Environmental Modelling & Software 23 (2008) 1240–1249

Contents lists available at ScienceDirect

Environmental Modelling & Software

journal homepage: www.elsevier.com/locate/envsoft

Using the Model Coupling Toolkit to couple earth system models

John C. Warner [a], Natalie Perlin [b,*], Eric D. Skyllingstad [b]

[a] U.S. Geological Survey, Coastal and Marine Geology Program, 384 Woods Hole Road, Woods Hole, MA 02543, USA
[b] College of Oceanic and Atmospheric Sciences, Oregon State University, Corvallis, OR 97331, USA

ARTICLE INFO

Article history:
Received 14 March 2007
Received in revised form 19 February 2008
Accepted 3 March 2008
Available online 23 May 2008

Keywords:
Model coupling
Model Coupling Toolkit
ROMS
COAMPS
SWAN
Sparse matrix interpolation

ABSTRACT

Continued advances in computational resources are providing the opportunity to operate more sophisticated numerical models. Additionally, there is an increasing demand for multidisciplinary studies that include interactions between different physical processes. Therefore there is a strong desire to develop coupled modeling systems that utilize existing models and allow efficient data exchange and model control. The basic system would entail model "1" running on "M" processors and model "2" running on "N" processors, with efficient exchange of model fields at predetermined synchronization intervals. Here we demonstrate two coupled systems: the coupling of the ocean circulation model Regional Ocean Modeling System (ROMS) to the surface wave model Simulating WAves Nearshore (SWAN), and the coupling of ROMS to the atmospheric model Coupled Ocean Atmosphere Prediction System (COAMPS). Both coupled systems use the Model Coupling Toolkit (MCT) as a mechanism for operation control and inter-model distributed memory transfer of model variables. In this paper we describe requirements and other options for model coupling, explain the MCT library, ROMS, SWAN and COAMPS models, methods for grid decomposition and sparse matrix interpolation, and provide an example from each coupled system. Methods presented in this paper are clearly applicable for coupling of other types of models.

1. Introduction

Significant investments have been made over the past couple of decades to develop numerical models for studying and prediction of environmental phenomena of various spatial and temporal scales. Any particular model, however, is intrinsically limited by the range of scales and corresponding processes it is able to represent. In Earth System modeling, for example, the cascade of resolving scales ranges from global, simulated using Global Circulation Models (GCM), to regional and local, using Limited-Area Models (LAM), down to Large-Eddy Simulation models (LES). For certain process-oriented studies, scale isolation could be useful for understanding of a particular environmental phenomenon. For more general model application and research, however, there is a need to address natural scale interactions. Various techniques have been developed for this purpose, such as downscaling or upscaling (Leung, 2005; Lau et al., 1999), multi-model modeling (Giustolisi et al., 2007), or, most recently developing, coupled modeling approach (Hermann et al., 2002; Militello and Zundel, 2002; Perlin et al., 2007; Jorba et al., 2008). Coupled modeling approach can be advantageous because it could provide two-way interaction between the models either operating on different scales, or simulating different set of interdependent processes.

A coupled modeling approach provides means to develop multidisciplinary application that incorporate more sophisticated physical processes and interactions in a numerical simulation. For example, studying the impacts of storms on coastal systems may require atmospheric winds, surface waves, river discharges, sediment transport, and biological tracers and address issues related to aquatic species and habitats. Some of these processes are self interacting such as the effects of waves on the currents and mutual response of currents on the waves. The circulation of tidal inlets involves the interactions of seaward flowing currents that can be strong enough to impede the landward propagation of wind-driven surface waves. This interaction can only be modeled correctly if a coupled ocean-wave modeling system is applied. The waves and currents drive sediment transport that cause changes to the seafloor morphology which feedback to the wave shoaling and water circulation. Simulation of this interaction requires a coupled sediment transport–ocean circulation–surface wave modeling system. In another coastal circulation system, wind-driven coastal upwelling, warm surface waters are driven offshore and replaced by cooler bottom ocean waters in the nearshore. These cold surface waters interact with the marine atmospheric

* Corresponding author.
E-mail addresses: jcwarner@usgs.gov (J.C. Warner), nperlin@coas.oregonstate.edu (N. Perlin), skylling@coas.oregonstate.edu (E.D. Skyllingstad).

1364-8152/$ – see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.envsoft.2008.03.002

boundary layer, alter its structure, stability, and the intensity of atmospheric forcing on the ocean. Many of these processes are typically simulated using different models. Therefore to simulate the interaction one would either have to parameterize the missing physical process in their model or dynamically couple to an existing model. With the increasing ease of acquiring multi-processor computers, the logical choice is often to couple with an already existing model. Coupling allows the models to run simultaneously and provide a method to transfer information in an efficient time stepping manner.

The coupling of models can occur with all the models running concurrently or sequentially. For the purposes of this paper we will consider the models to be running concurrently. Each model runs on its own set of processors that may reside on a common cluster, and there exists a driver for the coupled modeling system that controls execution and mediates data exchanges between the individual models. The use of clusters is increasing in popularity and cluster-based parallel computing is now widely affordable to scientists. Cluster computers give their owners (i) convenience since they are not sharing a large resource with lots of other users (i.e. jobs are not spending a lot of time waiting in a queue); (ii) flexibility since the processors and interconnect are often purchased separately; and (iii) extendibility because one can always buy more processors to extend the cluster as budgets allow. To take advantage of the multiple processors of a cluster, the models use the Message Passing Interface (MPI) distributed memory parallelism communication protocol. MPI provides means for information about the overlapping halo regions of the grid tiles to be passed between processors different nodes. The methodology described in this paper uses the MPI protocol method.

There exist several entities available to act as model coupling agents. One can certainly write their own routines to allow data transfer. The programmer would include calls for each processor to send and receive information from each grid segment or from each model at various stages in the model stepping. However, each programmer would inevitably develop their own data acquisition and dissemination method, and it would be cumbersome to easily couple different models from different developers. There needs to be a standard to foster a common protocol.

The Model Coupling Toolkit (MCT; described in more detail below) is an open source software library to construct parallel coupled models from individual models. The MCT data model constitutes three data types for storage of multi-field integer and real valued data, grids, and domain decompositions. Each separate model runs on its own set of processors and MCT provides the protocols for inter-model data coupling, allows efficient data transfer between the different models, and provides interpolation algorithms for the data fields that are transferred.

Other coupling methods include the Earth System Modeling Framework (ESMF; Hill et al., 2004) which is also an open source code product. ESMF defines architecture for composing coupled systems. Each component of the system is a physical domain, a function, a coupler, etc., that has a standard calling interface. This will allow each component to be completely replaced by another system. Eventually, ESMF functionality should exceed that of the MCT, but at the time of writing it is still in development with some initial applications.

Herein we demonstrate the coupling of an oceanographic circulation model to a surface wave model using the MCT, and the coupling of an oceanographic model to an atmospheric model using the MCT. In Section 2 we describe each of the models and the coupler. In Section 3 the methodology to couple the models is described in detail. Section 4 demonstrates a few basic applications of each coupled system. The discussion and conclusions are in Sections 5 and 6.

2. Models

2.1. Regional Ocean Modeling System (ROMS, v. 2.2)

ROMS is a public domain, free surface, hydrostatic, three-dimensional, primitive equation ocean circulation model (Shchepetkin and McWilliams, 2005; Haidvogel et al., 2008). The model solves the Boussinesq approximation to the Reynolds averaged form of the Navier Stokes equations on an orthogonal curvilinear Arakawa 'C' grid in the horizontal and uses stretched terrain following coordinates in the vertical. The model features second, third, and fourth order horizontal and vertical advection schemes for momentum and tracers, and can use splines to reconstruct vertical advection profiles. Along with temperature and salinity, ROMS can transport passive tracers, contains algorithms for suspended and bed load sediment transport, multiple choices for turbulence closures, biologic routines, and several types of boundary conditions.

We have implemented algorithms to include the effects of surface wind waves on the currents (Warner et al., in press) based on the method of Mellor (2003, 2005). For these effects ROMS requires information of wave energy, wave length, and wave direction. Other processes such as surface fluxes of turbulent kinetic energy due to breaking waves, bed-load sediment transport, and enhanced bottom friction due to waves require information of bottom orbital velocities, surface and bottom wave periods, and wave-energy dissipation. These parameters can be obtained directly from a wave model, such as SWAN.

2.2. Simulating Waves Nearshore (SWAN, v. 4041AB)

SWAN is a public domain spectral wave model that solves the spectral density evolution equation (Booij et al., 1999; Ris et al., 1999). SWAN simulates wind wave generation and propagation in coastal waters and includes the processes of refraction, diffraction, shoaling, wave–wave interactions, and dissipation due to white-capping, wave breaking, and bottom friction. SWAN allows input of wind forcing, water velocity, and water level, and we have added the input of changing bottom topography. These inputs allow the wave model to respond to changes in water currents, sea level, and morphological changes due to sediment transport on the sea floor to effect the generation and propagation of surface wind waves. Coupling of ROMS and SWAN is a natural formulation to allow mutual interactions of the waves and currents. During the coupling, the SWAN model is run in the non-stationary mode to allow temporal evolution of the model state.

2.3. Atmospheric component—COAMPS model (v. 3.1.1)

The atmospheric model component in the coupled system is the Naval Research Laboratory (NRL) COAMPS™ mesoscale model (Hodur, 1997). It is a nonhydrostatic, quasi-compressible atmospheric model with physics schemes and a variety of physical parameterizations of sub-grid scale processes for predicting meso and micro scales of motion. The model predicts wind momentum components (u, v, and w), surface pressure, dew point, precipitation, surface sensible and latent heat fluxes, relative humidity, and air temperature on a sigma-z vertical coordinate grid. COAMPS has been extensively used for operational forecasts, as well as for real-data and idealized research experiments (Burk et al., 1999; Samelson et al., 2002; Pickett and Paduan, 2003). The model surface fluxes of heat, momentum, and moisture are exchanged with the ocean model. In return, COAMPS requires water temperature that can be computed by the ocean component.

2.4. Model Coupling Toolkit (MCT, v. 2.1.0)

The MCT (Jacob et al., 2005; Larson et al., 2005) allows the transmission and transformation of various distributed data between component models that constitute a parallel coupled system. MCT is an open source software written in Fortran90 and works with the MPI communication protocol. It is compiled as a set of Fortran modules and libraries. The modules are used during the compilation and the libraries are linked to build the executable. Each model calls the MCT during execution to send and receive data. The MCT has been used to couple the Weather Research and Forecasting Model (WRF) to ROMS (Michalakes, 2003) and acts as the base for the Community Climate System Model coupler (Craig et al., 2005).

3. Methodology

The component models in a system coupled via MCT will be referred to as MODEL1 and MODEL2; these are ROMS and SWAN in one case, ROMS and COAMPS in another case. We present, in a general manner, the methodology implemented to couple these two sets of models. This provides a basic example that could be used for coupling of other types of models. A common approach used to develop both systems is that the data exchange between the individual component models is performed across the horizontal domain, i.e. exchanging only two-dimensional model fields.

3.1. Master program

Each coupled system was developed in a way to preserve the original MODEL1 and MODEL2 codes as much as possible; these models could be placed into separate directories. The individual models of the coupled system could be compiled separately as libraries and then linked together as one executable program. The main program that serves to control the flow and processor allocation of the coupled model system is called '**Master**'. This '**Master**' program was coded for concurrent coupling, i.e., in which the component models run simultaneously on several different processors. The MCT allows for other types of coupling as well (such as sequential); however, we chose the concurrent configuration as the best alternative for our applications.

At execution, the '**Master**' driver program distributes the total number of processors allocated for the job to the individual models, and calls each model component with an initialize, run, and finalize structure. The master program is organized as follows. '**Master**' initializes MPI using a standard call '**MPI_INIT**' to activate a common MPI communicator (*MPI_COMM_WORLD*) for the entire coupled system. '**Master**' knows the total number of models that are being coupled (for our examples this is two), and uses the '**MPI_COMM_SPLIT**' function to split the common communicator *MPI_COMM_WORLD* into multiple communicators *COMM1* and *COMM2* (one for each model). It now needs to delegate specific processors to each model. To accomplish this, '**Master**' determines the total number of processors requested for the submitted job using the '**MPI_COMM_SIZE**' command and reads an input file to determine the number of processors to be assigned to each model, for instance, 'M' processors to MODEL1 and 'N' processors to MODEL2. Then, each processor ID is determined using the '**MPI_COMM_RANK**' function; '**Master**' assigns processors to each model based on the processor ID and number of processors requested for that model. In our test cases with ROMS and SWAN we allocated a total of five processors, two of which are for ROMS, and three are for SWAN. For COAMPS–ROMS coupled tests we allocated a total of three processors, two for COAMPS and one for ROMS. These numbers are determined by the specific application and are limited only by the number of processors on the computer cluster.

The '**Master**' program then calls steps to initialize, run, and finalize on all processors for both MODEL1 and MODEL2. It is recommended that the individual modeling components be structured in this way to provide a consistent framework and to assist in determining the correct synchronization points. Additional subroutines are written for each model to follow steps for the MCT aspect of initialization, communication via MCT during the run phase, and terminate MCT properly during the finalize phase. These routines are organized in modules called ***ADD_MOD1*** and ***ADD_MOD2***, for MODEL1 and MODEL2, respectively, and look schematically as follows:

```
module ADD_MOD1
    use MCT_modules
    contains
        subroutine MCTinit_MOD1(COMM1,…)
        subroutine MCTrun_MOD1(COMM1,…)
        subroutine MCTend_MOD1
end module ADD_MOD1
```

The module ***ADD_MOD1*** is placed with the code for MODEL1 and compiled with that model. A similar module, ***ADD_MOD2*** is written for MODEL2, and compiled with that model. The structure of each step of initialize, run, and finalize is detailed below.

3.2. Initialize

During the initial step, each processor determines its grid segment (tile) for the model, allocates and initializes arrays and model variables. MCT is also initialized during this step. This occurs in the corresponding subroutine ***MCTinit_MODn*** (where *n* is 1 for MODEL1 and 2 for MODEL2). This subroutine is called to activate MCT connection, to perform MCT domain decomposition, to initialize MCT arrays and interpolation matrices (if needed). The basic structure of this subroutine is given below, followed by explanations:

```
subroutine MCTinit_MOD1(COMM1, ncomps, MOD1_ID, MOD2_ID)
    call MCTWorld_init(ncomps,MPI_COMM_WORLD,COMM1,MOD1_ID)
    call GlobalSegMap_init(GlobalSegMapMOD1,start,length,root,COMM1,
    MOD1_ID)
    call AttrVect_init(AV1_toMOD2,rList="M1var1:M1var2:M1var3",
    lsize=AV1size)
    call AttrVect_init(AV1_fromMOD2,rList="M2var1:M2var2:M2var3",
    lsize=AV2size)
    call Router_init(MOD2_ID, GlobalSegMapMOD1,COMM1,Router1)
end subroutine MCTinit_MOD1
```

First, the **MCTWorld_init** function registers a component model based on its communicator and model affiliation (*COMM1*, *MOD1_ID*), gets total number of components in a coupled system (*ncomps,* equals to two in our examples), determines total number of processes owned by a given component, and identifies local and global processor rank of each process for a given component and for the world communicator. Identification number of the second component (*MOD2_ID*), needs to be specified explicitly by the user in the subroutine to establish communication between the models.

When more than one processor is assigned to the model, a model decomposes its gridded domain into tiles (segments). In our coupling method each processor performs calculations in one tile of the individual model. The next step in our procedure is to initialize the global segment map (**GlobalSegMap_init**), reporting MCT what segment of the model grid the current processor is working on. To enable data exchange between the components of model fields scattered across multiple processors, MCT therefore needs to construct universal identification of grid point locations. This is accomplished using one-dimensional domain decomposition known as "virtual linearization" (Larson et al., 2005), in which multidimensional model arrays are linearized, and multiple indices are converted into a single unique index—a global ID number of the grid point. Special MCT data type object '*GlobalSegMap*' holds information on how a grid segment associated with each tile is orientated within the global MCT grid for that model, and is initialized by function call **GlobalSegMap_init**. The '*GlobalSegMap*' data object is determined by the arrays '*start*' and '*length*', which indicate starting indices of the linear grid segment on a global grid, and length of these linear segments, respectively.

As a simple illustration, Fig. 1 shows a computational rectangular grid with 24 points: 4 rows by 6 columns. Small numbers indicate linear indices of the global grid. Domain decomposition is two-dimensional for MODEL1 (ROMS model), and is achieved from user-specified number of tiles in x-direction (NTILEI) and y-direction (NTILEJ), resulting in a global grid subdivided into NTILEI × NTILEJ tiles. Each actual tile will have ghost (halo) points added around the perimeter of the tile that are often needed for the numerical algorithms, and contain repeated information (except along the perimeter of the domain). Halo points are only updated within each single component for each model's own state solution and the halo points are not referenced or transferred during the model coupling. In this simple example the MODEL1 will be allocated 2 processors and the grid is decomposed into 2 tiles (tile 0 and tile 1), one for each processor. Tile 0 for MODEL1 will have a '*GlobalSegMapMOD1*' identified with two '*start*' indices, grid point number 1 and grid point number 7, and two '*length*' values both equal to 6. Total length of the linearized data array for the tile 0 will be 12. Similarly, tile 1 for MODEL1 will have '*start*' indices 13 and 19, two '*length*' values both equal to 6; total length of the data segment equal 12.

The MODEL2 in this simple example has three processors and is decomposed into three tiles (numbered 0, 1, and 2). Grid decomposition is determined by dividing the domain into a number of sections equal to the number of processors allocated to MODEL2 (SWAN model). This could be accomplished by starting along the shorter direction of the grid (row or column) and taking full rows (or columns) for 1/(*number of processors*) of the grid length. In this example, '*GlobalSegMapMOD2*' for tile 0 has four elements in '*start*' array at indices 1, 7, 13, and 19, four elements in '*length*' all equal to two, and total length of one-dimensional data array equal to 8. Similar maps are designated for tiles 1 and 2. **GlobalSegMap_init** also uses information about the processor rank of the root processor (usually *root*=0), and component model communicator and identification (*COMM1, MOD1_ID* or *COMM2, MOD2_ID*).

Data storage in MCT is accomplished using a data type called an *attribute vector* ('*AV1_toMOD2*' as in the above example). Attribute vectors hold data for each

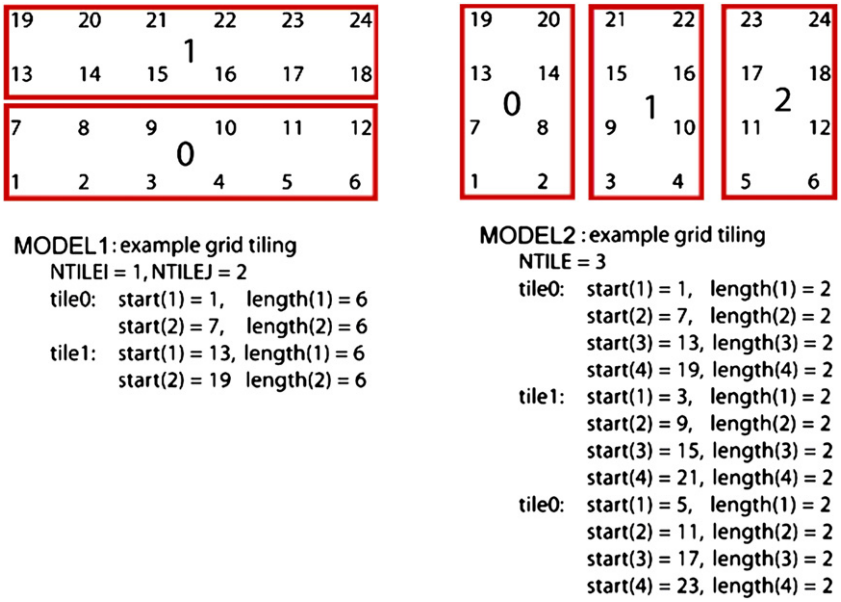

Fig. 1. Simple example of how different models may decompose the same grid. MODEL1 decomposition is user specified whereas MODEL2 decomposition is determined by grid orientation.

linearized domain segment as described by corresponding '*GlobalSegMap*' object for a given processor, and are initialized using **AttrVect_init** function that allocates storage space for the linearized array. Attribute vectors hold real and/or integer data segments of user-defined fields/variables specified in 'rList='*var1:var2:var3*' for real fields, or 'iList=…' for integer data. The 'rList' (or 'iList') is a list of string tokens delimited by colons that refer to field names. This formulation provides flexibility to easily adjust the fields that are being used in the coupling. All these data segments are of equal length, 'lsize=*AVsize*', corresponding to one-dimensional MCT domain decomposition. In the examples shown in Fig. 1, sizes of these data segments are 'lsize=12' for both tiles in MODEL1 domain, and 'lsize=8' for each of the three tiles in MODEL2 grid (although 'lsize' may vary for different tiles of a model domain). This establishes the size of data segments that are sent from one component MODEL1 (*MOD1_ID*) to MODEL2 (*MOD2_ID*) ('*AV1_toMOD2*'). A separate attribute vector is initialized for the data to be received from MODEL2 ('*AV1_fromMOD2*'). Similarly, attribute vectors *AV2_toMOD1* and *AV2_fromMOD1* are initialized in ***MCTinit_MOD2*** routine to allocate storage space for data computed by MODEL2 to be sent to MODEL1, and for the fields from the MODEL1 to be received by MODEL2, correspondingly.

MCT data transfer requires a special communication table called '*Router*'. An MCT '*Router*' is a data type object that is created from two components' respective domain decomposition descriptors (i.e., their corresponding '*GlobalSegMap*') to allow for parallel data transfer between model domain segments residing on different processors (Larson et al., 2005). For a given set of MODEL1 gridpoints on a given processor ('*GlobalSegMapMOD1*'), the '*Router1*' determines corresponding grid point locations of MODEL2 ('*GlobalSegMapMOD2*') on other processors, and provides the conduit for data that will be transferred. The '*Router1*' table for the MODEL1 is initialized by a **Router_init** function call in ***MCTinit_MOD1*** subroutine, which connects information about a second component '*MOD2_ID*', domain decomposition of the calling component '*GlobalSegMapMOD1*', and the communicator of the calling component '*COMM1*'. A similar procedure is employed to initialize the corresponding '*Router2*' communication table for the second model, in ***MCTinit_MOD2*** routine.

3.3. Run phase

Each model operates on its own set of processors according to split communicators *COMM1* and *COMM2*, and continues through its own time stepping loop, communicating amongst all the processors for that same model. At some point it reaches a user-defined model time, for instance *MCTtime*, where it communicates with the other models and exchanges data. Component models may have different time steps; the user has to make sure all component models have the information about *MCTtime* and are able to access it during model time integration. In our test cases, instantaneous model fields for a specific time were used for data exchange; MCT, however, has capabilities to perform time average of model forcing fields as well.

When each component model reaches *MCTtime*, it calls a corresponding subroutine ***MCTrun_MOD****n* (*n* is 1 or 2). This routine is called by each processor of the model and marks a convergence point where all the processors must synchronize for data exchange. Subroutine ***MCTrun_MOD1*** operates on attribute vectors *AV1_toMOD2* and *AV1_fromMOD2*. In a similar way, the second component operates on attribute vectors *AV2_toMOD1* and *AV2_fromMOD1*, to be sent to and received from MODEL1. A modeled data field to be transferred to the other component has to be linearized before loading into the MCT attribute vector. The linearization could be accomplished, for example, by loading a 2-D data from a domain segment into a working vector array '*avdata*' that is of the equivalent linear dimension as the 2-D data segment. Further, uploading this working array into a particular variable '*M1var1*' of attribute vector '*AV1_toMOD2*' could be done via a single function call:

call **AttrVect_importRAttr**(*AV1_toMOD2,"M1var1",avdata*)

As an example, MODEL1 fills its attribute vector with data to be transferred to MODEL2 (*AV1_toMOD2*) by importing all the variables ("M1var1"…"M1varn"). Then MODEL1 calls **MCT_Send** to transfer the attribute vector '*AV1_toMOD2*' to the MCT via the '*Router1*' previously established. MCT then redistributes the data using the '*GlobalSegMapMOD2*', and fills the attribute vectors *AV2_fromMOD1*. MODEL2 receives the attribute vector *AV2_fromMOD1* via its corresponding '*Router2*' by invoking **MCT_Recv.** In our implementation, this is a fundamental synchronization point of the coupled models, because **MCT_Send** and **MCT_Recv** are blocking commands. This means that the component model that sends the data with a special '*tag*' to MCT waits until the second component model receives that data with the same '*tag*', which completes a given data transfer session on both sides. Data transfer from MODEL2 to MODEL1 requires a separate set of **MCT_Send** and **MCT_Recv** calls. The form of data transfer sessions we implemented is as follows:

call **MCT_Send**(*AV1_toMOD2,Router1,tag1*)
(in ***MCTrun_MOD1,*** sending data via '*Router1*' computed by first model)
call **MCT_Recv**(*AV2_fromMOD1,Router2,tag1*)
(in ***MCTrun_MOD2,*** receiving the data by the second model)
call **MCT_Send**(*AV2_toMOD1,Router2,tag2*)
(in ***MCTrun_MOD2,*** sending data via '*Router2*' computed by second model)
call **MCT_Recv**(*AV1_fromMOD2,Router1,tag2*)
(in ***MCTrun_MOD1,*** receiving data by the first model)

Attribute vectors on a receiving end could be exported into a linear working array '*avdata*' for further use by a component in a way similar to the uploading process:

call **AttrVect_exportRAttr**(*AV2_fromMOD1,"M1var1",avdata, avlen*),

where *avlen* is the length of a linear data segment of '*M1var1*' variable, filling *avdata* working array.

After each component finishes its data transfer procedures, model integration resumes for another *MCTtime* period. The process is repeated for the remainder of the simulation.

3.4. Finalize phase

The last step is for all the components to finalize their calls and close all '*Routers*', attribute vectors, '*GlobalSegMap*'-s, MCT, and MPI communications. The MCT module ***ADD_MOD1*** may contain the following:

```
subroutine MCTend_MOD1
    call Router_clean(Router1)
    call AttrVect_clean(AV1_toMOD2)
    call AttrVect_clean(AV1_fromMOD2)
    call GlobalSegMap_clean (GlobalSegMapMOD1)
    call MCTWorld_clean()
end subroutine MCTend_MOD1
```

After model integration is completed for all components, and the corresponding MCT communications are closed, the execution returns to the '**Master**' program. It completes the simulation by closing all the MPI communications.

3.5. Sparse matrix interpolation

If the component model grids are not identical, MCT has the capability to perform matrix interpolation to remap the model fields from one horizontal grid to another. It is the user's responsibility, however, to provide corresponding matrices with remapping weights prior the simulation start. Separate programs could be used for that purpose, for example, SCRIP (Spherical Coordinate Remapping and Interpolation Package; Jones, 1999; http://climate.lanl.gov/Software/SCRIP/).

As the horizontal model grid meshes are linearized into vector arrays, the task or regridding becomes a matrix-vector multiplication, similar to $Y = M \cdot X$, where **Y** is a linearized vector array of global destination grid elements, **M** is multiplication matrix, and **X** is a vector of global source grid elements. If a horizontal global destination grid **Y** consists of *imax* by *jmax* grid elements, its linear size is *nRows*=*imax***jmax*; linear size of a destination grid defines a number of rows in the interpolations matrix **M**. Similarly, *nCols* is the linear size of a horizontal global source grid **X** that defines the number of columns in the matrix **M**. A total number of elements in interpolation matrix **M** is equal to *num_elements*=*nRows***nCols*. The interpolation stencil is generally fairly small (e.g., four adjacent points for bilinear), and thus **M** is very sparse matrix.

MCT creates a special data type object '*SparseMatrix*' to store sparse matrix weights in a compact form; it stores only non-zero elements with their corresponding column and row indices. First, user-supplied remapping weights and their index locations need to be loaded into working linear arrays *weights*, *rows*, *columns*. The sizes of these variables have to be identical, *size(weights)*, *size(rows)*, and *size(columns)*. Then, MCT sparse matrix '*sMat*' is created in the following way:

```
call SparseMatrix_init(sMat, nRows, nCols, num_elements)
call SparseMatrix_importGRowInd(sMat,rows,size(rows))
call SparseMatrix_importGColInd(sMat,columns,size(columns))
call SparseMatrix_importMatrixElem(sMat,weights,size(weights))
```

Next, we illustrate how to apply the matrix '*sMat*' for remapping. Consider an example that requires remapping data from the first component model grid to the second component model grid. Assume the interpolation is performed in the second component module (***MCTinit_MOD2***). Two sets of similar attribute vectors holding the same variables have to be initialized in the same module; one for the data on the source grid ('*AV_gridMOD1*'), and another for the remapped data on the second model grid ('*AV_gridMOD2*'). Remapping is performed on real fields only, associated with corresponding tokens in 'rList' of input and output attribute vectors. When input and output data vectors '*AV_gridMOD1*' and '*AV_gridMOD2*' are purely data-local, sparse matrix-vector multiplication could be achieved with a single function call as follows:

```
call sMatAvMult(AV_gridMOD1, sMat, AV_gridMOD2).
```

Here, the condition of total data locality implies that the input data vector contains all the values referenced by the local column indices of '*sMat*', and the output data vector contains all the values referenced by the local row indices of argument '*sMat*'. Note, that in this case '*sMat*', a '*SparseMatrix*' data object, holds information about remapping weights for the entire (global) domains of both component model grids.

When horizontal model grids are distributed over a number of processors, the input and output vectors will not be totally data-local. Then, matrix-vector multiplication process has to be explicitly distributed-memory parallel, performed on each processor for its domain segment. Another data type object, '*SparseMatrixPlus*', is defined to store the elements of **M** and all the information needed to coordinate data redistribution and reduction of partial sums.

Domain decomposition for two sets of grids needs to be defined by the corresponding '*GlobalSegMap*' objects, for example, '*GSMapMOD2_grid1*','*GSMapMOD2_grid2*'.

To perform distributed-memory parallel matrix-vector interpolation, i.e., interpolation performed on each processor for its domain segment, a data type '*SparseMatrixPlus*' is defined. It uses previously created '*sMap*' for global grid remapping, and information about domain decompositions, '*GSMapMOD2_grid1*' and '*GSMapMOD2_grid2*'. The corresponding function call to create this object, named below '*sMatPlus*', has the following form:

```
call SparseMatrixPlus_init(sMatPlus, sMat, 'GSMapMOD2_grid1',
    'GSMapMOD2_grid2', strategy, root, COMM2, MOD2_ID),
```

where '*strategy*' is a method of matrix decomposition by rows (*Xonly*), columns (*Yonly*), or both (*XandY*). As '*sMatPlus*' now encapsulates all the information necessary to perform a distributed-memory parallel matrix-vector multiplication, '*sMat*' data object is no longer needed, and it could be cleared from memory. The interpolation is then executed by a single function for matrix multiplication:

```
call MCT_MatVetMul('AV_gridMOD1', sMatPlus, 'AV_gridMOD2')
```

After this call, the attribute vector '*AV_gridMOD2*' holds data fields computed by the first component and remapped onto second component's grid. If attribute vectors '*AV_gridMOD1*' and '*AV_gridMOD2*' contain a number of attributes, as specified in their correspondent 'rList'-s, the function **MCT_MatVetMul** performs interpolation of multiple fields in a single library call.

4. Test cases

We have included several test cases to demonstrate the necessity for coupling and to compare the performance of the coupled system. The first example involves the ROMS–SWAN coupled system and demonstrates the significance of using two-way coupling for model prediction. The second example of COAMPS–ROMS code includes tests with variable model grids and resolutions, demonstrating advantages and limitations of remapping capabilities. Both coupled codes were implemented as single-executable systems, following flow chart shown in Fig. 2.

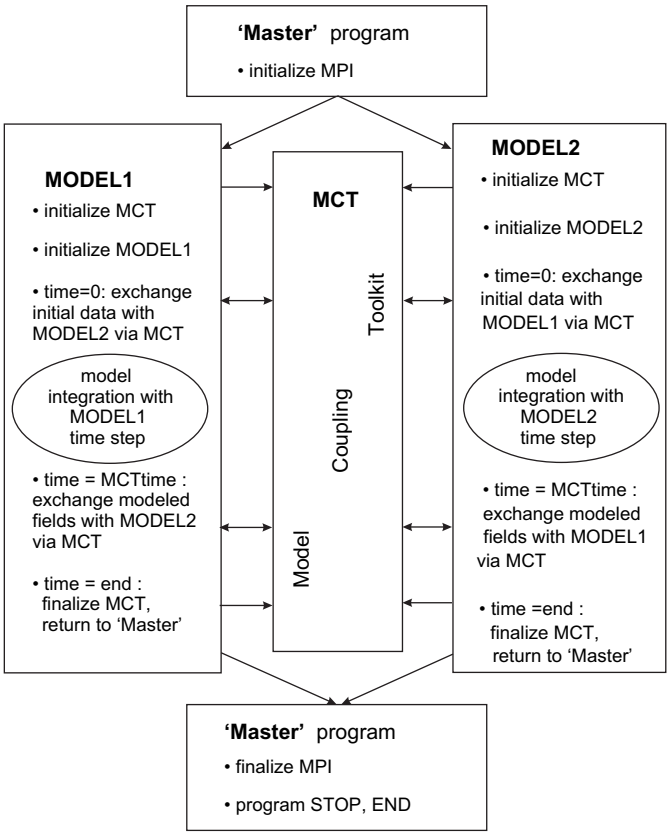

Fig. 2. A flow chart of the coupled code that forms a single-executable modeling system; its components run in concurrent mode.

4.1. ROMS–SWAN: Tidal inlet wave-current coupled system

This coupled example demonstrates the significance of wave–current coupling and includes ROMS coupled to SWAN. The test case is a simple tidal inlet system, based on a rectangular basin domain that is 15 km wide by 14 km in length. The entire basin is initialized with a uniform water depth of 4 m (Table 1). The seafloor is initialized as a 10 m thick bedlayer of uniform very fine 0.10 mm diameter sand. The backbarrier region (bottom) is enclosed with four walls, except for a 2 km wide inlet centered along the middle wall that connects the back region to the seaward domain. The seaward (top) region has northern, western, and eastern edges that are open (Fig. 3). The model is forced by oscillating the water level on the northern edge with a tidal amplitude of 1 m and a period of 12 h. Waves are also imposed on the northern edge with a height of 1 m, directed to the south with a period of 10 s. The water level oscillations drive the ocean circulation model and the wave amplitude drives the wave model. The model system is run with two configurations: (1) one-way coupled with wave information passed to the circulation model; (2) two-way fully coupled where the ocean model sends water velocities, water level, and bottom morphologic change to the wave model, and the wave model sends wave height, period, and wave length to the ocean model. For both configurations the modeling system is simulated for a period of 2 days. A morphologic scale factor of 10 provides an increased response of the seafloor evolution to the stresses imposed by the currents and waves. Thus the total simulation is considered to represent a 20-day period.

In the one-way coupled system the wave heights quickly evolve to a steady state. The wave heights decreasing southward toward the inlet and show no effect from the inlet currents (Fig. 3). The wave heights along the western and eastern edges are lower in amplitude because there were no waves imposed on these boundaries. The contour of wave height is straight across the inlet opening showing that there is no sign of the currents interacting with the waves, as specified for this one-way coupled application. At the peak of the ebb tide the combined wave-current bottom stresses on the seafloor are maximum near the location of maximum currents. As the tidal currents transport sediment, the bathymetry of the sea floor will evolve. Flood and ebb tidal shoals are produced with a larger ebb than a flood shoal. By contrast, the two-way coupled model results demonstrate that the wave heights are greatly increased in front of the inlet. The wave height contour lines show increased wave amplitudes in the inlet as the approaching wave interacts with an opposing current. The increased wave heights create combined bottom stresses that are greater than the one-way coupled system, and the peak bottom stresses are located near the maximum wave heights. The seafloor morphology evolves to produce a stronger ebb shoal than the one way coupled system due to the higher stresses and the shoal is displaced slightly further seaward. The coupling of the wave and ocean models demonstrates an important feature that the nearshore wave and circulation dynamics are mutually interactive and accurate modeling of this type of system should include these communications.

Table 1
Model parameters for the ROMS–SWAN test case 1

Model parameter	Variable	Value
length, width, depth	*Xsize, Esize, depth*	15000 m, 14000 m, 4.0 m
number of grid spacings	*Lm, Mm, Nm*	75, 70, 10
bottom roughness	z_{ob}	0.015 m
time step	*dt*	10 s
simulation steps	*Ntimes*	17280 steps (2 days)
morphology factor	*morph_fac*	10 (=20 day scaled simulation)
grain size	*Sd50*	0.10 mm
settle velocity	w_s	11.0 mm s^{-1}
erosion rate	E_0	5×10^{-3} kg m^{-2} s^{-1}
critical stresses	τ_{cd}, τ_{ce}	0.10 N m^{-2}
porosity	φ	0.50
bed thickness	*bed_thick*	10.0 m
northern edge tide	A, T_t	1.0 m, 12 h
northern edge wave height	H_{sig}	2 m
northern edge wave period	T	10 s
northern edge wave direction	θ	from 0° (from North)

4.2. ROMS–COAMPS

The coupled system for the next example includes ROMS as the ocean component and COAMPS as the atmospheric component. The simulations below analyze and compare tests with identical grids in both components vs. tests where model grid resolutions differ and MCT remapping capabilities are used. Domain location and model initial conditions were chosen to represent typical summertime conditions off the central Oregon coast with its relatively straight north-south oriented coastline. Strong and persistent northerly winds during the summer season in this area create conditions where warm surface waters are transported offshore and replaced by upwelled colder bottom waters, forming a narrow strip of cool surface temperatures within 10–50 km from the coast. The spatial scales of this upwelling are typically under-resolved by common operational forecast models. Because of the simple, linear configuration of the Oregon coast, we use an idealized two-dimensional simulation domain.

The control case has a horizontal domain 50 km in latitudinal direction (x-direction) and 20 km in the longitudinal (y-direction), with 1-km grid spacing ($dx = dy = 1$ km); grid spacing was similar in both model components. The simulations were numerically two-dimensional; the along-channel grid points (y-direction) were added only to assure proper numerical execution of the atmospheric model code. Test cases in the present paper are organized by their corresponding grid box sizes in atmosphere and ocean models in cross-shore (dx_{atm},dx_{oce}) in kilometers; given this scheme, the control case is further referred as **case(1,1)**. The horizontal domain is located over the coastal ocean and excludes land; the eastern boundary of the ocean domain is a "coastal wall" boundary condition (BC). The western boundary of the ocean domain has open BCs; the atmospheric model has open BCs for both east and west boundaries. Periodic north–south BCs in both models are used to form a periodic channel. Ocean bathymetry is uniform in alongshore y-direction; it changes linearly in the cross-shore x-direction from 10 m at the eastern edge of the domain to about 300 m at the western edge about 50 km offshore. Model time step was 3 s for the atmospheric model, and 300 s for the ocean model; data exchange via MCT was performed every 300 s (the larger time step of the two models).

The coupled model forecast is run for 72 h. Forcing in the atmospheric model is prescribed using a constant 15 m/s geostrophic northerly wind, whereas the ocean is started at rest. Horizontally-homogeneous initialization is chosen for temperature and moisture in the atmospheric model, and for temperature and salinity in the ocean model. For the atmospheric model, the vertical profile of potential temperature linearly increased from 287 K at sea level to 350 K at 9 km elevation. For the ocean model, the temperature was set at 14 °C for a surface mixed layer depth of 20 m, a sharp thermocline with a decrease in temperature to 10 °C at 40 m, and then a linear decrease in temperature to 6 °C at 300 m. The salinity reflected a similar shape as the thermocline and increased from 32.5 at the surface to nearly 34 at depth. During data exchange sessions via MCT, the ocean model receives predicted momentum (wind stress) and heat fluxes from the atmospheric model, and atmosphere receives sea surface temperature (SST) updates from the ocean model. More analysis of the similar case studies and the

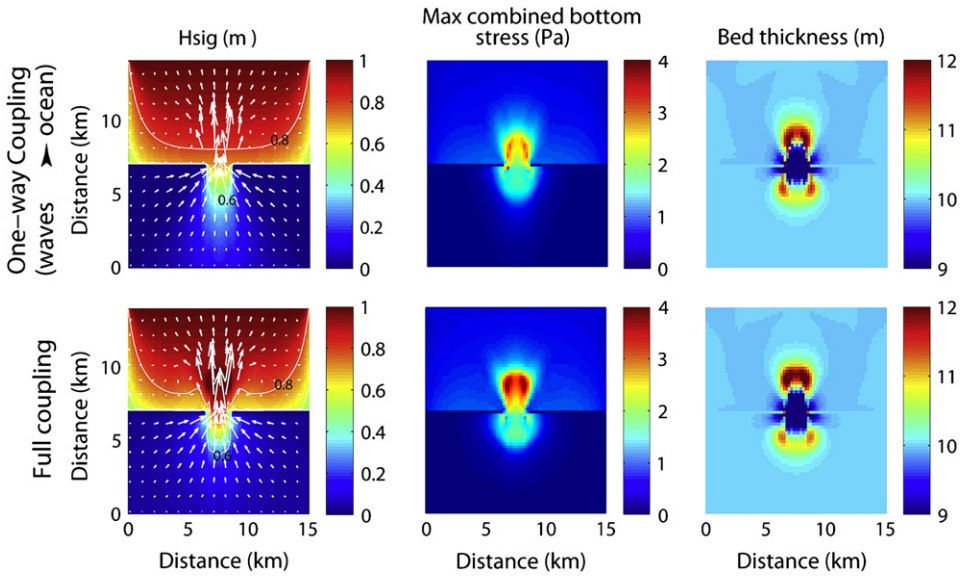

Fig. 3. ROMS–SWAN coupling tidal inlet test case. Comparison of significant wave height, maximum bottom stress, and bed thickness for a one-way coupled simulation (wave parameters sent to ocean model) to a fully coupled simulation (wave parameters to ocean model and ocean data to wave model). For the fully coupled system, the wave heights show effect of currents, the maximum bottom stress in enhanced due to increased wave heights, and bed thickness develops a stronger ebb shoal than the one-way coupled system.

effects of model coupling on ocean and atmosphere boundary layers' formation could be found in Perlin et al. (2007).

Additional cases were run with similar settings, but variable grid spacing. In some experiments, grid box size was similar in both atmosphere and ocean models; according to the naming convention accepted earlier, these simulations are labeled as **case(0.5, 0.5), case(3,3),** and **case(9,9**). Three other experiments smaller grid box size in cross-shore direction (*dx*) for the ocean model; these are labeled as **case(1,0.25)**, **case(3,1)**, and **case(9,3).** Grid box sizes in y-direction (*dy*) remained identical for ocean and atmosphere model pairs in each of the cases. To account for different model resolution in x-direction, remapping of fields from one model grid to another was needed. In some cases, the differences in model resolutions dictated choosing smaller or larger time steps accordingly, and also extension of model domain further offshore in coarser resolution simulations. These differences nevertheless allowed making direct comparisons between the cases in order to analyze the effects of remapping.

Results of 72-h forecasts show that upwelling developed at the eastern domain boundary in all the cases, with sea surface temperatures (SST) in the control **case(1,1)** about 4 °C colder than the offshore temperature (Fig. 4). One of the atmospheric responses to the cold water near the coast is reduced wind stress (Fig. 5); the primary atmospheric forcing for upwelling circulation has an important feedback to the ocean during the run. The control case resulted in a 42% reduction of the nearshore meridional wind stress in comparison with values 20 km or more offshore.

Decreasing grid spacing in half in **case(0.5,0.5)** as compared to **case(1,1)**, produced almost identical SST fields and meridional wind stress across the domain. This indicates that the intensity and spatial scale of the upwelling circulation were resolved sufficiently good with the 1-km grid. SST in **case(1,0.25)** was also similar to the control case, except for slight variations within 2–3 km of the coast. In **case(1,0.25)** a band of colder water consisting of several grid points extended almost all the way to the coast. Effects of decreasing resolution become notable in **case(3,3)**: the nearshore cold SST region was represented by a single grid point, and coastal wind stress reduction was 34% versus ~42% in the control case. However, with the identical atmospheric model resolution in **case(3,3)** and **case(3,1)**, decreased grid spacing in ocean model in the latter case produced SST values very similar to the control case. Further decrease in model resolution affected the SST prediction more notably. With the closest grid point center located at 4.5 km off the coast in **case(9,9)**, a critical nearshore region is excluded from the model

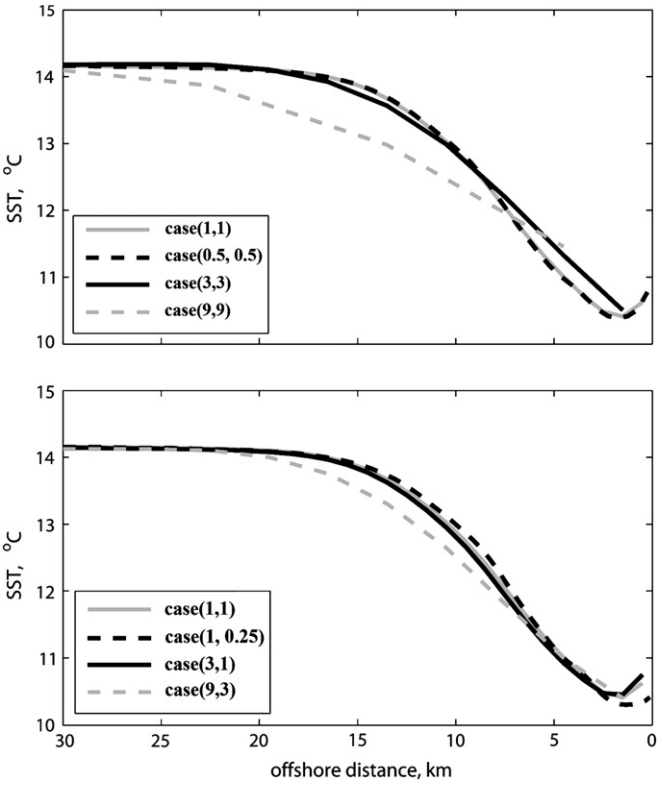

Fig. 4. Ocean model 72-h forecast of sea surface temperature (SST),°C, in cross-shore direction for COAMPS–ROMS test cases. Upper panel shows cases with similar resolutions and computational grids in both ocean atmosphere components; lower panel shows a control simulation, **case(1,1)**, and cases that involve interpolation for grid remapping. Plotted SST values correspond to the center of a grid box. Having the coastal wall at the eastern boundary, the closest grid point to the coast is plotted at 0.5 km offshore distance for 1-km grid box, at 1.5 km offshore for 3-km grid box, at 4.5 km offshore for 9-km grid box, and so on.

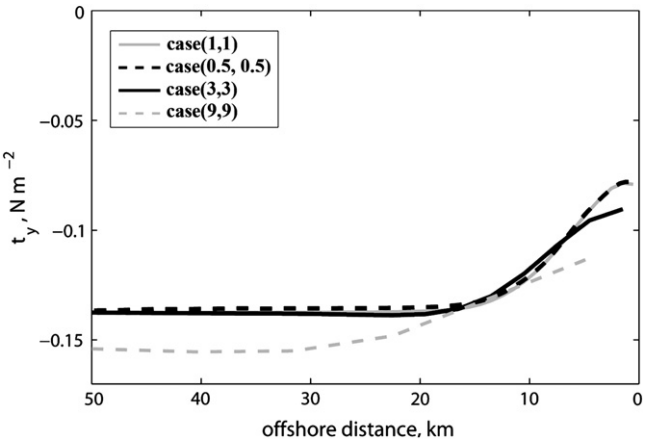

Fig. 5. Atmospheric model 72-h forecast of meridional wind stress (τ_y), N m^{-2}, in cross-shore direction for four test cases with different horizontal resolution in the atmospheric component of the COAMPS–ROMS coupled system.

calculations, and the lowest SST in this simulation is 0.8 °C warmer than in the control case. The spatial scale of the upwelling circulation in **case(9,9)** is larger than in other cases, and extends at least 30 km off the coast, as seen in both SST and meridional wind stress plots. Wind stress reduction was about 30%. Decreased ocean model grid box size in **case(9,3)** corrected the scale and intensity (SST) of upwelling. The coldest temperature at the last inshore point in **case(9,3)** was similar to minimum temperature in **case(1,1)**.

Comparison of ocean cross-sections of turbulent kinetic energy (TKE) at the end of the simulation (Fig. 6) serves another good illustration of useful remapping capabilities. Areas of elevated TKE near the surface and the bottom indicate locations of surface and bottom boundary layers, respectively. As water shoals near the coast, the two boundary regions tend to merge and form a mixed layer across the entire water depth. Note the similarities in TKE prediction in the nearshore 5 km in **case(1,1), case(0.5,0.5)**, and **case(3,1)**, whereas TKE in **case(9,9)** is overestimated. The TKE fields are similar for **case(3,3)** and **case(9,3)** which had identical ocean model resolution; merging of the boundary layers for the latter case occurred further offshore, as marked by "1" isoline, in comparison with the higher-resolution cases. The **case(9,9)** differs significantly from the other simulations because of poor resolution in the coastal shallow water region.

Notable improvement in SST prediction from increased ocean model resolution is a valuable result because the atmospheric model takes most of the computational time. The atmospheric model is more costly because of smaller time step required: 3 s in **case(1,1)**

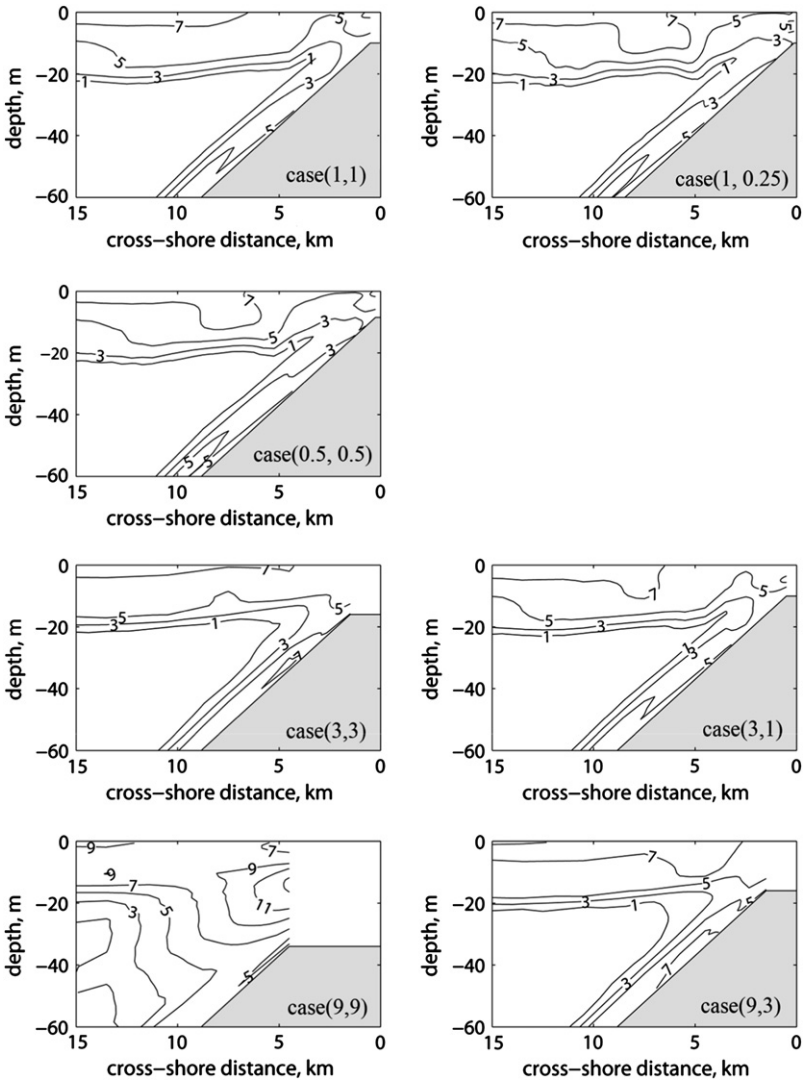

Fig. 6. Ocean model cross-section of turbulent kinetic energy, $10^4 * m^2 s^{-2}$, at the end of 72-h simulation. The coast is at "0" distance. Note that nearshore ocean bathymetry depends on model resolution, and data points indicate the middle of a grid box (for **case(9,9)** a nearshore grid point is 4.5 km from the coast).

versus a time step of 300 s in the ocean model. Use of a coarser resolution for the atmospheric model grid therefore helps to reduce computational costs of the coupled runs, without degradation of the ocean model prediction. However, decrease in atmospheric model resolution may lead to smoothing of smaller-scale circulation features, which may be important for estimation of turbulent properties in the upper ocean (mixing) layer.

5. Discussion

The MCT method provides a rather unintrusive technique to allow model coupling in that each model runs as a separate entity, with data exchanges at predetermined synchronization intervals. Each model operates on its own grid and with independent input and output functions. However, there are drawbacks to this formulation. For example each model may require redundant information, such as a surface wind field. In the current formulation the same data would be required multiple times but most likely in different file formats. A more intrusive method would be to use the core computational components from each model and develop new components to perform the I/O for the coupled system. This could potentially be more efficient, but would require fragmenting the individual models and require more code development.

MCT is easy to install and we have successfully compiled the libraries on systems ranging from a dual core PC using Windows XP and MPICH2 to a 76-processor AMD Linux cluster. Computational costs for the MCT component itself are typically small, but it does add to the overall computational effort of the coupled system. A single user may be able to develop a more efficient data reconstruction and dissemination formulation, but the MCT provides a standard method and allows different developers to share common formulations and increases portability.

The efficiency of the coupled system can be increased by determining optimal load balance on the processors. If MODEL1 reaches the synchronization point first and requires a long wait time, then this severely increases the total computation time. Processor allocation should be arranged to allocate processors amongst MODEL1 and MODEL2 so that both models reach the synchronization point as close to the same time as possible. It is difficult to predict exactly how the load balance should be allocated initially, as this is very dependent on the type of simulation being performed.

When two models are dynamically coupled the behavior of each model may be different than before. The dynamic feedback can cause the models to evolve into a state that is not expected. Users need to experiment with time stepping, spatial resolutions, and synchronization intervals to address the impact of coupling on their particular application. Additionally, boundary conditions on one model may now affect both models. Also, if two model grids are not identical then developers need to be cautious on how to handle the regions not computed by one of the models.

6. Conclusions

We used the Model Coupling Toolkit (MCT) to develop coupled modeling systems of MODEL1 running on "M" processors and MODEL2 running on "N" processors. In such a system, each model component is formulated in an initialize, run, and finalize structure. A '**Master**' program controls the execution of the modeling system and allocates processors to each model. A new module consisting of several subroutines needs to be written for each of the component models, and we describe in detail the structure and function of these modules and routines. The routines create global segment maps that describe how the grid is divided amongst different processors for each model, the creation of attribute vectors that hold the data, and routers that transfer the data. The models initialize, enroll into MCT, and transfer data using MPI-based MCT protocols to efficiently send and receive model fields during model execution.

We demonstrate the application of two coupled model systems: an ocean–wave and an ocean–atmosphere system. All the employed models had structured grids, and coupling was implemented on the horizontal plane. MCT can be adapted to handle unstructured grids and future implementations will provide interpolation in the vertical dimension. Results suggest that the coupled systems provide more realistic simulations. A set of simulations is presented demonstrating the capabilities of using sparse matrix interpolation, when the two models have different grid resolutions. The remapping requires the user to first obtain a matrix with interpolation weights, which can be computed with the SCRIP package. Future versions of the coupling can easily call the package during initialization. The MCT is found to be a relatively simple and non-intrusive communication library for dynamic coupling of existing earth system models.

References

Booij, N., Ris, R.C., Holthuijsen, L.H., 1999. A third-generation wave model for coastal regions, Part I, Model description and validation. Journal of Geophysical Research 104 (C4), 7649–7666.

Burk, S.D., Haack, T., Samelson, R.M., 1999. Mesoscale simulation of supercritical, subcritical, and transcritical flow along coastal topography. Journal of Atmospheric Sciences 56, 2780–2795.

Craig, A., Jacob, R., Kauffman, B., Bettge, T., Larson, J., Ong, E., Ding, C., He, Y., 2005. Cpl6: The new extensible, high-performance parallel coupler for the community climate system model. International Journal of High Performance Computing Applications 19, 309–327.

Giustolisi, O., Doglioni, A., Savic, D.A., Webb, B.W., 2007. A multi-model approach to analysis of environmental phenomena. Environmental Modelling & Software 22, 674–682.

Haidvogel, D.B., Arango, H.G., Budgell, W.P., Cornuelle, B.D., Curchitser, E., DiLorenzo, E., Fennel, K., Geyer, W.R., Hermann, A.J., Lanerolle, L., Levin, J., McWilliams, J.C., Miller, A.J., Moore, A.M., Powell, T.M., Shchepetkin, A.F., Sherwood, C.R., Signell, R.P., Warner, J.C., Wilkin, J., 2008. Ocean forecasting in terrain-following coordinates: formulation and skill assessment of the Regional Ocean Modeling System. Journal of Computational Physics 227, 3595–3624.

Hermann, A.J., Haidvogel, D.B., Dobbins, E.L., Stabeno, P.J., 2002. Coupling global and regional circulation models in the coastal Gulf of Alaska. Progress in Oceanography 53, 335–367.

Hill, C., DeLuca, C., Balaji, V., Suarez, M., da Silva, A., the ESMF Joint Specification Team, 2004. The architecture of the earth system modeling framework. Computing in Science and Engineering 6, 18–28.

Hodur, R.M., 1997. The Naval Research Laboratory's Coupled Ocean/Atmosphere Mesoscale Prediction System (COAMPS). Monthly Weather Review 125, 1414–1430.

Jacob, R., Larson, J., Ong, E., 2005. M×N Communication and Parallel Interpolation in Community Climate System Model Version 3 using the model coupling toolkit. International Journal of High Performance Computing Applications 19, 293–307.

Jones, P.W., 1999. First- and second-order conservative remapping schemes for grids in spherical coordinates. Monthly Weather Review 127, 2204–2210.

Jorba, O., Loridan, T., Jiménez-Guerrero, P., Pérez, C., Baldasano, J.M., 2008. Linking the advanced research WRF meteorological model with the CHIMERE chemistry-transport model. Environmental Modelling & Software 23, 1092–1094.

Larson, J., Jacob, R., Ong, E., 2005. The model coupling toolkit: A new Fortran90 toolkit for building multiphysics parallel coupled models. International Journal of High Performance Computing Applications 19, 277–292.

Lau, L., Young, R.A., McKeon, G., Syktus, J., Duncalfe, F., Graham, N., McGregor, J., 1999. Downscaling global information for regional benefit: coupling spatial models at varying space and time scales. Environmental Modelling & Software 14, 519–529.

Leung, L.R., 2005. Regional climate modeling: Research needs and direction. WRF/MM5 Users Workshop, Boulder, CO. June 27–30, 2005.

Mellor, G.L., 2003. The three-dimensional current and surface wave equations. Journal of Physical Oceanography 33, 1978–1989.

Mellor, G.L., 2005. Some consequences of the three-dimensional currents and surface wave equations. Journal of Physical Oceanography 35, 2291–2298.

Michalakes, J.G., 2003. Infrastructure development for regional coupled modeling environments, Final Project Report. Contract No. N62306-01-D-7110, Task Order No. 0041, Project ID: CWO-03-002, September 2003.

Militello, A., Zundel, A.K., 2002. Coupling of regional and local circulation models ADCIRC and M2D. Coastal and Hydraulics Engineering Technical Note ERDC/CHL CHETN-IV-42. U.S. Army Engineer Research and Development Center, Vicksburg, MS.

Perlin, N., Skyllingstad, E., Samelson, R., Barbour, P., 2007. Numerical simulation of air-sea coupling during coastal upwelling. Journal of Physical Oceanography 37, 2081–2093.

Pickett, M.H., Paduan, J.D., 2003. Ekman transport and pumping in the California Current based on the U.S. Navy's high-resolution atmospheric model (COAMPS). Journal of Geophysical Research 108 (C10), 3327, doi:10.1029/2003JC001902.

Ris, R.C., Holthuijsen, L.H., Booij, N., 1999. A third-generation wave model for coastal regions, Part II, Verification. Journal of Geophysical Research 104 (C4), 7667–7681.

Samelson, R., Barbour, P., Barth, J., Bielli, S., Boyd, T., Chelton, D., Kosro, P., Levine, M., Skyllingstad, E., Wilczak, J., 2002. Wind stress forcing of the Oregon coastal ocean during the 1999 upwelling season. Journal of Geophysical Research 107 (C5), 3034, doi:10.1029/2001JC000900.

Shchepetkin, A.F., McWilliams, J.C., 2005. The Regional Ocean Modeling System: A split-explicit, free-surface, topography-following coordinates ocean model. Ocean Model 9, 347–404, doi:10.1016/j.ocemod.2004.08.002.

Warner, J.C., Sherwood, C.R., Signell, R.P., Harris, C.K., Arango, H.G., in press. Development of a three-dimensional, regional, coupled wave, current, and sediment-transport model, Computers and Geosciences.