sends date to SOLVEG by calling MPI routine "MPI_SEND". The grid parameters and initial values of SOLVEG are sent once at the first time step. These variables are latitude, longitude, air density, soil bottom temperature, sea surface temperature, soil type, vegetation type, and soil moisture. Parameters soil type and vegetation type are defined by including MM5 parameter file "soilp.incl". Top boundary values sent to SOLVEG every time step are surface pressure, solar radiation, long-wave radiation, precipitation intensity, u-wind, v-wind, air temperature, and specific humidity. If the offline data output flag is on (IFLGOFF = 1), input data for SOLVEG offline calculation are output.

<Input variables>

INEST	integer	domain number
NY	integer	number of latitudinal grid
NX	integer	number of longitudinal grid
XTIME	real	calculation time (min)
PTOP	real	pressure of model top (Pa)
XLAT(MIX, MJX)	real	latitude of grid point (deg)
XLONG(MIX, MJX)	real	longitude of grid point (deg)
RHO1(MIX, MJX, MKX)	real	density of surface air (kg m^{-3})
TMN(MIX, MJX)	real	soil temperature (K)
TSS(MIX, MJX)	real	sea surface temperature (K)
MAVAIL(MIX, MJX)	real	volumetric soil water content (m^3 m^{-3})
PSB(MIX, MJX)	real	surface pressure of reference state (Pa)
PP3D(MIX, MJX, MKX)	real	pressure perturbation from reference value (Pa)
GLW(MIX, MJX)	real	downward long-wave radiation (W m^{-2})
RAINP(MIX, MJX)	real	precipitation intensity (cm in time step)
UA10(MIX, MJX)	real	u-wind at 10 m (m s^{-1})
VA10(MIX, MJX)	real	v-wind at 10 m (m s^{-1})
TA10(MIX, MJX)	real	temperature at 10 m (K)

QA10(MIX, MJX) real specific humidity at 10 m (kg kg^{-1})

where MKX is the number of maximum vertical grid of MM5.

4.2 SOLVEG

The module (stampi_mm5.f) to exchange data with MM5 consists of six subroutines coded by FORTRAN77. Since one subroutine is called in other subroutines, five subroutines are called from original SOLVEG routines. Positions to put sentences calling these subroutines are shown in Fig. 5.

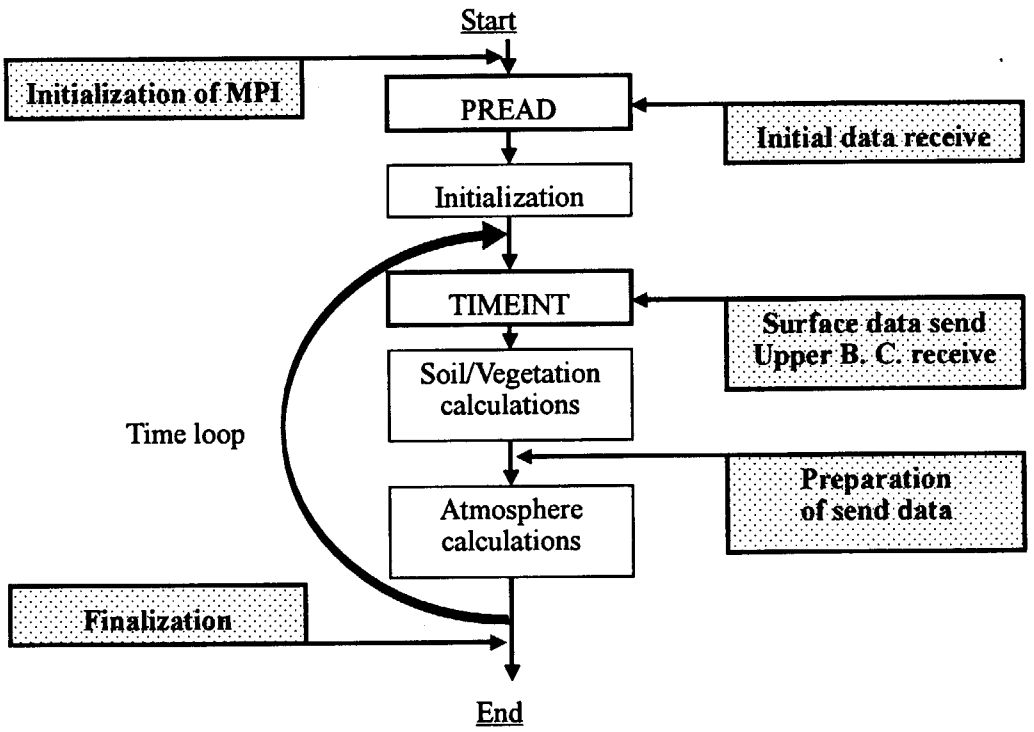

Fig. 5 Calculation flow of SOLVEG main routine and positions to put coupling routines.

1) Initialization routine: STAMPI_START

This routine is called at the beginning of the SOLVEG main routine (Fig. 5) and initializes the environment of MPI communication. The following variable determines on/off of coupling and is common among the coupling routines of SOLVEG.

<Variable for coupling on/off>

ISTAMPI integer coupling on/off flag (1/0)

2) Finalization routine: STAMPI_END

This routine is called at the end of the SOLVEG main routine (Fig. 5) and finalizes the MPI communication after calling the "communication finalization routine".

3) Communication finalization routine: REC_KILL_SIG

This routine makes confirmation of finalization between MM5 processes.

4) Send data make routine: STAMPI_MKSEND (FSH, FLQ, FLW, ALB)

This routine is called in the SOLVEG main routine (Fig. 5) and prepares dataset sent to MM5. These data are passed to "data send receive routine" through a common sentence. In this routine, the surface skin temperature is calculated from the upward long-wave radiation at the canopy top.

<Input variables>

FSH(NX, NY)	real*8	sensible heat flux (W m^{-2})
FLQ(NX, NY)	real*8	vapor flux (kg m^{-2} s^{-1})
FLW(NX, NY)	real*8	upward long-wave radiation (W m^{-2})
ALB(NX, NY)	real*8	albedo (0.0 to 1.0)

<Variables via common sentence>

SEND_SH(NX, NY)	real	sensible heat flux (W m^{-2})
SEND_QF(NX, NY)	real	vapor flux (kg m^{-2} s^{-1})
SEND_TS(NX, NY)	real	surface skin temperature (K)
SEND_AL(NX, NY)	real	albedo (0.0 to 1.0)

Where NX and NY are longitudinal and latitudinal grid numbers and defined by including SOLVEG parameter file "Inclnum".

5) Initial data receive routine: STAMPI_EXCHANGE1 (ISTAMPI1, FLAT, FLON, ROU, TBOTM, TWATR, STY, VTY, HWI, PHPI, RSOLI, RINFI, RRI, UI, VI, TI, QI)

This routine is called in the initial data read routine "PREAD" (Fig. 5) and receives grid parameters and initial values from MM5.

<Input variables>

ISTAMPI1	integer	coupling on/off flag (1/0)
FLAT(NX, NY)	real	latitude of grid point (deg)
FLON(NX, NY)	real	longitude of grid point (deg)
ROU(NX, NY)	real	density of surface air (kg m^{-3})
TBOTM(NX, NY)	real	bottom soil temperature (K)
TWATER(NX, NY)	real	sea surface temperature (K)
STY(NX, NY)	real	number of soil type (1.0 to 16.0)

VTY(NX, NY)	real	number of vegetation type (1.0 to 24.0)
HWI(NX, NY)	real	volumetric soil water content (m^3 m^{-3})
PHPI(NX, NY, 0:1)	real	air pressure (hPa)
RSOLI(NX, NY, 0:1)	real	global solar radiation (W m^{-2})
RINFI(NX, NY, 0:1)	real	downward long-wave radiation (W m^{-2})
RRI(NX, NY, 0:1)	real	precipitation intensity (mm h^{-1})
UI(NX, NY, 0:1)	real	u-wind at 10 m (m s^{-1})
VI(NX, NY, 0:1)	real	v-wind at 10 m (m s^{-1})
TI(NX, NY, 0:1)	real	temperature at 10 m (K)
QI(NX, NY, 0:1)	real	specific humidity at 10 m (kg kg^{-1})

6) Data send receive routine: STAMPI_EXCHANGE2 (ISTAMPI1, ITIME, TIMER, PHPI, RSOLI, RINFI, RRI, UI, VI, TI, QI)

This routine is called in the boundary data read routine "TIMEINT" (Fig. 5) and sends data to MM5. Also, it receives top boundary data from MM5. As described before, data sent to MM5 are prepared in other routine and passed through the common sentence.

<Input variables>

ISTAMPI1	integer	coupling on/off flag (1/0)
ITIME	integer	calculation time (sec)
TIMER(0:1)	real	time interval of data exchange (sec)
PHPI(NX, NY, 0:1)	real	surface air pressure (hPa)
RSOLI(NX, NY, 0:1)	real	global solar radiation (W m^{-2})
RINFI(NX, NY, 0:1)	real	downward long-wave radiation (W m^{-2})
RRI(NX, NY, 0:1)	real	precipitation intensity (mm h^{-1})
UI(NX, NY, 0:1)	real	u-wind at 10 m (m s^{-1})
VI(NX, NY, 0:1)	real	v-wind at 10 m (m s^{-1})
TI(NX, NY, 0:1)	real	temperature at 10 m (K)
QI(NX, NY, 0:1)	real	specific humidity at 10 m (kg kg^{-1})

4.3 Compile and Run

To compile and run the coupled model, the following changes are needed for model compilation and machine settings.

1) Compile

To use the Stampi library, compile commands for FORTRAN and C compilers are set as "jmpif90" and "jmpicc" instead of "frt" and "cc", respectively. For MM5, this change needs to be done for the file "MM5_root/configure.user". Also, the land surface model option for

OSU-LSM (ISOIL = 2) needs to be chosen in this file. For Fujitsu VPP machine, the MPI link options are changed to use the Stampi library by the following lines.

```
************************************
setenv MPIINCDIR "stampi_root/include"
setenv MPILIBS "stampi_root/lib"
************************************
```

Other changes for MM5 are made for make-files so as to include the coupling module. Since the coupling module is installed in the same directory as MM5 main-routine, the make-file "MM5_root/Run/Makefile" is changed. If the distributed memory parallel calculation option (MPP) is used, the RSL make-file "MM5_root/MPP/RSL/Makefile.RSL" also needs to be modified. These changes are as follows.

<MM5_root/Run/Makefile>
The following lines are modified to include the coupling module "stampi_solveg.F".

```
****** Original *****************
OBJS =¥
        mm5.o
SRC =¥
        mm5.i
SRCF =¥
        mm5.f
**********************************

****** Modified *****************
OBJS =¥
        mm5.o   stampi_solveg.o
SRC =¥
        mm5.i   stampi_solveg.i
SRCF =¥
        mm5.f   stampi_solveg.f
**********************************
```

<MM5_root/MPP/RSL/Makefile.RSL>
The following compile rule for the coupling module "stampi_solveg.F" is added to this file.

```
**********************************************************
#stampi
stampi_solveg.o : stampi_solveg.F
        $(MFC) -c $(FCFLAGS) $(INCLUDES) $*.F 2> $*.lis
**********************************************************
```

For SOLVEG, only the FORTRAN compile command is changed from "frt" to "jmpif90", and no other modifications are needed.

2) Run

The file ".rhosts" of each machine needs to be edited to allow the remote-shell access from the other machine. A sample shell-scripts for MM5 and SOLVEG executions are as follows.

<Sample shell-script for MM5>

```
************************************************************
#!/bin/csh -f
#########################################################
# STAMPI coupling: MM5 and SOLVEG
#########################################################
#@$-C  STAMPI_MM5_SOLVEG
#@$-r  RUN_mm5
#@$-q  vpa
#@$-lP 2
#########################################################
setenv VPP_MBX_SIZE 10000000
setenv JMPIRUN /stampi_root/bin/jmpirun
setenv MP_STDOUTMODE=0
cd $QSUB_WORKDIR
####### run MM5 #######
timex $JMPIRUN -np 2 mm5.mpp >& $QSUB_WORKDIR/mm5.print.out
************************************************************
```

< Sample shell-script for SOLVEG>

```
**************************************************************
#!/bin/csh -f
#######################################################
# Ver. 3: Grid calculation for whole grid
#######################################################
#@$-C  solveg     # program name
#@$-q  apl        # submit batch job class
#@$-r  SOLgo      # batch request name
#######################################################
set hmdir="/xxx/SOLVEG"
set otdir=${hmdir}"/OUTstampi1"
cd ${hmdir}
#go
timex zsolveg_stampi1.go < EXpara_stampi1 >& ${otdir}/outlist
**************************************************************
```

SOLVEG executable and shell-script are prepared for each calculation domain if the nesting calculation is used.

5. Test Calculations

Test calculations for two-domain, two-way nesting were carried out to validate this coupling method. Calculations were carried out for the Kanto Region of Japan. For the meteorological inputs of MM5, Grid Point Value by Regional Spectral Model (GPV/RSM) of Japan Meteorological Agency (JMA) was used. Since there are no soil moisture data, uniform volumetric soil water content (0.3 m^3 m^{-3}) was given as the initial soil water condition. For SOLVEG calculations, an assumed land-surface (winter wheat for all land) was used so as to show clearly the impacts of feed back from SOLVEG on MM5 calculation.

1) Calculation conditions

Conditions of test calculations are as follows.

General:

- Area — Kanto Region of Japan
- Center of domain — latitude: 36.5°N, longitude: 139.5°E
- Period — 24 hours from 00 UTC on 17 July, 2001

Input data:

- GPV/RSM of JMA

- Resolution surface: 0.2°×0.25°, upper: 0.4°×0.5°, 20 pressure levels
- Time interval surface: 1 hour, upper: 3 hours

MM5 options:

- Cumulus None
- Microphysics Reisner2
- Radiation Cloud
- PBL MRF
- LSM OSU-LSM

MM5 outer domain:

- Area 324 km square
- Grid 55×55×23
- Resolution 6 km
- Time increment 15 s

MM5 inner domain:

- Area 96 km square
- Grid 49×49×23
- Resolution 2 km
- Time increment 5 s

SOLVEG: (same for both domains)

- Atmosphere 10 layers, 10 m above the surface
- Soil 7 layers, 2 m below the land surface
- Vegetation lower 5 layers of atmosphere
- Time increment 5 s
- Assumed land winter wheat (height: 0.7 m, LAI: 5.6) for all land

Fujitsu scalar-parallel computer Primepower was used to execute 24-hour calculations. As shown in Fig. 2, at least three processors (one for MM5 and two for SOLVEG domains) are necessary for this case. Since the distributed memory parallel calculation of MM5 (MPP) can be used in this coupling method, both single and MPP calculations for MM5 were tested. Moreover, this coupling method has two coupling modes: one-way and two-way couplings. In one-way coupling, only MM5 sends data to SOLVEG and no feed back from SOLVEG is considered in MM5. In two-way coupling, the dynamical link between MM5 and SOLVEG is realized by mutual data exchanges between them. Both coupling modes were examined, and the validity of this method was confirmed by reasonable model outputs.

2) Results

Calculated skin temperatures at the time of 24 hours from the initial time, which is 09 JST (Japan Standard Time), are shown in Figs. 6 and 7 for the original MM5 and two-way coupling cases, respectively. The skin temperature by the coupling model is lower than that by

the original MM5. Although figures are not shown for sensible and latent heat fluxes, these values are also different between the original MM5 and the coupling model. While the sensible heat flux decreased, the latent heat flux increased in the calculation by the coupled model. It is because that SOLVEG calculates the land surface heat and water budget for the assumed land (winter wheat for all land). Since the efficiency of evapotranspiration is high at winter wheat field, the latent heat flux becomes higher than that at other land used in the original MM5. This higher latent heat flux resulted in the decrease in the sensible heat flux and skin temperature. Also, these results are consistent between calculations for outer and inner domains.

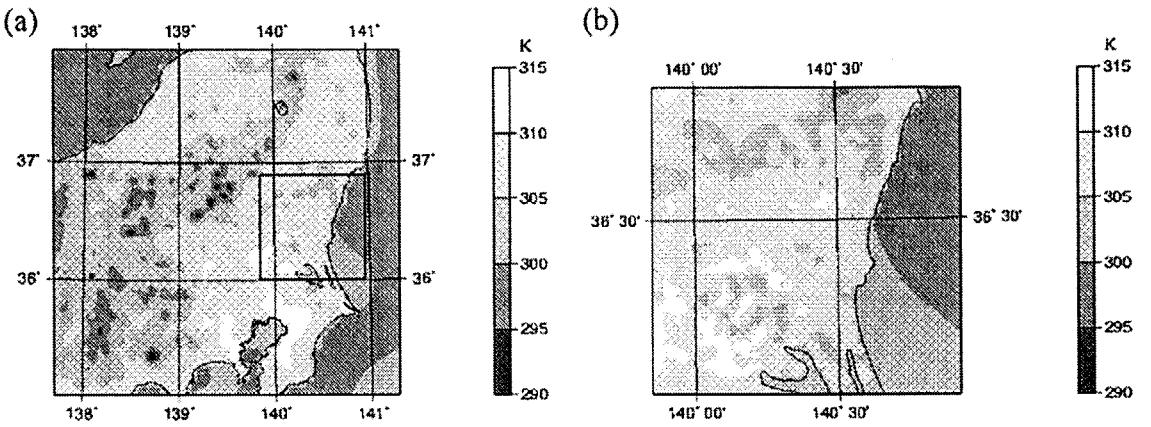

Fig. 6 Skin temperature distribution at 24-hour from the initial time (09 JST) calculated by the original MM5 for (a) outer domain and (b) inner domain.

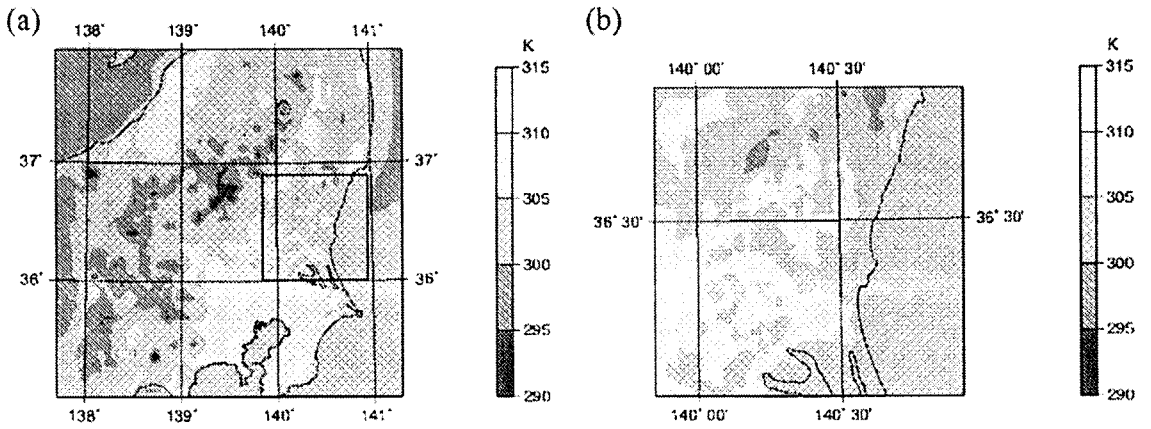

Fig. 7 Same as in Fig. 6 but calculated by MM5 in two-way coupling with SOLVEG.

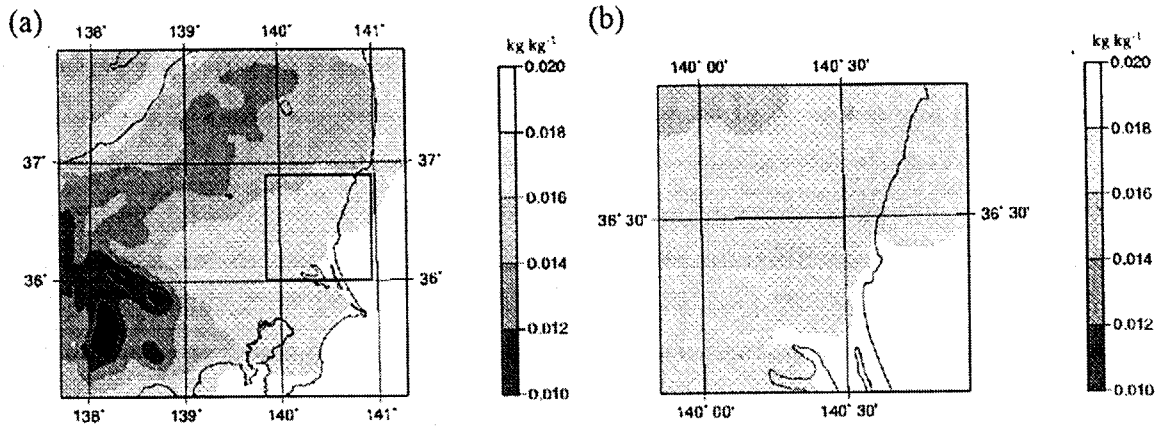

Fig. 8 Same as in Fig. 6 but for specific humidity at the atmospheric surface-layer calculated by the original MM5.

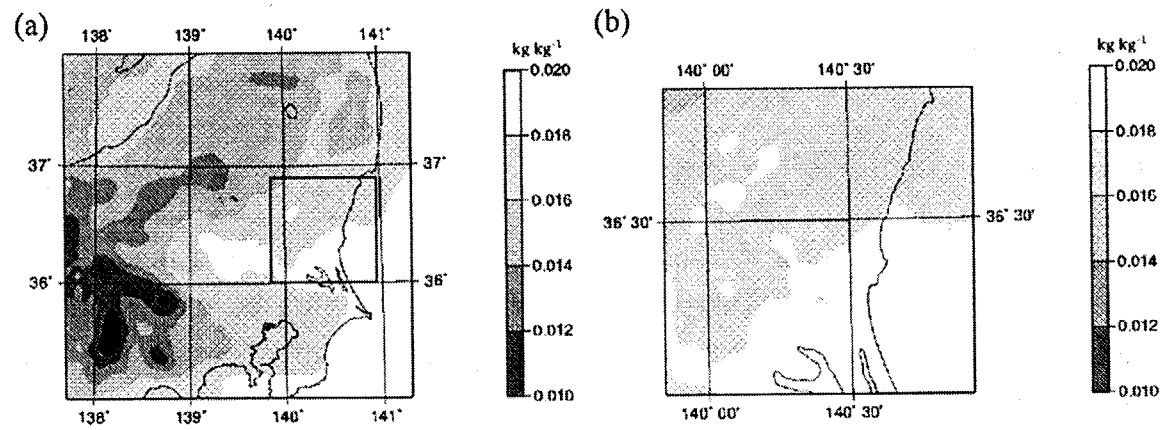

Fig. 9 Same as in Fig. 8 but calculated by MM5 in two-way coupling with SOLVEG.

Figures 8 and 9 show the specific humidity at the atmospheric surface-layer calculated by the original MM5 and the coupled model, respectively. It is clearly shown that the specific humidity calculated by the coupled model increased as the result of the higher evapotranspiration. This result indicates that the feed back from SOLVEG to MM5 was correctly carried out by using the coupling method. Furthermore, the change in the skin temperature and specific humidity at the atmospheric surface-layer affected the generation of cloud and, consequently, radiation flux. The influence of these changes is also seen in the land surface calculations. These results indicate that this coupling method can realize the dynamical coupling of MM5 and SOLVEG as if SOLVEG was incorporated into MM5 like the land-surface model OSU-LSM, which is included in the original MM5 code as one of the subroutines for physical processes.

6. Summary

The new method to couple models by using the Stampi, which is the extended version of MPI, was proposed and it was applied to the coupling of atmospheric model MM5 and land surface model SOLVEG. In this coupling, two models are calculated independently as parallel tasks and necessary data for each model are exchanged by MPI. By using the Stampi for MPI, the model coupling between different parallel machines are achieved. In test calculations, reasonable model outputs proved the validity of this method and its applicability to develop this kind of complex coupling model including interactions among the atmosphere, soil, and vegetation.

It is planed that more complex model coupling is achieved by using this coupling method. The coupled model includes atmosphere, ocean, and land models, and each model has a physical (dynamical) part and material transport part. This coupling system can predict the material transport continuously in the atmospheric, oceanic, and terrestrial environment, and contribute for understanding of processes of material transport and making solution of environmental problems.

References

1) Grell, G. A., J. Dudhia, and D. R. Stauffer, 1994: A description of the fifth-generation Penn State/NCAR Mesoscale Model (MM5), NCAR Tech. Note, NCAR/TN-398+STR, 122pp.
2) Nagai, H., 2002: Validation and sensitivity analysis of a new atmosphere-soil-vegetation model, J. Appl. Meteor., 41, 160-176.
3) Nagai, H., 2003: Validation and sensitivity analysis of a new atmosphere-soil-vegetation model. Part II: Impacts on in-canopy latent heat flux over a winter wheat field determined by detailed calculation of canopy radiation transmission and stomatal resistance, J. Appl. Meteor., 42, 434-351.
4) Imamura, T., H. Koide, and H. Takemiya, 1998: Stampi: A message passing library for distributed parallel computing – User's guide (in Japanese), JAERI-Data/Code 98-034, Japan Atomic Energy Research Institute, 29pp.

Appendix: Sample Source Codes for Coupling Modules

1) MM5 coupling module source code: stampi_solveg.F

```
***** stampi_solveg.F ***************************************************************
c+++++STAMPI+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c     Coupling to SOLVEG
c           Initialization
c           Finalization
c           Subroutines for Init and Final
c           Feed back in PBL
c           Data exchange 1 and 2
c
c      If this file is changed, update mm5.F file then make.
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c*****Initialization
      subroutine STAMPI_START
c
      IMPLICIT NONE
c
      integer N, NNEST
      parameter( NNEST = 2 )
c
      include 'mpif.h'
      integer RANK, SIZE, SERVER, ERR, CLIENT(NNEST)
     &      , STAT(MPI_STATUS_SIZE)
      common /MPI/ RANK, SIZE, SERVER, ERR, CLIENT, STAT
c
      integer NSTAMPI(NNEST), IFLGFEED(NNEST)
     &      , IFLGOFF(NNEST), ICPLNEST(NNEST)
      common /NSTAMPI/ NSTAMPI, IFLGFEED, IFLGOFF, ICPLNEST
c
c*****Set parameters for each domain
c-----Coupling ==> ICPLNEST = 1:yes , 0:no
      ICPLNEST(1) = 1
      ICPLNEST(2) = 1
c-----Feed back ==> IFLGFEED = 1:yes , 0:no
      IFLGFEED(1) = 1
      IFLGFEED(2) = 1
c-----Data output for off-line cal. ==> IFLGOFF = 1:yes , 0:no
      IFLGOFF(1) = 0
      IFLGOFF(2) = 0
c*****End of parameter setting
c
c-----STAMPI initialization
      call MPI_COMM_SIZE(MPI_COMM_WORLD, SIZE, ERR)
      call MPI_COMM_RANK(MPI_COMM_WORLD, RANK, ERR)
      call MPI_COMM_GET_PARENT(SERVER, ERR)
      write(6,*) 'STAMPI Initialize'
      write(6,*) 'MPI_COMM_WORLD = ',MPI_COMM_WORLD,'   RANK = ',RANK
     &          ,'   SIZE = ',SIZE
      write(6,*) 'SERVER = ',SERVER
      if( SERVER .eq. MPI_COMM_NULL ) then
        write(6,*) 'SERVER(TRUE) = ', SERVER
```

```
      end if
c
c-----Spawn client: SOLVEG
      do N = 1 , NNEST
        NSTAMPI(N) = 0
        if( ICPLNEST(N) .eq. 1 ) call INVOKE_CLIENT( N )
      enddo
c-----STAMPI
c
      return
      end
c
c
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c*****Finalization
      subroutine STAMPI_END
c
      IMPLICIT NONE
c
      integer N, NNEST
      parameter( NNEST = 2 )
c
      include 'mpif.h'
      integer RANK, SIZE, SERVER, ERR, CLIENT(NNEST)
     &      , STAT(MPI_STATUS_SIZE)
      common /MPI/ RANK, SIZE, SERVER, ERR, CLIENT, STAT
c
      integer NSTAMPI(NNEST), IFLGFEED(NNEST)
     &      , IFLGOFF(NNEST), ICPLNEST(NNEST)
      common /NSTAMPI/ NSTAMPI, IFLGFEED, IFLGOFF, ICPLNEST
c
      if( RANK .eq. 0 ) then
c-----STAMPI send finalize_sig to client
      write(6,*) 'STAMPI SERVER_FINALIZE'
      do N = 1 , NNEST
        if( ICPLNEST(N) .eq. 1 ) call SERVER_FINALIZE( N )
      enddo
      end if
      call MPI_BARRIER(MPI_COMM_WORLD,ERR)
c
c-----STAMPI finalize
      write(6,*) 'STAMPI MPI_FINALIZE'
c-----STAMPI
c
      return
      end
c
c
c+++++STAMPI subroutines++++++++++++++++++++++++++++++++++++++++++++++++++++++
c*****Spawn client
      subroutine INVOKE_CLIENT( N )
c
      IMPLICIT NONE
c
      integer N, NNEST
      parameter( NNEST = 2 )
```

```
c
      include 'mpif.h'
      integer RANK, SIZE, SERVER, ERR, CLIENT(NNEST)
     &      , STAT(MPI_STATUS_SIZE)
      common /MPI/ RANK, SIZE, SERVER, ERR, CLIENT, STAT
      integer INFO
      character SCRIPT*13
c
c
      write(6,*) 'STAMPI Spawn client'
      write(6,*) 'MPI_COMM_WORLD = ',MPI_COMM_WORLD,'  RANK = ',RANK
      write(6,*) 'SERVER = ',SERVER
c
      write(SCRIPT,'(A,I1,A)') 'go_stampi',N,'.sh'
      write(6,*) 'Spawn client: SOLVEG-',SCRIPT
      call MPI_INFO_CREATE(INFO,ERR)
      call MPI_INFO_SET(INFO,"host","xxxx",ERR)
      call MPI_INFO_SET(INFO,"user","xxxx",ERR)
      call MPI_INFO_SET(INFO,"path","/xxxx/SOLVEG",ERR)
      call MPI_INFO_SET(INFO,"wdir","/xxxx/SOLVEG",ERR)
      call MPI_COMM_SPAWN(SCRIPT,MPI_ARGV_NULL,1,INFO,0
     &     ,MPI_COMM_WORLD,CLIENT(N),MPI_ERRCODES_IGNORE,ERR)
      write(6,*) 'CLIENT',N,' = ',CLIENT(N),'  ERR = ',ERR
c
      return
      end
c
c
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c*****Send finalize_sig to client
      subroutine SERVER_FINALIZE( N )
c
      IMPLICIT NONE
c
      integer N, NNEST
      parameter( NNEST = 2 )
c
      include 'mpif.h'
      integer RANK, SIZE, SERVER, ERR, CLIENT(NNEST)
     &      , STAT(MPI_STATUS_SIZE)
      common /MPI/ RANK, SIZE, SERVER, ERR, CLIENT, STAT
      integer BUF
c
      write(6,*) 'STAMPI Finalize to client'
      write(6,*) 'MPI_COMM_WORLD = ',MPI_COMM_WORLD,'  RANK = ',RANK
      write(6,*) 'SERVER = ',SERVER
c
      BUF = -1
      call MPI_SEND(BUF,1,MPI_INTEGER,0,0,CLIENT(N),ERR)
      call MPI_RECV(BUF,1,MPI_INTEGER,0,0,CLIENT(N),STAT,ERR)
c
      return
      end
c
c
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
c*****Feed back in PBL
      subroutine STAMPI_FBACK( INEST, J, IST, IEN, HFX, QFX, TGB, ALB )
c
      IMPLICIT NONE
c
      integer N, NNEST
      parameter( NNEST = 2 )
c
#     include <parame.incl>
      integer I, J, IST, IEN, INEST
      real    HFX(MIX,MJX), QFX(MIX,MJX), TGB(MIX,MJX), ALB(MIX,MJX)
c
      real*4 RECV_SH(MIX,MJX), RECV_QF(MIX,MJX)
      real*4 RECV_TS(MIX,MJX), RECV_AL(MIX,MJX)
      common /STAMPI/ RECV_SH, RECV_QF, RECV_TS, RECV_AL
      integer NSTAMPI(NNEST), IFLGFEED(NNEST)
     &      , IFLGOFF(NNEST), ICPLNEST(NNEST)
      common /NSTAMPI/ NSTAMPI, IFLGFEED, IFLGOFF, ICPLNEST
c
      N = INEST
      if( ICPLNEST(N) .ne. 1 ) then
        return
      end if
c
      if( IFLGFEED(N) .eq. 1 .and. NSTAMPI(N) .gt. 0 ) then
        do I = IST , IEN
          HFX(I,J) = RECV_SH(I,J)
          QFX(I,J) = RECV_QF(I,J)
          TGB(I,J) = RECV_TS(I,J)
          ALB(I,J) = RECV_AL(I,J)
        enddo
      end if
c
      return
      end
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c*****Data exchange 1
      subroutine STAMPI_EXCHANGE1( INEST , NY , NX , KTAU , NTRAD
     &                           , GSW , ALB )
c
      IMPLICIT NONE
c
      integer N, NNEST
      parameter( NNEST = 2 )
c
#     include <parame.incl>
      integer I, J, KTAU, NTRAD, INEST, JS, JE, LEN, NPE
     &      , JSI(0:4,NNEST), JEI(0:4,NNEST)
      real    GSW(MIX,MJX), ALB(MIX,MJX)
      real    DATA1(MIX,MJX,0:3), DATA2(MIX,MJX,0:3)
c
      include 'mpif.h'
      integer RANK, SIZE, SERVER, ERR, CLIENT(NNEST)
     &      , STAT(MPI_STATUS_SIZE)
      common /MPI/ RANK, SIZE, SERVER, ERR, CLIENT, STAT
```

```
c
      integer NX, NY, SUMERR, NXY
      real*4 SEND_RS(MJX,MIX,NNEST)
      real*4 RECV_SH(MIX,MJX), RECV_QF(MIX,MJX)
      real*4 RECV_TS(MIX,MJX), RECV_AL(MIX,MJX)
      real*4 RECV_VAL(MJX*MIX)
      common /STAMPI/ RECV_SH, RECV_QF, RECV_TS, RECV_AL
      common /DSTAMPI/ SEND_RS
      common /ISTAMPI/ JSI, JEI
      integer NSTAMPI(NNEST), IFLGFEED(NNEST)
     &      , IFLGOFF(NNEST), ICPLNEST(NNEST)
      common /NSTAMPI/ NSTAMPI, IFLGFEED, IFLGOFF, ICPLNEST
c
c
      write(6,*) 'STAMPI Data exchange 1: Receive INEST =', INEST
c
      N = INEST
      if( ICPLNEST(N) .ne. 1 ) then
        return
      end if
c
      if( MOD(KTAU,NTRAD) .EQ. 0 ) then
c-----SOLAR RADIATION (W M-2)
        LEN = MIX*MJX
        call MPI_GATHER(GSW,LEN,MPI_REAL4
     &       ,DATA1(1,1,RANK),LEN,MPI_REAL4,0,MPI_COMM_WORLD,ERR)
        call MPI_GATHER(ALB,LEN,MPI_REAL4
     &       ,DATA2(1,1,RANK),LEN,MPI_REAL4,0,MPI_COMM_WORLD,ERR)
        call MPI_BARRIER(MPI_COMM_WORLD,ERR)
        if( RANK .eq. 0 ) then
        JSI(0,INEST) = 4
        do NPE = 0 , SIZE-1
          if( NSTAMPI(INEST) .eq. 0 ) then
          do J = 4 , MJX
            if( DATA2(1,J,NPE) .ne. 1.0 ) JE = J
          enddo
          JEI(NPE,INEST)   = JSI(NPE,INEST) + JE - 4
          JSI(NPE+1,INEST) = JEI(NPE,INEST) - 1
          end if
          JS  = JSI(NPE,INEST)
          JE  = JEI(NPE,INEST)
          do J = JS , JE
            do I = 1 , MIX
              DATA1(I,J,0) = DATA1(I,4+J-JS,NPE)
              DATA2(I,J,0) = DATA2(I,4+J-JS,NPE)
            enddo
          enddo
        enddo
        do J = 1 , NY-1
          do I = 1 , NX-1
            SEND_RS(I,J,N) = DATA1(J,I+3,0)/(1.0-DATA2(J,I+3,0))
          enddo
          SEND_RS(NX,J,N) = SEND_RS(NX-1,J,N)
        enddo
        do I = 1 , NX
          SEND_RS(I,NY,N) = SEND_RS(I,NY-1,N)
```

```
          enddo
          end if
        end if
c
c
        if( NSTAMPI(N) .gt. 0 ) then
          SUMERR = 0
c-----SENSIBLE HEAT FLUX (W M-2)                                        *
          if( RANK .eq. 0 ) then                                        *
          call MPI_RECV(RECV_VAL,NX*NY,MPI_REAL4,0,0,CLIENT(N),STAT,ERR)*
          SUMERR = SUMERR + ERR                                         *
          NXY = 0                                                       *
          do J = 1 , NY                                                 *
            do I = 1 , NX                                               *
              NXY = NXY + 1                                             *
              DATA1(J,I+3,0) = RECV_VAL(NXY)                            *
            Enddo                                                       *
          Enddo                                                         *
          do NPE = 0 , SIZE-1                                           *
            JS  = JSI(NPE,INEST)                                        *
            JE  = JEI(NPE,INEST)                                        *
            do J = JS , JE                                              *
              do I = 1 , MIX                                            *
                DATA1(I,4+J-JS,NPE) = DATA1(I,J,0)                      *
              enddo                                                     *
            enddo                                                       *
          enddo                                                         *
          end if                                                        *
          call MPI_BARRIER(MPI_COMM_WORLD,ERR)                          *
          LEN = MIX*MJX                                                 *
          call MPI_SCATTER(DATA1(1,1,RANK),LEN,MPI_REAL4                *
     &         ,RECV_SH,LEN,MPI_REAL4,0,MPI_COMM_WORLD,ERR)             *
```

~~~~~~

The same procedure as the above lines with asterisk on the right end is repeated for other receive-data from SOLVEG: water vapor flux (RECV_QF), surface temperature (RECV_TS), and albedo (RECV_AL).

~~~~~~

```
c
          if( SUMERR .gt. 0 ) then
            write(6,*) 'MPI_RECV ERROR: SUMERR = ',SUMERR
     &                ,'  NSTAMPI',N,' = ',NSTAMPI(N)
          end if
        end if
c
        return
        end
c
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c*****Data exchange 2
        subroutine STAMPI_EXCHANGE2( INEST , NY , NX , DT , XTIME , PTOP
     &                             , XLAT , XLONG , RHO1
     &                             , TMN , TSS , MAVAIL
     &                             , PSB , PP3D , GLW , RAINP
```

```
     &                              , UA10 , VA10 , TA10 , QA10 )
c
      IMPLICIT NONE
c
      integer N, NNEST
      parameter( NNEST = 2 )
c
#     include <parame.incl>
#     include <soilp.incl>
      integer I, J, INEST, JS, JE, LEN, NPE
     &      , JSI(0:4,NNEST), JEI(0:4,NNEST)
      real   PTOP, DT, XTIME
      real   XLAT(MIX,MJX), XLONG(MIX,MJX), RHO1(MIX,MJX,MKX)
      real   TMN(MIX,MJX), TSS(MIX,MJX), MAVAIL(MIX,MJX)
      real   PSB(MIX,MJX), PP3D(MIX,MJX,MKX)
      real   GLW(MIX,MJX), RAINP(MIX,MJX)
      real   UA10(MIX,MJX), VA10(MIX,MJX)
      real   TA10(MIX,MJX), QA10(MIX,MJX)
c
      real   DATA1(MIX,MJX,0:3), DATA2(MIX,MJX,0:3)
      integer IDATA(MIX,MJX,0:3)
c
      include 'mpif.h'
      integer RANK, SIZE, SERVER, ERR, CLIENT(NNEST)
     &      , STAT(MPI_STATUS_SIZE)
      common /MPI/ RANK, SIZE, SERVER, ERR, CLIENT, STAT
c
      integer NX, NY, SUMERR, NXY
      real*4 SEND_RS(MJX,MIX,NNEST)
      real*4 SEND_PH(MJX,MIX), SEND_RI(MJX,MIX), SEND_RR(MJX,MIX)
      real*4 SEND_UU(MJX,MIX), SEND_VV(MJX,MIX)
      real*4 SEND_TT(MJX,MIX), SEND_QQ(MJX,MIX)
      real*4 SEND_LAT(MJX,MIX), SEND_LON(MJX,MIX), SEND_ROU(MJX,MIX)
      real*4 SEND_TBS(MJX,MIX), SEND_TSS(MJX,MIX)
      real*4 SEND_STY(MJX,MIX), SEND_VTY(MJX,MIX)
      real*4 SEND_MSL(MJX,MIX)
      real*4 SEND_VAL(MJX*MIX)
      common /DSTAMPI/ SEND_RS
      common /ISTAMPI/ JSI, JEI
      integer NSTAMPI(NNEST), IFLGFEED(NNEST)
     &      , IFLGOFF(NNEST), ICPLNEST(NNEST)
      common /NSTAMPI/ NSTAMPI, IFLGFEED, IFLGOFF, ICPLNEST
c-----OFF LINE
      integer INTXTIME
      character FOFFLINE*90, CDATENEW*19, CDATEOFF*15, DOM*4
#     include <chardate.incl>
c-----OFF LINE
c
      write(6,*) 'STAMPI Data exchange 2: Send INEST =', INEST
c
      N = INEST
      if( ICPLNEST(N) .ne. 1 ) then
        return
      end if
c
      if( NSTAMPI(N) .eq. 0 ) then
```

```
      SUMERR = 0
c-----LATITUDE (deg.)                                                    *
      LEN = MIX*MJX                                                      *
      call MPI_GATHER(XLAT,LEN,MPI_REAL4                                 *
     &       ,DATA1(1,1,RANK),LEN,MPI_REAL4,0,MPI_COMM_WORLD,ERR)        *
      call MPI_BARRIER(MPI_COMM_WORLD,ERR)                               *
      if( RANK .eq. 0 ) then                                             *
      do NPE = 0 , SIZE-1                                                *
        JS  = JSI(NPE,INEST)                                             *
        JE  = JEI(NPE,INEST)                                             *
        do J = JS , JE                                                   *
          do I = 1 , MIX                                                 *
            DATA1(I,J,0) = DATA1(I,4+J-JS,NPE)                           *
          enddo                                                          *
        enddo                                                            *
      enddo                                                              *
      do J = 1 , NY-1                                                    *
        do I = 1 , NX-1                                                  *
          SEND_LAT(I,J) = DATA1(J,I+3,0)                                 *
        enddo                                                            *
        SEND_LAT(NX,J) = 2.0*SEND_LAT(NX-1,J)                            *
     &                     - SEND_LAT(NX-2,J)                            *
      enddo                                                              *
      do I = 1 , NX                                                      *
        SEND_LAT(I,NY) = 2.0*SEND_LAT(I,NY-1)                            *
     &                     - SEND_LAT(I,NY-2)                            *
      enddo                                                              *
      NXY = 0                                                            *
      do J = 1 , NY                                                      *
        do I = 1 , NX                                                    *
          NXY = NXY + 1                                                  *
          SEND_VAL(NXY) = SEND_LAT(I,J)                                  *
        enddo                                                            *
      enddo                                                              *
      call MPI_SEND(SEND_VAL(1),NX*NY,MPI_REAL4,0,0,CLIENT(N),ERR)       *
      SUMERR = SUMERR + ERR                                              *
      end if                                                             *
```

~~~~~~

The same procedure as the above lines with asterisk on the right end is repeated for other send-data to SOLVEG: longitude (XLON → SEND_LON), air density (RHO1 → SEND_ROU), soil bottom temperature (TMN → SEND_TBS), sea surface temperature (TSS → SEND_TSS), soil type (ISLTYP → SEND_STY), vegetation type (IVGTYP → SEND_VTY), and soil moisture (SMCA → SEND_MSL).

~~~~~~

```
c
      if( SUMERR .gt. 0 ) then
        write(6,*) 'MPI_SEND ERROR: SUMERR = ',SUMERR
     &             ,'  LAT, LON, ROU'
      end if
c
c-----OFF LINE
      if( IFLGOFF(N) .eq. 1 ) then
        write(DOM,'(A,I1)') 'dom',N
        FOFFLINE = 'offline/'//DOM//'/lat.dat'
```

```
      open( unit=81, file=FOFFLINE
     &     , status='unknown', form='formatted' )
      do J = 1 , NY
        write(81,'(200F8.2)') (SEND_LAT(I,J), I=1,NX)
      enddo
      close(81)
c
      FOFFLINE = 'offline/'//DOM//'/lon.dat'
      open( unit=81, file=FOFFLINE
     &     , status='unknown', form='formatted' )
      do J = 1 , NY
        write(81,'(200F8.2)') (SEND_LON(I,J), I=1,NX)
      enddo
      close(81)
c
      FOFFLINE = 'offline/'//DOM//'/rou.dat'
      open( unit=81, file=FOFFLINE
     &     , status='unknown', form='formatted' )
      do J = 1 , NY
        write(81,'(200F8.2)') (SEND_ROU(I,J), I=1,NX)
      enddo
      close(81)
c
      FOFFLINE = 'offline/'//DOM//'/TBSOIL.dat'
      open( unit=81, file=FOFFLINE
     &     , status='unknown', form='formatted' )
      do J = 1 , NY
        write(81,'(200F8.2)') (SEND_TBS(I,J), I=1,NX)
      enddo
      close(81)
c
      FOFFLINE = 'offline/'//DOM//'/TSEASFC.dat'
      open( unit=81, file=FOFFLINE
     &     , status='unknown', form='formatted' )
      do J = 1 , NY
        write(81,'(200F8.2)') (SEND_TSS(I,J), I=1,NX)
      enddo
      close(81)
c
      FOFFLINE = 'offline/'//DOM//'/SOILTX.dat'
      open( unit=81, file=FOFFLINE
     &     , status='unknown', form='formatted' )
      do J = 1 , NY
        write(81,'(200F8.2)') (SEND_STY(I,J), I=1,NX)
      enddo
      close(81)
c
      FOFFLINE = 'offline/'//DOM//'/LANDUSE.dat'
      open( unit=81, file=FOFFLINE
     &     , status='unknown', form='formatted' )
      do J = 1 , NY
        write(81,'(200F8.2)') (SEND_VTY(I,J), I=1,NX)
      enddo
      close(81)
c
      FOFFLINE = 'offline/'//DOM//'/SMOIST.dat'
```

```
          open( unit=81, file=FOFFLINE
     &        , status='unknown', form='formatted' )
          do J = 1 , NY
            write(81,'(200F8.2)') (SEND_MSL(I,J), I=1,NX)
          enddo
          close(81)
        end if
c-----OFF LINE
      end if
c
 7777 NSTAMPI(N) = NSTAMPI(N) + 1
      SUMERR = 0
c-----SURFACE PRESSURE (hPa)
      LEN = MIX*MJX
      call MPI_GATHER(PSB,LEN,MPI_REAL4
     &     ,DATA1(1,1,RANK),LEN,MPI_REAL4,0,MPI_COMM_WORLD,ERR)
      call MPI_GATHER(PP3D(1,1,MKX),LEN,MPI_REAL4
     &     ,DATA2(1,1,RANK),LEN,MPI_REAL4,0,MPI_COMM_WORLD,ERR)
      call MPI_BARRIER(MPI_COMM_WORLD,ERR)
      if( RANK .eq. 0 ) then
      do NPE = 0 , SIZE-1
        JS  = JSI(NPE,INEST)
        JE  = JEI(NPE,INEST)
        do J = JS , JE
          do I = 1 , MIX
            DATA1(I,J,0) = DATA1(I,4+J-JS,NPE)
            DATA2(I,J,0) = DATA2(I,4+J-JS,NPE)
          enddo
        enddo
      enddo
      do J = 1 , NY-1
        do I = 1 , NX-1
          SEND_PH(I,J) =
     &    (DATA1(J,I+3,0)+PTOP+DATA2(J,I+3,0)*0.001)*10.0
        enddo
        SEND_PH(NX,J) = SEND_PH(NX-1,J)
      enddo
      do I = 1 , NX
        SEND_PH(I,NY) = SEND_PH(I,NY-1)
      enddo
      NXY = 0
      do J = 1 , NY
        do I = 1 , NX
          NXY = NXY + 1
          SEND_VAL(NXY) = SEND_PH(I,J)
        enddo
      enddo
      call MPI_SEND(SEND_VAL,NX*NY,MPI_REAL4,0,0,CLIENT(N),ERR)
      SUMERR = SUMERR + ERR
      end if
c
c-----SOLAR RADIATION (W M-2)
      if( RANK .eq. 0 ) then
c-----Calculated in exchange 1
      NXY = 0
      do J = 1 , NY
```

```
          do I = 1 , NX
            NXY = NXY + 1
            SEND_VAL(NXY) = SEND_RS(I,J,N)
          enddo
        enddo
        call MPI_SEND(SEND_VAL,NX*NY,MPI_REAL4,0,0,CLIENT(N),ERR)
        SUMERR = SUMERR + ERR
        end if
c
c-------LONG-WAVE RADIATION (W M-2)                                        *
        LEN = MIX*MJX                                                      *
        call MPI_GATHER(GLW,LEN,MPI_REAL4                                  *
     &      ,DATA1(1,1,RANK),LEN,MPI_REAL4,0,MPI_COMM_WORLD,ERR)           *
        call MPI_BARRIER(MPI_COMM_WORLD,ERR)                               *
        if( RANK .eq. 0 ) then                                             *
        do NPE = 0 , SIZE-1                                                *
          JS  = JSI(NPE,INEST)                                             *
          JE  = JEI(NPE,INEST)                                             *
          do J = JS , JE                                                   *
            do I = 1 , MIX                                                 *
              DATA1(I,J,0) = DATA1(I,4+J-JS,NPE)                           *
            enddo                                                          *
          enddo                                                            *
        enddo                                                              *
        do J = 1 , NY-1                                                    *
          do I = 1 , NX-1                                                  *
            SEND_RI(I,J) = DATA1(J,I+3,0)                                  *
          enddo                                                            *
          SEND_RI(NX,J) = SEND_RI(NX-1,J)                                  *
        enddo                                                              *
        do I = 1 , NX                                                      *
          SEND_RI(I,NY) = SEND_RI(I,NY-1)                                  *
        enddo                                                              *
        NXY = 0                                                            *
        do J = 1 , NY                                                      *
          do I = 1 , NX                                                    *
            NXY = NXY + 1                                                  *
            SEND_VAL(NXY) = SEND_RI(I,J)                                   *
          enddo                                                            *
        enddo                                                              *
        call MPI_SEND(SEND_VAL,NX*NY,MPI_REAL4,0,0,CLIENT(N),ERR)          *
        SUMERR = SUMERR + ERR                                              *
        end if                                                             *
```

~~~~~~

The same procedure as the above lines with asterisk on the right end is repeated for other send-data to SOLVEG: precipitation (RAINP → SEND_RR), u-wind (UA10 → SEND_UU), v-wind (VA10 → SEND_VV), temperature (TA10 → SEND_TT), and specific humidity (QA10 → SEND_QQ).

~~~~~~

```
c
        if( SUMERR .gt. 0 ) then
          write(6,*) 'MPI_SEND ERROR: SUMERR = ',SUMERR
     &              ,'  NSTAMPI',N,' = ',NSTAMPI(N)
        end if
c
```

```
c-----OFF LINE
      if( IFLGOFF(N) .eq. 1 .and.
     &    MOD(NSTAMPI(N),NINT(3600.0/(DT/2.0))) .eq. 1 ) then
        write(DOM,'(A,I1)') 'dom',N
        INTXTIME = NINT(XTIME*60.)
        if( NSTAMPI(N) .gt. 1 ) INTXTIME = INTXTIME + NINT(DT/2.0)
        CALL GETH_NEWDATE( CDATENEW, CDATE, INTXTIME )
        write(6,*) 'OFF LINE DATA OUTPUT:',CDATENEW
        CDATEOFF = CDATENEW(1:13)//CDATENEW(15:16)
        FOFFLINE = 'offline/'//DOM//'/pressure/PR'//CDATEOFF
        open(unit=81,file=FOFFLINE,status='unknown',form='formatted')
        do J = 1 , NY
          write(81,'(200F8.2)') (SEND_PH(I,J),I=1,NX)
        enddo
        close(81)
c
        FOFFLINE = 'offline/'//DOM//'/sw_down/SW'//CDATEOFF
        open(unit=81,file=FOFFLINE,status='unknown',form='formatted')
        do J = 1 , NY
          write(81,'(200F8.2)') (SEND_RS(I,J,N),I=1,NX)
        enddo
        close(81)
c
        FOFFLINE = 'offline/'//DOM//'/lw_down/LW'//CDATEOFF
        open(unit=81,file=FOFFLINE,status='unknown',form='formatted')
        do J = 1 , NY
          write(81,'(200F8.2)') (SEND_RI(I,J),I=1,NX)
        enddo
        close(81)
c
        FOFFLINE = 'offline/'//DOM//'/precip/RA'//CDATEOFF
        open(unit=81,file=FOFFLINE,status='unknown',form='formatted')
        do J = 1 , NY
          write(81,'(200F8.2)') (SEND_RR(I,J),I=1,NX)
        enddo
        close(81)
c
        FOFFLINE = 'offline/'//DOM//'/u_wind/UW'//CDATEOFF
        open(unit=81,file=FOFFLINE,status='unknown',form='formatted')
        do J = 1 , NY
          write(81,'(200F8.2)') (SEND_UU(I,J),I=1,NX)
        enddo
        close(81)
c
        FOFFLINE = 'offline/'//DOM//'/v_wind/VW'//CDATEOFF
        open(unit=81,file=FOFFLINE,status='unknown',form='formatted')
        do J = 1 , NY
          write(81,'(200F8.2)') (SEND_VV(I,J),I=1,NX)
        enddo
        close(81)
c
        FOFFLINE = 'offline/'//DOM//'/air_temp/TA'//CDATEOFF
        open(unit=81,file=FOFFLINE,status='unknown',form='formatted')
        do J = 1 , NY
          write(81,'(200F8.2)') (SEND_TT(I,J),I=1,NX)
        enddo
```

```
        close(81)
c
        FOFFLINE = 'offline/'//DOM//'/spc_humidity/QH'//CDATEOFF
        open(unit=81,file=FOFFLINE,status='unknown',form='formatted')
        do J = 1 , NY
          write(81,'(200F8.2)') (1000.0*SEND_QQ(I,J),I=1,NX)
        enddo
        close(81)
      end if
c-----OFF LINE
c
      if( NSTAMPI(N) .le. 1 ) goto 7777
c
      return
      end
*******************************************************************************
```

2) SOLVEG coupling module source code: stampi_mm5.f

```
***** stampi_mm5.f ***************************************************************
c+++++STAMPI+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c     Coupling to MM5
c           Initialization
c           Finalization
c           Subroutine for Final
c           Make send data
c           Data exchange 1 and 2
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c*****Initialization
      subroutine STAMPI_START
c
      include 'mpif.h'
      integer RANK, SIZE, SERVER, ERR, CLIENT, STAT(MPI_STATUS_SIZE)
      common /MPI/ RANK, SIZE, SERVER, ERR, CLIENT, STAT
      integer ISTAMPI
      common /ISTAMPI/ ISTAMPI
c
c-----Set ISTAMPI = 1:STAMPI-cal. , 0:offline-cal.
      ISTAMPI = 1
c
      if( ISTAMPI .ne. 1 ) then
        return
      end if
c
c-----STAMPI initialization
      call MPI_INIT(ERR)
      call MPI_COMM_SIZE(MPI_COMM_WORLD,SIZE,ERR)
      call MPI_COMM_RANK(MPI_COMM_WORLD,RANK,ERR)
      call MPI_COMM_GET_PARENT(SERVER,ERR)
      write(6,*) 'MPI_COMM_WORLD = ',MPI_COMM_WORLD
     .          ,'  SERVER = ',SERVER
      if( SERVER .eq. MPI_COMM_NULL ) then
```

```
        write(6,*) 'SERVER(TRUE) = ', SERVER
      end if
c-----STAMPI
c
      return
      end
c
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c*****Finalization
      subroutine STAMPI_END
c
      include 'mpif.h'
      integer RANK, SIZE, SERVER, ERR, CLIENT, STAT(MPI_STATUS_SIZE)
      common /MPI/ RANK, SIZE, SERVER, ERR, CLIENT, STAT
      integer ISTAMPI
      common /ISTAMPI/ ISTAMPI
c
      if( ISTAMPI .ne. 1 ) then
        return
      end if
c
c-----STAMPI finalization
c-----Receive kill_sig from server-model
      write(6,*) 'STAMPI REC_KILL_SIG'
      call REC_KILL_SIG
      write(6,*) 'STAMPI MPI_FINALIZE'
      call MPI_FINALIZE(ERR)
c-----STAMPI
c
      return
      end
c
c
c+++++STAMPI subroutines+++++++++++++++++++++++++++++++++++++++++++++++++++++
c*****Receive kill_sig from server-model
      subroutine REC_KILL_SIG
c
      include 'mpif.h'
c
      integer RANK, SIZE, SERVER, ERR, CLIENT, STAT(MPI_STATUS_SIZE)
      common /MPI/ RANK, SIZE, SERVER, ERR, CLIENT, STAT
      integer BUF
c
      if( RANK .eq. 0 ) then
        call MPI_RECV(BUF,1,MPI_INTEGER,0,0,SERVER,STAT,ERR)
        if( BUF .lt. 0 ) then
          BUF = -1
          call MPI_SEND(BUF,1,MPI_INTEGER,0,0,SERVER,ERR)
        else
          write(6,*) 'Unexpected kill_sig BUF = ', BUF
        end if
      else
        call MPI_RECV(BUF,1,MPI_INTEGER,0,0,MPI_COMM_WORLD,STAT,ERR)
        if( BUF .lt. 0 ) then
          write(6,*) 'Kill_sig OK'
```

```
        else
          write(6,*) 'Unexpected kill_sig BUF = ', BUF
        end if
      end if
c
      return
      end
c
c
c+++++STAMPI subroutines++++++++++++++++++++++++++++++++++++++++++++++++++++++
c*****Make send data
      subroutine STAMPI_MKSEND( FSH , FLQ , FLW , ALB )
c
      include 'Inclnum'
c
      REAL*8 FSH(NX,NY), FLQ(NX,NY), FLW(NX,NY), ALB(NX,NY)
      real*4 SEND_SH(NX,NY), SEND_QF(NX,NY)
     .     , SEND_TS(NX,NY), SEND_AL(NX,NY)
      common /STAMPI/ SEND_SH, SEND_QF, SEND_TS, SEND_AL
      integer ISTAMPI
      common /ISTAMPI/ ISTAMPI
c
      if( ISTAMPI .ne. 1 ) then
        return
      end if
c
      SIGM = 5.670D-8
c
      do 1000 J = 1 , NY
      do 1000 I = 1 , NX
        SEND_SH(I,J) = FSH(I,J)
        SEND_QF(I,J) = FLQ(I,J)
        SEND_TS(I,J) = ( FLW(I,J)/SIGM )**0.25
        SEND_AL(I,J) = ALB(I,J)
 1000 continue
c
      return
      end
c
c
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c*****Data exchange 1
      subroutine STAMPI_EXCHANGE1( ISTAMPI1 , FLAT , FLON , ROU
     .                           , TBOTM , TWATR , STY , VTY , HWI
     .                           , PHPI , RSOLI , RINFI , RRI
     .                           , UI , VI , TI , QI )
c
      include 'Inclnum'
      real*8 FLAT(NX,NY), FLON(NX,NY), ROU(NX,NY)
      real*8 TBOTM(NX,NY), TWATR(NX,NY), STY(NX,NY), VTY(NX,NY)
      real*8 HWI(NX,NY)
      real*8 PHPI(NX,NY,0:1), RSOLI(NX,NY,0:1), RINFI(NX,NY,0:1)
      real*8 RRI(NX,NY,0:1), UI(NX,NY,0:1), VI(NX,NY,0:1)
      real*8 TI(NX,NY,0:1), QI(NX,NY,0:1)
c
      include 'mpif.h'
```

```
      integer RANK, SIZE, SERVER, ERR, CLIENT, STAT(MPI_STATUS_SIZE)
      common /MPI/ RANK, SIZE, SERVER, ERR, CLIENT, STAT
      real*4 RECV_PH(NX,NY), RECV_RS(NX,NY), RECV_RI(NX,NY)
     .     , RECV_RR(NX,NY), RECV_UU(NX,NY), RECV_VV(NX,NY)
     .     , RECV_TT(NX,NY), RECV_QQ(NX,NY)
      real*4 RECV_LAT(NX,NY), RECV_LON(NX,NY), RECV_ROU(NX,NY)
     .     , RECV_TBS(NX,NY), RECV_TSS(NX,NY)
     .     , RECV_STY(NX,NY), RECV_VTY(NX,NY), RECV_MSL(NX,NY)
      integer ISTAMPI
      common /ISTAMPI/ ISTAMPI
c
      ISTAMPI1 = ISTAMPI
      if( ISTAMPI .ne. 1 ) then
        return
      end if
c
c------LATITUDE (deg.)                                                  *
      call MPI_RECV(RECV_LAT,NX*NY,MPI_REAL4,0,0,SERVER,STAT,ERR)      *
      write(6,*) 'RECV_LAT(I,25): ERR = ',ERR                          *
      write(6,'(50F8.2)') (RECV_LAT(I,25),I=1,NX)                      *
      do 500 J = 1 , NY                                                *
      do 500 I = 1 , NX                                                *
        FLAT(I,J) = RECV_LAT(I,J)                                      *
  500 continue                                                         *
```

~~~~~~

The same procedure as the above lines with asterisk on the right end is repeated for other receive-data from MM5: longitude (RECV_LON), air density (RECV_ROU), soil bottom temperature (RECV_TBS), sea surface temperature (RECV_SST), soil type (RECV_STY), vegetation type (RECV_VTY), and soil moisture (RECV_MSL).

~~~~~~

```
c
c------SURFACE PRESSURE (hPa)                                           *
 3001 call MPI_RECV(RECV_PH,NX*NY,MPI_REAL4,0,0,SERVER,STAT,ERR)       *
      write(6,*) 'RECV_PH(I,25): ERR = ',ERR                           *
      write(6,'(50F8.1)') (RECV_PH(I,25),I=1,NX)                       *
      do 3002 J = 1 , NY                                               *
      do 3002 I = 1 , NX                                               *
        PHPI(I,J,1) = RECV_PH(I,J)                                     *
 3002 continue                                                         *
```

~~~~~~

The same procedure as the above lines with asterisk on the right end is repeated for other receive-data from MM5: solar radiation (RECV_RS), long-wave radiation (RECV_RI), precipitation (RECV_RR), u-wind (RECV_UU), v-wind (RECV_VV), temperature (RECV_TT), and specific humidity (RECV_QQ).

~~~~~~

```
c
      return
      end
c
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c*****Data exchange 2
```

```
      subroutine STAMPI_EXCHANGE2( ISTAMPI1 , ITIME , TIMER
     .                           , PHPI , RSOLI , RINFI , RRI
     .                           , UI , VI , TI , QI )
c
      include 'InclNum'
      integer ITIME
      real*8 TIMER(0:1)
      real*8 PHPI(NX,NY,0:1), RSOLI(NX,NY,0:1), RINFI(NX,NY,0:1)
      real*8 RRI(NX,NY,0:1), UI(NX,NY,0:1), VI(NX,NY,0:1)
      real*8 TI(NX,NY,0:1), QI(NX,NY,0:1)
c
      include 'mpif.h'
      integer RANK, SIZE, SERVER, ERR, CLIENT, STAT(MPI_STATUS_SIZE)
      common /MPI/ RANK, SIZE, SERVER, ERR, CLIENT, STAT
      real*4 RECV_PH(NX,NY), RECV_RS(NX,NY), RECV_RI(NX,NY)
     .     , RECV_RR(NX,NY), RECV_UU(NX,NY), RECV_VV(NX,NY)
     .     , RECV_TT(NX,NY), RECV_QQ(NX,NY)
      real*4 SEND_SH(NX,NY), SEND_QF(NX,NY)
     .     , SEND_TS(NX,NY), SEND_AL(NX,NY)
      common /STAMPI/ SEND_SH, SEND_QF, SEND_TS, SEND_AL
      integer ISTAMPI
      common /ISTAMPI/ ISTAMPI
c
      ISTAMPI1 = ISTAMPI
c
      if( ISTAMPI .ne. 1 ) then
        return
      end if
c
c-----STAMPI date send
      if( TIMER(0) .gt. 0.0 ) then
        NERR = 0
c-----SENSIBLE HEAT FLUX (W M-2)
        call MPI_SEND(SEND_SH, NX*NY, MPI_REAL4, 0, 0, SERVER, ERR)
        NERR = NERR + ERR
c-----WATER VAPOR FLUX (KG M-2 S-1)
        call MPI_SEND(SEND_QF, NX*NY, MPI_REAL4, 0, 0, SERVER, ERR)
        NERR = NERR + ERR
c-----SURFACE TEMPERATURE (K)
        call MPI_SEND(SEND_TS, NX*NY, MPI_REAL4, 0, 0, SERVER, ERR)
        NERR = NERR + ERR
c-----ALBEDO
        call MPI_SEND(SEND_AL, NX*NY, MPI_REAL4, 0, 0, SERVER, ERR)
        NERR = NERR + ERR
c
        if( NERR .gt. 0 ) then
          write(6,*) 'MPI_SEND ERROR: ERR_SUM = ', NERR
     .               ,'  ITIME = ', ITIME
        end if
      end if
c
c
c-----STAMPI data get
c-----SURFACE PRESSURE (hPa)                                          *
      NERR = 0                                                        *
 2001 call MPI_RECV(RECV_PH, NX*NY, MPI_REAL4, 0, 0, SERVER, STAT, ERR)  *
```

```
      NERR = NERR + ERR                                          *
      do 2002 J = 1 , NY                                         *
      do 2002 I = 1 , NX                                         *
        PHPI(I,J,1) = RECV_PH(I,J)                               *
 2002 continue                                                   *
```

~~~~~~

The same procedure as the above lines with asterisk on the right end is repeated for other receive-data from MM5: solar radiation (RECV_RS), long-wave radiation (RECV_RI), precipitation (RECV_RR), u-wind (RECV_UU), v-wind (RECV_VV), temperature (RECV_TT), and specific humidity (RECV_QQ).

~~~~~~

```
c
      if( NERR .gt. 0 ) then
        write(6,*) 'MPI_RECV ERROR: ERR_SUM = ',NERR
     .              ,'   ITIME = ',ITIME
      end if
c
      return
      end
************************************************************************************
```

国際単位系 (SI) と換算表

表 1　SI基本単位および補助単位

量	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
平面角	ラジアン	rad
立体角	ステラジアン	sr

表 2　SIと併用される単位

名称	記号
分，時，日	min, h, d
度，分，秒	°, ′, ″
リットル	l, L
トン	t
電子ボルト	eV
原子質量単位	u

1 eV = 1.60218 × 10^{-19} J

1 u = 1.66054 × 10^{-27} kg

表 5　SI接頭語

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

表 3　固有の名称をもつSI組立単位

量	名称	記号	他のSI単位による表現
周波数	ヘルツ	Hz	s^{-1}
力	ニュートン	N	$m \cdot kg/s^2$
圧力，応力	パスカル	Pa	N/m^2
エネルギー，仕事，熱量	ジュール	J	N·m
工率，放射束	ワット	W	J/s
電気量，電荷	クーロン	C	A·s
電位，電圧，起電力	ボルト	V	W/A
静電容量	ファラド	F	C/V
電気抵抗	オーム	Ω	V/A
コンダクタンス	ジーメンス	S	A/V
磁束	ウェーバ	Wb	V·s
磁束密度	テスラ	T	Wb/m^2
インダクタンス	ヘンリー	H	Wb/A
セルシウス温度	セルシウス度	℃	
光束	ルーメン	lm	cd·sr
照度	ルクス	lx	lm/m^2
放射能	ベクレル	Bq	s^{-1}
吸収線量	グレイ	Gy	J/kg
線量当量	シーベルト	Sv	J/kg

表 4　SIと共に暫定的に維持される単位

名称	記号
オングストローム	Å
バーン	b
バール	bar
ガル	Gal
キュリー	Ci
レントゲン	R
ラド	rad
レム	rem

1 Å = 0.1 nm = 10^{-10} m

1 b = 100 fm^2 = 10^{-28} m^2

1 bar = 0.1 MPa = 10^5 Pa

1 Gal = 1 cm/s^2 = 10^{-2} m/s^2

1 Ci = 3.7 × 10^{10} Bq

1 R = 2.58 × 10^{-4} C/kg

1 rad = 1 cGy = 10^{-2} Gy

1 rem = 1 cSv = 10^{-2} Sv

（注）

1. 表1－5は「国際単位系」第5版，国際度量衡局 1985年刊行による。ただし，1 eVおよび1 uの値はCODATAの1986年推奨値によった。
2. 表4には海里，ノット，アール，ヘクタールも含まれているが日常の単位なのでここでは省略した。
3. barは，JISでは流体の圧力を表わす場合に限り表2のカテゴリーに分類されている。
4. EC閣僚理事会指令ではbar，barnおよび「血圧の単位」mmHgを表2のカテゴリーに入れている。

換　算　表

力	N(=10^5 dyn)	kgf	lbf
	1	0.101972	0.224809
	9.80665	1	2.20462
	4.44822	0.453592	1

粘度　1 Pa·s(N·s/m^2) = 10 P(ポアズ)(g/(cm·s))

動粘度　1 m^2/s = 10^4 St(ストークス)(cm^2/s)

圧力	MPa(=10 bar)	kgf/cm^2	atm	mmHg(Torr)	lbf/in^2(psi)
	1	10.1972	9.86923	7.50062×10^3	145.038
	0.0980665	1	0.967841	735.559	14.2233
	0.101325	1.03323	1	760	14.6959
	1.33322×10^{-4}	1.35951×10^{-3}	1.31579×10^{-3}	1	1.93368×10^{-2}
	6.89476×10^{-3}	7.03070×10^{-2}	6.80460×10^{-2}	51.7149	1

エネルギー・仕事・熱量	J(=10^7 erg)	kgf·m	kW·h	cal(計量法)	Btu	ft·lbf	eV
	1	0.101972	2.77778×10^{-7}	0.238889	9.47813×10^{-4}	0.737562	6.24150×10^{18}
	9.80665	1	2.72407×10^{-6}	2.34270	9.29487×10^{-3}	7.23301	6.12082×10^{19}
	3.6×10^6	3.67098×10^5	1	8.59999×10^5	3412.13	2.65522×10^6	2.24694×10^{25}
	4.18605	0.426858	1.16279×10^{-6}	1	3.96759×10^{-3}	3.08747	2.61272×10^{19}
	1055.06	107.586	2.93072×10^{-4}	252.042	1	778.172	6.58515×10^{21}
	1.35582	0.138255	3.76616×10^{-7}	0.323890	1.28506×10^{-3}	1	8.46233×10^{18}
	1.60218×10^{-19}	1.63377×10^{-20}	4.45050×10^{-26}	3.82743×10^{-20}	1.51857×10^{-22}	1.18171×10^{-19}	1

1 cal = 4.18605 J（計量法）

= 4.184 J　（熱化学）

= 4.1855 J（15 °C）

= 4.1868 J（国際蒸気表）

仕事率　1 PS（仏馬力）

= 75 kgf·m/s

= 735.499 W

放射能	Bq	Ci
	1	2.70270×10^{-11}
	3.7×10^{10}	1

吸収線量	Gy	rad
	1	100
	0.01	1

照射線量	C/kg	R
	1	3876
	2.58×10^{-4}	1

線量当量	Sv	rem
	1	100
	0.01	1

（86年12月26日現在）

New Method for Model Coupling Using Stampi:Application to the Coupling of Atmosphere Model(MM5) and Land-surface Model(SOLVEG)