



JAX-SCM v1.0: a modern atmospheric single-column model for boundary layer research

Maximilian Pierzyna¹

¹Department of Geoscience and Remote Sensing, Delft University of Technology, Stevinweg 1, 2628 CN Delft, The Netherlands

Correspondence: Maximilian Pierzyna (mail@maximilian-pierzyna.de)

Abstract. We present JAX-SCM v1.0, an open-source atmospheric single-column model for boundary layer research, implemented in Python using the JAX computing library. The model solves for horizontal wind, potential temperature, and specific humidity, combined with prognostic turbulent kinetic energy and turbulent statistics parameterized by the Mellor-Yamada-Nakanishi-Niino level 2.5 (MYNN-2.5) turbulence closure. We verify the implementation against three well-established benchmark cases covering neutral (turbulent Ekman layer), stable (GABLS1), and convective (Wangara Day 33) conditions. Close agreement with reference solutions is demonstrated across all regimes. By building on JAX, the model benefits from just-in-time compilation and native GPU support. While JAX-SCM is not yet fully differentiable, basing it on JAX also lays the foundation for future integration with machine learning components. The model is designed for simplicity and modularity, lowering the barrier to entry for users and developers alike.

10 1 Introduction

The atmosphere is a system of complex processes that interact across many orders of magnitude, from global circulation ($\mathcal{O}(1000\text{ km})$) to turbulent motion ($\mathcal{O}(1\text{ cm})$). Numerically resolving processes at all these scales is computationally intractable (think, e.g., cloud microphysics or turbulence), so processes smaller than the numerical grid resolution are typically parameterized (Stensrud, 2007). Developing, tuning, and assessing these parameterizations within full three-dimensional (3D) models, however, is challenging. 3D models with parameterizations remain expensive to run, and their complexity makes it difficult to attribute model errors to specific processes or modeling assumptions (Randall et al., 1996). Therefore, atmospheric single-column models (SCMs) have emerged (Betts and Miller, 1986) as simplified one-dimensional (1D) test beds for parameterization. SCMs enable inexpensive atmospheric experiments under controlled conditions (Randall et al., 1996), such as parameter sweeps to assess the sensitivity of model outputs to these parameters (e.g., Betts and Miller (1986); Randall et al. (1996)). Still, when provided with appropriate initial conditions and forcings, SCMs yield realistic simulations of the atmospheric state (e.g., Yamada and Mellor (1975)). Modern SCMs have evolved to capture complex processes, including land-atmosphere interactions (Schulz et al., 2001), ocean-atmosphere interactions (Hartung et al., 2018), aerosol dynamics (Gettelman et al., 2019), and intermittent turbulence (Boyko and Vercauteren, 2024).



However, these SCMs are often implemented in traditional programming languages like Fortran, which hinders their ability to leverage modern computing hardware and techniques. For example, adapting these models to use graphical processing units (GPUs) for better performance, or integrating them with current machine learning (ML) tools to create hybrid physics-ML models, can be difficult. Compared to code written in contemporary languages like Python, Fortran also often appears less readable and harder to maintain (Dahm et al., 2023), which can negatively affect developer and user experience and may slow adoption within the modeling community. Python also enables model interaction through interactive notebooks, which lowers the barrier to experimentation and education.

We address these issues by proposing JAX-SCM, a modern atmospheric single-column model written in Python using the JAX computing library (Bradbury et al., 2018). JAX has recently been successfully used for similar tasks, for example, in computational fluid mechanics (Bezgin et al., 2023, 2025), 3D atmospheric simulations (Kochkov et al., 2024; Yuval et al., 2026; Zhu et al., 2026), or ocean modeling (Häfner et al., 2021; Meunier et al., 2025). On CPU, Häfner et al. (2021) report that the JAX implementation of their ocean model is on par with Fortran and outperforms NumPy (Harris et al., 2020), another popular Python computing library. On GPU, JAX outperforms Fortran for large domains. These results motivate our choice of JAX for JAX-SCM, suggesting high CPU performance for individual runs and high CPU/GPU performance for parallel runs, such as traditional parameter studies. More importantly, the aforementioned models employ JAX to leverage its automatic differentiation (AD) feature, enabling the coupling of numerical models with ML and gradient-based parameter tuning. While JAX-SCM is not yet fully differentiable, we envision using AD in the future, further motivating our choice of JAX as a future-proof platform. For an overview of the opportunities arising from differentiable numerical models, see Gelbrecht et al. (2023).

In addition to its technical advantages, JAX-SCM is designed to be simple and practical. The turbulence closure is the level-2.5 Mellor-Yamada-Nakanishi-Niino (MYNN-2.5) scheme (Mellor and Yamada, 1974, 1982; Nakanishi and Niino, 2006, 2009), which includes a prognostic turbulent kinetic energy (TKE) equation and is widely used operationally, making JAX-SCM immediately applicable to boundary layer research.

The paper is organized as follows. We begin by introducing the governing equations and numerical implementation in sect. 2. In sect. 3, we verify JAX-SCM by running three canonical test cases covering neutral, stably stratified, and convective atmospheric conditions. The current and envisioned advantages of implementing JAX-SCM in Python and JAX are discussed in sect. 4. Finally, sect. 5 concludes the paper.

2 Model description

JAX-SCM advances a one-dimensional (1D) mean state of the atmosphere (horizontal wind, potential temperature, specific humidity, and turbulent kinetic energy) in time under prescribed geostrophic wind, Coriolis force, and surface forcing. Turbulent processes are parameterized by a turbulence closure model, but not resolved. For this first release, the high-order level-2.5 Mellor-Yamada-Nakanishi-Niino turbulence closure scheme (MYNN-2.5) is used (Mellor and Yamada, 1974, 1982; Nakanishi and Niino, 2006, 2009). The governing SCM equations and the MYNN-2.5 scheme are discussed in sect. 2.1. Numerical implementation details are presented in sect. 2.2 followed by an introduction of the surface coupling in sect. 2.3.



2.1 Governing equations

The governing equations are a set of forced, one-dimensional vertical diffusion equations with turbulent fluxes parameterized using the MYNN-2.5 scheme (Nakanishi and Niino, 2009, abbreviated as NN09 in the following). We select MYNN-2.5 as the closure scheme for the initial release of JAX-SCM because it is well tested and used operationally (e.g., Olson et al. (2019)) and because it solves for TKE, which we consider an advantage for boundary layer research. Nevertheless, the modular design of JAX-SCM allows the easy implementation of other turbulence closures in the future.

JAX-SCM implements the NN09 equations unchanged except for the partial-condensation scheme to reduce the complexity of JAX-SCM v1.0. So, while JAX-SCM still solves for specific humidity, which allows moisture in the atmosphere, it does not yet support condensation or cloud formation. Throughout this manuscript, mean quantities are indicated with capital letters (e.g., wind components U and V), turbulent fluctuation quantities with lowercase letters (e.g., u and v), and (parameterized) turbulent statistics with angular brackets (e.g., $\langle w\theta \rangle$). An exception is the turbulent velocity scale q and the turbulent kinetic energy $q^2/2 = (\langle uu \rangle + \langle vv \rangle + \langle ww \rangle)/2$, which are mean statistical quantities but, following Mellor-Yamada notation, use lowercase letters.

The five equations governing the evolution of horizontal momentum (U, V), potential temperature Θ , specific humidity Q_v , and turbulent kinetic energy $q^2/2$ are

$$\frac{\partial U}{\partial t} = -\frac{\partial \langle uw \rangle}{\partial z} + f_c(V - V_g) \quad (1a)$$

$$\frac{\partial V}{\partial t} = -\frac{\partial \langle vw \rangle}{\partial z} - f_c(U - U_g) \quad (1b)$$

$$\frac{\partial \Theta}{\partial t} = -\frac{\partial \langle w\theta \rangle}{\partial z} + \frac{f_c \Theta_0}{g} \left(V \frac{\partial U_g}{\partial z} - U \frac{\partial V_g}{\partial z} \right) \quad (1c)$$

$$\frac{\partial Q_v}{\partial t} = -\frac{\partial \langle wq_v \rangle}{\partial z} \quad (1d)$$

$$\frac{\partial q^2}{\partial t} = -\frac{\partial}{\partial z} \left(-K_q \frac{\partial q^2}{\partial z} \right) + 2P_S + 2P_B - 2\varepsilon. \quad (1e)$$

where f_c is the Coriolis parameter, (U_g, V_g) are geostrophic wind components, Θ_0 is a user-defined reference temperature, and $g = 9.81 \text{ m s}^{-2}$ is Earth's acceleration. The forcing term on the right-hand side (RHS) of Θ (cf. Eq. (1c)) corresponds to horizontal advection of temperature due to thermal winds under the hydrostatic assumption (Yamada and Mellor, 1975; Nakanishi and Niino, 2009).

The TKE sources and sinks in Eq. (1e) are the shear production

$$P_S = -\left(\langle uw \rangle \frac{\partial U}{\partial z} + \langle vw \rangle \frac{\partial V}{\partial z} \right), \quad (2)$$

buoyancy production

$$P_B = \frac{g}{\Theta_0} \langle w\theta_v \rangle, \quad (3)$$

and eddy dissipation rate

$$\varepsilon = q^3 / (B_1 L_M). \quad (4)$$



Note that NN09 use *twice* the TKE $q^2 = 2e = \langle uu \rangle + \langle vv \rangle + \langle ww \rangle$ as prognostic variable, leading to the factors of two in front of P_S , P_B , and ε in Eq. (1e). As the TKE flux divergence is parameterized based on the gradient of q^2 , the factor of two is not needed there. Finally, the buoyancy flux in Eq. (3) is $\langle w\theta_v \rangle = \langle w\theta \rangle + 0.61\Theta \langle wq_v \rangle$ and the closure constant B_1 is $B_1 = 24$ (Nakanishi and Niino, 2009).

Vertical fluxes of a mean quantity Φ are parameterized through gradient diffusion

$$\langle w\phi \rangle = -K_\Phi \frac{\partial \Phi}{\partial z} \quad (5)$$

where ϕ denotes the turbulent fluctuation of Φ , with eddy diffusivity $K_\Phi = L_M q S_\Phi$, $q = \sqrt{q^2}$ the characteristic turbulent velocity scale, L_M the turbulent (master) length scale, and S_Φ the MYNN-2.5 stability function. Specifically, $K_\Phi = K_m$ for momentum ($\Phi \in \{U, V\}$), K_h for heat and moisture ($\Phi \in \{\Theta, Q_v\}$), and $K_q = 3K_m$ for TKE, with corresponding stability functions S_m and S_h . The master length scale L_M is the harmonic mean of characteristic surface (L_S), turbulent (L_T), and buoyancy (L_B) scales:

$$L_M = (L_S^{-1} + L_T^{-1} + L_B^{-1})^{-1}. \quad (6)$$

For brevity, we do not repeat the formulations of the stability functions and length scales here, but refer the interested reader to Nakanishi and Niino (2009).

2.2 Numerical implementation

To numerically solve the set of governing partial differential equations (PDE; cf. Eq. (1)), they need to be discretized in time and space. For spatial discretization, JAX-SCM uses a staggered vertical grid with fixed grid spacing Δz where prognostic variables $\Phi \in \{U, V, \Theta, Q_v, q^2\}$ are located at N full levels (cell centers), while turbulent fluxes $\langle w\phi \rangle$ and diffusivities K_Φ are located at $(N + 1)$ half levels (cell faces). The first full level has height $z_1 = \Delta z/2$ so that the lowest and highest half levels are located at the surface ($z_{(1/2)} = 0$) and the domain top ($z_{N+(1/2)} = H$). Vertical gradients are approximated using central finite differences between full levels, leading to second-order accuracy on half-levels:

$$\left. \frac{\partial \Phi}{\partial z} \right|_{i+(1/2)} \approx \frac{\Phi_{i+1} - \Phi_i}{\Delta z}. \quad (7)$$

To suppress small-scale numerical oscillations of the closure variables in the vertical, 1-2-1 filtering (Shapiro, 1971) of $\psi = \{L_M, K_m, K_h\}$ is implemented at every time step as $\psi_i = (\psi_{i-1} + 2\psi_i + \psi_{i+1})/4$ with the subscript denoting the vertical level and with $\psi_{i-1} = \psi_{i+1}$ for edge values.

Time integration is performed using either second-order-accurate explicit or implicit methods. More specifically, the user can choose between an explicit second-order Adams-Bashforth (AB2) scheme or a semi-implicit Crank-Nicolson (CN) scheme. The AB2 solver determines the solution of a prognostic variable Φ at the future step $n + 1$ (Jacobson, 2005) as

$$\Phi^{n+1} = \Phi^n + \Delta t \left(\frac{3}{2} f^n - \frac{1}{2} f^{n-1} \right) \quad (8)$$



where Δt is the time step in seconds, Φ^n and f^n are the state and tendency at the current step, and f^{n-1} is the tendency from the previous step. Here, “tendency” means the full RHS of the governing PDE. Also, for $n = 0$, it is $f^{n-1} = f^n$ leading to Euler integration as a warm-up step. The AB2 scheme is easy to implement but requires time steps smaller than $\Delta z^2 / \max_z K(z)$ to remain numerically stable (Jacobson, 2005). Consequently, very small time steps are needed for small grid spacings and strong mixing (large K), e.g., in the convective boundary layer, which can slow the simulation down.

The semi-implicit CN solver, on the other hand, is unconditionally stable, allowing larger Δt . The solver is called semi-implicit because the flux divergences and the TKE dissipation rate are treated implicitly, while all other tendencies in Eq. (1) are evaluated explicitly. Expressing the fluxes through gradient diffusion (cf. Eq. (5)), the CN step is

$$\frac{\Phi^{n+1} - \Phi^n}{\Delta t} = \frac{1}{2} \left[\frac{\partial}{\partial z} \left(K_{\Phi}^n \frac{\partial \Phi^n}{\partial z} \right) + \frac{\partial}{\partial z} \left(K_{\Phi}^n \frac{\partial \Phi^{n+1}}{\partial z} \right) \right] + s_{\Phi}^n. \quad (9)$$

The non-linear eddy diffusivities K_{Φ} and source and sink terms s_{Φ} are evaluated explicitly at current time n , leaving only state variables Φ to be solved implicitly. The resulting system of equations is linear in Φ and solved using a tridiagonal solver.

Special care is taken when solving for q^2 as the dissipation rate ε on the RHS also depends directly on q^2 . Janjić (1990) found that explicit treatment of ε can lead to instability, so we treat it semi-implicitly in the CN solver. To keep the system linear in q^2 , the dissipation rate is quasi-linearized by splitting the non-linear expression $\varepsilon = q^3 / (B_1 L_M)$ (cf. Eq. (4)) into a linear part evaluated at time $(n + 1)$ (implicit) and a non-linear part evaluated at n (explicit): $\varepsilon^{n+1} \approx (q^2)^{n+1} q^n / (B_1 L_M^n)$. This quasi-linearized dissipation is added as an additional semi-implicit contribution to the RHS of Eq. (9) as $(\varepsilon^{n+1} + \varepsilon^n) / 2$. Consequently, the explicit sources for q^2 only contain shear and buoyancy production at time n .

The choice of the time integration scheme is up to the user and depends on the case being simulated. AB2 with small time steps is a good choice for debugging and reference when implementing more complex solvers (e.g., other implicit schemes) due to its simplicity. The need for very small time steps, however, can make it impractical in high-resolution convective simulations. The CN solver allows larger time steps and is unconditionally stable, but it requires solving a system of linear equations at each time step, which is more computationally expensive than the simple sum of tendencies in AB2. Because of its stability, CN is a sensible default for production simulations, but its wall-clock time advantage over AB2, due to its larger Δt , is situation-dependent. For both time-stepping techniques, flux boundary conditions are imposed on the domain top and the surface, as described below.

2.3 Boundary conditions

Lower and upper boundary conditions (BCs) are needed for the flux-divergence terms in the governing equations. The upper BC for all prognostic variables is zero flux at the top, and the lower BCs are obtained from Monin-Obukhov similarity theory (MOST, Monin and Obukhov (1954)).

MOST provides surface momentum fluxes $\langle uw \rangle_s$ and $\langle vw \rangle_s$ depending on the wind magnitude $M_1 = (U_1^2 + V_1^2)^{1/2}$ at the lowest full level and a prescribed aerodynamic surface roughness z_{0m} . For surface heat coupling, the user prescribes either a surface sensible heat flux $\langle w\theta \rangle_s$ or a surface temperature Θ_s , both of which can vary in time, together with a surface temperature roughness z_{0h} . If Θ_s is provided, MOST takes Θ_1 at the lowest full level to determine a matching sensible heat flux $\langle w\theta \rangle_s$ as



150 lower BC. If $\langle w\theta \rangle_s$ is prescribed, it is directly used as BC, and Θ_s is diagnosed for convenience. Moisture Q_v is coupled to the surface through a user-defined moisture flux $\langle wq_v \rangle$ serving directly as BC.

Taking the model state at the first full level $z_1 = \Delta z/2$, MOST yields (Wilson, 2001)

$$u_* = \frac{\kappa M_1}{\ln(z_1/z_{0m}) - \Psi_m(z_1/L) + \Psi_m(z_{0m}/L)} \quad (10)$$

and

$$\langle w\theta \rangle_s = -\frac{(\Theta_1 - \Theta_s)\kappa u_*}{\ln(z_1/z_{0h}) - \Psi_h(z_1/L) + \Psi_h(z_{0h}/L)} \quad (11)$$

155 where $\kappa = 0.4$ and $L = -(\Theta_v u_*^3)/(\kappa g \langle w\theta_v \rangle_s)$ is the Obukhov length. The virtual potential temperature Θ_v in L is diagnosed as $\Theta_v = \Theta(1 + 0.61Q_v)$. As u_* and $\langle w\theta \rangle_s$ depend on L but L also depends on both, JAX-SCM determines both quantities iteratively. The surface stresses for the lower momentum BCs, $\langle uw \rangle_s$ and $\langle vw \rangle_s$, are obtained through projection of u_* along the wind components as $\langle uw \rangle_s = -u_*^2(U_1/M_1)$ and $\langle vw \rangle_s = -u_*^2(V_1/M_1)$.

160 The integrated similarity functions needed for Eqns. (10) and (11), $\Psi_m(\zeta)$ and $\Psi_h(\zeta)$ with $\zeta = z/L$, are user-configurable. Various expressions for similarity functions and their coefficients have been proposed (see reviews, e.g., by Wilson (2001); Foken (2006)), which can be implemented and tested in JAX-SCM in the future. Presently, we stick to the original Businger-Dyer relations (Dyer and Hicks, 1970; Businger et al., 1971) in their integrated form (Paulson, 1970) given as

$$\Psi_m = \begin{cases} 2 \ln[(1+x)/2] + \ln[(1+x^2)/2] & \zeta < 0 \\ -2 \arctan(x) + \pi/2 & \\ -b\zeta & \zeta \geq 0 \end{cases} \quad (12)$$

for momentum and

$$165 \Psi_h = \begin{cases} 2 \ln[(1+x^2)/2] & \zeta < 0 \\ -b\zeta & \zeta \geq 0 \end{cases} \quad (13)$$

for heat. In both equations, $x = (1 - \gamma\zeta)^{1/4}$, $\gamma = 16$ and $b = 5$.

Finally, the diagnosed TKE shear production (cf. Eq. (2)) requires the gradients of U and V at the lowest half-level $z_{1/2}$, but only the gradient at the second-lowest level $z_{3/2}$ can be computed through central differences. Because the gradients closest to the surface are the strongest, we choose not to extrapolate the gradients from $z_{3/2}$. Instead, we obtain them using MOST, similar to Moeng (1984). Using the gradient-form ϕ_m of the similarity function Ψ_m , one gets at $z_1 = \Delta z/2$

$$\frac{\partial M}{\partial z} \Big|_{z_1} = \frac{u_*}{\kappa z_1} \phi_m(z_1/L). \quad (14)$$

Here, $\phi_m(\zeta) = 1/x(\zeta)$ for $\zeta < 0$ and $\phi_m(\zeta) = 1 + b\zeta$ for $\zeta \geq 0$. Projecting the gradient onto the wind directions, we obtain $(\partial U/\partial z)|_{z_1} = (\partial M/\partial z)|_{z_1} U_1/M_1$ and $(\partial V/\partial z)|_{z_1} = (\partial M/\partial z)|_{z_1} V_1/M_1$, respectively.

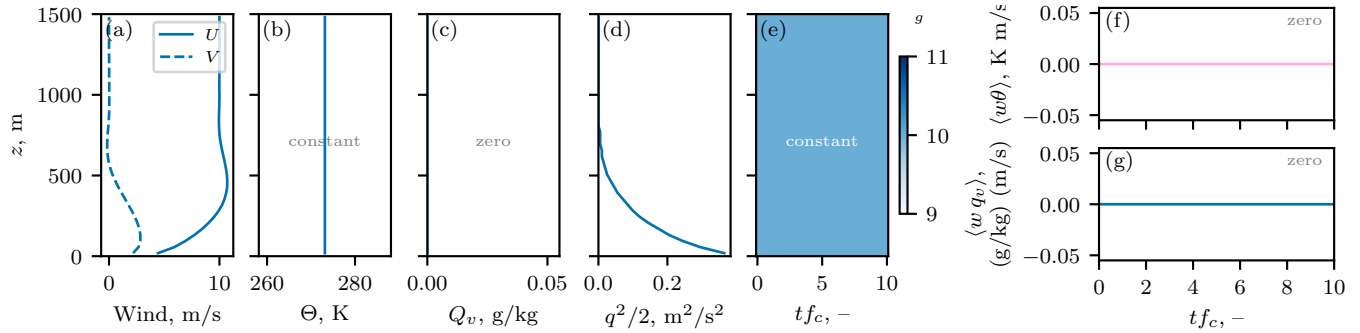


Figure 1. Initial and boundary conditions for the neutral turbulent Ekman layer test case (Andren et al., 1994). Initial wind (a) and TKE (d) profiles are prescribed together with constant geostrophic forcing (e). All other initial profiles and surface forcings are zero (constant for Θ). Domain height is $H = 1500$ m, discretized into $N = 100$ levels ($\Delta z = 15$ m).

3 Test cases

175 To verify the correct implementation of JAX-SCM, we run three well-known benchmark cases that cover the neutral, stable, and convective atmospheric stability regimes, in order of increasing complexity. The first case is the turbulent neutral Ekman layer discussed by Andren et al. (1994) and presented in sect. 3.1. The case is synthetic and forced only by constant geostrophic wind and Coriolis force, but has zero surface fluxes. As a stable test case, we take the GABLS1 SCM intercomparison of Cuxart et al. (2006) and compare JAX-SCM to their results in sect. 3.2. The case is a dry synthetic case with constant surface cooling leading to increasing stratification. Finally, day 33 of the 1967 Wangara field campaign (Clarke et al., 1971) is reproduced and discussed in sect. 3.3. This case is the most complex and also the most realistic of the three. The simulation is initialized with observed wind, temperature, and humidity profiles and forced with time-varying surface sensible heat and moisture fluxes. This case demonstrates the real-world applicability of JAX-SCM.

3.1 Andren et al. (1994): neutral turbulent Ekman layer

185 The goal of the neutral turbulent Ekman layer is to evaluate how JAX-SCM behaves in a simple synthetic case: a flow driven by a constant large-scale pressure gradient in a rotating system (Andren et al., 1994, A94 from now on). The large-scale pressure gradient is imposed in the simulation as a constant geostrophic wind forcing of $U_g = 10 \text{ m s}^{-1}$. The Coriolis frequency is set to $f_c = 10^{-4}$, corresponding to ca. 45°N latitude, and the surface has a prescribed roughness of $z_{0m} = 0.1$ m. As this case targets neutral atmospheric stratification, no sensible heat and moisture fluxes are prescribed at the surface. The initial conditions for u , v , and q^2 profiles are taken from Table A.1 of A94 and are visualized in Fig. 1, panels (a)–(d). The constant forcings are presented in panels (e), (f), and (g). The domain height is set to $H = 1500$ m and is discretized into $N = 100$ full levels, leading to $\Delta z = 15$ m. The simulation is run for 10 inertial periods, i.e., $t = 10/f_c \approx 27.78$ h.

The simulated output of JAX-SCM (orange) is compared to the ensemble of simulations intercompared in A94 (grey) in Fig. 2. Only plots relevant to verifying the SCM performance are presented here, and discussions in A94 related to the original

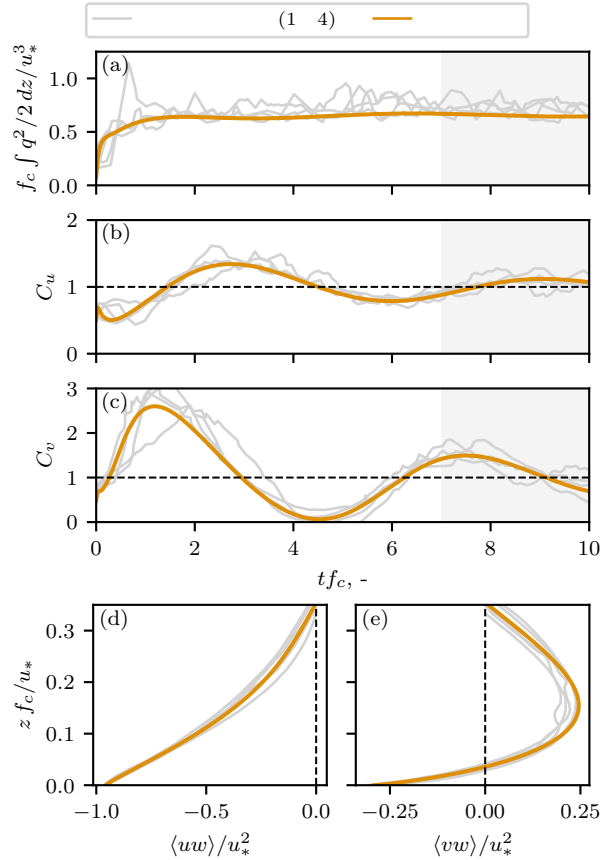


Figure 2. Comparison of JAX-SCM and Andren et al. (1994) reference results in terms of (a) column-integrated normalized TKE, normalized deviation from steady state of mean wind profiles of (b) U and (c) V , and (d+e) averaged normalized momentum flux profiles. Averaging window of momentum fluxes (d+e) is $t f_c \in [7, 10]$ as visualized by grey shading in (a)–(c).

195 benchmark of large eddy simulation (LES) are skipped. We also treat the four codes in A94 as a single reference ensemble, as we only aim to verify the correct implementation of JAX-SCM.

Panel (a) of Fig. 2 shows the column-integrated TKE, normalized with f_c and the surface friction velocity u_* from MOST; panels (b) and (c) assess the convergence of the simulation toward a steady state solution by plotting the non-stationarity of the velocity fields at every time step as

$$200 \quad C_u = -\frac{f_c}{\langle uw \rangle_s} \int_0^H (V - V_g) dz \quad (15a)$$

$$C_v = +\frac{f_c}{\langle vw \rangle_s} \int_0^H (U - U_g) dz \quad (15b)$$

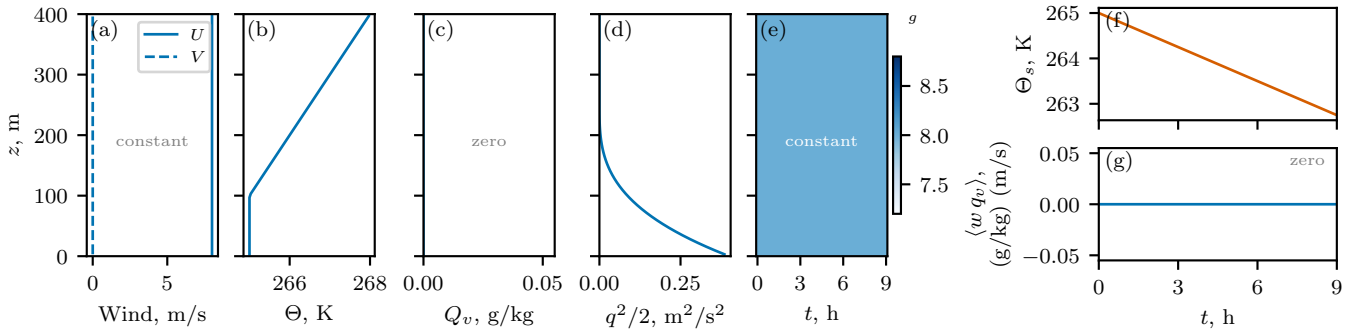


Figure 3. Initial and boundary conditions for dry stable boundary layer test case (GABLS1, Cuxart et al. (2006)). Initial profiles for (a) wind, (b) potential temperature, and (d) TKE are prescribed together with (e) constant geostrophic wind and (f) constant cooling of the surface. Domain height is $H = 400$ m, discretized into $N = 64$ levels ($\Delta z = 6.25$ m).

where $\langle uw \rangle_s$ and $\langle vw \rangle_s$ are the surface momentum fluxes from MOST; and panels (d) and (e) present the normalized vertical profiles of momentum fluxes. Following A94, the momentum flux profiles are averaged over the last three inertial periods as indicated by the grey shading in panels (a)–(c). In all five panels, JAX-SCM agrees closely with the A94 reference simulation. The JAX-SCM curves are smoother because the contribution of turbulence to the flow is fully parameterized, whereas the LES runs partially resolve turbulence, leading to greater variability. Along this line of thought, it is particularly encouraging for future turbulence research that the integrated TKE and the MYNN-2.5-parameterized momentum flux profiles match the LES results so well. In conclusion, JAX-SCM simulates the neutral Ekman layer very well, verifying the applicability to shear-driven flows.

210 3.2 GABLS1: stable boundary layer

The next step is to assess JAX-SCM’s performance in simulating a stably stratified atmospheric boundary layer. The test case employed here comes from the GABLS1 SCM intercomparison conducted by Cuxart et al. (2006), who derived it from Kosović and Curry (2000). Similar to A94, the simulation is subject to constant geostrophic forcing ($U_g = 8 \text{ m s}^{-1}$) but experiences a constant cooling of 0.25 K h^{-1} at the surface. JAX-SCM directly supports this surface forcing as the user can prescribe either a heat flux or the surface temperature (used here) as the lower BC. The initial profiles and forcings following Cuxart et al. (2006) are visualized in Fig. 3. The initial wind (panel a) is constant with height, humidity (panel c) is zero, and the potential temperature profile shows a neutral layer of depth 100 m with stable stratification on top. The simulation runs for 9 h of simulated time in a 400 m domain, discretized into $N = 64$ levels.

The comparison of JAX-SCM against GABLS1 is presented in Fig. 4. As for A94, we do not consider the SCMs benchmarked in GABLS1 individually, but only verify JAX-SCM against the ensemble. The profiles in panels (a)–(f) correspond to the state at the end of the simulation, after 9 h, showing horizontal wind magnitude $M = (U^2 + V^2)^{1/2}$, potential temperature, parameterized sensible heat flux, parameterized turbulent stress τ , and the momentum and heat eddy diffusivities. The two panels (g) and (h) show the temporal evolution of the boundary layer height h and the surface friction velocity, as determined

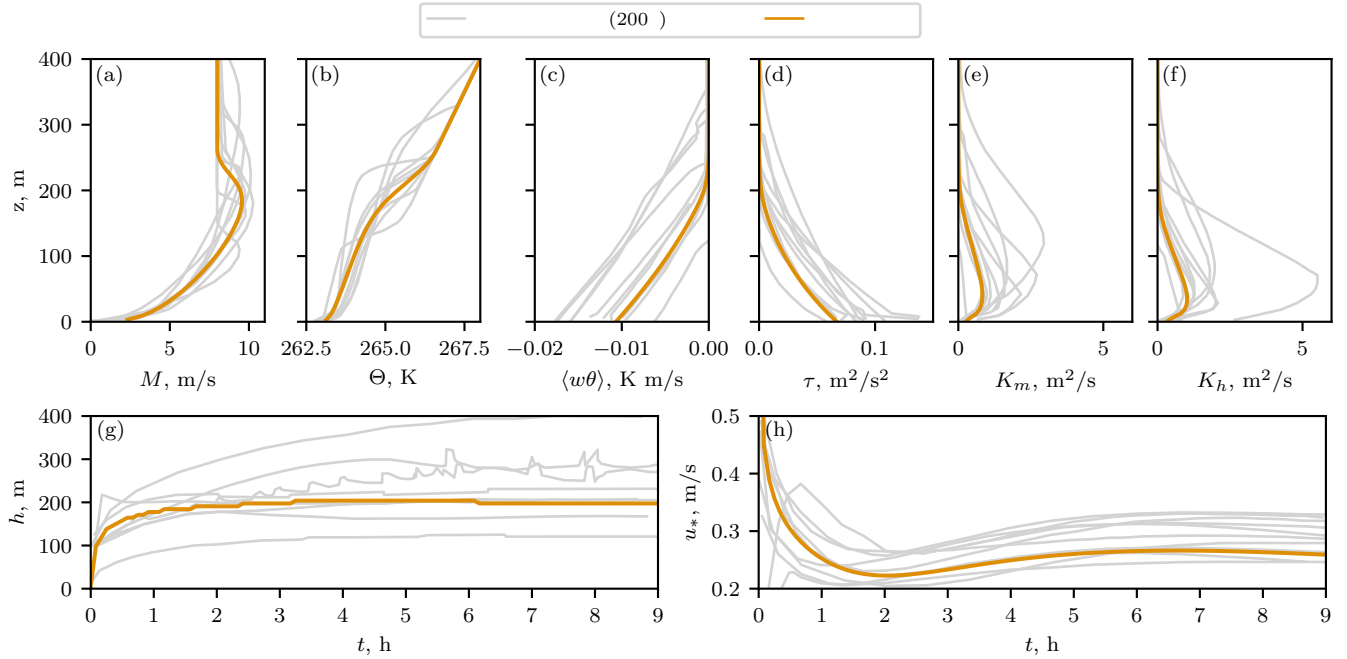


Figure 4. Comparison of JAX-SCM and GABLS1 SCM reference results (Cuxart et al., 2006) with respect to (a-f) profiles at $t = 9h$ and temporal evolutions of (g) boundary layer height h and (h) surface friction velocity. Panel (a) shows wind magnitude $M = (U^2 + V^2)^{1/2}$ and panel (d) the total turbulent stress $\tau = (\langle uw \rangle^2 + \langle vw \rangle^2)^{1/2}$.

by MOST. The total turbulent stress τ in panel (d) is computed from the parameterized vertical momentum flux profiles as

$$225 \quad \tau = \left(\langle uw \rangle^2 + \langle vw \rangle^2 \right)^{1/2}, \quad (16)$$

and h in panel (g) is defined as the height where τ vanishes. More specifically, h is computed by determining the height where τ falls to 5% of its surface value, followed by a linear extrapolation to where $\tau \approx 0$ (Cuxart et al., 2006; Beare et al., 2006): $h = z_{\tau,5\%}/0.95$ where $\tau(z)/\tau_s \approx 0.05$ is met at $z_{\tau,5\%}$.

Similar to A94, the JAX-SCM outputs align well with the GABLS1 reference. All curves demonstrate the expected behavior, including the profiles of turbulent quantities ($\langle w\theta \rangle$, τ , K_m , K_h) parameterized by MYNN-2.5. Like the reference models, JAX-SCM successfully exerts surface friction on the initially constant wind profiles, decelerating wind at the surface and forming a jet at $z \approx 180$ m (cf. Fig. 4a). The constant surface cooling successfully yields a stratified atmosphere (cf. Fig. 4b) and a realistic surface sensible heatflux of $\langle w\theta \rangle \approx -0.01$ K m s⁻¹ (cf. Fig. 4c). JAX-SCM's option to prescribe the surface heat flux indirectly through the surface temperature is crucial for accurately simulating moderately stable to very stable conditions due to the duality of the heat flux (Malhi, 1995; Basu et al., 2008a).

In conclusion, we consider the GABLS1 verification successful, demonstrating that JAX-SCM can simulate stable atmospheric boundary layers.

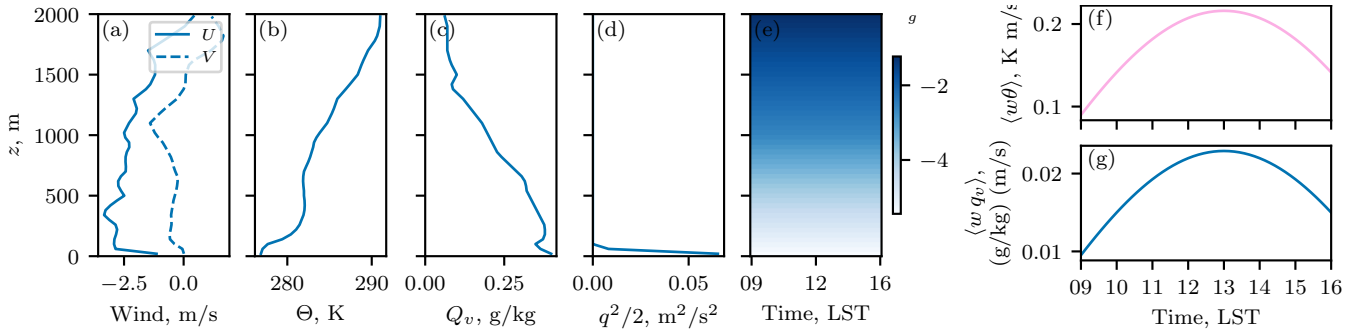


Figure 5. Initial and boundary conditions for convective boundary layer test case, day 33 of the Wangara field campaign (Clarke et al., 1971; Nakanishi and Niino, 2009). Initial profiles of (a) wind, (b) temperature, and (c) moisture are obtained from 09:00 soundings at Wangara. Simulation is forced with (e) constant-in-time geostrophic wind profiles and time-dependent (f) sensible heat flux and (g) moisture flux at the surface. Domain height is $H = 2000$ m, discretized into $N = 100$ levels ($\Delta z = 20$ m).

3.3 Wangara, day 33: convective boundary layer

The final test case is the simulation of a moist convective boundary layer with time-dependent surface forcing and observation-based initial conditions. More specifically, we simulate day 33 of the 1967 Wangara field campaign (Clarke et al., 1971), which has been used as a test case in numerous publications (e.g., Wyngaard and Coté (1974); Yamada and Mellor (1975); Pielke and Mahrer (1975); Chen and Cotton (1983); Musson-Genon (1995); Kim et al. (2003); Basu et al. (2008b)). Crucially, Wangara Day 33 (WG33) is also the test case employed by Nakanishi and Niino (2009), whose closure model JAX-SCM implements.

The initial conditions and forcings are summarized in Fig. 5. Averaged soundings of mean wind, potential temperature, and specific humidity at 9:00 local time of campaign day 33 (16 August 1967) serve as initial conditions (Clarke et al., 1971, data page 227). As neither Clarke et al. (1971) nor NN09 specify an initial profile of q^2 , we estimate a (twice) TKE surface value of q^2 as $q_s^2 = B_1^{2/3} u_*^2$ (Mellor and Yamada, 1982) and prescribe a vertical power-law decay of that value as

$$q^2(z) = \begin{cases} q_s^2 [1 - (z/h_0)]^3 & z \leq h_0 \\ 0 & z > h_0. \end{cases} \quad (17)$$

From the observed profile of Θ (cf. Fig. 5b), we estimate an initial boundary layer height of $h_0 = 100$ m for the equation above. The TKE surface value is estimated as $q_s^2/2 \approx 0.1274 \text{ m}^2 \text{ s}^{-2}$ based on an observed $u_* \approx 0.175 \text{ m s}^{-1}$ (Hicks, 1981) and $B_1 = 24$ (Nakanishi and Niino, 2009). The domain height is set to $H = 2000$ m, in agreement with the maximum balloon height and previous studies. We choose to discretize the domain into $N = 100$ levels, leading to a grid spacing of $\Delta z = 20$ m. Time integration is performed from 9:00 on day 33 until 16:00 on the same day, matching NN09.

Forcing enters the simulation through prescribed geostrophic wind, surface fluxes, and the Coriolis force. Following NN09, the geostrophic wind forcing is height-dependent but constant in time (cf. Fig. 5e), and the Coriolis parameter of $f_c \approx 8.26 \times 10^{-5} \text{ s}^{-1}$ is set to match Wangara's latitude of 34.5°S . The surface fluxes of moisture and heat are time-dependent and take the

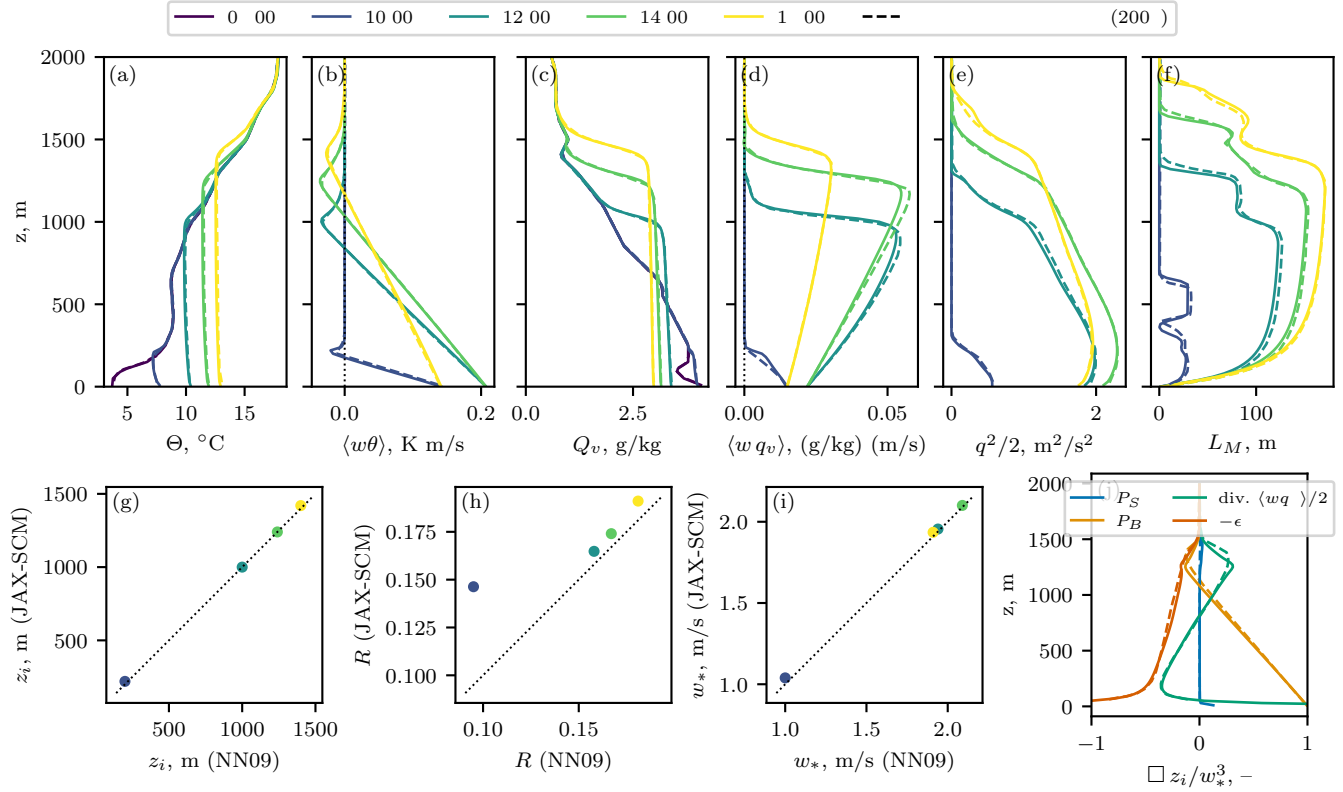


Figure 6. Comparison of JAX-SCM and reference results from Nakanishi and Niino (2009) for Wangara day 33. Temporal evolutions of profiles of mean and parameterized turbulent quantities are presented in (a) to (f). Mixed layer parameters are displayed in panels (g) to (i) with (g) inversion height z_i , (h) ratio R of $\langle w\theta \rangle$ at z_i and surface, (i) and convective velocity scale w_* . Panel (j) shows the normalized TKE budget over height at 14:00 LST.

form

$$\langle w\phi \rangle_s = c \cos\left(\frac{t_h - 13}{11} \pi\right) \quad (18)$$

where t_h is the simulated time in hours since midnight of day 33, and $c = 2.16 \times 10^{-1} \text{ K m s}^{-1}$ for $\langle w\theta \rangle_s$ and $c = 2.29 \times 10^{-5} \text{ m s}^{-1}$ for $\langle wqv \rangle_s$, respectively.

In contrast to the previous two cases, the WG33 test case is quite realistic. The simulation is initialized from observations and subject to simplified but realistic time-dependent forcing. Now, we also simulate the evolution of the specific humidity Q_v , verifying the final prognostic equation of JAX-SCM that has not been addressed so far. Thus, this test employs all of JAX-SCM's features and also demonstrates its real-world applicability.

The comparison of JAX-SCM and the reference results from NN09 is presented in Fig. 6. The first row displays the temporal evolution of profiles of potential temperature, specific humidity, their fluxes, TKE, and the master length scale throughout the



day. The reference solution of NN09 is given as dashed lines. Panels (g)–(i) in the second row compare three mixed-layer parameters at different simulation times. The inversion height z_i in panel (g) is defined as the height where $\langle w\theta \rangle$ is minimal. The strength of $\langle w\theta \rangle$ at that height relative to the surface value is quantified as $R = -\langle w\theta \rangle_{z_i} / \langle w\theta \rangle_s$ (cf. panel (h)). The convective velocity scale $w_* = (g/\Theta_0 z_i \langle w\theta_v \rangle_s)^{1/3}$ with surface buoyancy flux $\langle w\theta_v \rangle_s$ is compared against NN09 values in (i). Finally, panel (j) illustrates the TKE budget at 14:00, normalized by z_i and w_* , respectively.

Overall, JAX-SCM matches all profiles very well, as expected given that it implements MYNN-2.5 according to NN09. We attribute the minor differences visible between our code and the reference to the numerical implementation of the models. It is also likely that our TKE initial profile differs from that used by NN09, leading to differences, especially at the beginning of the simulation (e.g., $\langle wq_v \rangle$ and L_M at 10:00). Similarly, the notable deviation of R at 10:00 in Fig. 6h is attributed to model spin-up and differences in vertical discretization, which may affect the resolution of the thin low entrainment layer at the beginning of the simulation. As the related values z_i and w_* are captured very well, we do not consider this deviation problematic.

In general, we view the WG33 test case as captured very well. Despite the minor differences discussed, the key behavior of the atmospheric column – boundary layer growth, drying, and TKE budget – is simulated realistically and accurately. We consider the verification of JAX-SCM complete, having demonstrated the successful simulation of a neutral, a stably stratified, and a convective boundary layer.

4 Technical details

Having verified the correctness of JAX-SCM, we now discuss two technical aspects of its implementation that are relevant to users and developers. The composable Python interface is presented in sect. 4.1 using the A94 simulation as an example. The choice of JAX as the numerical backend and its current and envisioned advantages are discussed in sect. 4.2.

4.1 Modern composable Python interface

The simple, composable Python interface of JAX-SCM is briefly presented by discussing the code needed to set up the A94 simulation (cf. sect. 3.1), which is given in listing 1.

After importing the necessary packages, the script begins with setting up the staggered grid, followed by defining the forcings in lines 14–32. As all forcings of the A94 case are constant in time, simple lambda functions are used to return the same vectors of geostrophic wind at every time step t_s as well as constant values of zero for surface sensible heat flux and moisture flux. For more complex forcings, such as the time-dependent sine-like forcing of WG33 (cf. Eq. (18)), users can implement a simple Python function taking the current simulation time in seconds as input and returning the corresponding flux. For the WG33 sensible heat flux, for example, the function could read

```
def w_thl_fn(t_s: float) -> float:
    t_h = t_s / 3600 # seconds to hours
    return 2.16e-1 * jnp.cos((t_h - 13) / 11 * jnp.pi)
```



Listing 1. Setup of initial and boundary conditions for Andren et al. (1994) neutral case. Listing 2 presents how to run the simulation.

```
1: import jax.numpy as jnp
2: import pandas as pd
3:
4: from scm import convert
5: from scm.grid import StaggeredGrid
6: from scm.interfaces import Simulation, Forcing
7: from scm.mo import MOSettings
8: from scm.mynn.interfaces import ProgVarsMYNN
9:
10: # Grid: 100 levels, 1500m domain height -> dz=15m
11: Nz = 100
12: grid = StaggeredGrid(Nz=Nz, H=1500)
13:
14: # Compute Coriolis parameter at 45 degree latitude
15: f_c = convert.get_fc(lat_deg=45) # ca. 1e-4 1/s
16:
17: # Constant geostrophic wind
18: u_g = jnp.ones(Nz) * 10
19: v_g = jnp.zeros(Nz)
20:
21: # Surface friction
22: mo_settings = MOSettings(z0h=0.1, z0m=0.1)
23:
24: # Collect all forcing terms into a Forcing object
25: forcing = Forcing(
26:     u_geo=lambd t_s: u_g,
27:     v_geo=lambd t_s: v_g,
28:     f_c=f_c,
29:     w_th_s=lambd t_s: jnp.array(0.0), # no surface heat flux
30:     w_qv_s=lambd t_s: jnp.array(0.0), # no surface moisture flux
31:     dth_dz_top=0.0, # no inversion at domain top
32: )
33:
34: # Read initial profiles from csv file (Andren et al. 1994, Table A1)
35: df = pd.read_csv("ref/andren94_tab_A1.csv")
36:
37: # Interpolate profiles to model grid
38: u = jnp.interp(grid.z, df["z"].values, df["u"].values)
39: v = jnp.interp(grid.z, df["z"].values, df["v"].values)
40: qke = jnp.interp(grid.z, df["z"].values, df["tke"].values) * 2 # QKE = 2 * TKE
41:
42: # Collect all initial conditions into an object of prognostic variables
43: init = ProgVarsMYNN(
44:     u=u,
45:     v=v,
46:     th=jnp.ones(Nz) * 273.15, # constant potential temperature profile
47:     qv=jnp.zeros(Nz), # no humidity
48:     qke=qke,
49: )
50:
51: # Assemble simulation object with all components
52: sim = Simulation(
53:     name="Andren1994",
54:     init=init,
55:     forcing=forcing,
56:     mo_settings=mo_settings,
57:     grid=grid,
58:     th_ref=273.15,
59:     t_start_s=0, # simulation start time
60:     t_end_s=int(10 / f_c), # simulate for 10 inertial periods
61: )
```



Listing 2. Running a JAX-SCM simulation. See listing 1 for an example setup of a `sim` object and listing 3 for an example namelist file.

```
1: from scm.config import load_namelist
2: from scm.io.local import out_to_ds
3: from scm.mynn.model import init_model
4: from scm.time_stepping import simulate
5:
6: # Setup simulation and load config
7: sim = ...
8: cfg = load_namelist("namelist.yaml")
9:
10: # Setup model and run simulation
11: model = init_model(sim, cfg)
12: out = simulate(model=model, sim=sim, cfg=cfg)
13:
14: # Save output to netcdf
15: ds = out_to_ds(out, sim)
16: ds.to_netcdf("andren1994.nc")
```

Listing 3. Namelist file to configure semi-implicit Crank-Nicolson time stepping.

```
1: time_int: "implicit"
2: dt_s: .5
3: dt_s_out: 300.0
```

and would be assigned to `w_th_s` in l. 29 instead of the lambda expression.

300 The initial conditions are read from a text file using the pandas library (l. 35; pandas-dev (2026)) and interpolated to the model grid (ll. 37–40). We stress that relying on a well-tested library like pandas simplifies our setup and enables JAX-SCM to ingest initial conditions in diverse formats. The initial conditions, the forcings, and simulation metadata are all collected in one consistent `Simulation` object. This object behaves like a tree, which can be passed around in Python code without the risk of “losing” or mismatching any settings along the way.

305 Once the simulation object is created, running JAX-SCM requires only a few steps as presented in listing 2. After loading the namelist file (l. 8) and setting up the model (l. 11), the simulation is executed in line 12. The namelist file enables the user, e.g., to configure the simulation output frequency or select the time integration method. An example namelist file configuring the CN solver with an internal time step of $\Delta t = 0.5$ s and an output frequency of 5 min is given in listing 3. The simulation outputs are converted to an xarray dataset in line 15. Xarray (Hoyer and Joseph, 2017) is another popular data-handling library that interfaces with various storage backends, such as netcdf (cf. l. 16) or the modern zarr format (zarr-developers, 2026). Being 310 able to use such an established library for data output is another benefit of the Python ecosystem.

Overall, we believe that JAX-SCM strikes a good balance between numerical performance via JAX and usability with Python. The technical advantages of utilizing JAX are presented in the following section.



4.2 GPU runs and automatic differentiation with JAX

As motivated in the introduction, JAX-SCM is implemented in Python to improve accessibility and enable the use of modern GPU hardware and ML-ready software tools. However, as an interpreted programming language, plain Python trades performance for ease of use, making it much slower than compiled languages like Fortran. To still keep the numerical components fast and efficient, we leverage the computing library JAX (Bradbury et al., 2018), which performs just-in-time compilation of Python code into fast XLA (Accelerated Linear Algebra; OpenXLA (2026)) code. XLA supports compilation for multiple hardware backends, notably standard CPU and modern GPUs. Häfner et al. (2021), for example, document on-par CPU performance between JAX-based and Fortran-based versions of their ocean model and a ca. 2x speedup of the JAX-GPU version over Fortran. Just-in-time compilation is transparent to the user, so JAX-SCM runs on CPUs and GPUs alike without any code changes. While a single JAX-SCM run does not require the parallelization offered by GPUs, GPUs can greatly accelerate tasks such as parameter sweeps or ensemble simulations with perturbed initial conditions to quantify uncertainties.

Additionally, JAX natively supports automatic differentiation (AD), which enables computing derivatives of implemented functions with respect to any of their inputs, down to machine precision (Baydin et al., 2018). Currently, JAX-SCM is not yet fully differentiable, as special care must be taken to avoid breaking gradients (Meunier et al., 2025). In MYNN-2.5, for example, computing the turbulent velocity scale $q = \sqrt{q^2}$ breaks gradients near $q \rightarrow 0$, as $d/dx(\sqrt{x}) = 1/(2\sqrt{x})$ goes to infinity at $x = 0$. In contrast to standard machine learning functions, physical parameterizations are not designed with differentiability in mind and therefore require modifications to avoid such gradient issues. We target full differentiability in future work, but would still like to briefly introduce AD as a key feature of JAX and motivate the envisioned use cases.

Gradients are computed by AD in two steps (Baydin et al., 2018). First, during model execution, the forward pass, the chain of elementary numerical operations (e.g., addition, division, exponentiation, sine, cosine, ...) used to produce the solution is stored in a computational graph. Analytical derivatives of all these elementary operations are known and implemented in JAX. To compute the gradient/derivatives of the solution with respect to the inputs, the computation graph is traced backward, and gradients are computed by evaluating the elementary derivatives and applying the chain rule. This reverse-mode AD is equivalent to the adjoint state method, which is typically challenging to apply for models implemented in traditional languages. In JAX, however, the adjoint is available “for free”, provided all gradients are well-behaved.

Once JAX-SCM is fully differentiable, AD can be used, for example, to tune the SCM so that its simulated state trajectory more closely matches a reference trajectory (Gelbrecht et al., 2023). This is needed, for example, to tune the parameters of turbulence closure models efficiently or to perform data assimilation, where initial conditions are optimized to match observed states. Moreover, such fully-differentiable numerical models enable the integration of machine learning components into the numerical solver (cf. e.g., (Kochkov et al., 2021, 2024; Sanderse et al., 2024)). As the Python-JAX setup generally enables these applications, we believe that JAX-SCM will be a powerful platform for future hybrid physics–machine-learning approaches.



5 Conclusions

345 We presented JAX-SCM v1.0, a modern open-source atmospheric single-column model (SCM) implemented in Python using the JAX library (Bradbury et al., 2018). The model solves for mean wind, temperature, humidity, and turbulent kinetic energy (TKE) using the level-2.5 Mellor-Yamada-Nakanishi-Niino (MYNN-2.5) turbulence closure scheme (Nakanishi and Niino, 2009), and its modular design allows alternative closure schemes to be integrated in the future. The Python interface aims to be a simple, modern, and accessible interface for users and developers alike. By choosing Python, JAX-SCM can be combined
350 with the extensive ecosystem of scientific packages. The JAX backend adds performance competitive with compiled Fortran via just-in-time compilation (Häfner et al., 2021), transparent GPU execution for computationally demanding tasks such as parameter sweeps and ensemble simulations, and native automatic differentiation (AD) as a path toward future hybrid physics–machine learning approaches (Gelbrecht et al., 2023).

We verified the correct implementation of JAX-SCM by simulating three well-known benchmark cases: the neutral turbulent
355 Ekman layer (Andren et al., 1994), the dry stable boundary layer of GABLS1 (Kosović and Curry, 2000; Cuxart et al., 2006), and the convective boundary layer of Wangara day 33 (Clarke et al., 1971; Yamada and Mellor, 1975). JAX-SCM matched the reference solutions closely in all three cases, confirming its correct implementation across all atmospheric stability regimes. Parameterized turbulent quantities, such as heat and momentum fluxes and TKE, were well captured throughout, establishing JAX-SCM as an immediately applicable tool for atmospheric boundary layer research. The close agreement with Wangara
360 day 33, initialized from observed profiles and forced with realistic time-dependent surface fluxes, further demonstrates its real-world applicability.

Looking ahead, two main directions are planned for future releases. First, a partial-condensation scheme (e.g., Nakanishi and Niino (2009)) will be implemented to enable cloud formation, opening JAX-SCM to the historically central use case of developing and testing cloud parameterizations (Betts and Miller, 1986; Randall et al., 1996). Second, and most importantly, we
365 aim to make JAX-SCM fully differentiable. Once differentiable, JAX-SCM will support gradient-based parameter optimization and data assimilation, as well as direct integration of machine-learning components within the solver (Gelbrecht et al., 2023). We envision similar hybrid physics–ML approaches demonstrated at the global scale (Kochkov et al., 2024, 2021) also for the atmospheric boundary layer.

In conclusion, JAX-SCM is a modern and accessible SCM that is immediately applicable to boundary layer research and is
370 designed from the ground up as a platform for future hybrid physics–machine-learning approaches in the atmospheric boundary layer.

Code and data availability. JAX-SCM is available on GitHub (https://github.com/mpierzyna/jax_scm) or on Zenodo (Pierzyna (2026); <https://doi.org/10.5281/zenodo.20314409>). The validation data were digitized without modification from the respective original publications and are also available on GitHub and Zenodo.

<https://doi.org/10.5194/egusphere-2026-2916>

Preprint. Discussion started: 11 June 2026

© Author(s) 2026. CC BY 4.0 License.



375 *Author contributions.* MP has conceived the idea for JAX-SCM, implemented it, validated it, and wrote the manuscript.

Competing interests. MP does not declare any competing interests.

Acknowledgements. MP receives funding from the FREE project (P19-13) of the research programme TTW-Perspectief which is (partly) financed by the Dutch Research Council (NWO). MP is grateful to Sukanta Basu, Yi Dai, and André van Ginkel for discussions and feedback regarding JAX-SCM. Generative large language models were used to edit the draft manuscript and to optimize the JAX-SCM codebase.



380 References

- Andren, A., Brown, A. R., Mason, P. J., Graf, J., Schumann, U., Moeng, C.-H., and Nieuwstadt, F. T. M.: Large-eddy Simulation of a Neutrally Stratified Boundary Layer: A Comparison of Four Computer Codes, *Quarterly Journal of the Royal Meteorological Society*, 120, 1457–1484, <https://doi.org/10.1002/qj.49712052003>, 1994.
- Basu, S., Holtslag, A. A. M., Van De Wiel, B. J. H., Moene, A. F., and Steeneveld, G.-J.: An Inconvenient “Truth” about Using Sensible Heat Flux as a Surface Boundary Condition in Models under Stably Stratified Regimes, *Acta Geophysica*, 56, 88–99, <https://doi.org/10.2478/s11600-007-0038-y>, 2008a.
- Basu, S., Vinuesa, J.-F., and Swift, A.: Dynamic LES Modeling of a Diurnal Cycle, *Journal of Applied Meteorology and Climatology*, 47, 1156–1174, <https://doi.org/10.1175/2007JAMC1677.1>, 2008b.
- Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M.: Automatic Differentiation in Machine Learning: A Survey, *Journal of Machine Learning Research*, 18, 1–43, 2018.
- Beare, R. J., Macvean, M. K., Holtslag, A. A. M., Cuxart, J., Esau, I., Golaz, J.-C., Jimenez, M. A., Khairoutdinov, M., Kosovic, B., Lewellen, D., Lund, T. S., Lundquist, J. K., McCabe, A., Moene, A. F., Noh, Y., Raasch, S., and Sullivan, P.: An Intercomparison of Large-Eddy Simulations of the Stable Boundary Layer, *Boundary-Layer Meteorology*, 118, 247–272, <https://doi.org/10.1007/s10546-004-2820-6>, 2006.
- Betts, A. K. and Miller, M. J.: A New Convective Adjustment Scheme. Part II: Single Column Tests Using GATE Wave, BOMEX, ATEX and Arctic Air-Mass Data Sets, *Quarterly Journal of the Royal Meteorological Society*, 112, 693–709, <https://doi.org/10.1002/qj.49711247308>, 1986.
- Bezgin, D. A., Buhendwa, A. B., and Adams, N. A.: JAX-Fluids: A Fully-Differentiable High-Order Computational Fluid Dynamics Solver for Compressible Two-Phase Flows, *Computer Physics Communications*, 282, 108 527, <https://doi.org/10.1016/j.cpc.2022.108527>, 2023.
- Bezgin, D. A., Buhendwa, A. B., and Adams, N. A.: JAX-Fluids 2.0: Towards HPC for Differentiable CFD of Compressible Two-Phase Flows, *Computer Physics Communications*, 308, 109 433, <https://doi.org/10.1016/j.cpc.2024.109433>, 2025.
- Boyko, V. and Vercauteren, N.: Simulating the Unsteady Stable Boundary Layer With a Stochastic Stability Equation, *Journal of Geophysical Research: Atmospheres*, 129, e2023JD039 370, <https://doi.org/10.1029/2023JD039370>, 2024.
- Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Katariya, Y., Leary, C., Maclaurin, D., Nacula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q.: JAX: Composable Transformations of Python+NumPy Programs, 2018.
- Businger, J. A., Wyngaard, J. C., Izumi, Y., and Bradley, E. F.: Flux-Profile Relationships in the Atmospheric Surface Layer, *Journal of the Atmospheric Sciences*, 28, 181–189, [https://doi.org/10.1175/1520-0469\(1971\)028<0181:FPRITA>2.0.CO;2](https://doi.org/10.1175/1520-0469(1971)028<0181:FPRITA>2.0.CO;2), 1971.
- Chen, C. and Cotton, W. R.: Numerical experiments with a one-dimensional higher order turbulence model: Simulation of the Wangara day 33 case, *Boundary-Layer Meteorology*, 25, 375–404, 1983.
- Clarke, R., Dyer, A., Brook, R., Reid, D., and Troup, A.: The Wangara Experiment: Boundary Layer Data, Technical Paper 19, Commonwealth Scientific and Industrial Research Organization, Australia, 1971.
- Cuxart, J., Holtslag, A. A. M., Beare, R. J., Bazile, E., Beljaars, A., Cheng, A., Conangla, L., Ek, M., Freedman, F., Hamdi, R., Kerstein, A., Kitagawa, H., Lenderink, G., Lewellen, D., Mailhot, J., Mauritsen, T., Perov, V., Schayes, G., Steeneveld, G.-J., Svensson, G., Taylor, P., Weng, W., Wunsch, S., and Xu, K.-M.: Single-Column Model Intercomparison for a Stably Stratified Atmospheric Boundary Layer, *Boundary-Layer Meteorology*, 118, 273–303, <https://doi.org/10.1007/s10546-005-3780-1>, 2006.



- Dahm, J., Davis, E., Deconinck, F., Elbert, O., George, R., McGibbon, J., Wicky, T., Wu, E., Kung, C., Ben-Nun, T., Harris, L., Groner, L., and Fuhrer, O.: Pace v0.2: A Python-based Performance-Portable Atmospheric Model, *Geoscientific Model Development*, 16, 2719–2736, <https://doi.org/10.5194/gmd-16-2719-2023>, 2023.
- Dyer, A. J. and Hicks, B. B.: Flux-Gradient Relationships in the Constant Flux Layer, *Quarterly Journal of the Royal Meteorological Society*, 420 96, 715–721, <https://doi.org/10.1002/qj.49709641012>, 1970.
- Foken, T.: 50 Years of the Monin–Obukhov Similarity Theory, *Boundary-Layer Meteorology*, 119, 431–447, <https://doi.org/10.1007/s10546-006-9048-6>, 2006.
- Gelbrecht, M., White, A., Bathiany, S., and Boers, N.: Differentiable Programming for Earth System Modeling, *Geoscientific Model Development*, 16, 3123–3135, <https://doi.org/10.5194/gmd-16-3123-2023>, 2023.
- 425 Gettelman, A., Truesdale, J. E., Bacmeister, J. T., Caldwell, P. M., Neale, R. B., Bogenschutz, P. A., and Simpson, I. R.: The Single Column Atmosphere Model Version 6 (SCAM6): Not a Scam but a Tool for Model Evaluation and Development, *Journal of Advances in Modeling Earth Systems*, 11, 1381–1401, <https://doi.org/10.1029/2018MS001578>, 2019.
- Häfner, D., Nuterman, R., and Jochum, M.: Fast, Cheap, and Turbulent—Global Ocean Modeling With GPU Acceleration in Python, *Journal of Advances in Modeling Earth Systems*, 13, e2021MS002717, <https://doi.org/10.1029/2021MS002717>, 2021.
- 430 Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E.: Array Programming with NumPy, *Nature*, 585, 357–362, <https://doi.org/10.1038/s41586-020-2649-2>, 2020.
- Hartung, K., Svensson, G., Struthers, H., Deppenmeier, A.-L., and Hazeleger, W.: An EC-Earth Coupled Atmosphere–Ocean Single-Column 435 Model (AOSCM.v1_EC-Earth3) for Studying Coupled Marine and Polar Processes, *Geoscientific Model Development*, 11, 4117–4137, <https://doi.org/10.5194/gmd-11-4117-2018>, 2018.
- Hicks, B. B.: An Analysis of Wangara Micrometeorology: Surface Stress, Sensible Heat, Evaporation, and Dewfall, NOAA Technical Memorandum TM ERL ARL-104, National Oceanic and Atmospheric Administration, Oak Ridge, Tennessee, USA, 1981.
- Hoyer, S. and Joseph, H.: Xarray: N-D Labeled Arrays and Datasets in Python, <https://doi.org/10.5334/jors.148>, 2017.
- 440 Jacobson, M. Z.: *Fundamentals of Atmospheric Modeling*, Cambridge University Press, Cambridge, second edition edn., <https://doi.org/10.1017/CBO9781139165389>, 2005.
- Janjić, Z. I.: The Step-Mountain Coordinate: Physical Package, *Monthly Weather Review*, 118, 1429–1443, [https://doi.org/10.1175/1520-0493\(1990\)118<1429:TSMCPP>2.0.CO;2](https://doi.org/10.1175/1520-0493(1990)118<1429:TSMCPP>2.0.CO;2), 1990.
- Kim, S., Yamaguchi, K., Kondo, A., and Soda, S.: A comparative study of the Mellor–Yamada and $k-\epsilon$ two-equation turbulence models in 445 atmospheric application, *Journal of Wind Engineering and Industrial Aerodynamics*, 91, 791–806, 2003.
- Kochkov, D., Smith, J. A., Alieva, A., Wang, Q., Brenner, M. P., and Hoyer, S.: Machine Learning–Accelerated Computational Fluid Dynamics, *Proceedings of the National Academy of Sciences*, 118, e2101784118, <https://doi.org/10.1073/pnas.2101784118>, 2021.
- Kochkov, D., Yuval, J., Langmore, I., Norgaard, P., Smith, J., Mooers, G., Klöwer, M., Lottes, J., Rasp, S., Düben, P., Hatfield, S., Battaglia, P., Sanchez-Gonzalez, A., Willson, M., Brenner, M. P., and Hoyer, S.: Neural General Circulation Models for Weather and Climate, 450 *Nature*, 632, 1060–1066, <https://doi.org/10.1038/s41586-024-07744-y>, 2024.
- Kosović, B. and Curry, J. A.: A Large Eddy Simulation Study of a Quasi-Steady, Stably Stratified Atmospheric Boundary Layer, *Journal of the Atmospheric Sciences*, 57, 1052–1068, [https://doi.org/10.1175/1520-0469\(2000\)057<1052:ALESSO>2.0.CO;2](https://doi.org/10.1175/1520-0469(2000)057<1052:ALESSO>2.0.CO;2), 2000.



- Malhi, Y. S.: The Significance of the Dual Solutions for Heat Fluxes Measured by the Temperature Fluctuation Method in Stable Conditions, *Boundary-Layer Meteorology*, 74, 389–396, <https://doi.org/10.1007/BF00712379>, 1995.
- 455 Mellor, G. L. and Yamada, T.: A Hierarchy of Turbulence Closure Models for Planetary Boundary Layers, *Journal of the Atmospheric Sciences*, 31, 1791–1806, [https://doi.org/10.1175/1520-0469\(1974\)031<1791:AHOTCM>2.0.CO;2](https://doi.org/10.1175/1520-0469(1974)031<1791:AHOTCM>2.0.CO;2), 1974.
- Mellor, G. L. and Yamada, T.: Development of a Turbulence Closure Model for Geophysical Fluid Problems, *Reviews of Geophysics*, 20, 851–875, <https://doi.org/10.1029/RG020i004p00851>, 1982.
- Meunier, E., Ouala, S., Frezat, H., Sommer, J. L., and Fablet, R.: Towards Fully Differentiable Neural Ocean Model with Veros, <https://doi.org/10.48550/arXiv.2511.17427>, 2025.
- 460 Moeng, C.-H.: A Large-Eddy-Simulation Model for the Study of Planetary Boundary-Layer Turbulence, *Journal of the Atmospheric Sciences*, 41, 2052–2062, [https://doi.org/10.1175/1520-0469\(1984\)041<2052:ALESMF>2.0.CO;2](https://doi.org/10.1175/1520-0469(1984)041<2052:ALESMF>2.0.CO;2), 1984.
- Monin, A. S. and Obukhov, A. M.: Basic Laws of Turbulent Mixing in the Surface Layer of the Atmosphere, *Contrib. Geophys. Inst. Acad. Sci.*, 151, 1954.
- 465 Musson-Genon, L.: Comparison of different simple turbulence closures with a one-dimensional boundary layer model, *Monthly weather review*, 123, 163–180, 1995.
- Nakanishi, M. and Niino, H.: An Improved Mellor–Yamada Level-3 Model: Its Numerical Stability and Application to a Regional Prediction of Advection Fog, *Boundary-Layer Meteorology*, 119, 397–407, <https://doi.org/10.1007/s10546-005-9030-8>, 2006.
- Nakanishi, M. and Niino, H.: Development of an Improved Turbulence Closure Model for the Atmospheric Boundary Layer, *Journal of the Meteorological Society of Japan. Ser. II*, 87, 895–912, <https://doi.org/10.2151/jmsj.87.895>, 2009.
- 470 Olson, J. B., Kenyon, J. S., Angevine, Wayne. A., Brown, J. M., Pagowski, M., and Sušelj, K.: A Description of the MYNN-EDMF Scheme and the Coupling to Other Components in WRF–ARW, Technical Report OAR GSD-61, Earth System Research Laboratory (U.S.), Global Systems Division, 2019.
- OpenXLA: [Openxla/Xla](https://openxla.org/), OpenXLA, 2026.
- 475 pandas-dev: [Pandas-Dev/Pandas](https://pandas.pydata.org/): Pandas, <https://doi.org/10.5281/zenodo.3509134>, 2026.
- Paulson, C. A.: The Mathematical Representation of Wind Speed and Temperature Profiles in the Unstable Atmospheric Surface Layer, *Journal of Applied Meteorology*, 9, 857–861, [https://doi.org/10.1175/1520-0450\(1970\)009<0857:TMROWS>2.0.CO;2](https://doi.org/10.1175/1520-0450(1970)009<0857:TMROWS>2.0.CO;2), 1970.
- Pielke, R. A. and Mahrer, Y.: Representation of the heated planetary boundary layer in mesoscale models with coarse vertical resolution, *Journal of Atmospheric Sciences*, 32, 2288–2308, 1975.
- 480 Pierzyna, M.: JAX-SCM v1.0: a modern atmospheric single-column model for boundary layer research, <https://doi.org/10.5281/zenodo.20314409>, 2026.
- Randall, D. A., Xu, K.-m., Somerville, R. J. C., and Iacobellis, S.: Single-Column Models and Cloud Ensemble Models as Links between Observations and Climate Models, *Journal of Climate*, 9, 1683–1697, [https://doi.org/10.1175/1520-0442\(1996\)009<1683:SCMACE>2.0.CO;2](https://doi.org/10.1175/1520-0442(1996)009<1683:SCMACE>2.0.CO;2), 1996.
- 485 Sanderse, B., Stinis, P., Maulik, R., and Ahmed, S. E.: Scientific Machine Learning for Closure Models in Multiscale Problems: A Review, <https://doi.org/10.48550/arXiv.2403.02913>, 2024.
- Schulz, J.-P., Dümenil, L., and Polcher, J.: On the Land Surface–Atmosphere Coupling and Its Impact in a Single-Column Atmospheric Model, *Journal of Applied Meteorology*, 40, 642–663, [https://doi.org/10.1175/1520-0450\(2001\)040<0642:OTLSAC>2.0.CO;2](https://doi.org/10.1175/1520-0450(2001)040<0642:OTLSAC>2.0.CO;2), 2001.
- Shapiro, R.: The Use of Linear Filtering as a Parameterization of Atmospheric Diffusion, *Journal of the Atmospheric Sciences*, 28, 523–531, [https://doi.org/10.1175/1520-0469\(1971\)028<0523:TUOLFA>2.0.CO;2](https://doi.org/10.1175/1520-0469(1971)028<0523:TUOLFA>2.0.CO;2), 1971.
- 490



- Stensrud, D. J.: Why Study Parameterization Schemes?, in: Parameterization Schemes: Keys to Understanding Numerical Weather Prediction Models, pp. 1–11, Cambridge University Press, 1 edn., <https://doi.org/10.1017/CBO9780511812590>, 2007.
- Wilson, D. K.: An Alternative Function For The Wind And Temperature Gradients In Unstable Surface Layers, *Boundary-Layer Meteorology*, 99, 151–158, <https://doi.org/10.1023/A:1018718707419>, 2001.
- 495 Wyngaard, J. C. and Coté, O.: The evolution of a convective planetary boundary layer—A higher-order-closure model study, *Boundary-Layer Meteorology*, 7, 289–308, 1974.
- Yamada, T. and Mellor, G.: A Simulation of the Wangara Atmospheric Boundary Layer Data, *Journal of the Atmospheric Science*, 32, 2309–2329, 1975.
- Yuval, J., Langmore, I., Kochkov, D., and Hoyer, S.: Neural General Circulation Models for Modeling Precipitation, *Science Advances*, 12, 500 eadv6891, <https://doi.org/10.1126/sciadv.adv6891>, 2026.
- zarr-developers: Zarr-Developers/Zarr-Python, Zarr Developers, 2026.
- Zhu, X., Qu, Y., and Shi, X.: LEX v1.6.0: A New Large-Eddy Simulation Model in JAX with GPU Acceleration and Automatic Differentiation, *Geoscientific Model Development*, 19, 1103–1120, <https://doi.org/10.5194/gmd-19-1103-2026>, 2026.