The Climate Machine (CLIMA)

Climate Modeling Alliance

November 10, 2018

1 Overview

To achieve a step change in the accuracy and precision of climate simulations and predictions, we are developing CLIMA, an Earth system model (ESM) that learns automatically from diverse data sources. Our goal is to use observational data along with modern computational methods not simply to evaluate and test models, as is current practice, but to systematically reduce model-data mismatches and quantify uncertainties. To accomplish this goal, we will harness, simultaneously and self-consistently, orders-of-magnitude more data than are currently used in model development, exploiting new techniques from data assimilation (DA) and machine learning (ML).

1.1 Automated Learning From Observations and High-Resolution Simulations

We will design an ESM platform with subgrid-scale (SGS) process models that learn automatically from two sources of information (for more details and references, see *Schneider et al.* (2017a), from which several passages of this document are taken):

1. *Global observations.* We live in the golden age of Earth observations from space. A suite of satellites is streaming coordinated and nearly simultaneous measurements of variables such as temperature, humidity, clouds, ocean surface currents, and sea ice cover, with global coverage for more than a decade. Space-based measurements of biogeochemical tracers and processes, such as measurements of column-average CO2 concentrations, of ocean biomass, and of photosynthesis in ecosystems, are also available, and so are more detailed observations of the cryosphere. To date, only a minute fraction of the data (mostly large-scale energy fluxes) has been directly used in ESM development (as opposed to ESM evaluation). CLIMA will learn directly from global data, augmented and validated with more detailed local observations where available.

2. *Local high-resolution simulations.* Some SGS processes in ESMs are in principle computable, only the globally achievable resolution precludes their explicit computation. For example, the turbulent dynamics of clouds can be computed with high fidelity in limited domains in large-eddy simulations (LES) with mesh sizes of meters to tens of meters. Increased computational performance has made

LES domain widths of 10–100 km feasible in recent years, while the horizontal mesh size in climate models has shrunk, to the point that the two scales have converged (*Schneider et al.*, 2017b). Thus, while global LES that reliably resolve low clouds will not be feasible for decades, it is now possible to nest LES in limited areas of atmosphere models and conduct targeted local high-fidelity simulations of cloud dynamics in them. Local high-resolution simulations of ocean turbulence or sea ice dynamics can be conducted similarly. CLIMA will learn from such nested high-resolution simulations.

Simultaneously exploiting global observations and local high-resolution simulations with new DA/ML tools presents the key opportunity for progress in Earth system modeling. Replacing the inefficient and sub-optimal manual tuning process and the offline fitting of parameterization schemes to data from few locations, as is currently common, CLIMA will autotune itself and quantify its uncertainties based on statistics of a much larger range of available data. It will adapt as new observations come online. It will also generate targeted high-resolution simulations on demand—akin to targeted observations in weather forecasting (*Palmer et al.*, 1998; *Lorenz and Emanuel*, 1998)—to reduce and quantify uncertainties in SGS models of computable processes. This will increase the amount of data to which SGS models are fitted by orders of magnitude.

1.2 Optimization Over Aggregate Climate Statistics

The automated learning from observations and high-resolution simulations in CLIMA will use *statistics accumulated in time* (e.g., over seasons) to:

1. Minimize model biases, especially biases that are known to correlate with the climate response of models. This amounts to minimizing mismatches between time averages of ESM-simulated quantities and data.

2. Minimize model-data mismatches in higher-order Earth system statistics. This includes covariances such as cloud-cover/surface temperature covariances, or ecosystem carbon uptake/surface temperature covariances, which are known to correlate with the climate response of models ("emergent constraints," or fluctuation-dissipation relations). It can also include higher-order statistics involving direct targets for prediction goals, such as high percentiles of the rainfall distribution.

By optimizing how well an ESM simulates climate statistics, our approach directly targets success metrics that are relevant for climate projections. Optimizing over climate statistics ameliorates problems arising from ill-posedness of the inverse problem (underdetermination of SGS models given data), and it avoids difficulties caused by sensitive dependencies on atmospheric initial conditions and small-scale roughness. These difficulties arise when optimizing over snapshots of Earth system states, as in numerical weather prediction. For example, the challenge of simulating when and where clouds occur, with temporal and spatial accuracy (e.g., hours to minutes in time and kilometers to hundred meters in space), has prevented the routine assimilation of space-based radar and lidar observations of clouds in numerical weather predictions (*Stephens et al.*, 2018). But if these same data are aggregated, for example, over seasons, precisely when and where individual clouds occur, and their sensitive dependence on atmospheric initial conditions, become less important. The aggregated cloud statistics vary smoothly in time and space, and minimizing mismatches in them directly targets what matters for climate projections. This is where the key untapped opportunity lies for CLIMA to radically improve upon existing models.

The problem of minimizing model-data mismatches in climate statistics is computationally challenging because accumulating statistics from an ESM is costly: each evaluation of the target statistics requires an ESM simulation at least over a season. But the computational problems are just beginning to be tractable, for example, with the ensemble-based inversion methods we will develop and employ.

1.3 Computable and Noncomputable Parameters

Learning from local high-resolution simulations and observations is aimed at determining two different kinds of parameters in parameterization schemes: *computable* and *non-computable* parameters. (Since parameters and parametric functions of state variables play essentially the same role in our discussion, we simply use the term parameter, with the understanding that this can include parametric functions and even nonparametric functions.) Computable parameters are those that can in principle be inferred from high-resolution simulations alone. They include parameters in radiative transfer schemes, which can be inferred from detailed line-by-line calculations; dynamical parameters in cloud turbulence parameterizations, such as entrainment rates, which can be inferred from LES; or parameters in ocean mixing parameterizations, which can be inferred from high-resolution simulations. Non-computable parameters are parameters that, currently, cannot be inferred from high-resolution simulations, either because computational limitations make it necessary for them to also appear in parameterization schemes in high-resolution simulations, or because the microscopic equations governing the processes in question are unknown. They include parameters in cloud microphysics parameterizations, which are still necessary to include in LES, and many parameters characterizing ecological and biogeochemical processes, whose governing equations are unknown. Cloud microphysics parameters will increasingly become computable through direct numerical simulation, but ecological and biogeochemical parameters will remain non-computable for the foreseeable future. We will denote computable parameters by $\boldsymbol{\theta}_c$ and non-computable parameters by $\boldsymbol{\theta}_n$. Jointly, they form the parameter vector $\boldsymbol{\theta} = (\boldsymbol{\theta}_c, \boldsymbol{\theta}_n)$.

Both computable and non-computable parameters can, in principle, be learned from observations; the only restrictions to their identifiability come from the well-posedness of the learning problem and its computational tractability. But only computable parameters can be learned from targeted high-resolution simulations. To be able to learn computable parameters, it is essential to represent non-computable aspects of a parameterization scheme consistently in the high-resolution simulation and in the parameterization scheme that is to learn from the high-resolution simulation. For example, radiative transfer and microphysical processes need to be represented consistently in a high-resolution LES and in a parameterization scheme if the parameterization scheme is to learn computable dynamical parameters such as entrainment rates from the LES. Otherwise, the optimization over climate statistics will reduce mismatches between statistics from LES and parameterization schemes that arise from differences in the representation of radiative transfer or microphysical processes by erroneously adjusting computable parameters, leading to aliasing errors.

1.4 Fresh Model Architecture

CLIMA will have a fresh model architecture (see the schematic in Fig. 1). Current ESM architectures make it difficult to carry out the hundreds to thousands of ESM simulations that are necessary for DA/ML based on climate statistics, and current parameterization schemes are not well suited for DA/ML approaches (e.g., because they contain many correlated parameters). We will need to design CLIMA to learn efficiently from observations and to run nested high-resolution simulations on demand, and we will need to replace several existing parameterization schemes by flexible SGS process models that learn effectively from diverse data sources. The SGS process models need to be systematically refine-able as more data become available, and they need to treat subgrid-scale motions (e.g., boundary layer turbulence, shallow convection, deep convection) in a unified manner. Achieving this will be a central focus of our development effort.

To achieve rapid payoffs in terms of reduced uncertainties in climate projections, we will focus the early development on the most uncertain components for which we have already begun development and prototyping of new SGS process models (e.g., for clouds, convection, and turbulence). Initially, we will use other components (e.g., radiative transfer schemes) from existing models. However, we seek to eventually replace most current parameterization schemes by more flexible SGS process models designed for DA/ML approaches.

To be fast and to run at the highest resolutions feasible, CLIMA will also need to effectively exploit the computing architectures that are currently emerging, including heterogeneous many-core architectures that combine traditional CPUs with hardware accelerators such as graphical processing units (GPUs) or Field Programmable Gate Arrays (FPGAs). Targeted high-resolution simulations nested in a global ESM are ideally suited for running on accelerators, which shine at solving computational problems that can be decomposed into largely independent subtasks that are comparable in computational effort. Many high-resolution simulations—potentially hundreds or thousands—can be run concurrently with the coarse-resolution ESM, avoiding idle computational threads that otherwise limit accelerator efficiency. We may also want to exploit emerging AI accelerator designs, when they are broadly available, in the DA/ML components of the ESM platform.

2 CLIMA Layers and Goals

CLIMA consists of several layers (Figs. 1 and 2). Wrapped around all components of what traditionally would be the ESM is a DA/ML layer, whose function it is to connect the ESM with observations and targeted high-resolution simulations, to learn about uncertain SGS processes in the ESM. For example, the DA/ML layer allows the ESM to learn about uncertain SGS models of ocean turbulence through matching simulated statistics of ocean turbulence to those observed by satellite altimeters. Or, the DA/ML

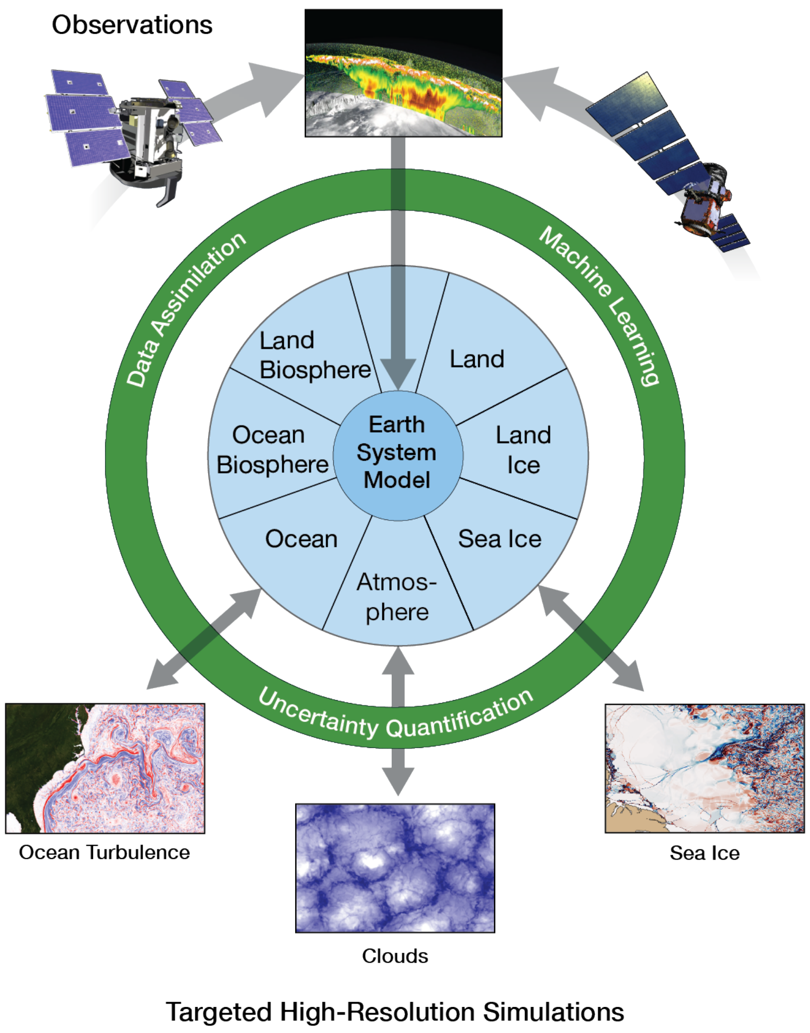

Figure 1: **Schematic of CLIMA.** Observations of the Earth system (e.g., from space) and output from targeted high-resolution simulations (e.g., of ocean turbulence) are passed through a data assimilation/machine learning (DA/ML) layer, which wraps around an Earth system model (ESM) with its component models. The high-resolution simulations that inform the ESM through the DA/ML layer are spun off from component models such as the atmosphere or ocean models. The DA/ML layer not only optimizes the ESM components but also quantifies uncertainties about model processes.

layer allows the ESM to learn about uncertain SGS models of clouds, convection, and atmospheric turbulence through matching ESM-simulated statistics of cloud cover and cloud condensate to those obtained from LES spun-off from the atmosphere model.

To design CLIMA to learn from diverse data sources, we will employ the key concepts and pursue the goals described in what follows, moving layerwise from the outside to the computational core.

2.1 Observations

Concepts

- Observations are primarily space-based or ground-based observations with large spatial coverage.
- Observations can also come from field campaigns, which provide more detailed data that are local in space and time. It remains to be seen whether we want to learn from such local data online in CLIMA, or whether we primarily want to use local data in off-line development and testing of SGS models and to provide prior information for learning in CLIMA.
- Observables $\boldsymbol{y}$ are linked to state variables $\boldsymbol{x}$ of the ESM through a map $\mathcal{H}$ representing an observing system, so that

  $$\boldsymbol{y}(t) = \mathcal{H}\big(\boldsymbol{x}(t)\big). \tag{1}$$

  The observables $\boldsymbol{y}$ might represent surface temperatures, cloud cover, or spectral radiances emanating from the TOA. The map $\mathcal{H}$ projects ESM state variables $\boldsymbol{x}$ to observables at the locations and times at which actual observations, denoted by $\tilde{\boldsymbol{y}}$, are available. For complex observing systems (e.g., satellites), the map $\mathcal{H}$ represents an observing system simulator and hence can be complicated. The observations $\tilde{\boldsymbol{y}}$ are independent of the parameters $\boldsymbol{\theta}$.

Goals

- For the atmosphere, focus first on learning from reanalysis data, including seasonally and spatially varying statistics of variables such as atmospheric temperature and precipitation for the satellite era (from 1980 onward). Reanalyses provide easily accessible and homogenized data, for which the map $\mathcal{H}$ is a simple projection operator. This will allow us to prototype and develop the CLIMA platform more quickly than would be possible if we were to integrate a large set of diverse observational data sources from the outset, which each requires a separate observing system simulator $\mathcal{H}$.
- For the oceans, focus first on the ECCO ocean reanalysis together with boundary layer statistics generated from the ARGO program. ECCO assimilates most global ocean observations into the MIT global ocean model and generates a time evolving ocean state estimate spanning the last fifteen years. The ARGO floats

provide continuous profiles of upper-ocean statistics that are particularly relevant for Earth's climate, such as sea-surface temperature, salinity, and boundary layer depth. While the ECCO product assimilates the ARGO profiles, its vertical resolution is much coarser and degrades that of the original profiles. We will use statistics from the higher-resolution ARGO data that are available as gridded data.[1]

- As soon as possible, incorporate selected satellite data products where reanalysis data do not provide adequate information about uncertain SGS processes. Initially, this will likely include cloud data products from platforms such as CloudSat, CALIPSO, and MODIS.
- Over time (likely years 4–5), incorporate other observations (e.g., satellite data products) in CLIMA.

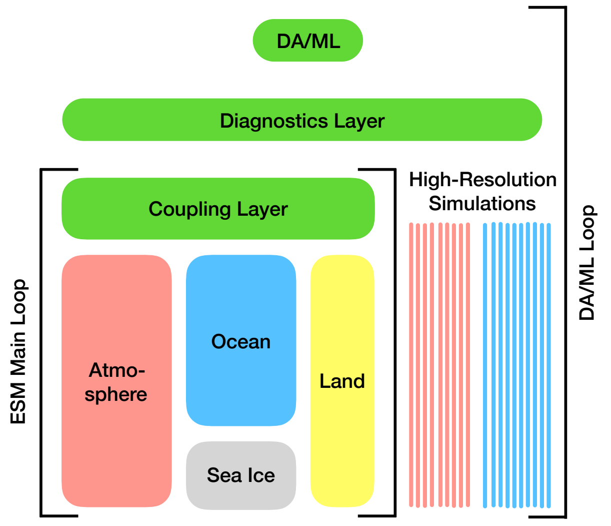

Figure 2: **Schematic of a possible CLIMA computing architecture.** Time increases downward, and processors are indicated across. The different horizontal and vertical extents of the components suggest the degree of parallelism and time of execution (not to scale). The coupling layout that is suggested here is one possibility: parallel coupling of atmosphere, ocean, and land (component concurrency), sequential coupling of ocean and sea ice, and concurrent high-resolution simulations, e.g., for atmosphere and ocean. Other layouts are possible, and the final layout remains to be decided. Also, other components in addition to those indicated (e.g., a land ice model) may eventually be incorporated into CLIMA.

[1]http://www.argo.ucsd.edu/Gridded_fields.html

2.2 Targeted High-Resolution Simulations

Concepts

- Local high-resolution simulations can provide information about uncertain computable parameters $\boldsymbol{\theta}_c$ in SGS processes where observations alone do not provide sufficient information about them. This can include high-resolution simulations of atmospheric clouds, convection, and turbulence, of ocean turbulence both on the mesoscale and submesoscale, and of sea ice dynamics.

- Local high-resolution simulations can also be used to provide detailed climate information where needed, e.g., for local climate impact projection, or as detailed boundary conditions over ice sheets.

- Local high-resolution simulations are nested within small areas of the respective ESM component models. For example, atmospheric LES are spun off from the atmosphere model component, nested in small areas (e.g., a grid column) of the global model. Ocean mesoscale-resolving simulations are spun off from the ocean model, in ocean patches that represent a small fraction of the globe, yet are large enough that nonlocal effects (e.g., advection by low-frequency flow components) can be meaningfully represented.

- One-way nesting suffices for the high-resolution simulations, which simplifies the embedding of the high-resolution simulations. That is, the global component model drives local high-resolution simulations, but the high-resolution simulations do not feed back onto the global model (except through the information they provide on SGS models). However, even one-way nesting requires the time-evolving large-scale conditions of the host model as driving and boundary conditions for the high-resolution simulations.

- Conceptually, local high-resolution simulations may be viewed as a time-dependent map $\mathcal{L}$ from ESM state variables $\boldsymbol{x}$ to simulated state variables $\tilde{\boldsymbol{z}}$,

  $$\tilde{\boldsymbol{z}}(t) = \mathcal{L}(\boldsymbol{\theta}_n, t; \boldsymbol{x}). \tag{2}$$

  The map $\mathcal{L}$ is parameterized by time t and by parameters $\boldsymbol{\theta}_n$ that are not computable in the high-resolution simulation and are inherited from the global model (e.g., microphysical parameters in an atmospheric LES). The map $\mathcal{L}$ can depend on the time-history of the state variables $\boldsymbol{x}$ up to time t, e.g., as time-evolving large-scale driving and boundary condition for the high-resolution simulation.

- The vector of simulated state variables $\tilde{\boldsymbol{z}}$ contains high-resolution variables aggregated over grid boxes of the global model, such as the mean cloud cover or liquid water content in a grid box. Their counterparts in the global model are computed by parameterization schemes.

- The corresponding variables $\boldsymbol{z}$ in the global model are obtained by a time-dependent map $\mathcal{S}$ that takes state variables $\boldsymbol{x}$ and parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}_c, \boldsymbol{\theta}_n)$ to $\boldsymbol{z}$,

  $$\boldsymbol{z}(t) = \mathcal{S}(\boldsymbol{\theta}, t; \boldsymbol{x}). \tag{3}$$

The map $\mathcal{S}$ typically represents a single grid column of the ESM with its parameterization schemes, taking as input $\boldsymbol{x}$ from the ESM. It is structurally similar to $\mathcal{L}$. Mismatches between $\boldsymbol{z}$ and $\tilde{\boldsymbol{z}}$ can be used to learn about computable parameters $\boldsymbol{\theta}_c$ because $\tilde{\boldsymbol{z}}$ does not depend on them.

- The high-resolution simulated state variables $\tilde{\boldsymbol{z}}$ can also be compared to observations $\tilde{\boldsymbol{y}}$. To do so, a map $\mathcal{Z}$ takes high-resolution simulated state variables to observables,
$$\boldsymbol{y}(t) = \mathcal{Z}\big(\tilde{\boldsymbol{z}}(t)\big), \tag{4}$$
e.g., cloud condensate amounts from LES to observable cloud condensate amounts. Mismatches between observables $\mathcal{Z}(t, \tilde{\boldsymbol{z}})$ and actual observations $\tilde{\boldsymbol{y}}$ can be used to learn about non-computable parameters $\boldsymbol{\theta}_n$ because high-resolution simulated state variables $\tilde{\boldsymbol{z}}$ depend on the non-computable parameters, but the observations $\tilde{\boldsymbol{y}}$ do not.

- Models of non-computable processes in the high-resolution simulation $\mathcal{L}$ and in the single-column model $\mathcal{S}$ need to be represented consistently: the non-computable processes in $\mathcal{S}$ need to be a consistently coarse-grained version of those in $\mathcal{L}$. This is necessary to be able to learn about the computable parameters $\boldsymbol{\theta}_c$ without the aliasing errors that would arise if non-computable processes are represented inconsistently across the hierarchy. It is also necessary to be able to use mismatches between $\mathcal{Z}(t, \tilde{\boldsymbol{z}})$ and observations $\tilde{\boldsymbol{y}}$ to learn about non-computable parameters $\boldsymbol{\theta}_n$ in the high-resolution model, and use the information so obtained in the ESM. The same code base can be used for non-computable processes in the coarse model and in the high-resolution model. For example, numerical quadrature methods may then be used to coarse-grain the high-resolution process model by sampling from the implied distributions in the coarse model.

Goals

- Design the high-resolution simulations to run efficiently on accelerators and in distributed computing environments. For example, it may be desirable to have one high-resolution simulation per GPU. But more flexible layouts (e.g., one high-resolution simulation spread over several GPUs) should also be possible.

- Design the high-resolution simulations to either run concurrently with the global host model (as in Fig. 2), with the host model providing time-evolving driving and boundary conditions, or in standalone mode, e.g., with the fixed and/or idealized boundary conditions that are common in atmospheric LES studies.

- Represent non-computable processes in the ESM component models (e.g., microphysics in atmosphere model) and in the high-resolution simulations (e.g., microphysics in atmospheric LES) consistently. That is, the non-computable process models in the ESM need to be consistently coarse-grained version of those in the high-resolution simulations, and the high-resolution simulations need to inherit the non-computable parameters $\boldsymbol{\theta}_n$ from the global model. Use the same code base for non-computable processes in the coarse model and in the high-resolution model, to reduce potentials for errors.

- Target a vertical resolution in atmospheric LES of about 10 m or finer, and a horizontal resolution of 50 m or finer. These are the minimum resolutions necessary to begin to resolve the challenging dynamics of stratocumulus clouds.
- Target the seasonal cycle of the boundary layer depth in ocean LES simulations, using a vertical resolution around 10 m and a horizontal resolution around 50 m, as in the atmosphere. These resolutions are necessary to resolve the boundary layer turbulence generated by surface cooling and wind stresses. The domain will need to be up to 10 km wide to capture the instabilities that develop along lateral density fronts and have been shown to modify the boundary layer depth.
- Make it easily addressable from the DA/ML layer at which physical locations high-resolution simulations are nested in the global model, to be able to target high-resolution simulations where they have the greatest impact on learning about parameters and reducing uncertainties.

2.3 DA/ML Layer

Concepts

- The DA/ML layer implements the algorithms for simultaneous learning about parameters (and parameteric functions) $\boldsymbol{\theta}$ in all component models, for example, learning about parameters in the atmosphere, ocean, land, ice, and biosphere models at the same time. The DA/ML layer allows for learning about the parameters both from observations and from targeted high-resolution simulations. It sits at the interface of models and data and fuses both. It also targets high-resolution simulations where they have large impact on learning about parameters.
- It is a crucial innovation of our design that the DA/ML layer wraps around all model components, allowing for simultaneous learning about all SGS processes from data. This makes it possible to exploit statistics of variables such as temperature and precipitation, which depend on many SGS processes. Treating SGS processes in isolation would make it difficult to exploit statistics that depend on many of these processes, and it would inevitably lead to overfitting of components and/or compensating errors among them. Simultaneous learning about all SGS processes will allow us to obtain and investigate the structure of correlations among SGS parameters across different process models.
- From the perspective of the DA/ML layer, the ESM is a function $\mathcal{G}$, parameterized by time t, that maps parameters $\boldsymbol{\theta}$ characterizing uncertain processes to climate state variables $\boldsymbol{x}$:

  $$\boldsymbol{x}(t) = \mathcal{G}(\boldsymbol{\theta}, t). \tag{5}$$

  The state variables $\boldsymbol{x}$ can include temperatures, humidity variables, and cloud, cryosphere, and biogeochemical variables, and the map $\mathcal{G}$ may depend on initial conditions and time-evolving boundary or forcing conditions. The vector of parameters $\boldsymbol{\theta}$ can include:

  - traditional parameters in SGS models (e.g., entrainment rates in convection parameteriations, turbulent diffusivities etc.);
  - parameters appearing in parametric functions in SGS models about which we want to learn from data (e.g., functions encoding how entrainment rates depend on environmental variables); or
  - parameters characterizing nonparametric functions in SGS models, such as Gaussian process models (*Rasmussen and Williams*, 2006) included as a flexible representation of error in the explicitly modeled processes.

- The map $\boldsymbol{y}(t) = \mathcal{H}\big(\boldsymbol{x}(t)\big)$ takes ESM state variables $\boldsymbol{x}$ to observables $\boldsymbol{y}$. Because $\boldsymbol{y}$ is parameterized by $\boldsymbol{\theta}$, while the actual observations $\tilde{\boldsymbol{y}}$ are independent of the parameters $\boldsymbol{\theta}$, mismatches $\boldsymbol{y} - \tilde{\boldsymbol{y}}$ can be used to learn about the uncertain parameters $\boldsymbol{\theta}$.

- Similarly, the map $\boldsymbol{z}(t) = \mathcal{S}(\boldsymbol{\theta}, t; \boldsymbol{x})$ (a single-column model) takes state variables $\boldsymbol{x}$ to variables $\boldsymbol{z}$ that are computable in high-resolution simulations $\mathcal{L}$. Crucially, the map $\mathcal{S}$ generally depends on all parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}_c, \boldsymbol{\theta}_n)$, while $\mathcal{L}$ only depends on non-computable parameters $\boldsymbol{\theta}_n$. Thus, mismatches $\boldsymbol{z}(t) - \tilde{\boldsymbol{z}}(t)$ can be used to learn about the computable parameters $\boldsymbol{\theta}_c$.

- Additionally, the map $\boldsymbol{y}(t) = \mathcal{Z}\big(\tilde{\boldsymbol{z}}(t)\big)$ takes high-resolution state variables $\tilde{\boldsymbol{z}}$ to observables $\boldsymbol{y}$. Because $\tilde{\boldsymbol{z}}$ is parameterized by non-computable parameters $\boldsymbol{\theta}_n$, but $\tilde{\boldsymbol{y}}$ is not, mismatches $\mathcal{Z}\big(\tilde{\boldsymbol{z}}(t)\big) - \tilde{\boldsymbol{y}}$ can be used to learn about the non-computable parameters $\boldsymbol{\theta}_n$.

- We generally define objective functions using time-averaged statistics. We denote the time average of a function $\phi(t)$ over the time interval $[t_0, t_0 + T]$ by

$$\langle \phi \rangle_T = \frac{1}{T} \int_{t_0}^{t_0+T} \phi(t)\, dt. \tag{6}$$

  (Extensions to data that are not averaged in time, e.g., to assimilate ocean states, will be considered as needed.)

- The observational objective function can then be written in the generic form

$$J_o(\boldsymbol{\theta}) = \frac{1}{2} \| \langle \boldsymbol{f}(\boldsymbol{y}) \rangle_T - \langle \boldsymbol{f}(\tilde{\boldsymbol{y}}) \rangle_T \|^2_{\Sigma_y} \tag{7}$$

  with the 2-norm

$$\| \cdot \|_{\Sigma_y} = \| \Sigma_y^{-1/2} \cdot \| \tag{8}$$

  normalized by standard deviations of and covariance information about errors in $\langle \boldsymbol{f}(\tilde{\boldsymbol{y}}) \rangle_T$ captured in Σ_y. The choice of variance/covariances will have a significant impact on the effectiveness of the learning, as it assigns relative weights to different sources of data. The relevant components of Σ_y may be chosen very small for quantities that are used as constraints on the ESM (e.g., the requirement of a closed global energy balance at TOA).

- The function $\boldsymbol{f}$ of the observables typically involves first- and second-order quantities, for example,

$$\boldsymbol{f}(\boldsymbol{y}) = \begin{pmatrix} \boldsymbol{y} \\ y_i' y_j' \end{pmatrix}, \tag{9}$$

  where, for any observable ϕ, $\phi'(t) = \phi(t) - \langle \phi \rangle_T$ denotes the fluctuation of ϕ about its mean $\langle \phi \rangle_T$. With $\boldsymbol{f}$ given by (9), the objective function penalizes mismatch between the vectors of mean values $\langle \boldsymbol{y} \rangle_T$ and $\langle \tilde{\boldsymbol{y}} \rangle_T$ and between the covariance components $\langle y_i' y_j' \rangle_T$ and $\langle \tilde{y}_i' \tilde{y}_j' \rangle_T$ for some indices i and j. However, $\boldsymbol{f}$ may also include higher-order statistics, such as high percentiles of the precipitation distribution, to penalize mismatches between simulated and observed statistics of precipitation extremes.

- Similarly, for the mismatch between the ESM and high-resolution simulations, we define an objective function analogously to that for the observations through

$$J_s(\boldsymbol{\theta}_c) = \frac{1}{2} \| \langle \boldsymbol{g}(\boldsymbol{z}) \rangle_T - \langle \boldsymbol{g}(\tilde{\boldsymbol{z}}) \rangle_T \|_{\Sigma_z}^2. \tag{10}$$

  Like the function $\boldsymbol{f}$ above, the function $\boldsymbol{g}$ typically involves first- and second-order quantities, and Σ_z encodes error variances and covariances. (This assumes that high-resolution simulations in any location are run over the same interval $[t_0, t_0 + T]$ over which ESM statistics are accumulated. This is how we will implement the learning algorithms for now. The assumption may be relaxed later.)

- An objective function $J_z(\boldsymbol{\theta}_n)$ for the mismatch between high-resolution simulations $\mathcal{Z}(t, \tilde{\boldsymbol{z}})$ and observations $\tilde{\boldsymbol{y}}$ is defined analogously.

- Learning about parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}_c, \boldsymbol{\theta}_n)$ in CLIMA proceeds through minimizing $J_o(\boldsymbol{\theta})$, $J_s(\boldsymbol{\theta_c})$, and $J_z(\boldsymbol{\theta_n})$. We will have to determine how to formalize the interaction between the different objective functions in this multi-objective optimization problem.

Goals

- Design the DA/ML layer to be as general purpose as possible, so that its algorithms can also be used for statistical learning problems in other sciences.

- Use gradient-free optimization algorithms (to avoid the stasis in model development that often arises when manually manipulated adjoints are required).

- Develop, test, and implement algorithms for optimizing parameters $\boldsymbol{\theta}$ that require as few ESM runs as possible—$O(1000)$ ESM runs, with embedded high-resolution simulations, over timescales of seasons are feasible.

- Develop, test, and implement algorithms for quantifying uncertainties about parameters $\boldsymbol{\theta}$, via emulation of the functions $\boldsymbol{f}$, $\boldsymbol{g}$ etc. (*Kennedy and O'Hagan*, 2001; *O'Hagan*, 2006).

2.4 Diagnostics Layer

Concepts

- Given observations and model output, both from the global model and high-resolution simulations, the diagnostics layer computes the statistics of model variables and observations that the DA/ML layer requires.
- Specifically, the diagnostics layer needs to provide the functions $\boldsymbol{f}$, $\boldsymbol{g}$ etc. and time averages $\langle\cdot\rangle_T$. These functions should be usable with input data from observations $\tilde{\boldsymbol{y}}$, global model state variables $\boldsymbol{x}$, high-resolution state variables $\tilde{\boldsymbol{z}}$, and single-column model variables $\boldsymbol{z}$.
- Because I/O can be slow and may generate unwieldy amounts of data, the diagnostics layer should be able to accumulate the statistics the DA/ML layer needs from the ESM and from the targeted high-resolution simulations on the fly and in parallel, without intermediate write-outs to disk.

Goals

- Design and implement a diagnostics layer that can accumulate statistics from global host models, targeted high-resolution simulations, and observations. Which statistics to accumulate should be easily addressable from the DA/ML layer.
- Design and implement a broad-purpose API for the diagnostics layer, so it can be used with diverse sources of input data.
- Provide a (possibly web-based) graphical user interface that allows users to easily examine output from the various models and compare it with observations.

2.5 Coupling Layer

Concepts

- The coupling layer lies at the interface between the DA/ML and diagnostics layers, on the one hand, and the component models, on the other hand (Fig. 2). It coordinates the exchange of data between component models such as the atmosphere, land, and ocean models.
- The coupling layer needs to allow for conservative exchange of data (momentum, water, energy, and tracer fluxes) across the component models' grids. To facilitate this exchange, it will be helpful to co-align grid nodes across the component models to the extent possible (e.g., atmosphere and ocean, and atmosphere and land, should share some nodes, even if they have different resolutions).
- Making internal data structures consistent across the component models will facilitate design of the coupling layer.

- The coupling layer needs to accommodate the parallel computing layout of the model components and needs to allow parallel data exchange between the model components (e.g., *Larson et al.*, 2005).
- It may be desirable to run model components concurrently in parallel (*Balaji et al.*, 2016), e.g., running the atmosphere model in parallel with the ocean model (as in Fig. 2). However, numerical coupling instabilities can arise when doing so (*Hallberg*, 2014). The timestepping strategy of the coupling needs to be chosen to guarantee stable coupling.

Goals

- Devise parallel coupling layout and a stable time-stepping strategy for the component coupling.
- Design and implement a parallel coupling layer with conservative exchanges of mass, water, momentum, and tracers across model components.

2.6 Component Models

The component models at the center of the ESM should function in a coupled configuration, and learning about parameters in them should occur simultaneously, to ameliorate problems arising from correlated errors among model components. Yet each model component should also be able to function as a standalone executable. The component models should share as much code (e.g., fluid dynamics operators, compute kernels) and inputs (e.g., topography/bathymetry databases) as possible, to avoid unnecessary growth of the total code base and make adaptation of the code to different planetary configurations as easy as possible.

2.6.1 Atmosphere Model (CLIMA-atmos)

Concepts

- The atmosphere model provides approximate solutions to the equations of fluid mechanics and thermodynamics on a rotating sphere. It consists of a dynamical core and parameterizations. The dynamical core discretizes the 3D compressible Euler equations and thermodynamic equations on a sphere. Parameterizations are needed principally for radiative transfer and for dynamical processes at scales below the resolution of the global grid, such as SGS turbulence, convection, cloud microphysics, and gravity waves.
- Large-eddy simulations provide approximate solutions to the same (non-hydrostatic) equations as the atmosphere dynamical core, just at higher resolution and, because of the large computational expense, in limited domains. LES resolutions needed to resolve the dynamics of cumulus clouds are in the range of $O(100\ \mathrm{m})$. To resolve the dynamics of stratocumulus clouds under a sharp inversion (a nearly discontinuous thermodynamic interface common in the subtropics), LES require resolutions of at least 50 m in the horizontal and 10 m in

the vertical. Avoiding spurious mixing in the numerical solution of the equations of motion (caused, e.g., by spuriously oscillatory numerics interacting with dissipative SGS schemes) is essential for faithful simulations of the climatically important stratocumulus clouds (*Pressel et al.*, 2017).

- The atmosphere dynamical core needs to be able to run at resolutions ranging from hundreds of kilometers in the horizontal (for prototyping and testing) to kilometers (where deep convection begins to be resolved).
- Scalability of the atmosphere model on a variety of hardware architectures, including accelerators, is paramount. This may require redesigning existing parameterizations, e.g., for radiative transfer, which currently involve extensive nonlocal operations (vertical integrals).
- For historical reasons, distinct parameterizations have been employed for processes lying on a continuous spectrum, such as boundary-layer turbulence, shallow (non-precipitating) convection, and deep (precipitating) convection. Parameterizations for processes such as microphysics (e.g., rain droplet formation) and radiative transfer typically have not strongly interacted with parameterizations for dynamical processes, although in nature these processes do strongly interact. This has led to correlated and compensating errors between the parameterization schemes; it leads to identifiability problems in a DA/ML approach to parameterizations. For our DA/ML approaches, it will be important to devise process-informed parameterization schemes that represent processes that lie on a continuous spectrum in a unified manner, with as few parameters as possible. Parameterizations also need to be able to adapt to different host model resolutions (be "scale aware") and to refine themselves as more data informing the processes become available. Error models (e.g., Gaussian processes) may also be incorporated directly in parameterization schemes and be estimated along with parameters and parametric functions in the parameterization schemes.
- Parameterizations should be developed in the form of continuous differential equations, which can then be discretized consistently with the host model (i.e., use the same compute kernels). However, parameterizations may employ resolutions and time steps that differ from those in the host model.
- It is desirable (but not necessary) for the global atmosphere model, parameterizations, and LES to share the same prognostic variables. Prognostic variables that mix linearly and that are conserved in adiabatic and inviscid flow are desirable. Using total energy or moist conserved variables (e.g., liquid-ice static energies) throughout may be advantageous. Variables that are nonlinear functions of temperature and other state variables (e.g., entropies) should be avoided, because they make it computationally expensive to retrieve temperature. (Potential temperatures have the added problem that they are not extensive and do not mix linearly, making it difficult to justify, e.g., diffusive SGS models for them.)

Goals

- Develop an atmosphere dynamical core that employs state-of-the art numerics, can exploit emerging computing architectures such as heterogenous many-core architectures, and can run embedded high-resolution simulations (LES) on demand.
- Develop atmosphere parameterizations suitable for DA/ML approaches, first focusing on turbulence, convection, and clouds, and then expanding to gravity waves and (if needed) radiative transfer.
- Demonstrate automated learning about parameterizations from observational data and high-resolution simulations nested in the atmosphere model.

2.6.2 Ocean Model (CLIMA-ocean)

Concepts

- The ocean model shares many of the same elements as the atmosphere model. It consists of a dynamical core that solves, on a sphere, the 3D Euler equations and the thermodynamic equations for nearly incompressible seawater. Parameterizations are needed for dynamical processes at scales below the resolution of the global grid, such as SGS boundary layer turbulence, frontal instabilities, and interior mesoscale turbulence. In the ocean, there is no equivalent to the microphysics and phase transitions in the atmosphere. The parameterization problem is strictly one of representing SGS dynamics whose governing equations are well known in terms of variables resolved on the model's global grid.
- Large-eddy simulations will be used to provide solutions of the ocean dynamics at higher resolution than the global model. LES resolutions needed to resolve boundary layer turbulence are in the range of O(10 m) in the horizontal and a few meters in the vertical. Lateral domains of O(1 km) are sufficient to resolve the vertical fluxes generated by boundary layer turbulence. However, lateral fluxes generated by surface density fronts require domains of O(10–50 km). Even larger domains of O(1000 km), at resolutions of kilometers, are necessary to resolve mesoscale turbulence. We plan to run the global ocean model at resolutions of 10–50 km and will need to parameterize boundary layer turbulence, frontal instabilities, and SGS mesoscale turbulence. [ok?]
- Like for the atmospheric model, scalability will also be required for the ocean model to run on a variety of architectures, including accelerators.
- Parameterization of ocean processes share the same historical idiosyncrasies as the those developed for atmospheric models. Parameterizations for boundary layer processes have been developed before and independently of those for mesoscale turbulence. However, the two processes strongly interact in the upper ocean: boundary layer processes such as convection and wind-driven turbulence weaken the ocean stratification, while mesoscale eddies tend to strengthen it. Indeed, when *Gent and McWilliams* (1990) implemented the first parameterization of mesoscale turbulence in an ocean model, it resulted in the nearly complete

suppression of boundary layer turbulence, a result clearly at odds with observations. Their ad-hoc solution was to turn off the mesoscale parameterization in the upper ocean, a nonphysical and unsatisfactory approach that has been adopted in climate models ever since. For physical reasons and for our DA/ML approaches, it will be important to devise parameterization schemes that represent all SGS processes in a unified manner and properly capture their interactions, rather than selectively suppressing them. Much like in the atmospheric case, the parameterizations also need to be "scale aware" and to refine themselves as more data informing the processes become available.

- The ocean parameterizations will follow the same guiding principles as the atmospheric ones, namely, they will be developed in the form of continuous differential equations, and they will share the same prognostic variables as the global ocean model and the LES.

Goals

- Develop an ocean dynamical core that shares much of the state-of-the-art numerics of the atmospheric dynamical core, to facilitate coupling of the two models. The model will have the capability of running embedded high-resolution simulations (LES) on demand.
- Develop ocean parameterizations suitable for DA/ML approaches, first focusing on upper-ocean boundary layer turbulence, and then expanding to mesoscale turbulence (as needed).
- Adopt the DA/ML approach to be developed at Caltech to train parameterization schemes with observational data and high-resolution simulations nested in the ocean model.

2.6.3 Sea Ice Model (CLIMA-sea-ice)

[Raf: please review and edit/amend as needed]

Concepts

- Sea ice models are coupled to the ocean underneath and to the atmosphere above, through exchanges of water, energy, and stresses. Typical sea ice models consist of a two-dimensional (horizontal) continuum model for the motion and deformation of sea ice and a one-dimensional (vertical) thermodynamic model (e.g.. *Semtner*, 1976) for sea ice thickness and energy fluxes through the ice.
- An important choice in a sea ice model concerns the rheology of the sea ice, which determines how ice deforms and breaks, opening up leads. Options include visco-plastic rheologies [refs] and elasto-brittle rheologies [cite Veronique Veronique Dansereau's 2016 paper].

- Around 40 km horizontal resolution is necessary to capture the energy fluxes through ice and leads accurately, which is essential for a climate model (cite Rampal paper showing convergence of fluxes).

- Because sea ice forms leads (breaks), discontinuity-capturing numerical discretization schemes (e.g., discontinuous Galerkin schemes) are essential for sea ice. While Lagrangian advection schemes are in use (ref), in the context of a climate model, Eulerian schemes are preferable: they facilitate coupling to atmosphere and ocean. Ideally, the sea ice model would share discretization strategies and compute kernels with the atmosphere and ocean, to harness the performance of the compute kernels and facilitate coupling with both atmosphere and ocean.

- While sea ice models are often implemented as part of an ocean model, there seems to be no compelling reason to do so. Conservative coupling to atmosphere and ocean is essential, but this can be achieved as for other model components through a conservative coupler (which, for sea ice, needs to couple at the top and the bottom, to atmosphere and ocean). Using a shared coupling layer across all model components including sea ice has the advantage that the machinery for parallel coupling of components with different resolutions can be shared, and does not need to be developed and implemented separately for a sea ice model. It will also facilitate running targeted high-resolution simulations for sea ice. [Raf: do you agree? I really see no reason, other than history, why sea ice and ocean are glued together the way they are today. If you do not do that, you can have component concurrency in the coupled model, giving additional parallel efficiency.]

- One can learn about uncertain SGS parameters in sea ice models (e.g, rheological parameters) from observations, e.g., of statistics of seasonally varying sea ice cover and ice deformations (fro microwave and synthetic aperture radar measurements from space).

Goals

- Develop and implement a sea ice model with a two-dimensional rheology and one-dimensional thermodynamics, using the same compute kernels as the atmosphere and ocean, to achieve scalable performance on emerging computing architectures.

- Test different rheologies in terms of their numerical performance and convergence properties (ideally, in observationally constrained benchmarks). (is there a clear ground truth benchmark?)

- Demonstrate automated learning, e.g., about rheological parameters from observational data and high-resolution simulations nested in the atmosphere model.

2.6.4 Land Model (CLIMA-land)

Concepts

- Land models consist of models for the land biosphere and for the hydrology of ground water and surface water (river runoff, snow hydrology). These two types of models interact closely (e.g., through water exchange and through plant effects on soil properties and water transport). But they should be modularized because we know the equations for modeling the physics of water flow, and we can coarse-grain and approximate these equations systematically; we may want to refine the hydrology modeling when the need arises, without changing the biosphere model. However, the biosphere must be modeled more empirically; biosphere models are not refineable in the same way as physical models, where we have a well-defined notion of averaging across scales.

- The atmosphere provides the upper boundary condition for the land model, providing precipitation and radiative fluxes and, e.g., the wind velocity, temperature, and specific humidity of the near-surface air, on which sensible heat fluxes and evapotranspiration at the surface depend. In turn, the land model provides the fluxes of momentum, of atmospheric constituents such as carbon dioxide, and of energy (radiative energy fluxes that depend on land albedo and emissivity, as well as sensible and latent heat fluxes). These fluxes provide the lower boundary condition for the atmosphere over land; they need to be consistent with the surface-layer fluxes in the atmosphere (e.g., as determined by Monin-Obukhov similarity theory, consistently both over land and oceans).

- The resolution of the land model should be adequate for its purposes. It should reflect the heterogeneity of the land surface (e.g., vegetation or soil heterogeneity) that needs to be resolved. The land model resolution does not need to be linked to the atmosphere resolution. But it is important that the fluxes that are being exchanged between different land and atmosphere grids in the coupling layer are aggregrated conservatively.

- Land modeling is at the cusp of a transformation, as new data from space-based platforms such as OCO-2, OCO-3, ECOSTRESS, and GRACE for the first time enable a comprehensive global characterization of land features such as gross primary productivity, of large-scale water fluxes, and of carbon exchange between land and the atmosphere. The potential of these new datasets to inform land models is only beginning to be tappped. Machine learning approaches to land biosphere modeling (e.g., at the MPI Jena) have shown promise but achieve good fits to the available data at the expense of strong overparameterization, as is typical for deep learning approaches. For good predictive capabilities, a process-driven model with a smaller number of fitted degrees of freedom is desirable.

- These characteristics lead to the following requirements for the land model:

    1. Given atmospheric conditions and fluxes just above the surface, provide energy, water, and carbon exchanges needed as boundary conditions for an atmospheric model.
    2. Given evaporation and precipitation, provide river fluxes into oceans to close the water budget over land and salinity budget of the oceans.

3. Provide surface roughness needed to calculate exchanges of momentum, energy, water, and tracers (e.g., carbon dioxide) with the atmosphere.
4. Design the land model to be "optimally complex and flexible" to (a) be able to represent land/atmosphere feedbacks from diurnal to centennial scales and (b) harness currently available and future expected data.

Goals

- Develop a scalable, parallel land model that is process driven yet simple enough that it can harness a wide range of available data effectively, without overfitting that may degrade its predictive capabilities. The complexity of the land model is to be decided; JPL's CARbon DAta MOdel fraMework (CARDAMOM; *Bloom et al.* (2016) provides a good starting point for exploration for the biosphere model.

- Design a modular land model that separates hydrology and biosphere modeling to the extent possible, but allows for the essential direct coupling between the two, in such a way that the biosphere and hydrology model can be updated and refined separately.

- Adopt the DA/ML approach to be developed at Caltech to optimize the land model using observables such as solar induced fluorescence (OCO-2 and OCO-3) and gravity measurements from GRACE.

- Empirically determine the complexity of the land model optimal for decadal predictions.

2.7 Compute Kernels

Concepts

- As of June 2018, three of the top five computers on the TOP500 (list of fastest supercomputers) are heterogeneous compute environments with many-core devices. All of the top twenty-three computers on the Green500 (list of the most efficient supercomputers in terms of flops per watt) are heterogeneous compute environments. Heterogeneous compute environments are here to stay for the foreseeable future.

- Placing compute intensive code into chunks, called kernels, allows for offloading these kernels to accelerators.

- Compute kernels provide discrete approximations of continuous differential and integral operators. They should be shared among component models (atmosphere, ocean, sea ice, land hydrology, including SGS process models) to the extent possible, to reduce code redundancy and isolate hardware-specific implementation issues in as few places as possible.

- We distinguish (1) non-performant and (2) performant compute kernels. The non-performant kernels are constructed with the goal of maximizing readability. Once tested and accepted, they will be redesigned to be performant (e.g., maximized for vectorization, ported to accelerators, and restructured to minimize communication).
- For the atmosphere model, we will use the discontinuous Galerkin method for the spatial discretization (*Abdi and Giraldo*, 2016). We are exploring using the same for the ocean model. To do so in the most straightforward manner, we will exploit tensor product bases functions such that the two-dimensional (horizontal) discrete operators, required for the ocean model, will be as similar as possible to the three-dimensional discrete operators. In addition, one-dimensional discrete operators will be required for some vertical processes, e.g., in SGS schemes.

Goals

- Develop our code so that it takes advantage of the many-core accelerators found in current and emerging supercomputers.
- Isolate all compute-intensive code in compute kernels with simple, array-based interfaces. This will allow for the code to be transitioned to different compute environments/backends.
- Minimize use of data outside of compute kernels as that may incur a performance penalty when the kernels are offloaded. In other words, be mindful of data accesses.
- Implement non-performant and then performant compute kernels in one, two, and three dimensions. We need kernels for mesh generation, common differential operators, and vertical integration (e.g., for radiative transfer and the barotropic mode in ocean), to be used by the atmosphere dynamical core and SGS process models, the ocean dynamical core and SGS process models, and by the sea ice and land hydrology models.
- Benchmark the performance of the new kernels against existing compute kernels (e.g., in Fortran 95). Roofline plots can also be useful in measuring performance against expected performance.
- Strive to make the CPU and GPU kernels readable and look as similar as possible. (Ideally and in the longer run, they may be one and the same, but for now they are separate but similar, as long as we use array-based implementations).

2.8 Parallel Communications/Services

Concepts

- There are many levels of parallelism in modern many-core based supercomputers. This includes intra-node parallelzation such as vector instructions, multiple cores in a device, and multiple devices. There is also inter-node parallelism where nodes are interconnected.

- Access to memory is not uniform on modern supercomputers. For example, both CPUs in a two-CPU node may be able to access all the memory but not at the same speeds. Some memory is “closer” than others and thus faster to access.
- Asynchronous communication allows for computation to be overlapped with communication. Further, many times latency is high but bandwidth is also high for inter-node communication. This is also true for communication between the host and compute device.
- Using a common abstraction for parallelism for each component will ease the coupling of components and minimize costly data transfers.

Goals

- Exploit the parallelism available in current and emerging supercomputers.
- To get good performance, allocate memory “near” the unit of parallelism that is going to use it.
- To minimize communication costs, group asynchronous communication together and try to hide its cost (latency) by overlapping it with computation.
- To ease the coupling of components, coordinate the selection of abstractions and implementations used for expressing parallelism in the code.

The above are more like general guidelines than concrete goals for a discrete piece of code. Should we eliminate this section and work it into the general guidelines? Or will there some piece of code that can be associated with this section?

3 General Guidelines

- Implement best practices for scientific computing (e.g., *Wilson et al.*, 2014), including
    1. Make names consistent, distinctive, and meaningful. For variable names, follow CMIP conventions[2] where possible and practicable).
    2. Use a build tool to automate workflows.
    3. Make incremental changes, with version control (GitHub).
    4. Modularize code rather than copying and pasting (DRY–Don’t Repeat Yourself).

[2]http://clipc-services.ceda.ac.uk/dreq/index.html

5. Integrate continuously and employ unit testing (try to develop tests before developing the code, and focus on short unit tests, which give more valuable information when they fail; make unit tests meaningful, e.g., check for conservation of properties that should be conserved).
6. Make code correct first and fast second (then use profilers to identify bottlenecks).
7. Document design and purpose (document interfaces and embed documentation in code).
8. Collaborate (pre-merge code reviews, pair programming to bring new people up to speed or for tricky problems; use issue tracking on GitHub).
9. Design APIs for the simplest cases first, add options for more flexible use.
10. Shared code ownership is the goal; siloed knowledge is bad.

- Limit external package dependency as much as possible to facilitate portability. Where package dependencies are essential, use common packages with a broad user base (e.g., MPI, NetCDF library)
- Collect *all* parameters (e.g., physical constants, input data) in one central place, where they are easy to modify and are shared by all components of the code. No hard-coded parameters anywhere in the code. Input parameters should not be multiply defined for different code components. For example, there should be one and only one topography/bathymetry dataset that all model components (atmosphere, ocean, land) use. Depending on their resolution, some components (e.g., the atmosphere model) may need to smooth the input topography; but there should only be one input dataset, so topography can easily and consistently be changed for all model components.
- Make use of Continuous Integration (CI) tools as much as possible. For example, on GitHub we may use Travis as our CI tool to test 100% of the code. Test the software across various platforms and operating systems (e.g., Linux, MacOS, and Windows). Some CI testing may be required to be performed on specific hardware; examples include testing the software on specific GPU hardware on local computers (rather than what is available on the cloud via GitHub). At NPS we may use Gitlab with either runners or interleaved with Jenkins CI.
- [Add and expand!]

4 Objectives and Key Results for Years 1–3 (2018–2021)

We will pursue several key objectives in years 1–3 (by 2021), and we will measure progress by achievement of the key results listed for each objective.[3] The key results listed are intentionally ambitious. Some may be a stretch to achieve. We can consider ourselves successful if we achieve about 70% of the key results.

Key results to be achieved in year 1 (by Q3/2019) are highlighted green.

[3]To read more about objectives and key results, or OKRs, for goal settings, see https://rework.withgoogle.com/guides/set-goals-with-okrs/steps/introduction/.

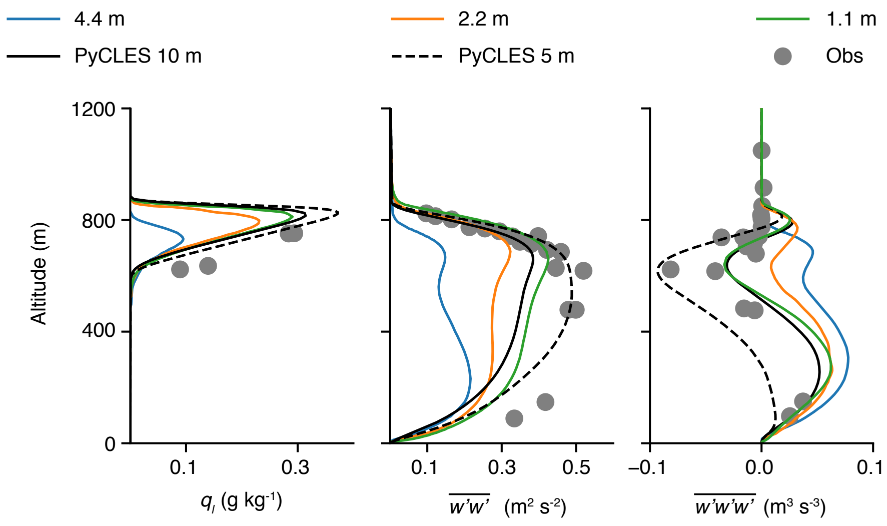

Figure 3: **Comparison of atmospheric LES in DYCOMS stratocumulus benchmark.** The figure shows liquid water specific humidity (q_l), vertical velocity variance ($\overline{w'w'}$), and the third cumulant of vertical velocity ($\overline{w'w'w'}$) for LES of the DyCOMS-II RF01 benchmark (*Stevens et al.*, 2003, 2005). Simulations with the PyCLES code with weighted essentially non-oscillatory (WENO) numerics and an implicit LES approach at $50\ \mathrm{m} \times 10\ \mathrm{m}$ and $35\ \mathrm{m} \times 5\ \mathrm{m}$ resolution (solid and dashed black lines) compare favorably with observations (circles) (*Pressel et al.*, 2015, 2017). The PyCLES simulations show higher fidelity to observations than simulations with oscillatory numerics at more than a factor 4 higher resolution and hence much greater computational expense (*Mellado et al.*, 2018). Although the WENO schemes are slower by an $O(1)$ factor than the oscillatory schemes at a given nominal resolution, they achieve comparable fidelity to observations at lower resolution, and hence at much reduced computational cost for a given fidelity to observations. From *Schneider et al.* (2018).

1. **Develop scalable atmosphere and ocean dynamical cores that exploit emerging computing architectures, can run high-resolution simulations on demand, and share numerical methods and computational architectures to the extent possible.**

   Key results:

   (a) Demonstrate well-tested compute kernels for common differential operators in 1, 2, and 3 dimensions, both for CPUs and GPUs, and to be shared by atmosphere and ocean dynamical cores.

   (b) Demonstrate scalability and performance of the atmosphere dynamical core on CPUs and GPUs comparable to NUMA (*Abdi et al.*, 2018a,b; *Müller et al.*, 2016).

   (c) Demonstrate time-to-solution of global simulations in the NGGPS benchmark comparable to (within factor 3) the best non-hydrostatic weather prediction models in the U.S. at a given nominal resolution.[4] (Note that our effective resolution may be higher than the nominal resolution used by some models with lower-order numerics. So a factor 3 slower performance in the NGGPS is acceptable to us because it will be compensated if we can achieve the same fidelity of a simulation at 30% lower nominal resolution. See Fig. 3 for an example.)

   (d) Demonstrate an ocean dynamical core that resembles the atmosphere core, has performance comparable with or superior to that of the MITgcm on CPUs, and can be run on GPUs.

2. **Develop LES that can be run in standalone mode or be nested in the global atmosphere and ocean models and use essentially the same code base as the global models (but possibly with differences, e.g., in time stepping).**

   Key results:

   (a) Implement fully 3D implicit-explicit (IMEX) methods with scalable iterative solvers and preconditioners, to replace the classical horizontally-explicit-vertical-implicit (HEVI) methods typically used in global nonhydrostatic models.

   (b) Implement and test "bullet proof" stable methods, such as entropy-stable DG methods (e.g., *Chan*, 2018).

   (c) Implement different choices of LES SGS models (e.g., Smagorinski-Lilly, dynamic models, implicit LES) and test them in canonical benchmark cases in atmosphere and oceans.

   (d) Demonstrate successful simulations of a stratocumulus-topped boundary layer in the atmosphere with standalone LES, with fidelity of simulations comparable with or superior to PyCLES with WENO schemes (Fig. 3).

   (e) Ocean LES demonstration?

   (f) Demonstrate at least 100 concurrent LES running alongside global atmosphere and ocean models and show that, if run with prescribed lower boundary conditions, they locally match observations (e.g., of subtropical low clouds and mixed-layer and sea ice seasonal cycle in oceans). Use the LES to establish database for training parameterizations.

3. **Develop atmosphere process models suitable for DA/ML approaches.**

   Key results:

   (a) Demonstrate, in single-column mode, physical skeleton of a unified parameterization for atmospheric turbulence, convection, and clouds (TCC) (including microphysics) that can at least qualitatively capture boundary layer and cloud regimes from stable boundary layers, over shallow convection and stratocumulus-topped boundary layers, to deep convection.

[4]For the NGGPS benchmark results, see https://www.weather.gov/media/sti/nggps/AVEC%20Level%201%20Benchmarking%20Report%2008%20

   (b) Demonstrate, in a perfect-model setting, learning about parameters, parametric functions, and (possibly) non-parameteric functions in the unified TCC parameterization.

   (c) Implement TCC parameterization in atmosphere model and demonstrate learning about its parameters in a perfect-model setting with simulated global observations.

   (d) Demonstrate a gravity-wave parameterization that can learn from observations.

4. **Develop ocean process models suitable for DA/ML approaches (some of them may be conceptually similar to the atmospheric parameterization, e.g., in unifying diffusive and mass flux approaches).**

   Key results:

   (a) Develop unified parameterization framework for mixed-layer turbulence, convection, and mesoscale and submesoscale eddies.

   (b) Demonstrate that the unified ocean parameterization can reproduce the seasonal cycle of mixed layers in (1) the high-latitude North Atlantic (where restratification in spring is not captured by present models) and (2) in the Southern Ocean (where mixed layer deepening in winter due to inertial shear is not included in present parameterizations).

   (c) Develop parameterizations for heat and salt fluxes under sea ice and test them against observations of seasonal extent/thickness of sea ice in the Artcic and Antarctic Oceans.

   (d) Implement new parameterizations in ocean model and demonstrate learning about its parameters in a perfect-model setting with simulated global observations.

5. **Develop fast and efficient DA/ML algorithms.**

   Key Results:

   (a) Develop a derivative-free, ensemble-based optimization scheme, for learning parameters in SGS models of the atmosphere and oceans, requiring on the order of 1000 ESM runs. Demonstrate and test the method in a global model in a perfect-model setting.

   (b) Develop and demonstrate machine learning surrogates for the ESM, trained on the output of the optimizer, and use them for Bayesian uncertainty quantification of parameters; develop appropriate optimal experimental design approaches for this purpose.

   (c) Investigate and exploit the trade-off between statistical information and computational tractability to develop mini-batch-type algorithms to learn parameters in a filtering approach.

   (d) Demonstrate online optimal experimental design methods to help locate LES turbulence simulations to generate maximally informative small-scale data.

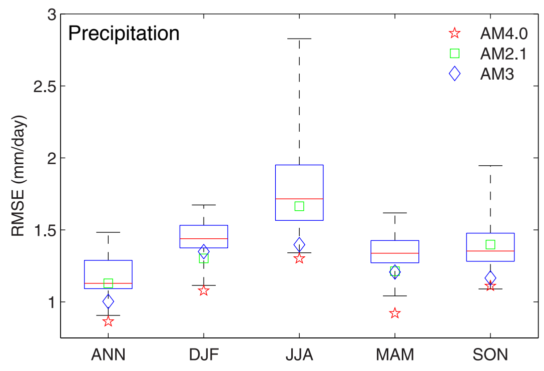

Figure 4: **Precipitation errors in current climate models.** Global rms error in precipitation time-averaged AMIP simulations for 1980–2014 with CMIP5 models. Red lines indicate the median and whiskers the maximum and minimum values of the CMIP5 model ensemble. Plotting symbols are for various versions of the GFDL atmosphere model (AM). The observational data are from the Global Precipitation Climatology Project. From *Zhao et al.* (2018).

6. **Demonstrate atmosphere and ocean models that learn from observational data (primarily reanalysis) and nested high-resolution simulations.**

   (a) Demonstrate atmosphere-only simulations that learn automatically from observations (reanalysis data for the past decades) and nested high-resolution simulations, in two configurations: (i) driven by observed sea surface temperatures (SST), i.e., in Atmospheric Model Intercomparison Project (AMIP) mode, and (ii) coupled to a slab ocean with observed ocean energy fluxes. (Configuration (ii) will clearly expose biases in the surface energy balance resulting, e.g., from biases in cloud cover, which may remain hidden in configuration (i).)

   (b) Demonstrate a substantial reduction in errors over seasonal cycle relative to current (CMIP5) simulations.

      i. Reduce global rms error in precipitation in AMIP runs to below $0.8\ \mathrm{mm\ day}^{-1}$ in any season (more than factor 2 reduction relative to CMIP5 model median, see Fig. 4).

      ii. Reduce global rms errors in TOA net, shortwave, and longwave radiative energy fluxes in AMIP simulations to below $5\ \mathrm{W\ m}^{-2}$ in any season (more than factor 3 reduction relative to CMIP5 model median).

      iii. Reduce tropical low-cloud cover biases in coupled runs to less than 10% (compare Fig. 6 for current model biases, which are around 50% and larger).

      iv. Reduce polar temperature biases to less than 2 K and sea ice cover biases to less than $2 \times 10^6\ \mathrm{km}^2$ in coupled runs (compare Fig. 7 for

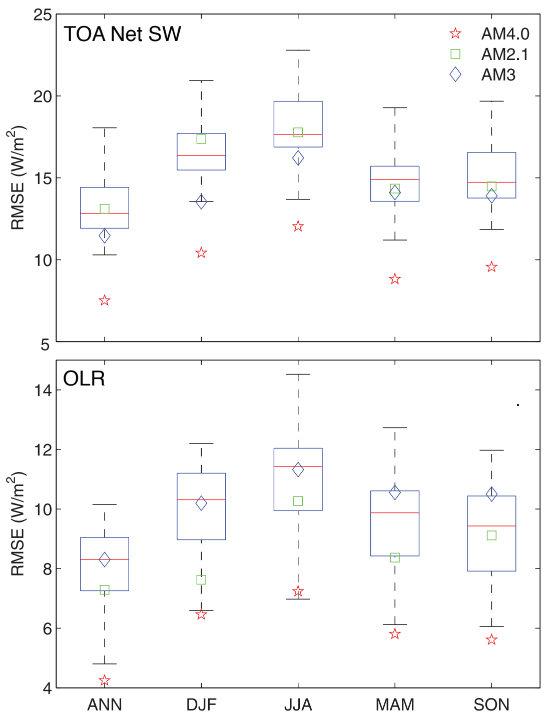

Figure 5: **Top-of-atmosphere energy flux errors in current climate models.** Global rms error in time-averaged net shortwave radiation and outgoing longwave radiation in AMIP simulations for 1980–2014 with CMIP5 models. Plotting conventions as in Fig. 4. Observational data from CERES. From *Zhao et al.* (2018).

CMIP5 model biases).

(c) Demonstrate reproduction of the 20th-century changes with the atmosphere model, coupled to a slab ocean (without having used the entire 20th-century record, with the exception of data for recent decades, for training the model).

(d) Demonstrate learning about parameterizations in ocean model from observations (ocean reanalysis and Argo floats) paired with LES.

(e) observational ocean targets? ML depth?

It should go without saying that these error reduction targets must not be achieved at the expense of a severe deterioration in other aspects of the climate simulations.

7. **Proof-of-concept of uncertainty quantification in process models and climate predictions.**

   (a) Demonstrate accuracy of UQ in perfect-model settings in idealized (e.g., aquaplanet) contexts, where the accurate but expensive UQ with MCMC is

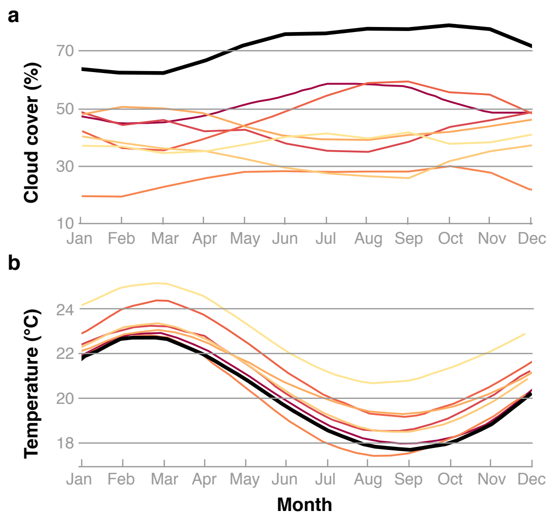

Figure 6: **Cloud cover bias in current climate models.** Annual cycle of (a) cloud cover and (b) sea surface temperature (SST) off the coast of subtropical South America from observations (black) and in current climate models (colors). Data from *Lin et al.* (2014).

still feasible.

(b) Demonstrate UQ in atmosphere-only models (driven by observed SST, or coupled to slab ocean).

(c) Demonstrate global UQ in climate predictions through ensemble of simulations drawn from posterior density of parameters in process models.

(d) Demonstrate local climate predictions with ensemble of targeted high-resolution simulations, with parameters drawn from posterior in process models.

(e) oceans?

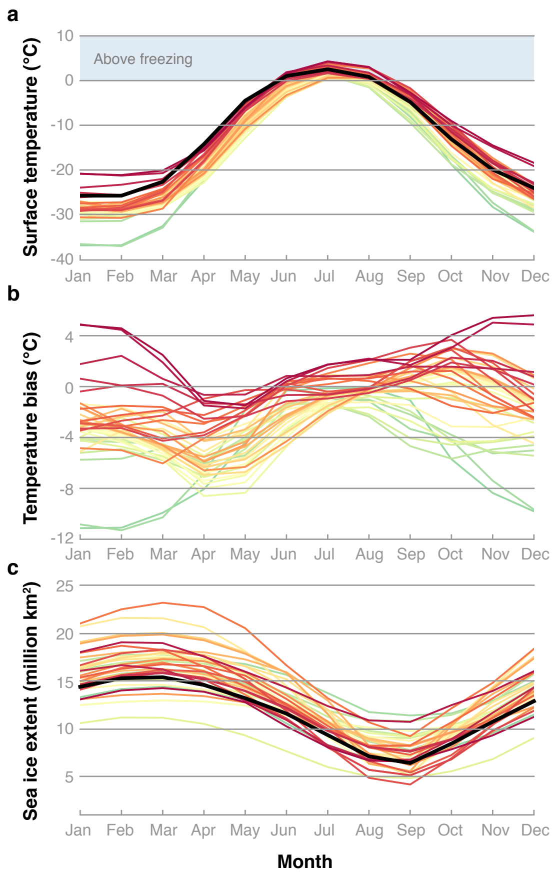

Figure 7: **Annual cycle of observed and simulated Arctic temperatures and sea ice extent.**; (a) Surface temperature (65°–90°N), (b) surface temperature bias (difference between simulations minus observations), and (c) sea ice extent from satellite observations (black) and in climate models (colors).

References

Abdi, D., and F. X. Giraldo (2016), Efficient construction of unified continuous and discontinuous galerkin formulations for the 3d euler equations, *J. Comp. Phys.*, *320*(1), 46–68.

Abdi, D., L. Wilcox, T. Warburton, and F. X. Giraldo (2018a), A gpu accelerated continuous and discontinuous galerkin non-hydrostatic atmospheric model, *International Journal of High-Performance Computing*, *(in press)*(1).

Abdi, D., F. Giraldo, E. Constantinescu, L. Carr III, L. Wilcox, and T. Warburton (2018b), Acceleration of the implicit-explicit non-hydrostatic unified model of the atmosphere (numa) on manycore processors, *International Journal of High-Performance Computing*, *(in press)*(1).

Balaji, V., R. Benson, B. Wyman, and I. Held (2016), Coarse-grained component concurrency in Earth system modeling: parallelizing atmospheric radiative transfer in the GFDL AM3 model using the Flexible Modeling System coupling framework, *Geosci. Model Dev.*, *9*, 3605–3616.

Bloom, A. A., J.-F. Exbrayat, I. R. van der Velde, L. Feng, and M. Williams (2016), The decadal state of the terrestrial carbon cycle: Global retrievals of terrestrial carbon allocation, pools, and residence times, *Proc. Natl. Acad. Sci.*, *113*, 1285–1290.

Chan, J. (2018), On discretely entropy conservative and entropy stable discontinuous Galerkin methods, *J. Comp. Phys.*, *362*, 346–374.

Gent, P. R., and J. C. McWilliams (1990), Isopycnal mixing in ocean circulation models, *J. Phys. Oceanogr.*, *20*, 150–155.

Hallberg, R. (2014), Numerical instabilities of the ice/ocean coupled system, *CLIVAR Exchanges*, *19*, 38–.

Kennedy, M. C., and A. O'Hagan (2001), Bayesian calibration of computer models, *J. Roy. Statist. Soc. B*, *63*, 425–464.

Larson, J., R. Jacob, and E. Ong (2005), The model coupling toolkit: a new Fortran90 toolkit for building multiphysics parallel coupled models, *Int. J. High Perf. Comp. App.*, *19*, 277–292.

Lin, J.-L., T. Qian, and T. Shinoda (2014), Stratocumulus clouds in Southeastern Pacific simulated by eight CMIP5–CFMIP global climate models, *J. Climate*, *27*, 3000–3022.

Lorenz, E. N., and K. A. Emanuel (1998), Optimal sites for supplementary weather observations: Simulation with a small model, *J. Atmos. Sci.*, *55*, 399–414.

Mellado, J. P., C. S. Bretherton, B. Stevens, and M. C. Wyant (2018), DNS and LES for simulating stratocumulus: Better together, *J. Adv. Model. Earth Sys.*, *10*, 1421–1438, doi:10.1029/2018MS001312.

Müller, A., M. A. Kopera, S. Marras, L. Wilcox, T. Isaac, and F. X. Giraldo (2016), Strong scaling for numerical weather prediction at petascale with the atmospheric model numa, *International Journal of High-Performance Computing*, *in press*, , doi:10.1016/j.jcp.2012.10.038.

O'Hagan, A. (2006), Bayesian analysis of computer code outputs: A tutorial, *Reliability Eng. Sys. Safety*, *91*, 1290–1300.

Palmer, T. N., R. Gelaro, J. Barkmeijer, and R. Buizza (1998), Singular vectors, metrics, and adaptive observations, *J. Atmos. Sci.*, *55*, 633–653.

Pressel, K. G., C. M. Kaul, T. Schneider, Z. Tan, and S. Mishra (2015), Large-eddy simulation in an anelastic framework with closed water and entropy balances, *J. Adv. Model. Earth Sys.*, *7*, 1425–1456, doi:10.1002/2015MS000496.

Pressel, K. G., S. Mishra, T. Schneider, C. M. Kaul, and Z. Tan (2017), Numerics and subgrid-scale modeling in large eddy simulations of stratocumulus clouds, *J. Adv. Model. Earth Sys.*, *9*, 1342–1365.

Rasmussen, C. E., and C. K. I. Williams (2006), *Gaussian Processes for Machine Learning*, MIT Press, Cambridge, MA.

Schneider, T., S. Lan, A. Stuart, and J. Teixeira (2017a), Earth system modeling 2.0: A blueprint for models that learn from observations and targeted high-resolution simulations, *Geophys. Res. Lett.*, *44*, 12,396–12,417, doi:10.1002/2017GL076101.

Schneider, T., J. Teixeira, C. S. Bretherton, F. Brient, K. G. Pressel, C. Schär, and A. P. Siebesma (2017b), Climate goals and computing the future of clouds, *Nature Climate Change*, *7*, 3–5.

Schneider, T., C. M. Kaul, and K. G. Pressel (2018), Possible climate transitions from breakup of stratocumulus decks under greenhouse warming, *Nature Geosci.*

Semtner, A. J. (1976), A model for the thermodynamic growth of sea ice in numerical investigations of climate, *J. Phys. Oceanogr.*, *6*, 379–389.

Stephens, G., D. Winker, J. Pelon, C. Trepte, D. Vane, C. Yuhas, T. L'Ecuyer, and M. Lebsock (2018), CloudSat and CALIPSO within the A-Train: Ten years of actively observing the Earth system, *Bull. Amer. Meteor. Soc.*, *99*, 569–581.

Stevens, B., D. H. Lenschow, G. Vali, H. Gerber, A. Bandy, B. Blomquist, J.-L. Brenguier, C. S. Bretherton, F. Burnet, T. Campos, S. Chai, I. Faloona, et al. (2003), Dynamics and chemistry of marine stratocumulus–DYCOMS-II, *Bull. Amer. Meteor. Soc.*, *84*, 579–593.

Stevens, B., C.-H. Moeng, A. S. Ackerman, C. S. Bretherton, A. Chlond, S. de Roode, J. Edwards, J.-C. Golaz, H. Jiang, M. Khairoutdinov, M. O. Kirkpatrick, D. C. Lewellen, A. Lock, F. Müller, D. E. Stevens, E. Whelan, and P. Zhu (2005), Evaluation of large-eddy simulations via observations of nocturnal marine stratocumulus, *Mon. Wea. Rev.*, *133*, 1443–1462, doi:10.1175/MWR2930.1.

Wilson, G., D. Aruliah, C. T. Brown, N. P. C. Hong, M. Davis, R. T. Guy, S. H. D. Haddock, K. D. Huff, I. M. Mitchell, M. D. Plumbley, B. Waugh, E. P. White, and P. Wilson (2014), Best practices for scientific computing, *PLoS Biology*, *12*, e1001,745.

Zhao, M., J.-C. Golaz, I. M. Held, H. Guo, V. Balaji, R. Benson, J.-H. Chen, X. Chen, L. J. Donner, J. P. Dunne, K. Dunne, J. Durachta, S.-M. Fan, S. M. Freidenreich, S. T. Garner, P. Ginoux, L. M. Harris, L. W. Horowitz, J. P. Krasting, A. R. Langenhorst, Z. Liang, P. Lin, S.-J. Lin, S. L. Malyshev, E. Mason, P. C. D. Milly, Y. Ming, V. Naik, F. Paulot, D. Paynter, P. Phillipps, A. Radhakrishnan, V. Ramaswamy, T. Robinson, D. Schwarzkopf, C. J. Seman, E. Shevliakova, Z. Shen, H. Shin, L. G. Silvers, J. R. Wilson, M. Winton, A. T. Wittenberg, B. Wyman, and B. Xiang (2018), The GFDL global atmosphere and land model AM4.0/LM4.0: 2. Model description, sensitivity studies, and tuning strategies, *J. Adv. Model. Earth Sys.*, *10*, 735–769, doi:10.1002/2017MS001209.