Interoperable Physics Driver (IPD) and Common Community Physics Package (CCPP)

Jimy Dudhia (NCAR/MMM) on behalf of DTC's GMTB IPD Design Team (Laurie Carson, Grant Firl, Don Stark with input from Jim Rosinski and Ligia Bernadet)

NGGPS Physics Workshop 11/09/2016

Definitions

- IPD - Interoperable Physics Driver – An interface that separates the physics from the dynamical core in such a way that the physics is swappable for a given dycore and the dycore is swappable for a given physics suite
  - Also to be usable in standalone mode (single-step column or unit testing), or as single-column model (time-evolving forcing)
- CCPP – Common Community Physics Package – Suites of physics designed to be called from the IPD

History

- NUOPC PI Team provided guidelines
- Initial implementation was done for NGGPS to port GFS physics to various dycores (Patrick Tripp)
- GMTB, EMC, NUOPC PI teams worked on requirements
- Here, GMTB is proposing a design to meet the updated requirements
- Development of this driver will start at GMTB in close connection with partners

Overview

- IPD is geared to column physics only (not Land) to allow for limited definable set of state input variables and outputs
  - *Q: How to deal with 3d physics that either needs area averages as input or distributes tendencies over areas on output?*
  - *Requirements do not address 3d physics yet*
  - *A: We will write a column IPD first (since all existing proposed physics is columnwise).*
- IPD functions as an intermediary providing a *generic* interface for both dynamics and physics
  - *Generic* in the sense that it has a specified set of input arguments that must be filled by the dycore/solver and *necessary* output arguments that must be provided by the CCPP in addition to *diagnostic* outputs
    - *Necessary* outputs such as tendencies or updates are required by the model to advance time steps
    - *Diagnostic* outputs may be dependent more on the CCPP rather than considered necessary for the running of the model, and are in this sense optional and probably more variable
- IPD Layer is independent of both the dynamical core and (as much as possible) the underlying CCPP
  - Therefore needs converter layers
    - Dynamics to IPD
    - IPD to CCPP
  - Has its own intermediate state variable names independent of dycore and CCPP
- IPD's main operation is to call physics or sub-sets of it as specified by its own call

Overview

- IPD is geared to column physics only (not Land) to allow for limited definable set of state input variables and outputs
  - *Q: How to deal with 3d physics that either needs area averages as input or distributes tendencies over areas on output?*
  - *Requirements do not address 3d physics yet*
  - *A: We will write a column IPD first (since all existing proposed physics is columnwise).*

Overview

- IPD functions as an intermediary providing a *generic* interface for both dynamics and physics
  - *Generic* in the sense that it has a specified set of input arguments that must be filled by the dycore/solver and *necessary* output arguments that must be provided by the CCPP in addition to *diagnostic* outputs
    - *Necessary* outputs such as tendencies or updates are required by the model to advance time steps
    - *Diagnostic* outputs may be dependent more on the CCPP rather than considered necessary for the running of the model, and are in this sense optional and probably more variable

Overview

- IPD Layer is independent of both the dynamical core and (as much as possible) the underlying CCPP
  - Therefore needs converter layers
    - Dynamics to IPD
    - IPD to CCPP
  - Has its own intermediate state variable names independent of dycore and CCPP
- IPD's main operation is to call physics or sub-sets of it as specified by its own call

NEMS System Architecture Overview

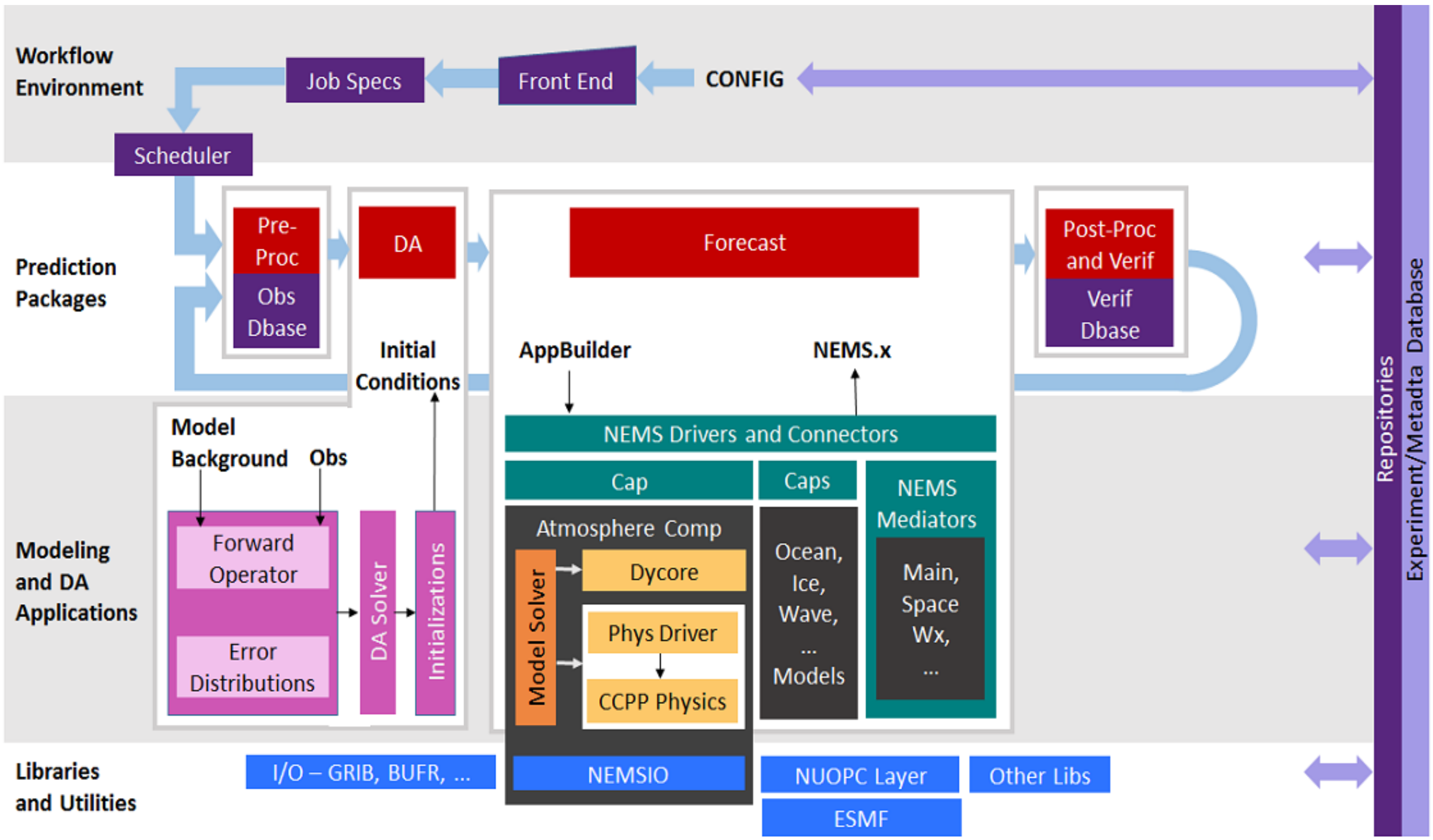


Diagram created by NGGPS Overarching System (OAS) Team

NEMS System Architecture Overview

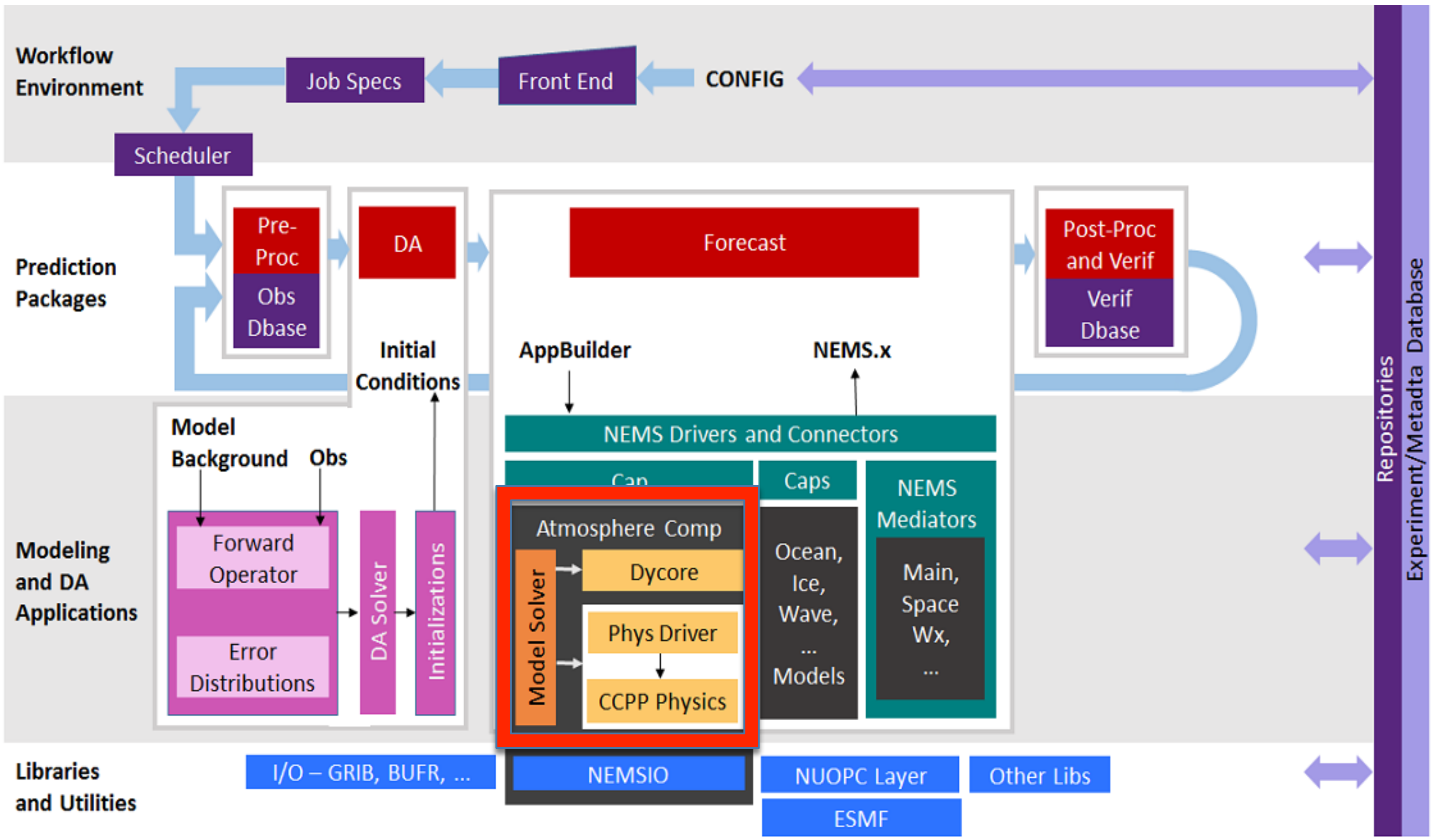


Diagram created by NGGPS Overarching System (OAS) Team

Developmental Testbed Center

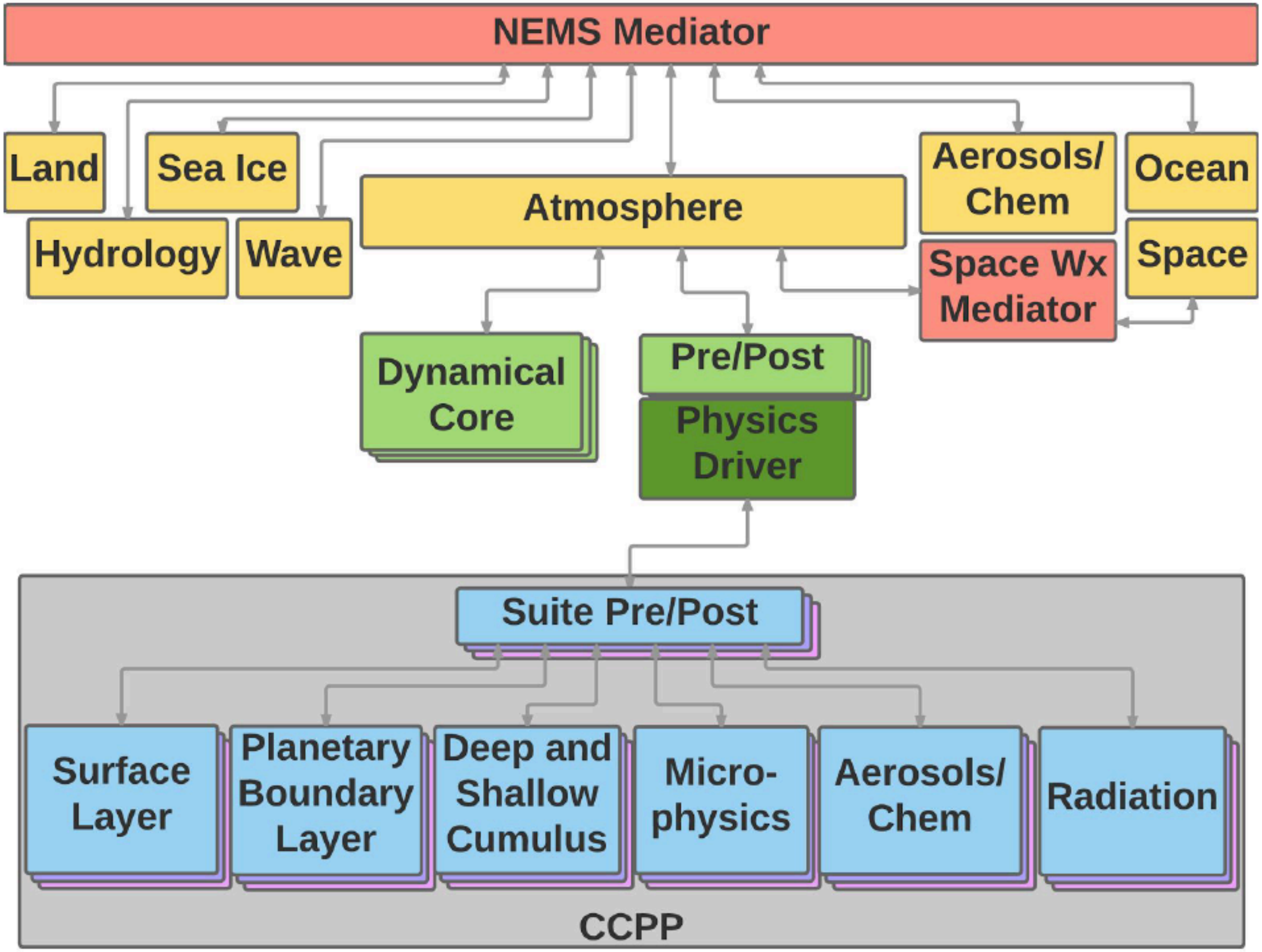
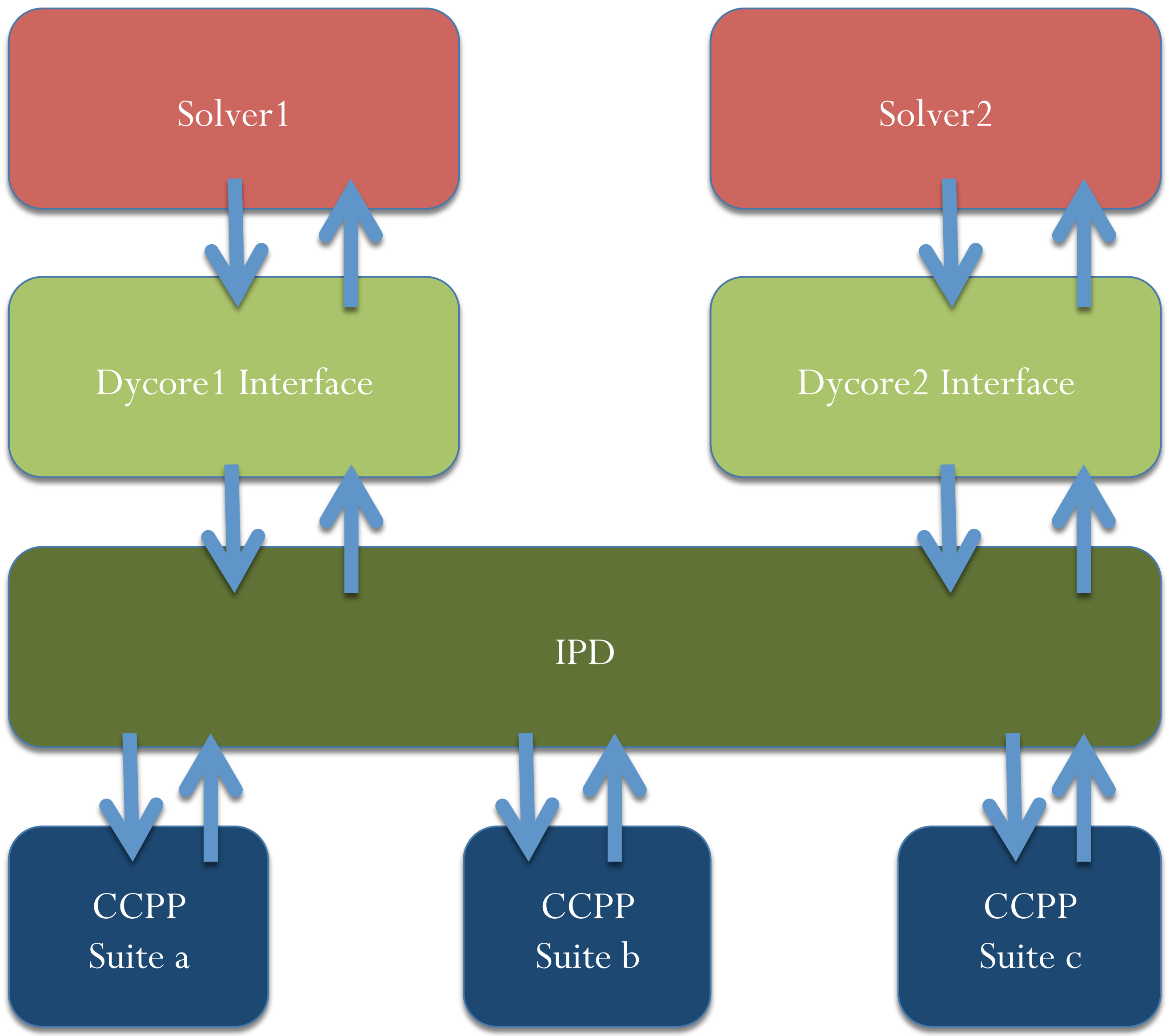


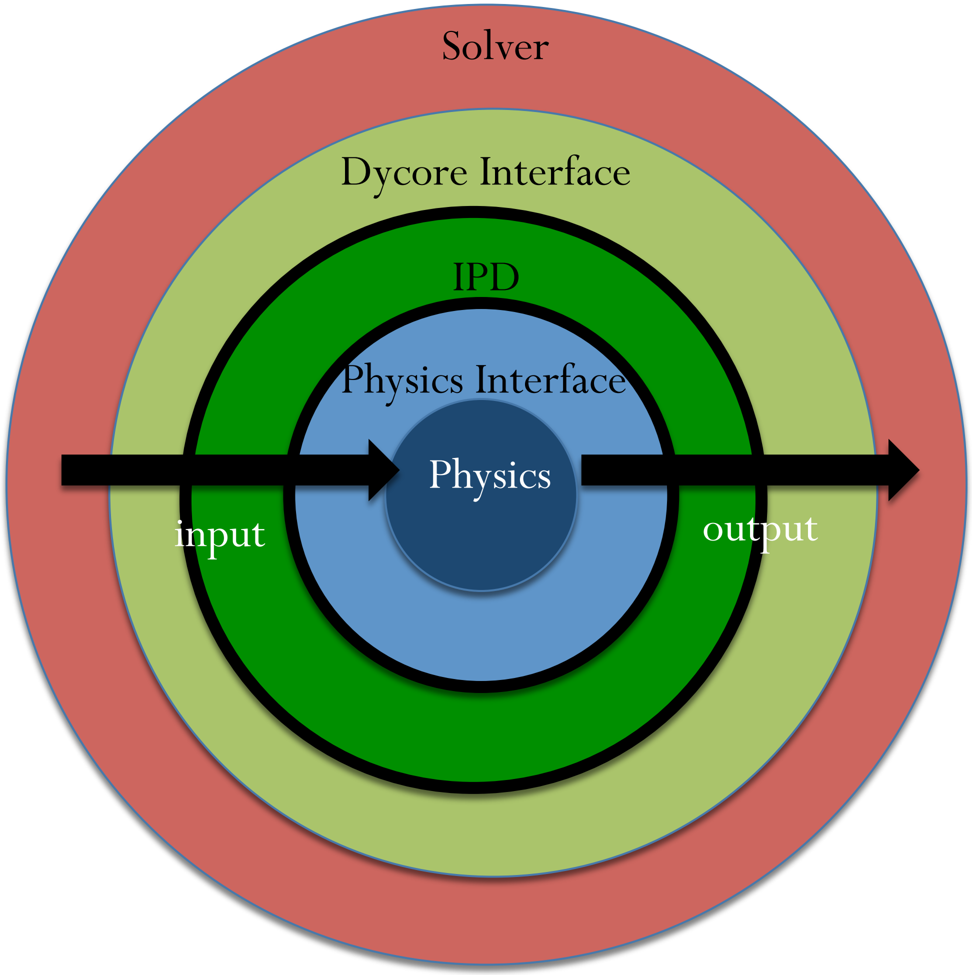
Diagram from IPD and CCPP: Goals and Requirements Document

Hierarchical View



Wrapper View

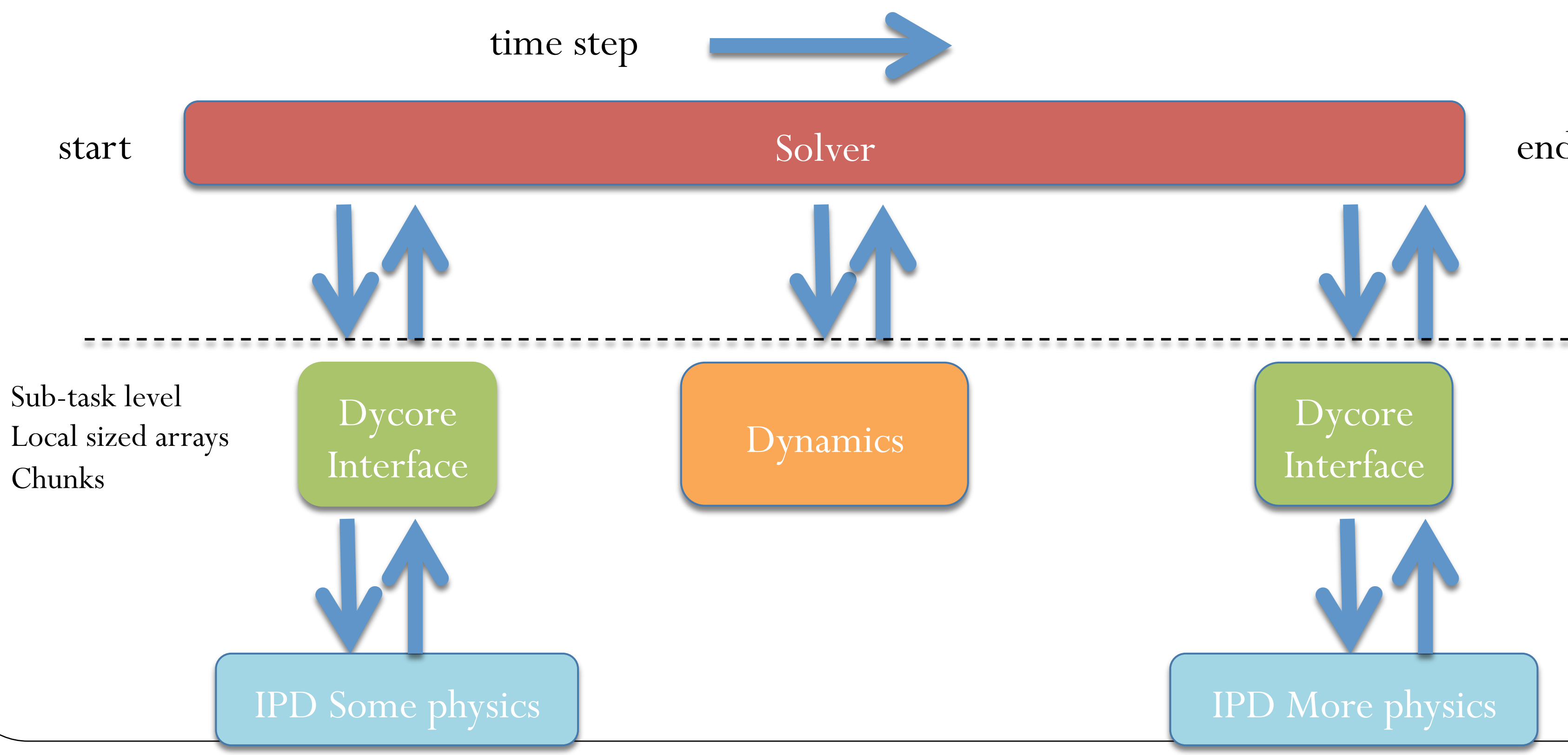
A "wrapper" is a code layer that has functions both before and after its main call(s)



- Note that each layer has an input and output function
- IPD has its own pre-defined input and output variables represented by thicker line – these serve to insulate the physics from the dycore

Calling

- IPD can be called at the sub-task (thread/chunk) level
  - Dycore Interface could include multiple calls to IPD depending on physics required in each call



IPD Design Diagram

Dycore Interface

Input dycore variables

Subsetting 3d to (i,k), etc.

Conversions to IPD(=units, flipping, staggering, combining etc.)

Fill IPD input state

Calling IPD

Conversions back to dycore(…combining tendencies, etc.)

Fill 3d dycore tendency/update variables

Output dycore variables

IPD

Input IPD variables

Calling CCPP interface(s)

Outputs IPD tendencies/updates

CCPP Interface (individual physics driver)

Input IPD variables

Conversions to physics

Fill physics input args

Calling physics

Conversions to IPD

Fill IPD tendency/update arrays

Outputs IPD physics tendencies/updates

Physics Scheme

Input required variables

Outputs physics tendencies/updates

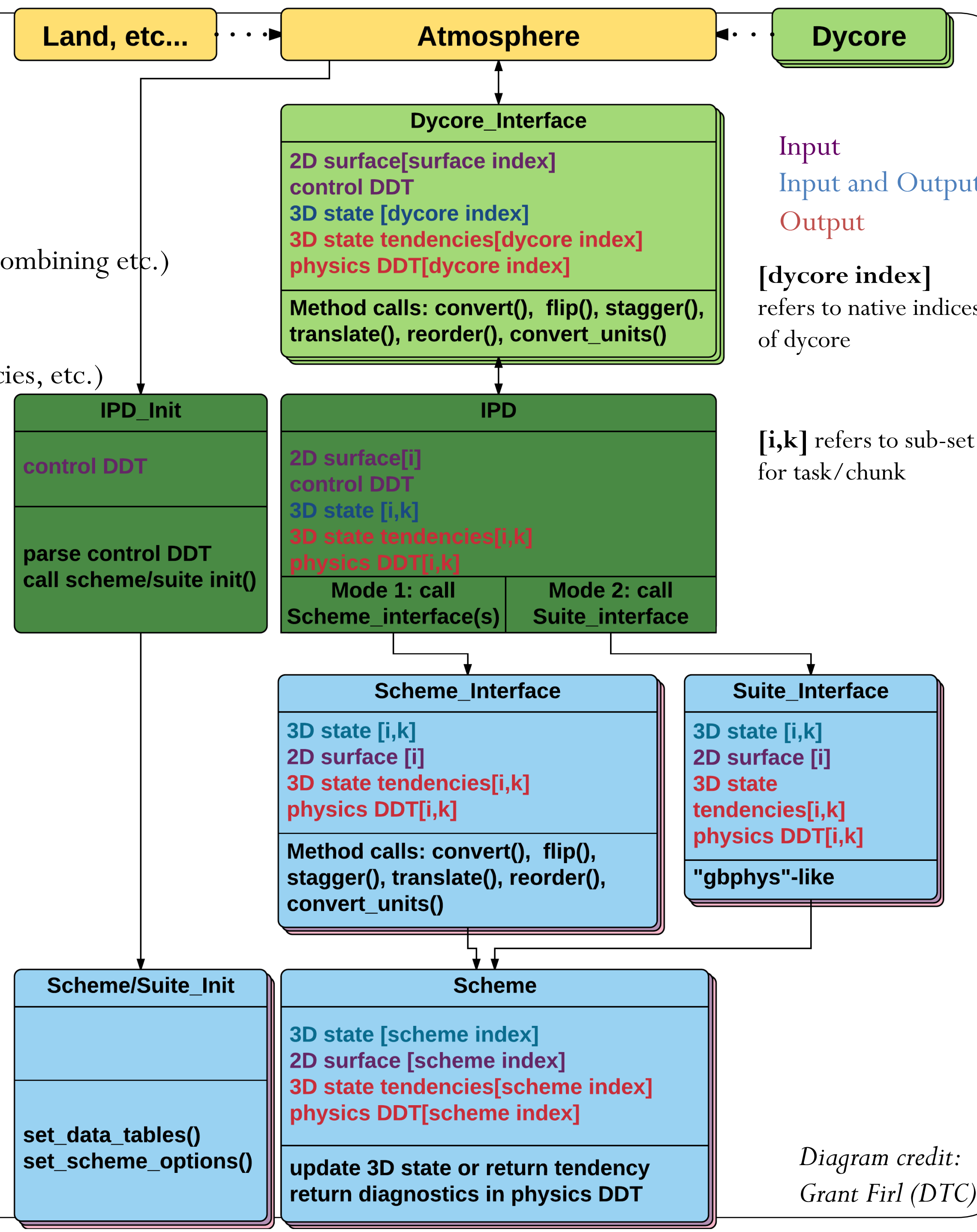


Diagram credit: Grant Firl (DTC)

Dycore Interface Conversions (Dycore/ IPD)

- IPD contains only task-sized (chunked) local arrays (i,k) or (i)
- Dycore state arrays are 3d (i,j,k) or (i,k,j), or 2d (i,j)
- "conversions" needed in dycore interface include
  - Subsetting 3d arrays to 2d slices
  - Flipping vertical index (if necessary, maybe IPD can inherit vertical index direction from dycore)
  - Translating variable names to IPD names
  - Calculating derived variables (e.g. RH)
  - Converting units (e.g. g/kg to kg/kg)
  - De-staggering velocities to physics column center
  - Interpolating to interface levels (e.g. T, z, and p)
- Reverse of some of these is needed after IPD, e.g. operating on tendencies/updates
- Note that many of these operations can be combined for efficiency, e.g. translating, flipping and sub-setting

Physics Interface Conversions (IPD/ Physics)

- IPD contains only task-sized local arrays (i,k) or (k) where k index should be in pre-defined direction, e.g. bottom-top
  - Their shape would depend on how the dycore does threads
- Physics interface may need to convert (i,k) to (k) if physics is only columnwise
- "conversions" needed in physics interface should be minimal but may include
  - Subsetting 2d slices to 1d columns
  - Flipping vertical index (if physics k is opposite to IPD and/or dycore)
  - Translating variable names to physics names (can be done through call)
  - Calculating any more needed derived variables for that physics (e.g. θ_v)
  - Converting units (e.g. MKS to cgs)
- Reverse of some of these is needed after physics, e.g. operating on tendencies
- Note again that many of these operations can be combined for efficiency, e.g. translating, flipping and sub-setting

Calling Control

- IPD can be called in "*init*" mode before run to call initialization routines of physics
  - e.g., reading in tables, creating look-up tables, initializing specific arrays and constants
- In "*run*" mode, IPD can be called with logicals that define which physics is to be executed at that time-step and for that IPD call
  - May also define whether IPD provides updates to state variables or just physics tendencies

IPD Generic Schematic

Subroutine IPD (*input args, output args, control args, …*)

Input args: *IPD state variables, e.g. T, U,V, etc.*

Output args: *IPD tendencies and/or updated IPD state variables*

Control args: *controls which physics to call*

IPD call

- *Call individual physics interfaces (IPD state input, IPD tendencies/updates output, diagnostics output, internal and external exchange variables output)*
- *OR Call suite interface if physics shares same variables already*

End Subroutine IPD

args would be collected logically into several Derived Data Types (DDTs), such as **ipd%statein**, and the variables within these DDTs could be *ipd%statein%temperature*, or *ipd%stateout%rad_temp_tend*, or *ipd%land_exchange%heat_flux*, or *ipd%diagnostic%2m_temperature*, but naming has not been decided yet.

IPD Schematic with Example

Subroutine IPD (*input args, output args, control args, …*)

Input args: *IPD state variables, e.g. T, U,V, etc.*

Output args: *IPD tendencies and/or updated IPD state variables*

This is an example sequence as might be used by WRF

1st IPD call

- *If radiation step - Call radiation driver* (IPD state input, IPD tendencies output)*
- *Call surface-layer driver (IPD state input, Land-specific output)*
  - *Return to solver for Land call (Atmosphere-Land coupling)*

2nd IPD call

- *Call pbl driver (IPD state input, Land-specific input(fluxes), IPD tendencies output)*
- *If cumulus step - Call cumulus driver (IPD state input, IPD tendencies output)*
  - *Return to solver for dynamics*

3rd IPD call

- *Call microphysics driver (IPD state input, IPD update output)*

End Subroutine IPD

** The term 'driver' is used for the CCPP-specific physics option's interface and its only role is to call the single CCPP physics routine of this class*

IPD Schematic with 2nd Example

Subroutine IPD (*input args, output args, control args, …*)

Input args: *IPD state variables, e.g. T, U, V, etc.*

Output args: *IPD tendencies and/or updated IPD state variables*

This is an example sequence as might be used by GFS with its embedded Land component

1st IPD call

- *If radiation step - Call radiation driver (IPD state input, IPD tendencies output)*
  - *Radiation and ozone*

2nd IPD call

- *Call GFS suite driver (IPD state input, IPD updates/tendencies output)*
  - *Suite includes surface-layer, land, pbl, gravity wave drag, cumulus*

End Subroutine IPD

IPD Variable Categories

- State Variables – Atmospheric model state variables or directly derived from these variables
- Tendency Variables – outputs from IPD required to advance state variables (can also be updated state variables)
- Internal Exchange Variables – variables passed between column physics components
- External Exchange Variables – variables passed to/from other model components (e.g. Land, Ocean, Chemistry)
- Diagnostic Variables – variables output from column physics but not necessary for advancing model state
- Constants – IPD should also pass through physical constants common to the dynamics and physics

Common Community Physics Package (CCPP)

Goals

- To contain physics packages usually as part of suites in a plug-compatible way for the IPD
- Plug-compatibility requires following coding rules and standards
- Physics schemes are each called by their own driver that interfaces to the IPD. This driver
  - Inputs state variables, exchange variables from other parts of physics (e.g. heat fluxes), constants
  - Outputs state variables (e.g. tendencies or updated variables), exchange variables to other physics (e.g. cloud information), diagnostics
  - Calls the main physics

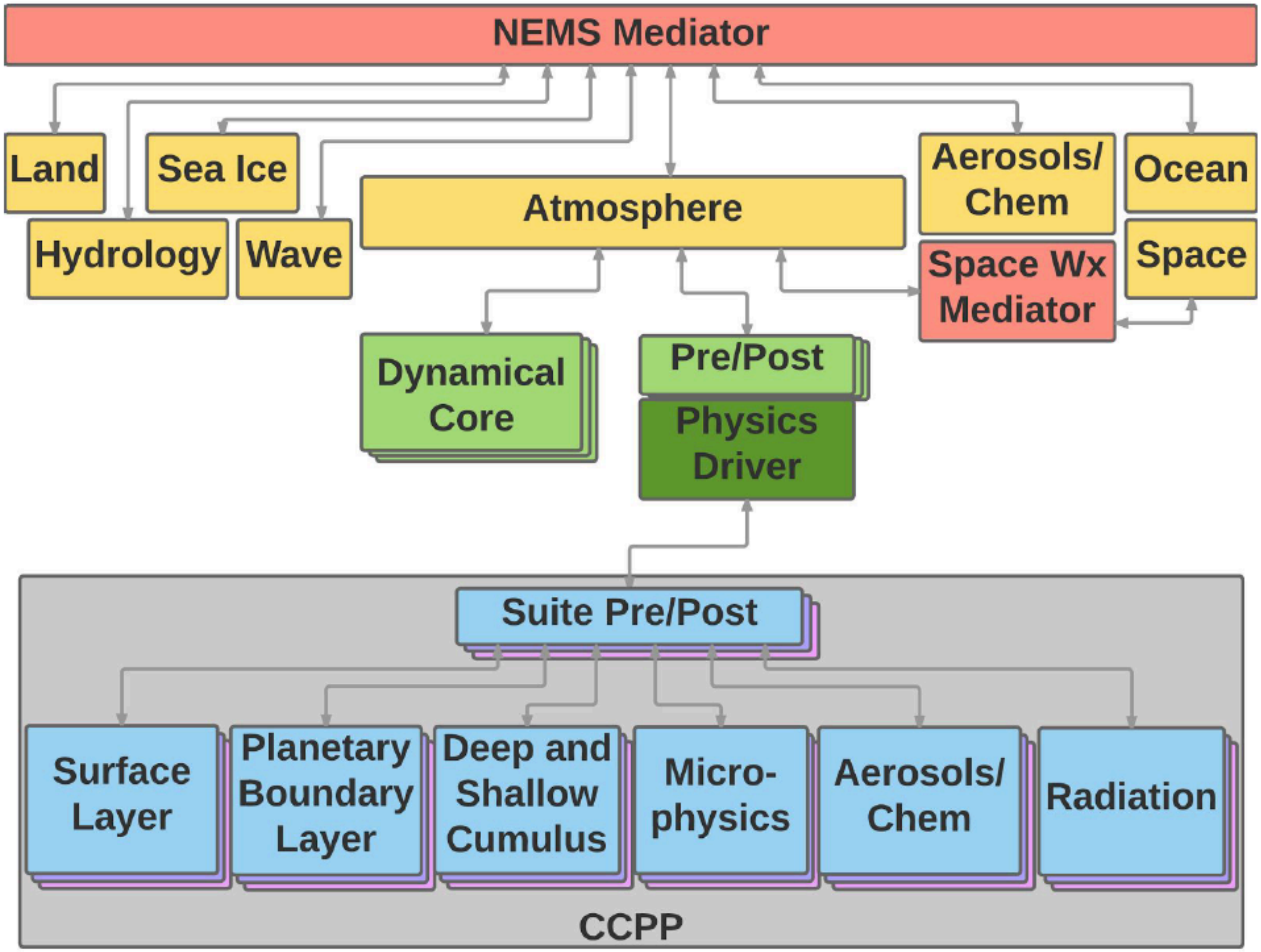


Diagram from IPD and CCPP: Goals and Requirements Document

Summary of IPD and CCPP Aims

- IPD itself should have no operations apart from calling physics suites or sub-sets of physics
  - It serves to pass through variables in both directions – input state down from the dycore's solver and tendencies/diagnostics back up from the physics
- IPD call for a specific CCPP suite should look the same from any dycore's interface
- IPD's physics interface calls should look as much as possible the same for any CCPP suite
  - Differences that can't be avoided will be due to exchange variables and diagnostics – but these differences could be just within DDTs
  - Commonality will be the IPD state and tendency inputs and outputs that the dycore provides and needs

Next Steps

- Solicit input on general plan from all interested groups before implementation
- Implementation of IPD
  - IPD code itself is very short and would be done first after DDTs are defined
  - This defines how its call(s) should look from the dycore interface
  - This will also have calls to the physics interface routines that can be used to develop a matching physics interface routine

Extra slides

- Example variable lists for DDTs
- Summary of IPD Requirements list

IPD I/O Variables

State 3d input

Pressure (hydrostatic)
Pressure (thermodynamic)
Height
Temperature
Theta
Density
Dry Density
Exner Function
Winds (u and v)
Vertical Velocity
Pressure Velocity
Water vapor mixing ratio
Hydrometeor mixing ratio
Relative Humidity
Specific Humidity
*Advective tendencies**

State 2d input

Heat Flux (from Land)
Moisture Flux (from Land)
Surface Pressure
Surface (Ground) Temperature
Albedo(s) (from Land)
Emissivity (from Land)
Roughness Length (from Land)

State 3d output

Radiation (lw/sw) tendencies/updates

- *Temperature*

Cumulus (deep/shallow) tendencies/updates

- *Temperature*
- *Vapor*
- *Hydrometeors*
- *Momentum*

PBL/vertical diffusion tendencies/updates

- *Temperature*
- *Vapor*
- *Hydrometeors*
- *Momentum*

Microphysics tendencies/updates

- *Temperature*
- *Vapor*
- *Hydrometeors*

State 2d output

Exchange coefficients (for Land)

- *Heat*
- *Moisture*

Precipitation (for Land)
Downward radiative fluxes (for Land)
Surface stress (for Ocean)

IPD Internal Exchange Variables

Depend on CCPP suite

examples

Microphysics particle radii for radiation

PBL tke for cumulus

Surface-layer stress for PBL

Cloud fractions for radiation

- Microphysics/RH diagnostic
- Cumulus (deep or shallow)

IPD External Exchange Variables

Depend on CCPP suite

Examples

Land-Physics

- Heat and moisture fluxes
- Surface properties (e.g. albedo, roughness length)

Ocean/Sea-Ice/Wave-Physics

- Sea-surface temperature
- Sea-ice fraction
- Wave stress

Chemistry-Physics interactions

- Aerosols as CCN and IN for microphysics
- Aerosol optical depth for radiation
- Ozone, trace gases, for radiation

IPD Diagnostic Outputs

Accumulated fields

Radiative fluxes (2d, possibly 3d)

- Longwave/shortwave, Top/bottom, upward/downward, clear/total

Precipitation (2d)

- Total, rain, snow, graupel from microphysics/cumulus

Surface fluxes of heat and moisture (land/ocean) (2d)

Physics tendencies (outputs accumulated over time) (3d)

Physics individual processes (e.g. latent heating from condensation) (3d)

Diagnostics

2m T and q, 10m wind

Reflectivity (3d)

Radiances (e.g. cloud-top brightness temperatures) (2d)

Solar energy (diffuse, direct surface components)

Wind energy (hub-height wind)

Max/min/mean/std

Surface temperature, wind, moisture, rainrate (e.g. daily for climate runs)

Storm tracking (max updraft, reflectivity, helicity, hail size) (e.g. hourly)

Requirements

- D1: Agnostic of dynamic core
- D2: Easily configurable entry point for passing information to/from physics parameterizations
- D3: expandable
- D4: can switch individual parameterizations
- D5: can activate parameterizations as a suite or individually
- D6: order and frequency of calls to individual parameterizations is configurable

Requirements

- D7: can share constants between dycore and physics
- D8: documentation
- D9: modern coding practices and standards, performance, portability
- D10: able to drive parameterizations in "offline" mode
- D11: able to work on "chunks" of dycore/solver variables
- D12: able to provide diagnostic variables for output

Requirements

- D13: able to provide physics variables to/from external models (coupled land, ocean, etc.)
- D14: IPD will not modify answers produced by parameterizations
- D15: ability for runtime changing of physics parameters
- D16: unambiguous naming
- D17: scientist-friendly coding