

# Description of a Python Package for Initializing an Idealized Baroclinic Wave in the WRF model

Daniel Lloveras

May 2021

## 1 Introduction

This document describes how to create initial conditions for the simulation of a realistic idealized mid-latitude cyclone with the Weather Research and Forecasting (WRF) model. Specifically, this document describes how to conduct this initialization using the Python script `bwave_wrfinput.py` and its associated package `bwave_ideal_wrf`. In order to run the Python scripts documented here, the user must have up-to-date versions of Numpy, Numba, and netCDF4-Python.

The scripts described here are designed to resemble the `em_b_wave` test case that is built into the WRF model. The main script `bwave_wrfinput.py` creates a netCDF file named `wrfinput_d01` which, with a properly-compiled WRF model, can be used to run a numerical simulation of an idealized mid-latitude cyclone developing within a zonally-periodic  $f$ -plane channel. In this way, these scripts play the role of the WRF executable `ideal.exe`. So, upon creating the input file, the user may simply run `wrf.exe` with an appropriate `namelist.input` file, and the model will produce a highly realistic mid-latitude cyclone.

There are four main steps to the initialization of the cyclone. First, a two-dimensional field of Ertel's potential vorticity (EPV) is inverted in order to obtain a realistic mid-latitude jet with a corresponding north-south temperature gradient. Second, a three-dimensional ball of quasi-geostrophic potential vorticity (QGPV) is inverted in order to obtain perturbation fields which trigger baroclinic growth. Third, the zonally-symmetric jet variables and three-dimensional perturbation fields are interpolated onto the WRF grid and added together, a moisture profile is prescribed, and the remaining WRF fields are computed. Lastly, all of the variables and attributes that WRF needs to run the model are written into a netCDF file. The four modules detailed below correspond to these four steps.

## 2 2-D EPV Inversion (epv\_jet.py)

This module contains functions which are used to conduct the two-dimensional EPV inversion in order to create a zonally-uniform jet with a meridional temperature gradient. This module is based on the initialization methodology described in Rotunno et al. (1994), and on code written by Riwal Plougonven and edited by various researchers, notably Max Menchaca and Lydia Tierney.

### 2.1 Functions and Formulas

Firstly, the functions `y_grid` and `pi_grid` initialize the meridional and vertical grids for the EPV inversion. The vertical coordinate used for the inversion is the Exner function, which is defined as:

$$\Pi = c_p \left( \frac{p}{p_0} \right)^{R_d/c_p} \quad (1)$$

where  $p$  is the atmospheric pressure,  $c_p = 1004 \text{ J K}^{-1} \text{ kg}^{-1}$  is the specific heat of air at constant pressure,  $R_d = 287 \text{ J K}^{-1} \text{ kg}^{-1}$  is the ideal gas constant of dry air, and  $p_0 = 1000 \text{ hPa}$  is a reference pressure. This coordinate thus has units of  $\text{J K}^{-1} \text{ kg}^{-1}$ .

The WRF vertical grid, which is based on the hydrostatic pressure-following coordinate  $\eta$ , is defined using the function `eta_grid`. The vertical coordinate is defined as:

$$\eta = \frac{p_h - p_{h,top}}{p_{h,bot} - p_{h,top}} \quad (2)$$

where  $p_h$  is the hydrostatic pressure,  $p_{h,top}$  is the hydrostatic pressure at the model top, and  $p_{h,bot}$  is the hydrostatic surface pressure. Thus,  $\eta$  is defined as 1 at the surface and 0 at the top of the atmosphere. At each vertical level  $k$ , the vertical coordinate is prescribed as:

$$\eta(k) = \frac{(e^{-2k/100} - e^{-2})}{(1 - e^{-2})} \quad (3)$$

There are two linear interpolation functions defined in this module, `interp_increasing` for a coordinate that increases in magnitude with each grid point (such as  $y$ ) and `interp_decreasing` for a coordinate that decreases in magnitude with each grid point (such as  $\eta$ ). The two functions are defined differently to reduce the amount of if statements, but both use the same basic formula for linear interpolation:

$$y = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) \quad (4)$$

where  $y$  is the value of the variable on the new grid,  $x$  is the value of the coordinate on the new grid, and subscripts refer to the values on the original grid.

The potential temperature ( $\theta$ ) field at the top of the atmosphere acts as a boundary condition for the EPV inversion. It is defined as a function of  $y$  (meridional distance in meters) with the function `theta_top` as:

$$\theta_{TOP}(y) = \begin{cases} \theta_m - \Delta\theta_e & \text{if } \sigma < -\frac{\pi}{2} \\ \theta_m + \Delta\theta_e \sin(\sigma) & \text{if } -\frac{\pi}{2} \leq \sigma \leq \frac{\pi}{2} \\ \theta_m + \Delta\theta_e & \text{if } \sigma > \frac{\pi}{2} \end{cases} \quad (5)$$

where  $\theta_m$  is the average  $\theta_{TOP}$ ,  $\Delta\theta_e$  is the maximum deviation from the average  $\theta_{TOP}$ ,  $\sigma = 1.5(y - \frac{1}{2}y_{tot})/\Delta y_{th}$ ,  $y_{tot}$  is the meridional domain length, and  $\Delta y_{th}$  is the meridional scale for the  $\theta_{TOP}$  transition.

The height of the tropopause is defined in  $\Pi$  as a function of  $y$  using the function `trop_shape`:

$$\Pi_{TP}(y) = \begin{cases} \Pi_m - \Delta\Pi_e & \text{if } \epsilon < -\frac{\pi}{2} \\ \Pi_m + \Delta\Pi_e \sin(\epsilon) & \text{if } -\frac{\pi}{2} \leq \epsilon \leq \frac{\pi}{2} \\ \Pi_m + \Delta\Pi_e & \text{if } \epsilon > \frac{\pi}{2} \end{cases} \quad (6)$$

where  $\Pi_m$  is the average height of the tropopause,  $\Delta\Pi_e$  is the maximum deviation from the average tropopause height,  $\epsilon = 2(y - \frac{1}{2}y_{tot})/\Delta y_e$ , and  $\Delta y_e$  is the meridional scale for the tropopause height transition.

The two dimensional EPV distribution is defined using the function `pv_dist` as:

$$PV(y, \Pi) = \frac{PV_T \cdot a + PV_S \cdot b}{2} + \frac{PV_T \cdot a - PV_S \cdot b}{2} \cdot \tanh\left(2\frac{\Pi - \Pi_{TP}}{\Delta\Pi_{TP}}\right) \quad (7)$$

$$a(y, \Pi) = \begin{cases} 1 + 3\gamma^2 & \text{if } \Pi \geq \Pi_{TP} \\ 1 & \text{if } \Pi < \Pi_{TP} \end{cases} \quad (8)$$

$$b(y, \Pi) = \begin{cases} 1 & \text{if } \Pi \geq \Pi_{TP} \\ 1 + 21\beta^2 & \text{if } \Pi < \Pi_{TP} \end{cases} \quad (9)$$

where  $\gamma = (\Pi_0 - \Pi)/(\Pi_0 - \Pi_{TP})$ ,  $\beta = (\Pi_{TP} - \Pi)/(\Pi_{TP} - \Pi_{TOP})$ ,  $\Pi_0 = 1004 \text{ J K}^{-1} \text{ kg}^{-1}$  is the height corresponding to the reference pressure  $p_0 = 1000 \text{ hPa}$ , and  $\Pi_{TOP}$  is the height at the top of the domain.

The inversion is solved using the function `solve_PV_inversion`, which calls the functions `pv_inv`, `u_calc`, `theta_calc`, and `pv_calc`. Note: it is most computationally effective to first run this solver on a low-resolution grid and then interpolate this solution onto the model grid (using `interp_decreasing` and `interp_increasing`) for use as a "first guess" at the full solution. This reduces the amount of iterations needed to achieve convergence to a solution, and it is the methodology used in `bwave_wrfinput.py`.

The function `solve_PV_inversion` solves the nonlinear PV equation in  $\Pi$  coordinates for the geopotential  $\phi$ :

$$PV(y, \Pi) = \frac{g\kappa c_p^{1/\kappa}}{p_0} \Pi^{1-1/\kappa} \left( f_0 \frac{\partial^2 \phi}{\partial \Pi^2} + f_0^{-1} \frac{\partial^2 \phi}{\partial y^2} \frac{\partial^2 \phi}{\partial \Pi^2} - f_0^{-1} \left( \frac{\partial^2 \phi}{\partial y \partial \Pi} \right)^2 \right) \quad (10)$$

where  $g = 9.81 \text{ m s}^{-2}$  is the gravitational constant,  $\kappa = R_d/c_p$  and  $f_0 = 10^{-4} \text{ s}^{-1}$  is the constant Coriolis parameter. The inversion is solved by plugging in second-order finite difference approximations for the derivatives, which are given by:

$$\frac{\partial^2 \phi}{\partial y^2}(j, k) = \frac{\phi(j+1, k) - 2\phi(j, k) + \phi(j-1, k)}{\Delta y^2} \quad (11)$$

$$\frac{\partial^2 \phi}{\partial \Pi^2}(j, k) = \frac{\phi(j, k+1) - 2\phi(j, k) + \phi(j, k-1)}{\Delta \Pi^2} \quad (12)$$

$$\frac{\partial^2 \phi}{\partial y \partial \Pi}(j, k) = \frac{\phi(j+1, k+1) - \phi(j+1, k-1) - \phi(j-1, k+1) + \phi(j-1, k-1)}{4\Delta y \Delta \Pi} \quad (13)$$

where  $j$  and  $k$  represent grid points in the  $y$  and  $\Pi$  directions, respectively. This results in an equation to be solved for  $\phi(j, k)$ :

$$\begin{aligned} & \phi^2(j, k) - \frac{1}{2} [\phi(j+1, k) + \phi(j-1, k) + \phi(j, k+1) + \phi(j, k-1) + (f\Delta y)^2] \phi(j, k) \\ & - \frac{1}{4} [PV(j, k) f (\Delta y \Delta \Pi)^2 \frac{p_0}{g\kappa c_p^{1/\kappa}} \Pi^{-(1-1/\kappa)} \\ & + \frac{1}{16} (\phi(j+1, k+1) - \phi(j-1, k+1) - \phi(j+1, k-1) + \phi(j-1, k-1))^2 \\ & - (\phi(j+1, k) + \phi(j-1, k)) (\phi(j, k+1) + \phi(j, k-1)) \\ & - (f\Delta y)^2 (\phi(j, k+1) + \phi(j, k-1))] = 0 \end{aligned} \quad (14)$$

The function `pv_inv` solves this quadratic at each grid point using the quadratic formula, resulting in a  $\phi^*$  value at each point. The inversion is iterated 1000 times using a successive over-relaxation (SOR) method with a coefficient of 1.8. At each iteration  $n$ ,  $\phi^*$  is calculated and  $\phi$  is updated using `pv_inv`:

$$\phi^n(j, k) = \phi^{n-1}(j, k) + 1.8(\phi^*(j, k) - \phi^{n-1}) \quad (15)$$

The solver function `solve_PV_inversion` computes the EPV distribution, tropopause height, and top potential temperature, and then calls `pv_inv` for its 1000 iterations. At each iteration, the boundary conditions are applied.  $\frac{\partial \phi}{\partial y} = 0$  at the lateral boundaries and  $\phi = 0$  along the bottom boundary. At the top boundary,  $\phi$  is constrained by the potential temperature at the top of the atmosphere,  $\theta_{TOP}$ , such that  $\frac{\partial \phi}{\partial \Pi} = \theta_{TOP}$ .

The solution results in a full geopotential field  $\phi$ , which is used by `solve_PV_inversion` to compute the two-dimensional zonal wind ( $u$ ) and potential temperature fields by applying geostrophic and hydrostatic balance with the functions `u_calc` and `theta_calc`:

$$u(y, \Pi) = -\frac{1}{f_0} \frac{\partial \phi}{\partial y} \quad (16)$$

$$\theta(y, \Pi) = \frac{\partial \phi}{\partial \Pi} \quad (17)$$

Finally, PV is computed using `pv_calc` to check to see if the inversion was solved correctly (that is, that the solved geopotential field, when plugged into the nonlinear PV equation (10), leads to a field similar to the original prescribed PV distribution).

The function `eta_calc` computes the jet variables on WRF's  $\eta$  vertical grid using linear interpolation. First, pressure levels are computed using the unstaggered  $\eta$  grid values:

$$p(j, k) = \eta(k)(p_{surf} - p_{top}) + p_{top} \quad (18)$$

where  $p_{top}$  is the pressure at the top of the atmosphere and  $p_{surf}$  is the surface pressure. Then, the potential temperature and zonal wind values are linearly interpolated onto these pressure levels using `interp_decreasing`. These values are used to calculate the density field  $\rho$ , which is in turned used to calculate the height field using hydrostatic balance:

$$\rho = \frac{p_0}{R_d \theta} \left( \frac{p}{p_0} \right)^{c_v/c_p} \quad (19)$$

$$\frac{dp}{dz} = -\rho g \quad (20)$$

where  $c_v = 717 \text{ J K}^{-1} \text{ kg}^{-1}$  is the specific heat of air at constant volume. Also, `eta_calc` computes the change in potential temperature with height  $\frac{d\theta}{dz}$  and the Brunt-Vaisala frequency  $N$ :

$$N = \sqrt{\frac{g}{\theta} \frac{d\theta}{dz}} \quad (21)$$

## 2.2 Output Variables

Variable	Description	Dimensions
p_jet	Pressure in Pa, $p$ in (18)	(ny, nz)
p_surf_jet	Surface pressure in Pa, $p_{surf}$ in (18)	(ny)
theta_jet	Potential temperature in K, $\theta$ after interpolation	(ny, nz)
u_jet	Zonal wind in $\text{m s}^{-1}$ , $u$ after interpolation	(ny, nz)
rho_jet	Density in $\text{kg m}^{-3}$ , $\rho$ in (19)	(ny, nz)
z_jet	Height in m, $z$ in (20)	(ny, nz)
dtdz_jet	$d\theta/dz$ in $\text{K m}^{-1}$	(ny, nz)
n_jet	Brunt-Vaisala frequency in $\text{s}^{-1}$ , $N$ in (21)	(ny, nz)
znw	Staggered $\eta$ levels, $\eta$ in (3)	(nz)
znu	Untaggered $\eta$ levels	(nz)

## 3 3-D QGPV Inversion (qgpv\_pert.py)

This module contains functions which are used to conduct the three-dimensional QGPV inversion in order to create perturbation fields which trigger the baroclinic growth of a mid-latitude cyclone. This module is based on code written by Greg Hakim and edited by various researchers, notably Max Menchaca and Lydia Tierney.

### 3.1 Functions and Formulas

The QGPV anomaly is initialized in Cartesian coordinates ( $x, y, z$ , in meters), and the inversion is solved spectrally in the horizontal direction (where  $k$  is the zonal wavenumber and  $l$  is the meridional wavenumber) using a matrix method. So, the first step is to initialize the grids using the functions `cartesian_mesh` and `spectral_mesh`, which make use of the Numpy function `numpy.meshgrid`. Next, the anomaly is initialized in Cartesian space and then transformed into spectral space using the function `qgpv_anom`. The anomaly  $Q'$  is created with the following formula:

$$Q'(x, y, z) = Q_0 e^{-(s/\delta_h)^2} e^{-[(z-z_c)/\delta_\nu]^2} \quad (22)$$

where  $(x_c, y_c, z_c)$  is the location of the center of the anomaly,  $Q_0$  is the maximum magnitude of the anomaly in  $\text{s}^{-1}$ ,  $\delta_\nu$  is the vertical decay scale,  $\delta_h$  is the horizontal decay scale, and  $s^2 = (x - x_c)^2 + (y - y_c)^2$ . The anomaly is transformed into spectral space using the Numpy function `numpy.fft.fft2`, which conducts a two-dimensional fast Fourier transform.

The anomaly is inverted using the function `qgpv_inversion`, which solves the Boussinesq,  $f$ -plane form of the QGPV equation, which is given by:

$$q = \nabla_H^2 \psi + f_0 + \frac{\partial}{\partial z} \left( \frac{f_0^2}{N^2} \frac{\partial \psi}{\partial z} \right) \quad (23)$$

where  $\psi = \phi/f_0$  is the quasi-geostrophic streamfunction. If we take  $Q' = q - f_0$  and assume that  $\frac{\partial N}{\partial z}$  is negligible, we can rewrite this equation as:

$$Q' = \frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} + \frac{f_0^2}{N^2} \frac{\partial^2 \psi}{\partial z^2}. \quad (24)$$

To solve this equation spectrally, we assume wave-like solutions in the horizontal direction such that:

$$\psi(x, y, z) = \psi(z) e^{i(kx - ly)} \quad (25)$$

$$Q'(x, y, z) = Q'(z) e^{i(kx - ly)} \quad (26)$$

Plugging these into (24) results in a second order linear ODE to solve for  $\psi(z)$  at each spectral grid point:

$$b(z) \frac{d^2 \psi}{dz^2} - (k^2 + l^2) \psi(z) = Q'(z) \quad (27)$$

Note that the factor  $b(z) = f_0^2/N^2$  varies with height, so this is a variable-coefficient equation. If we use a simple centered finite difference approximation for the derivative:

$$D^2 \Psi_i = \frac{1}{\Delta z^2} (\Psi_{i-1} - 2\Psi_i + \Psi_{i+1}) \quad (28)$$

we can write the ODE as  $A\Psi = Q$ , where the tridiagonal matrix  $A$  is written as:

$$A = \frac{1}{\Delta z^2} \begin{bmatrix} -\Delta z^2(k^2 + l^2) - 2b_1 & b_1 & & & \\ b_2 & -\Delta z^2(k^2 + l^2) - 2b_2 & b_2 & & \\ & & \ddots & & \\ & & & b_{n-1} & \\ & & & b_n & -\Delta z^2(k^2 + l^2) - 2b_n \end{bmatrix} \quad (29)$$

The equation  $A\Psi = Q$  is solved at each spectral gridpoint  $(k, l)$  using the Numpy function `numpy.linalg.solve`. The boundary conditions are also applied such that  $\partial \Psi / \partial z = 0$  along the top and bottom boundaries. This gives a solution for  $\Psi(k, l, z)$ .

This solution is transformed back into Cartesian space by conducting a two-dimensional inverse fast Fourier transform using the Numpy function `numpy.fft.ifft2`. Subsequently, the perturbation wind, potential temperature, and density fields are computed from the solution for  $\Psi$  using geostrophic and hydrostatic balance. The computations of  $u'$ ,  $v'$ , and  $\rho'$  are trivial, as:

$$u' = -\frac{\partial \psi}{\partial y} \quad (30)$$

$$v' = \frac{\partial \psi}{\partial x} \quad (31)$$

$$\rho' = -\frac{\rho_0 f_0}{g} \frac{\partial \psi}{\partial z} \quad (32)$$

where  $\rho_0 = 1.0 \text{ kg m}^{-3}$  is a reference density. The computation of  $\theta'$  involves a form of the QGPV equation that is equivalent to (22):

$$q = \nabla_H^2 \psi + f_0 + \frac{\partial}{\partial z} \left( f_0 \left( \frac{d\bar{\theta}}{dz} \right)^{-1} \theta \right). \quad (33)$$

We obtain an expression for  $\theta'$  by equating the  $\frac{\partial}{\partial z}$  terms in (23) and (33):

$$\theta' = \frac{f_0}{N^2} \frac{d\bar{\theta}}{dz} \frac{\partial \psi}{\partial z} \quad (34)$$

where  $\frac{d\bar{\theta}}{dz}$  is obtained from the EPV inversion described above.

## 3.2 Output Variables

Variable	Description	Dimensions
u_pert	Zonal wind anomaly in $\text{m s}^{-1}$ , $u'$ in (30)	(nx, ny, nz)
v_pert	Meridional wind anomaly in $\text{m s}^{-1}$ , $v'$ in (31)	(nx, ny, nz)
rho_pert	Density anomaly in $\text{kg m}^{-3}$ , $\rho'$ in (32)	(nx, ny, nz)
theta_pert	Potential temperature anomaly in K, $\theta'$ in (33)	(nx, ny, nz)

## 4 WRF Variables (`wrf_fields.py`)

This module contains functions which are used to add the QGPV anomaly to the zonally-symmetric initial jet, add moisture to the initial state, and compute all of the variables required to run WRF. Several functions within this module are very similar to those in the `module_initialize_b_wave.F` script that is called by `ideal.exe` when running the built-in WRF test case. For that reason, if the user finds the description of this module unclear, it is recommended that the user seek out official WRF documentation.

### 4.1 Functions and Formulas

WRF grid parameters are computed using the function `wrf_grid`. This includes various grid spacings and extrapolation constants.



The interpolation function used in this module is `interp_0`, which uses the same linear interpolation formula given by (4), but in a different form to incorporate both increasing and decreasing grid values.

The function `add_pert` extends the two-dimensional initial jet variables into three dimensions, interpolates the QGPV perturbation variables onto the WRF grid, and adds the perturbed fields to the initial jet fields. This results in complete three-dimensional fields of  $u$ ,  $v$ ,  $\rho$ ,  $\theta$ , and  $z$ .

Moisture is added to the initial state by computing soundings at each horizontal grid point using the function `calc_sounding`. The pressure and temperature soundings are computed as follows:

$$p = p_0 \left( \frac{R_d \theta \rho}{p_0} \right)^{c_p/c_v} \quad (35)$$

$$T = \theta \left( \frac{p}{p_0} \right)^{R_d/c_p} \quad (36)$$

The water vapor mixing ratio ( $q_v$ ) profile is then calculated using Bolton's formula for the saturation vapor pressure of water ( $e_s$ ), and then the potential temperature profile is updated, assuming that the input potential temperature is equal to the equivalent potential temperature ( $\theta_e$ ):

$$e_s = \begin{cases} 611.2 \exp \left( 17.67 \left( \frac{T-273.15}{T-29.65} \right) \right) & \text{if } T > 273.15 \text{ K} \\ 611.2 \exp \left( 21.87 \left( \frac{T-273.15}{T-7.66} \right) \right) & \text{if } T \leq 273.15 \text{ K} \end{cases} \quad (37)$$

$$q_v = RH \frac{R_d}{R_v} \left( \frac{e_s}{(p - e_s)} \right) \quad (38)$$

$$\theta = \theta_e / (1 + 0.61 q_v) \quad (39)$$

where  $R_v = 461.6 \text{ J K}^{-1} \text{ kg}^{-1}$  is the ideal gas constant of water vapor and  $RH$  is a prescribed approximation of the meridionally-averaged relative humidity. Then, the density, moist pressure ( $p_m$ ), and dry pressure ( $p_d$ ) profiles are updated using hydrostatic balance (note that to establish moist hydrostatic balance, the calculations of the density and moist pressure profiles are iterated 10 times per vertical level):

$$\frac{dp_d}{dz} = -\rho g \quad (40)$$

$$\frac{dp_m}{dz} = -\rho g (1 + q_v) \quad (41)$$

$$\rho = \frac{p_0}{R_d \theta (1 + \frac{R_v}{R_d} q_v)} \left( \frac{p_m}{p_0} \right)^{c_v/c_p} \quad (42)$$

The function `middle_sounding` runs `calc_sounding` in the center of the horizontal domain and computes the following base-state variable values:

$$\mu_{base} = p_{bot} - p_{top} \quad (43)$$

$$p_{base} = \eta(p_{bot} - p_{top}) + p_{top} \quad (44)$$

$$\theta_{init} = \theta - 300 \quad (45)$$

$$\alpha_{base} = \frac{R_d \theta}{p_0} \left( \frac{p_0}{p_{base}} \right)^{c_v/c_p} \quad (46)$$

$$\frac{d\phi_{base}}{d\eta} = -\mu_{base} \alpha_{base} \quad (47)$$

where  $\mu$  is the dry air mass per unit area in a column of the atmosphere (note that  $\eta = (p - p_{top})/\mu = (p - p_{top})/(p_{bot} - p_{top})$ ), and  $\alpha$  is the inverse density.

The function `full_domain` runs `calc_sounding` at each horizontal grid point and computes the perturbation fields:

$$\mu_{pert} = (1 + q_{v,top})(p_{bot} - p_{top}) - \mu_{base} \quad (48)$$

$$\frac{dp_{pert}}{d\eta} = (1 + q_v)\mu_{pert} + q_v\mu_{base} \quad (49)$$

$$\alpha = \frac{R_d \theta}{p_0} \left( 1 + \frac{R_v}{R_d} q_v \right) \left( \frac{p_0}{p_{base} + p_{pert}} \right)^{c_v/c_p} \implies \alpha_{pert} = \alpha - \alpha_{base} \quad (50)$$

$$\frac{d\phi_{pert}}{d\eta} = -\alpha_{pert}(\mu_{base} + \mu_{pert}) - \alpha_{base}\mu_{pert} \quad (51)$$

Finally, `full_sounding` interpolates  $u$  and  $v$  onto the moist grid, imposes zonal periodicity of these variables, computes the values of potential temperature, mixing ratio, and zonal and meridional velocity at  $(x, y) = (0, 0)$  and writes them to base-state variables, and computes the surface temperature and writes it to a variable.

## 4.2 Output Variables

Variable	Description	Dimensions
u_field	Total zonal wind in $\text{m s}^{-1}$	(nx, ny, nz)
v_field	Total meridional wind in $\text{m s}^{-1}$	(nx, ny, nz)
t	Perturbation potential temperature in K	(nx, ny, nz)
ph	Perturbation geopotential in $\text{m}^2 \text{s}^{-2}$	(nx, ny, nz)
mu	Perturbation dry air mass in column in Pa	(nx, ny)
p	Perturbation pressure in Pa	(nx, ny, nz)
moist	Water vapor mixing ratio in $\text{kg kg}^{-1}$	(nx, ny, nz)
t_base	Base state potential temperature in K	(nz)
qv_base	Base state water vapor mixing ratio in $\text{kg kg}^{-1}$	(nz)
u_base	Base state zonal wind in $\text{m s}^{-1}$	(nz)
v_base	Base state meridional wind in $\text{m s}^{-1}$	(nz)
tsk	Surface skin temperatue in K	(nx, ny)
tmn	Soil temperature at lower boundary in K	(nx, ny)
mu_base	Base state dry air mass in column in Pa	(nx, ny)
pb	Base state pressure in Pa	(nx, ny, nz)
t_init	Initial potential temperature in K	(nx, ny, nz)
phb	Base state geopotential in $\text{m}^2 \text{s}^{-2}$	(nx, ny, nz)
dnw	Staggered vertical ( $\eta$ ) grid spacing	(nz)
rdnw	Staggered inverse vertical ( $\eta$ ) grid spacing	(nz)
dn	Unstaggered vertical ( $\eta$ ) grid spacing	(nz)
rdn	Unstaggered inverse vertical ( $\eta$ ) grid spacing	(nz)
fnp	Lower weight for vertical stretching	(nz)
fnn	Upper weight for vertical stretching	(nz)
cfn	Extrapolation constant	(1)
cfn1	1 - extrapolation constant	(1)
cf1	First second-order extrapolation constant	(1)
cf2	Second second-order extrapolation constant	(1)
cf3	Third second-order extrapolation constant	(1)
rdx	Inverse zonal grid spacing	(1)
rdy	Inverse meridional grid spacing	(1)

## 5 WRF netCDF Input File (write\_wrfinputd01.py)

This module contains only one function called `write` which, given that a netCDF file has been opened for writing using the netCDF4-Python function `netCDF4.Dataset`, writes to this file all of the variables and attributes needed to run WRF. Many of the variables and attributes written here are set to 0 because they are not required for running a

simulation using this set-up. The non-zero variables that are written to this file have been described above. The attributes that the author has deemed useful for editing are described in the section below about user parameters. If the user has more questions about the variables and attributes in a WRF input file, it is recommended that the user seek out official WRF documentation.

## 6 User Parameters in the Main Script

The main script is effectively the user interface for the initialization. The first section of the script contains various parameters which the user may adjust to produce the initial state desired. These parameters are described in the tables below.

### 6.1 Grids

Parameter	Description	Example Value
nx	Number of grid points in zonal (x) direction	2000
ny	Number of grid points in meridional (y) direction	1800
nz	Number of grid points in vertical ( $\eta$ or z) direction	100
hres	Horizontal grid spacing, in kilometers	4
zl	Height at top of atmosphere, in kilometers	20
pbot	Surface pressure, in Pascals	101000
ptop	Pressure at the top of the atmosphere, in Pascals	5000

## 6.2 Zonally-Uniform Jet

Parameter	Description	Example Value
pibot	Reference surface value of $\Pi$ , in $\text{J K}^{-1} \text{kg}^{-1}$	1004
pibtop	$\Pi$ at top of atmosphere for EPV inversion	410
npi_h	Number of grid points in $\Pi$ for high-res run	180
npi_l	Number of grid points in $\Pi$ for low-res run	40
ny_l	Number of grid points in $y$ for low-res run	30
thtop	$\theta_m$ in (5), in K	520
dth	$\Delta\theta_e$ in (5), in K	20
dyth	$\Delta y_{th}$ in $\sigma$ in (5), in meters	2.5e6
pim	$\Pi_m$ in (6), in $\text{J K}^{-1} \text{kg}^{-1}$	670
dpitr	$\Delta\Pi_e$ in (6), in $\text{J K}^{-1} \text{kg}^{-1}$	40
dytr	$\Delta y_e$ in $\epsilon$ in (6), in meters	1.5e6
pvs	$PV_S$ in (7), in $\text{m}^2 \text{K s}^{-1} \text{kg}^{-1}$	1.8e-6
pvt	$PV_T$ in (7), in $\text{m}^2 \text{K s}^{-1} \text{kg}^{-1}$	0.22e-6
dpipv	$\Delta\Pi_{TP}$ (7), in $\text{J K}^{-1} \text{kg}^{-1}$	15
nit	Number of iterations in SOR method	1000
om	SOR coefficient	1.8

## 6.3 QGPV Perturbation

Parameter	Description	Example Value
qgpv_mag	$Q_0$ in (22), in $\text{s}^{-1}$	3.0e-5
theta_mag	Perturbation $\theta$ at boundary of domain (Neumann BC)	0
x_pert	x grid point of center of anomaly	500
y_pert	y grid point of center of anomaly	750
z_pert	z grid point of center of anomaly	32
az	Vertical decay scale of anomaly, in kilometers	3
axy	Horizontal decay scale of anomaly, in kilometers	1000

## 6.4 WRF Fields

Most of the parameters listed below appear in WRF's namelist.input file. For details on these WRF parameters, it is recommended that the user seek out official WRF documentation.

Parameter	Description	Example Value
avg_rh	RH in (38), unitless	0.7
title_str	Title of netCDF file	WRF INPUT
time_str	Time of initial state	2001-06-11_00:00:00
dt	Time step of integration, in seconds	20
mp_physics	WRF microphysics option	17
ra_lw_physics	WRF longwave radiation option	0
ra_sw_physics	WRF shortwave radiation option	0
sf_sfclay_physics	WRF surface layer option	1
sf_surface_physics	WRF land surface option	0
bl_pbl_physics	WRF boundary layer option	1
cu_physics	WRF cumulus option	0
num_soil_layers	WRF soil layers option	5
diff_opt	WRF turbulence and mixing option	1
km_opt	WRF eddy coefficient option	3
damp_opt	WRF upper-level damping option	3
dampcoef	WRF damping coefficient	0.4
khdif	WRF horizontal diffusion constant	0
kvdif	WRF vertical diffusion constant	0
spec_bdy_width	WRF rows for boundary value nudging	8
sf_lake_physics	WRF lake physics option	0
surface_input_source	WRF surface input source	1
hypsoetric_opt	WRF hypsometric option	2
num_land_cat	WRF number of land categories	24
num_soil_cat	WRF number of soil categories	16

## 7 Example Initial State

Running the main script using the example parameter values indicated in section 6 creates a WRF input file that leads to a simulation of a highly realistic idealized mid-latitude cyclone.

The zonal wind and water vapor mixing ratio fields produced by this initialization are shown in Figure 1. The jet maximum is located at about 12 km, with a speed of about  $35 \text{ m s}^{-1}$ . The water vapor mixing ratio reaches a maximum of about  $12 \text{ g kg}^{-1}$  near the surface in the southernmost part of the domain; this corresponds to a relative humidity of about 80%. The mixing ratio decreases with both height and amplitude from this point. The initial potential temperature field is shown in Figure 2. The surface temperature ranges from about 290 K in the southernmost part of the domain to about 260 K in the northernmost. The tropopause is located near 12 km in the southernmost part of the domain and near 10 km in the northernmost.

The state of the cyclone at 4.5 days into the simulation is shown in Figures 3 and 4. By this time, the cyclone is well-developed, with a minimum surface pressure of about 960 hPa. There are also clear warm, cold, and occluded fronts present at the surface which match the typical structure of a well-developed mid-latitude cyclone. In addition, the surface temperature field shows evidence of downstream development beginning to take place. The cloud field is also highly realistic, as there is a thin region of clouds along the cold front, and a large swath of high clouds rising above the warm front and wrapping around the low pressure center.

For more information about the realism of this initialization, please contact the author.

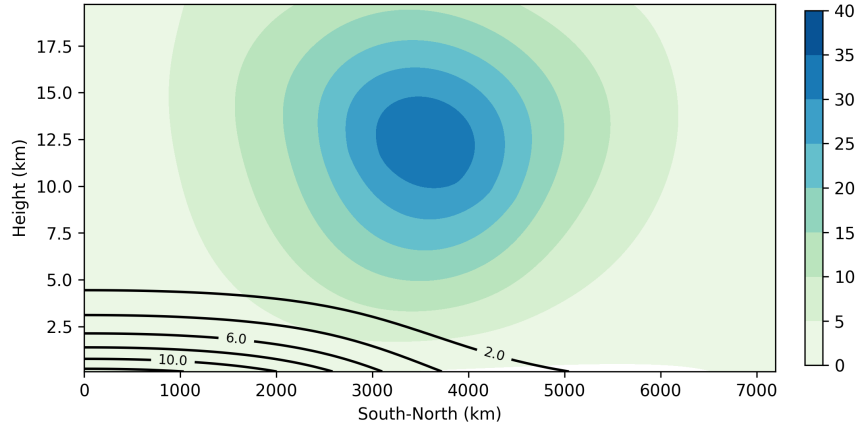


Figure 1: Cross section of the initial zonal wind velocity  $u$  (colors at  $5 \text{ m s}^{-1}$  intervals) and water vapor mixing ratio (black contours at  $2 \text{ g kg}^{-1}$  intervals).

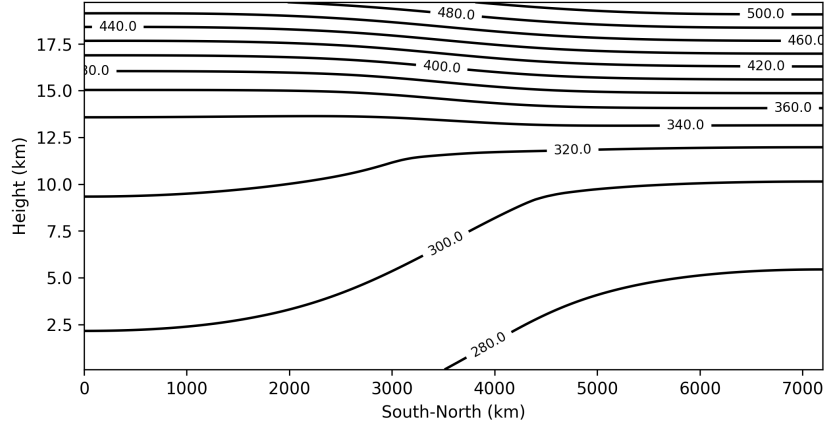


Figure 2: Cross section of the initial potential temperature (black contours at 20 K intervals).

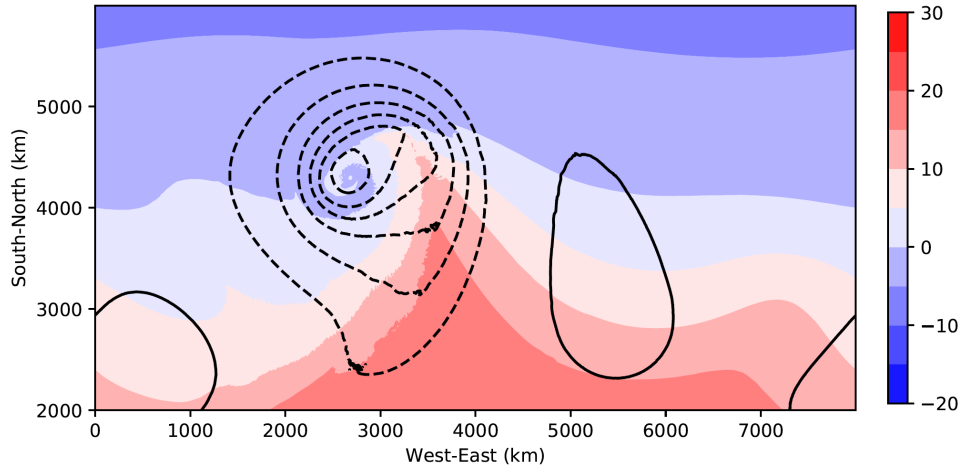


Figure 3: Surface temperature (colors at 5 K intervals) and pressure (black contours at 8 hPa intervals) at day 4.5 of the simulation. Dashed contours correspond to pressures 1000 hPa or less, while lined contours correspond to pressures greater than 1000 hPa.

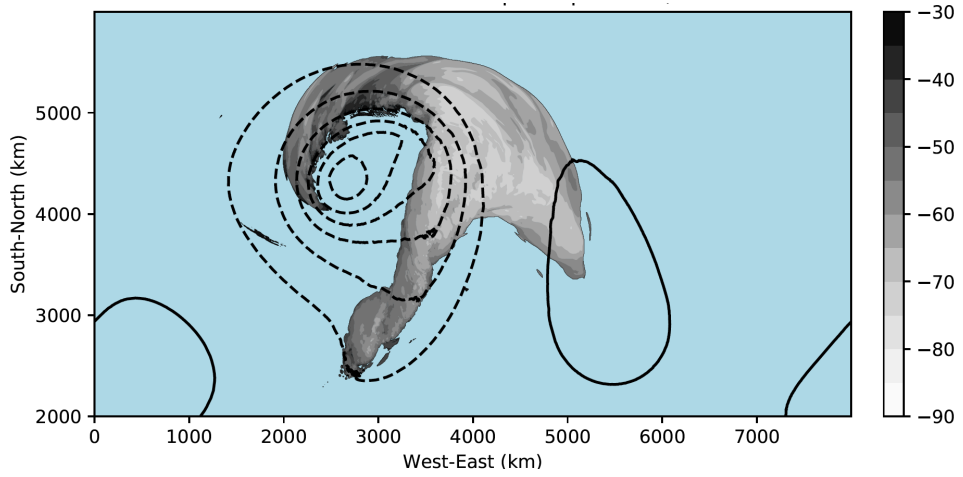


Figure 4: Surface pressure (black contours at 8 hPa intervals, as in Figure 3) and cloud top temperature (grey-scale colors contoured every 10 °C) at day 4.5 of the simulation.