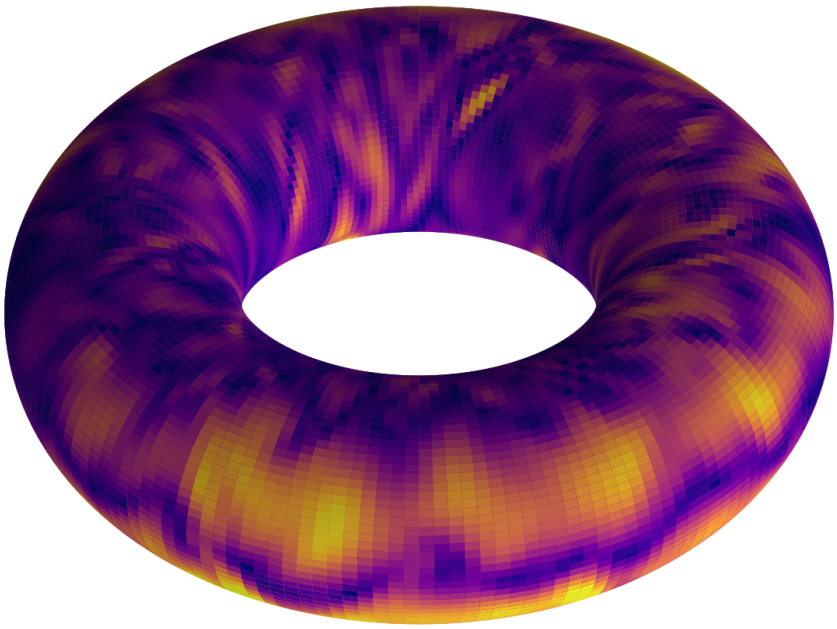
# Shallow water equations on a toroidal planet in the domain-specific language GT4Py

Killian P. Brennan, Dana Grund, Joren Janzing, and Franco Lee

Course Project: High-Performance Computing for Weather and Climate

August 2023



# Contents

# 1 Introduction

This project analyses the performance of a `GT4Py`-based implementation of the shallow water equations (SWE) on a toroidal planet. After moving a given SWE model to the torus, the runtime for different `GT4Py` backends is compared to the provided implementation in `numpy`. It is found that all `GT4Py` backends (except the `numpy` backends) result in a considerable acceleration of the simulation. In particular, the use of GPUs speeds it up by a factor of about 8.

## 1.1 Atmospheric dynamics on toroidal planets

Over the past few years, the existence of numerous exoplanets has been verified (i.e. Jenkins et al., 2015). Among these are a significant number of rocky exoplanets along with their associated atmospheres (Seager and Deming, 2010). These planets display a wide array of physical and dynamic conditions, influenced by their diverse orbital characteristics such as period, eccentricity, and obliquity, as well as their masses and the amount of stellar energy they receive (Showman et al., 2013, and references therein).

While most studies around planetary atmosphere aim to discuss the atmospheric dynamics on spherical exoplanets, spheres or spheroids are not the only theoretically allowed planet shape. A toroidal planet is a theoretical category of terrestrial exoplanet characterized by its torus structure. While a concrete theoretical explanation for the natural formation of toroidal planets remains elusive, the configuration itself holds the potential for quasi-stability (i.e., Dyson and Thomson, 1997; Ansorg et al., 2003). Naturally, the question of atmospheric dynamics on a torus emerges. Few studies have investigated the fluid dynamics on a torus (Irigi, 2010; Anders, 2014; Kumar, 2022), and we aim to add insights to this emerging field by implementing the shallow water equations (SWE) on a torus.

## 1.2 Shallow water equations on the sphere

First, we will discuss how the shallow water equations work on a sphere. Let the longitudinal coordinate be $\phi$, the latitudinal one $\theta$, and horizontal velocity $\mathbf{v} = (u, v)$. Following the notation in Williamson, 1992, the horizontal gradient of a scalar $s$ in Cartesian coordinates $x$ (longitude) and $y$ (latitude) is given by

$$\begin{aligned} \partial_x s &= \tfrac{1}{a\cos\theta}\tfrac{\partial s}{\partial \lambda}, \\ \partial_y s &= \tfrac{1}{a}\tfrac{\partial s}{\partial \theta}, \end{aligned} \tag{1}$$

with the Earth radius $a$. The shallow water equations (SWEs) in flux form can then be expressed as

$$\begin{aligned} \tfrac{\partial h\mathbf{v}}{\partial t} + \nabla\cdot(\mathbf{v}h\mathbf{v}) &= -f\mathbf{k}\times h\mathbf{v} - gh\nabla h, \\ \tfrac{\partial h}{\partial t} + \nabla\cdot(h\mathbf{v}) &= 0, \end{aligned} \tag{2}$$

where the model variables are horizontal height flux $h\mathbf{v} = (hu, hv)$ and fluid height $h$, the vertical unit vector is $\mathbf{k}$, acceleration due to gravity $g$, and the Coriolis parameter $f = 2\Omega\sin\theta$, with the earth rotation rate $\Omega$. Note that we assume a flat domain and therefore do not distinguish between the fluid height above terrain and above the sphere.

The implementation of the SWE at hand is based on Cartesian coordinates `x` and `y`. The sphere is discretized uniformly in spherical coordinates, excluding areas of up to 5 degrees north and south to avoid pole problems. Given inputs `M` and `N`, the physical domain has shape `domain=(M, N-2)`. To account for periodic boundary conditions in the longitudinal direction, two halo points and one overlapping point are added. In the latitudinal direction, an outflow condition is employed by copying the penultimate latitudes to the last ones. This leads to a full computational domain of shape `domain_halo=(M+3,N)`.

## 1.3 Discretization with the Lax-Wendroff scheme in `numpy`

The provided code we build upon employs a Lax-Wendroff finite difference scheme of Richtmyer type to discretize the SWE. This scheme is of second order and does not require the computation of the full Jacobian (LeVeque, 1992). We take as an example the following one-dimensional partial differential equation:

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0. \tag{3}$$

The scheme is then decomposed into two steps as follows:

1. **Staggered step** $t^{n+1/2}$**.** $U_{j+1/2}^{n+1/2} = \frac{1}{2}(U_j^n + U_{j+1}^n) - \frac{\Delta t}{2\Delta x}\left(f(U_{j+1^n}) - f(U_j^n)\right)$,
2. **Unstaggered step** $t^{n+1}$**.** $U_J^{n+1} = U_j^n - \frac{\Delta t}{\Delta x}\left(f(U_{j+1/2}^{n+1/2}) - f(U_{j-1/2}^{n+1/2})\right)$,

Here, $U_j^n$ is the discretization of $u$ at location $x_j$ and time $t^n$. It extends naturally to two spatial dimensions, where the staggered step is repeated in both spatial dimensions for each model variable. For example, the x-staggered fluid height `hMidx` at time $t^{n+1/2}$ is computed as[1]

```
hMidx = 0.5 * (h[1:,1:-1] + h[:-1,1:-1]) - \
        0.5 * self.dt / self.dx[:,1:-1]  * \
              (hu[1:,1:-1] - hu[:-1,1:-1])
```

Computations of similar nature are conducted for other variables such as `hMidy`, `huMidx`, and so on. Note that the latitudinal halo `[:,1:-1]` is excluded from the computation and updated separately, together with the longitudinal one, after each time step. The final update then takes the following form:

```
hnew = h[1:-1,1:-1] - \
       self.dt / self.dxc  * (huMidx[1:,:]  - huMidx[:-1,:]) - \
       self.dt / self.dy1c * (hvMidy[:,1:]  * cMidy[:,1:]    - \
                              hvMidy[:,:-1] * cMidy[:,:-1])
```

A correction scaling is applied to latitudinal fluxes to account for the anisotropic shape of the grid cells in this direction:

```
y1    = a * sin(theta)
dy1   = y1[:,1:] - y1[:,:-1]
dy1c  = 0.5 * (dy1[1:-1,:-1] + dy1[1:-1,1:])
cMidy = cos(0.5 * (theta[1:-1,1:] + theta[1:-1,:-1]))
```

[1] When displaying code snippets within the report, we will focus on the relevant parts and leave out details such as `self` inside the `Solver` class or repetitive computations within one function or stencil.

An additional correction is employed to ensure no flux on the staggered grid where the fluid height is zero:

```
VxMid = np.where(hMidx > 0.0, \
                 hvMidx * huMidx / hMidx, 0.0)
```

After computing one Richtmyer step in flux form `h, hu, hv`, the variables are converted back to `h, u, v` by division.

There is the option to add diffusion with fluid viscosity $\nu$ to these latter variables. They are extended by an extra halo in both dimensions and centered differences are employed twice. As the Laplacian was discussed in detail during the course, we will not focus on these computations in this report.

## 1.4 Test cases

The initial conditions IC0 and IC1 provided are taken from the set of test cases for spherical SWE by Williamson, 1992 (sixth and second), where details on the formulations can be found. For lack of better options and previously published test cases, the spherical test cases were directly adapted to the torus, only changing the initial fluid height to $h_0 = 200\,\text{km}$ and adding the uniform cases IC2 and IC3. Kumar, 2022 introduces a chess board-type test case for a toroidal planet, however their setup does not include toroidal gravity and the test case is poorly documented and as such could not be implemented in our model.

**IC0: Rossby-Haurwitz Wave.** Rossby-Haurwitz Waves travel with wave number $R = 4$ travel eastwards in a non-divergent flow. Their spatial pattern stays constant on a sphere (see Fig. 1).

**IC1: Global Steady State Nonlinear Zonal Geostrophic Flow.** A steady zonal flow is simulated sustaining a steady, non-flat distribution of the water height (see Fig. 2). By varying the Coriolis parameter $f$ in both longitude and latitude, this test case accounts for the earth rotation axis not being aligned with the poles.

**IC3: Global zonal uniform flow with added noise.** The model is initialized with globally uniform zonal flow ($u_0 = 10\,\text{ms}^{-1}$) with added noise ($h_{noise} = 1000\,\text{m}$, $u, v_{noise} = 0.1\,\text{ms}^{-1}$). IC2 is equal to IC3 without the noise, but not discussed here.

## 1.5 Research questions

To investigate the behaviour of a fluid on a rotating toroidal planet, and the potential to accelerate its simulation with the domain specific language (DSL) `GT4Py`, we formulate the following research questions that we aim to answer in the scope of this project:

1. What are main characteristics of shallow water flow on a toroidal planet?
   - What is the equilibrium water surface shape of the fluid surface on a torus?
   - How do different initial conditions evolve on a toroidal planet over the span of 100 days?
   - What are features of the synoptic circulation patterns on a torus?
2. How can the Lax-Wendroff scheme of Richtmyer type be ported to `GT4Py`, and does it yield a speed-up without major performance optimization?

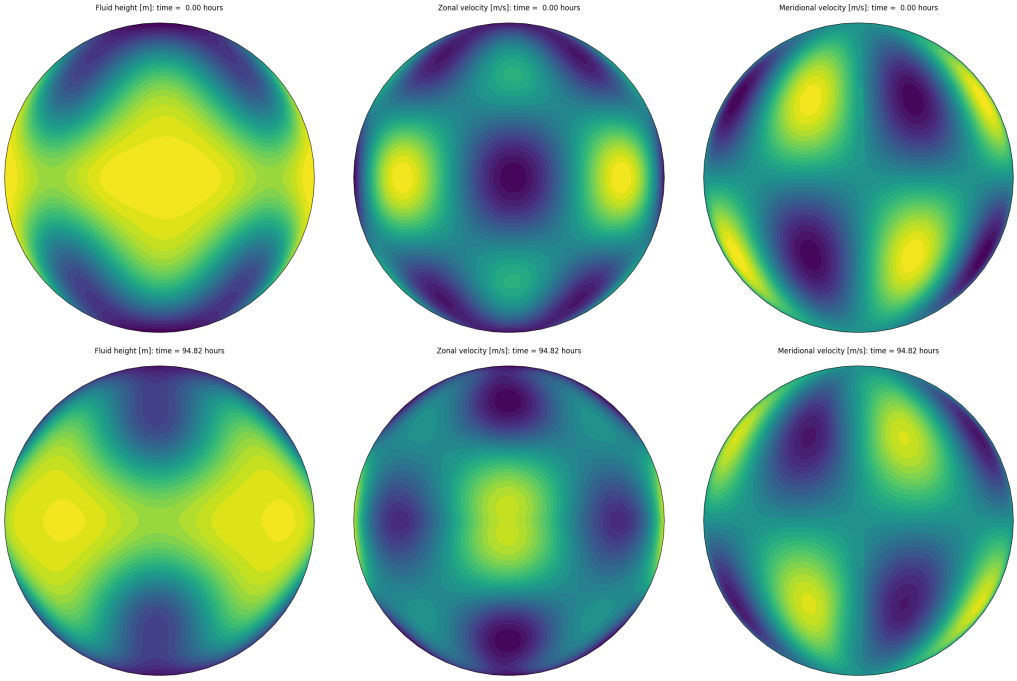



Figure 1: IC0: Rossby-Haurwitz Wave with wave number $R = 4$. Initial condition (top) and solution after $T = 4$ days (bottom) for the fluid height $h$, zonal velocity $u$ and meridional velocity $v$ (left to right).

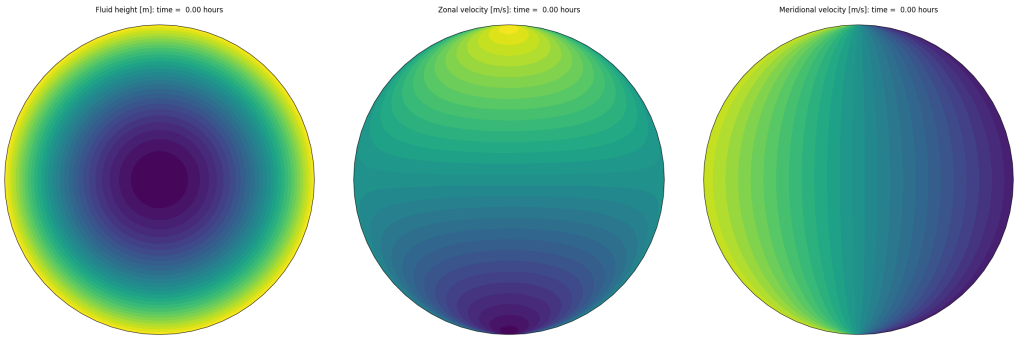



Figure 2: IC1: Global Steady State Nonlinear Zonal Geostrophic Flow. Steady-state initial condition for the fluid height $h$, zonal velocity $u$ and meridional velocity $v$ (left to right).

# 2 Methods

This section presents the approach employed to port the SWE model to `GT4Py`, as well as the physical foundations to move it to a toroidal planet.

## 2.1 Porting the model to GT4Py

The given `numpy` implementation of the SWE was ported to `gt4py` for acceleration and use on GPU. As pointed out in section 1.3, the Lax-Wendroff update is split into a staggered first step in x and y, and an unstaggered second step. As these three computations are conducted using different stencils, they are implemented in separate parallel `gt4py` computations called `x_staggered_first_step`, `y_staggered_first_step`, and `combined_last_step`. In addition, the fluxes are computed as products of full fields in `compute_temp_variables` at the beginning of each time step. The respective stencil configurations are

```
shape              = (   nx,    ny,  nz) # full domain size w/ halo
shape_default      = (nx-2, ny-2,  nz) # physical domain, w/o halo
shape_staggered_x  = (nx-1,    ny,  nz) # full but staggered in x
shape_staggered_y  = (   nx, ny-1,  nz) # full but staggered in y

origin_default     =  (0,0,0)
origin_staggered   = (0,0,0)
```

The origin is always chosen to be zero, as all stencil computations are one-sided only and the halos are included in the staggered computations for simplicity. A non-zero origin could be used to compute two-sided stencils such as the central difference in the diffusion part of the code, as displayed schematically in Fig. 3. All input and output fields, as well as all fields passed between the stencil operations, are moved to the `gt4py` backend before the computations, adding a dummy dimension z of size `nz=1` to fit the three-dimensional `gt4py` format. The output fields are then brought back to `numpy` format before returning. In the following, the `gt4py` pipeline is presented for the examples from section 1.3.

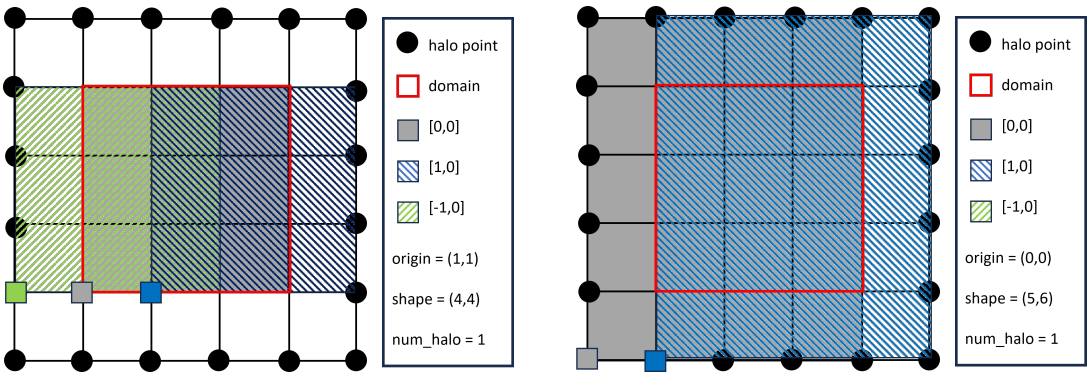



Figure 3: Schematic stencil computations with `gt4py` in x directions. Left: central difference with one halo point. Right: Computation of an x-staggered variable at points $x_{i+1/2} = \frac{1}{2}(x_i + x_{i+1})$ in the Lax-Wendroff scheme.

### 2.1.1 First step (staggered)

Outside the `Solver` class, the x-staggered stencil is defined with the help of `gt4py` functions for readability. The code lines relevant to compute the x-staggered fluid height `hMidx` are[2]

```
@gtscript.function
def compute_hMidx(dx, dt, h, hu):
    return 0.5 * (h[1, 0,0] + h[0,0,0]) - \
           0.5 * dt / dx[0,0,0] * (hu[1,0,0] - hu[0,0,0])

def x_staggered_first_step(
    h:      gtscript.Field[float],
    hu:     gtscript.Field[float],
    dx:     gtscript.Field[float],
    hMidx: gtscript.Field[float],
    *,
    dt: float):
    from __gtscript__ import PARALLEL, computation, interval
    from __externals__ import compute_hMidx

    with computation(PARALLEL), interval(...):
        hMidx = compute_hMidx(dx=dx, dt=dt, h=h, hu=hu)
```

Within the initialization of `Solver`, the stencils are compiled passing the external functions as

```
x_staggered = gtscript.stencil(
    definition=x_staggered_first_step,
    backend=backend,
    externals={
        "compute_hMidx":compute_hMidx,
    },
    rebuild=False,
    **kwargs
)
```

The stencils are then evaluated within the time stepping loop as

```
x_staggered(
    h=h, hu=hu, hMidx=hMidx,
    dx=dx, dt=dt,
    origin=origin_default,
    domain=shape_staggered_x
)
```

The model variables `h` and `hu` are full fields of shape `(nx,ny,nz)`, while the x-staggered output variable `hMidx` and the space increments `dx` are of shape `(nx-1,ny,nz)`. Note that all variables passed between the stencils have to be initialized, as the program moves back to pure CPU-based python in between the stencil computations in case a GPU-based `GT4Py` backend is used.

[2] Again, we leave out computations concerning other variables and the `self` attribute for readability within the report. For the full computations, please refer to the code itself.

### 2.1.2 Second step (unstaggered)

In the same fashion as for the first step, the stencil `combined_last_step` is set up using the function `compute_hnew`. It takes as input staggered fields such as `huMidx`, `hvMidx`, and `dxc`, outputs full fields such as `hnew`, and has the form

```
def combined_last_step(
    h:       gtscript.Field[float],
    huMidx:  gtscript.Field[float],
    hvMidy:  gtscript.Field[float],
    cMidy:   gtscript.Field[float],
    dxc:     gtscript.Field[float],
    dy1c:    gtscript.Field[float],
    hnew:    gtscript.Field[float],
    *,
    dt: float,
):
    from __gtscript__ import PARALLEL, computation, interval
    from __externals__ import compute_hnew, compute_hunew, compute_hvnew

    with computation(PARALLEL), interval(...):

        hnew=compute_hnew(h, dt, dxc, huMidx, dy1c, hvMidy, cMidy)
```

Within the update of the moments in `combined_last_step`, the correction using `np.where()` is computed using element-wise basic logic operations as

```
VxMid = ( \
    hvMidx[0,0,0] * huMidx[0,0,0] / hMidx[0,0,0] \
    ) if (hMidx[0,0,0] > 0.0) else 0.0
```

The boundary conditions are applied in `numpy` as in the original version.

## 2.2 Toroidal planetary physics

This section covers the derivation of the physical processes governing fluid dynamics on a toroidal planet.

### 2.2.1 Shallow water equations on a torus

A torus can be describe and defined by the major radius $R$ and minor radius $r$ (Fig. 5). The ratio of the minor radius to the major radius ($\frac{r}{R}$) is referred to as the aspect ratio $\alpha$. On the surface of the torus, one can define a toroidal coordinate system (Fig. 5) using angles $\phi$ and $\theta$ together with the radial distance from the centre of the toroidal tube $r'$.

Although a toroidal planet is theoretically possible, only some of the combinations of the mass distribution and angular momentum are stable. The gravitational field and centrifugal acceleration together determine the equilibrium shape of the planet. The resulting cross-sections of the toroidal tube from these equilibria are not necessarily circles and the exact shapes are not trivial. This complicates the problem and the cross-section was approximated to be circular for simplicity in this project. This approximation would work if the equilibrium shape is not highly deformed,

which in our case needs some testing to figure out a good set of parameters. Also, the effective gravitation force (accounting for centrifugal acceleration) is not normal to the surface under such approximation since the surface of the torus is no longer an isosurface for the potential. Hence, there is a tangential component of the gravitation force at the surface (in $\theta$ direction).

The shallow water equations (SWEs) on a toroidal planet therefore have to be adapted and in toroidal coordinates they can be expressed as

$$\begin{array}{lllllllll}
\frac{\partial hu}{\partial t} & + & \frac{1}{r}\frac{\partial huv}{\partial \theta} & + & \frac{1}{R+r\cos\theta}\frac{\partial (hu^2+\frac{1}{2}g_{r'}h^2)}{\partial \varphi} & - & (f+\frac{u\sin\theta}{R+r\cos\theta})hv & = & 0 \\
\frac{\partial hu}{\partial t} & + & \frac{1}{r}\frac{\partial (hv^2+\frac{1}{2}g_{r'}h^2)}{\partial \theta} & + & \frac{1}{R+r\cos\theta}\frac{\partial huv}{\partial \varphi} & + & (f+\frac{u\sin\theta}{R+r\cos\theta})hu + g_\theta h & = & 0 \\
\frac{\partial h}{\partial t} & + & \frac{1}{r}\frac{\partial hv}{\partial \theta} & + & \frac{1}{R+r\cos\theta}\frac{\partial hu}{\partial \varphi} & & & = & 0,
\end{array}$$

where h is the fluid height, (u, v) are the velocity components in $\varphi$ and $\theta$ directions, $\Omega$ is the rotation rate of the planet, $f = 2\Omega\sin\theta$ is the Coriolis parameter and $(g_{r'},\ g_\theta)$ are the $r'$ and $\theta$ component of the acceleration due to effective gravity. The coordinate systems employed and the implementation of the terms in the equations will be discussed in greater details below.

### 2.2.2 Coordinate systems and planetary parameters

Different coordinate systems were employed to implement different terms in the SWEs on a torus. Gravity, centrifugal acceleration and the Coriolis force can be solved most efficiently in cylindrical coordinates $(\tilde{r}, z, \varphi)$, which has the z-axis aligned with the rotation axis of the torus, but need to then be translated to toroidal coordinates $(r', \theta, \varphi)$ (Fig. 5). When unwrapping and flattening the toroid along the inner equator and the $\varphi = 0$ section for numerical implementation, the toroidal coordinates smoothly translate to cartesian coordinates $(z, y, x)$.

Several planetary parameters are also required to define the toroidal planet. The first set relates to the shape and mass distribution, which includes the major radius, minor radius and mass density of the planet. The total mass $m = 5.97 \times 10^{24}$ kg and density $\rho = 5.52 \times 10^3\,\mathrm{kgm}^{-3}$ are chosen to be the same as the Earth, which means the torus will have the same volume as the Earth. The major (R) and minor (r) radii can then be defined when an aspect ratio $\alpha$ is set with the following equations:

$$\begin{aligned} r &= \sqrt[3]{\tfrac{2\alpha}{3\pi}} R_{Earth} \\ R &= \tfrac{r}{\alpha}, \end{aligned} \tag{4}$$

with $R_{Earch}$ =6371.22 km being the Earth's radius. With these paramters, the gravitational field can be calculated. Currently, $r = 2222.38$ km and $R = 11\,111.91$ km with aspect ratio $\alpha = 0.2$ Another important parameter is related to the rotation rate of the planet, which would affect the centrifugal acceleration and the Coriolis effect. If the rotation rate is too fast, the centrifugal acceleration will exceed gravity and fluid on the planet will not gravitate towards the planet, which set a physical upper limit of the rotation rate. With the current geometric parameters of the toroidal planet, the rotation rate is set to $\Omega = 3.551 \times 10^{-4}\,\mathrm{rads}^{-1}$, which is around 4.87 times of that of the Earth.

### 2.2.3 Gravity

The gravitational field near a torus was approximated by aggregating the mass within the torus to a ring of point masses with diameter $R$ (Pages, 2023). By integrating over the ring of point masses for every query point, a gravitational field can be estimated numerically. From the rotational symmetry

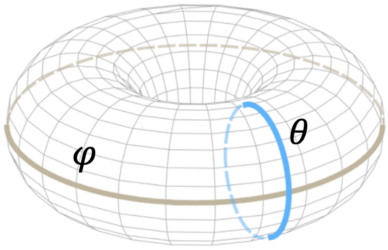



Figure 4: Gravitational components (cylindrical coordinates) near a toroid ($z = 0.2$) approximated as point masses distributed along a ring with radius $r = 1$ (arbitrary units).

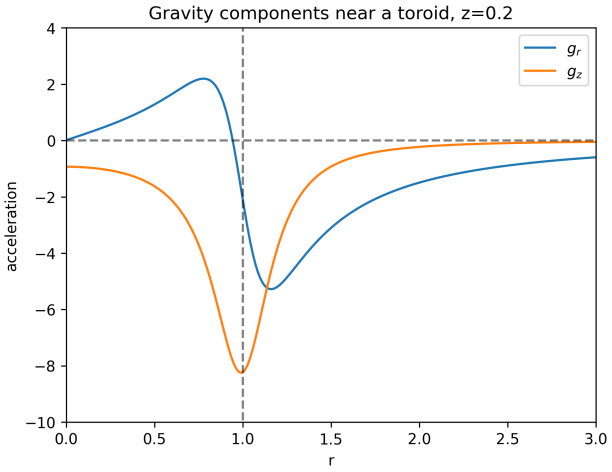



Figure 5: Toroidal coordinate system with toroidal and poloidal directions. The radial direction (not depicted) is normal to the toroid surface. Major and minor toroidal radii, $R$ and $r$, revolve around $\varphi$ and $\theta$ respectively. From Kumar, 2022.

of the torus, cylindrical coordinates offer the most straightforward mathematical solution. To start with, the toroidal planet is set to have the same mass and density (and hence volume) as the earth. For each discretized section of the toroidal tube, the total mass is approximated by a point mass placed at the centre of the cross-section (at major radius). The surface gravity is then estimated through numerical integration of the gravitational force exerted by each section. The test case from Pages, 2023 compares well to our implementation of the toroidal gravity approximation in the $r$-direction (Fig. 4). The $z$-direction is less consistent with the online resource, that resource however, has to be taken critically, because when tested, the stated equations did not yield the gravitational fields described therein.

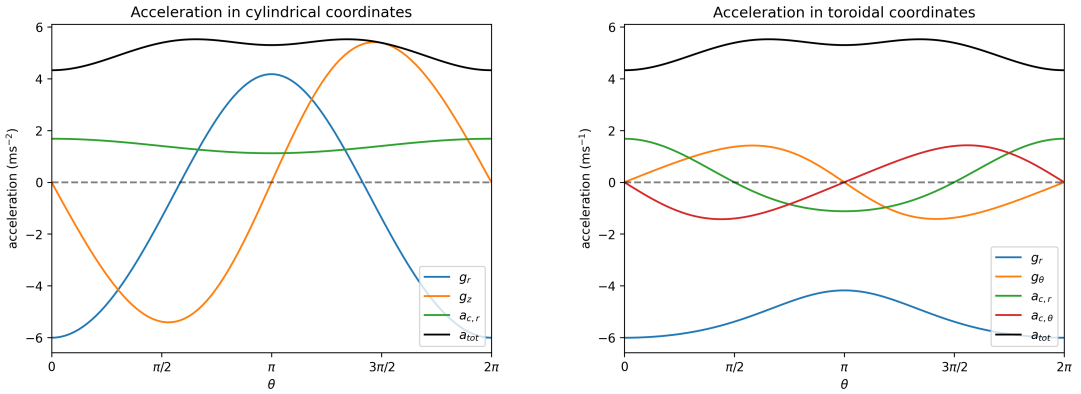
As discussed, unlike the gravitational field on a spherical planet, the gravity vector on a toroid is not normal ($g_\theta = 0$) to the surface at the equator but possesses a horizontal component ($g_\theta \neq 0$) outside the equatorial regions (Fig. 6b) due to our circular cross-section approximation. This horizontal component of the gravitational field then leads to an additional term ($g_\theta h$) in the $\theta$-component equation.

### 2.2.4 Centrifugal acceleration

The centrifugal acceleration in cylindrical coordinates only has a nonzero component in the $\tilde{r}$ direction which is given by:

$$a_{\tilde{r}} = \Omega^2 \tilde{r}, \tag{5}$$

where $\Omega$ is the rotation rate of the planet, and $\tilde{r}$ is the radius from the rotational axis. For simplicity, the centrifugal acceleration will be combined with the gravitational acceleration to form the effective gravity from here on (Fig. 6a).




(a) Cylindrical coordinates

(b) Toroidal coordinates

Figure 6: Surface acceleration on a toroid due to gravity and centrifugal forces on the torus surface along the toroidal axis $\theta$. Where $g_r$ and $g_z$ are the cylindrical components of the gravitational field, $a_{c,r}$ is the centrifugal acceleration and $a_{tot}$ is the magnitude of the combined gravitational and centrifugal field. Torus with aspect ratio $\alpha = 0.2$.

### 2.2.5 Coriolis acceleration

For the Coriolis term, the rotation vector expressed in the rotating toroidal coordinate system ($\omega$) is identical to that on the sphere (Kumar, 2022) , solely the latitude is replaced with the toroidal coordinate component $\theta$:

$$\omega = \Omega \begin{pmatrix} 0 \\ \cos\theta \\ \sin\theta \end{pmatrix}, \tag{6}$$

where $\Omega$ is the rotation rate of the torus.

The velocity of movement in this local coordinate system in toroidal coordinates is:

$$v = \begin{pmatrix} v_\varphi \\ v_\theta \\ v_r \end{pmatrix} \tag{7}$$

The resulting Coriolis acceleration ($a_C$) is given as:

$$a_C = \begin{pmatrix} a_{C\varphi} \\ a_{C\theta} \\ a_{Cr} \end{pmatrix} = -2\omega \times v = 2\Omega \begin{pmatrix} v_\theta \sin\theta - v_r \cos\theta \\ -v_\varphi \sin\theta \\ v_\varphi \cos\theta \end{pmatrix}. \tag{8}$$

Assuming a small magnitude of radial velocity and neglecting the radial component of the Coriolis acceleration, as its magnitude is dwarfed by gravity, the equations can be simplified as:

$$a_C = 2\Omega \begin{pmatrix} v_\theta \sin\theta \\ -v_\varphi \sin\theta \\ 0 \end{pmatrix}. \tag{9}$$

### 2.2.6 Toroidal geometry

Going from a sphere to a torus, the boundary condition in the $\theta$-direction is first changed to periodic and implemented with the addition of extra halo grid points. Due to the toroidal geometry, the curvature terms are also adapted for the toroidal shape by replacing the spherical radius $a$ to $R + r\cos\theta$. Moreover, as the grid cells are not retangular, with the spacing in $\varphi$ direction being larger in the outer part of the torus and smaller in the inner part. The effect of this curvilinear grid has to be accounted for in the terms containing fluxes in $\theta$ direction. The correction factor for velocity in the $\theta$ direction is $\frac{R+r\cos\theta}{R+r}$ and the corresponding distance metric is $y_1 = \frac{Rr\theta + r^2\sin\theta}{R+r}$. These corrected quantities are used for terms that contains derivatives of the fluxes in $\theta$ direction, i.e. the $\frac{1}{r}\frac{\partial huv}{\partial\theta}$ and $\frac{1}{r}\frac{\partial(hv^2+\frac{1}{2}g_{r'}h^2)}{\partial\theta}$ terms in $\varphi$- and $\theta$-component equations respectively.

# 3 Results

## 3.1 SWE on the torus

After initiating the model with "flat" water initially ($h = h_0$), water quickly flows towards the equilibrium governed by the gravitational and centrifugal forces (Fig. 7), answering the first research question. The finer scale behavior of the fluid on the torus depends on the test cases investigated. We attempt to answer the remaining two research questions in the following paragraphs.

**IC0: Rossby-Haurwitz Wave.** Unlike in the spherical setup, where it leads to spatially consistent flow, test case IC0 leads to a very different flow pattern that periodically repeats with a period in the order of 1.5 d (Fig. 9). This periodic flow pattern emerges after a chaotic spin up time of 40 d. The flow pattern exhibits a zonal ridge that propagates meridionally (in the positive $\theta$ direction, and distorts as it leaves the equatorial regions. There are persistent equatorial jets with magnitutes in excess of 600 $\mathrm{ms}^{-1}$, that flow "eastwards" (positive $\theta$ direction) at the inner equator and "westwards" (negative $\theta$ direction) at the outer equator. During periods when the ridge is in the equatorial regions, the flow is mostly zonal, whereas the flow becomes more chaotic with added meridional components as the ridge passes over the top and bottom of the torus. Further, global scale vortices can be observed as the ridge passes through the "polar" regions (Fig. 9). This behavior can be explained by the Coriolis terms, which approach zero in the equatorial regions and reach their maxima in the "polar" rings.

**IC1: Global Steady State Nonlinear Zonal Geostrophic Flow.** Test case IC1 shows a very different behavior to test case IC0. IC1 is not only stationary on a sphere but also seems to be stationary on a torus. There is a high-frequency signal that appears globally in the water height field, superimposed onto the stationary synoptic pattern. Two distinct synoptic rotations evolve around the outer-equatorial $h$-minima in opposite directions, while the flow at the inner equator is less organized (Fig. 10). Due to the aspect ratio of the torus, this vorticity is stretched in the $\varphi$-direction and appears as a jet-like feature. IC1 remains quasi-steady state up to the tested run time of 200 d.

**IC3: Global zonal uniform flow with added noise.** Test case IC3 displays behavior similar to IC0 (Fig. 11). However, the periodic flow pattern takes longer to develop than with IC0 (70 d).

Unlike on the sphere, wind velocity reaches magnitudes in excess of 400 $\mathrm{ms}^{-1}$ for test case IC1 and even up to 700 $\mathrm{ms}^{-1}$ for the other test cases on the torus (Fig. 8). The high velocities are likely due to the substantial Coriolis forces caused by the increased planetary rotation $\omega$.

The fact that we arrive at the same flow pattern with two very different initial conditions (IC0 & IC3) is promising evidence towards the kind of modes that might be expected for fluid flowing on a toroidal planet.

From the resulting equilibrium fluid height on a torus (Fig. 7), it could be argued that fluid heights approaching 300 km fall outside the validity of shallow water equations. Addressing this shortcoming falls outside the scope of this work, however, we can still formulate some recommendations for future studies.

Firstly, considering that the rocky toroidal planet would behave like a fluid during planetary formation, the torus cross-section would not be circular but rather flattened as in Fig. 7, depending on the planetary constants. As such, the model could account for this distortion by being initialized with a topography that follows the distorted cross-sectional shape. This would substantially reduce fluid heights and increase the validity of employing shallow water equations.

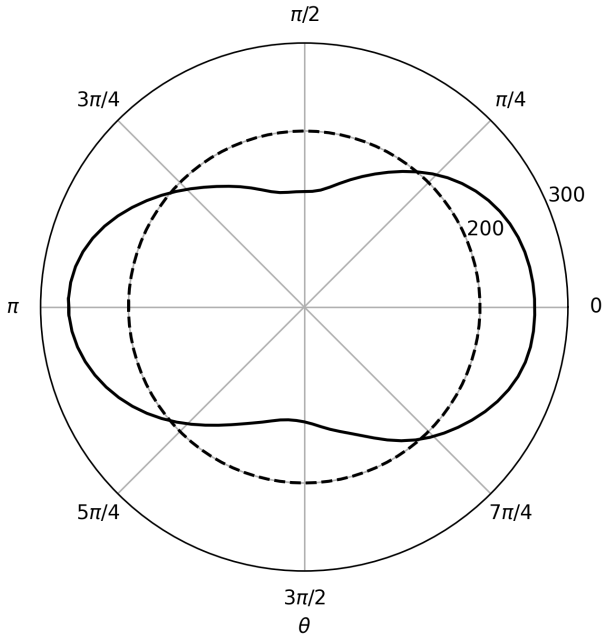



Figure 7: Polar plot of zonal mean equilibrium water height $\bar{h}$ in km. Initial water height (black, dashed line) and IC1 100 days after model initiation (black, solid line).

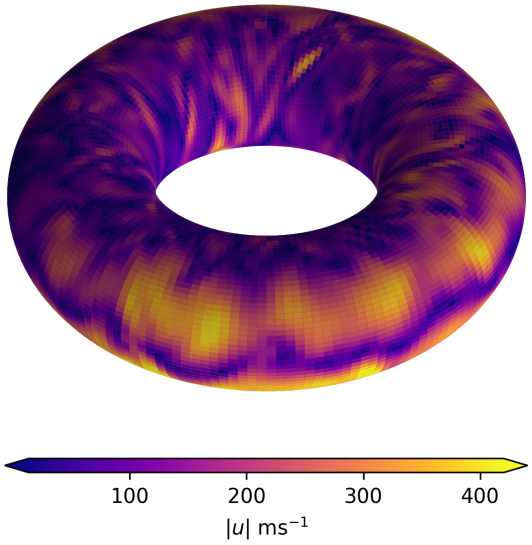



Figure 8: Velocity magnitude $|u|$ of test case IC1 at $t = 100\,\mathrm{d}$. The 2d-velocity field is projected onto a torus with aspectratio $\alpha = 0.4$ for better visibility of the features. Same figure as on the title page.

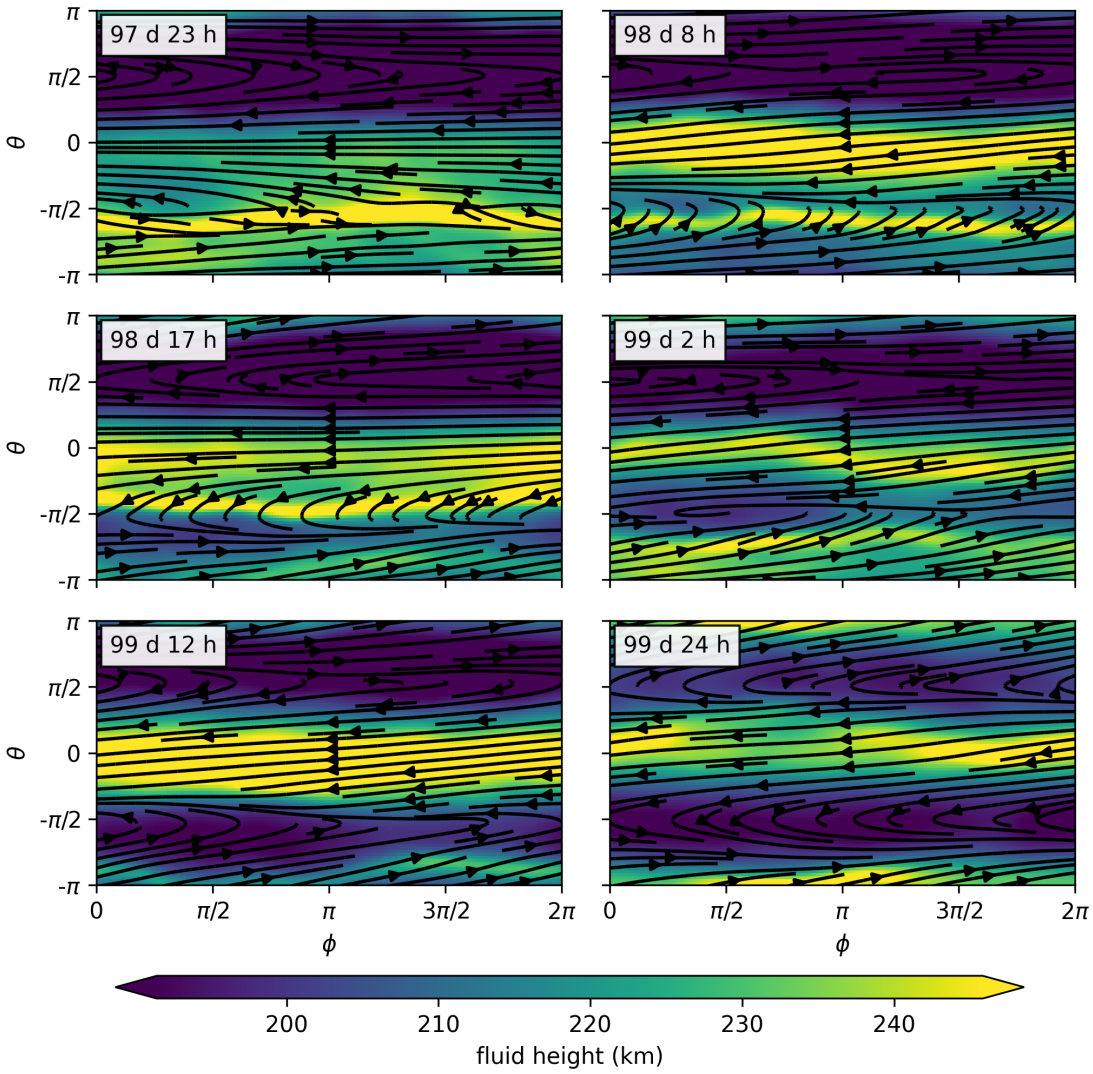



Figure 9: Evolution of the fluid height and stream lines on a torus from test case IC0, after 97 days spin-up.

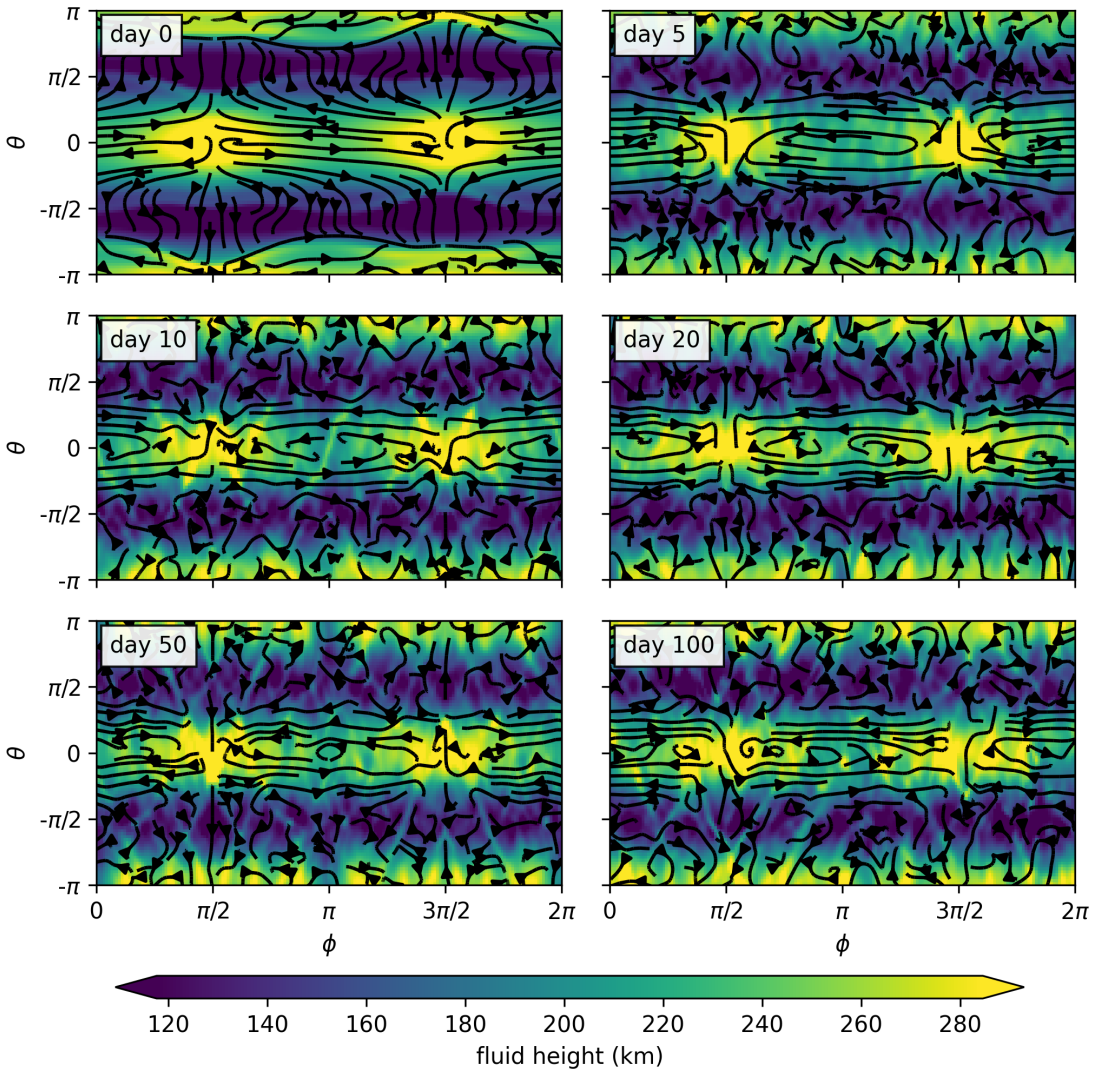



Figure 10: Evolution of the fluid height and stream lines on a torus from test case IC1, throughout a period of 100 days.

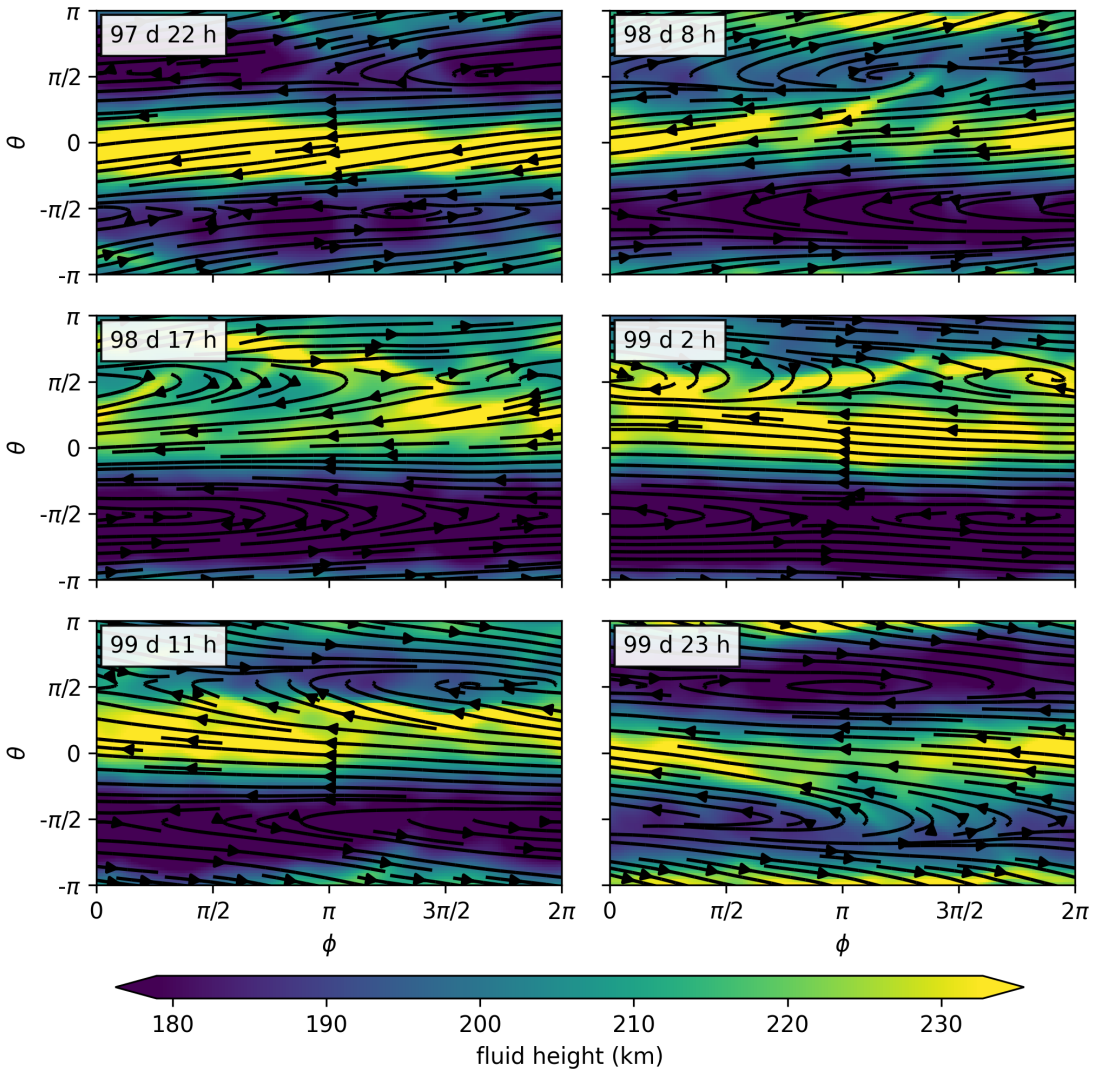



Figure 11: Evolution of the fluid height and stream lines on a torus from test case IC3, after 97 days spin-up.

Secondly, a multi-layer shallow water equation would also be a more fitting approach to modeling the vast fluid height calculations, as the individual layers would remain shallow.

Thirdly, the planetary parameters have to be carefully tuned in unison with the initial water height $h_0$. Especially with toroidal aspect ratios exceeding $\alpha > 0.3$ the parameters are very difficult to tune. It is speculated that with larger $\alpha$ the polar regions start to run dry, which is not fully implemented in the model. Dry spots also appear at the outer equator, for $\alpha > 0.1$ if the planetary rotation frequency $\omega$ is not sufficiently increased. Vice versa the inner equator runs dry if $\omega$ is too high and centrifugal forces push the water towards the outer equator.

Due to the lack of published test cases for toroidal planets, we could not validate our model against existing solutions.

## 3.2 Performance analysis on the torus

The SWEs solver for a toroidal planet is also ported to `gt4py` as described in section 2.1 with an aim to speed up the simulations. The performances of the solvers implemented using `numpy` and `gt4py` are then compared using a test case using IC1 described in section 3.1 at increasing resolutions. The number of grid points (nx×ny) increases from `180×90`, `360×90`, `360×180`, `720×180` to `720×360`. To make the performance assessment practically feasible, the length of the simulation is set to 24 hours. The results are shown in Fig. 12. Experiments for the `gt4py` solver using different backends will be referred to as `gt4py-"backend"` below. The experiments were performed on Piz-Daint twice, which showed very similar results. Therefore, only the first experiment is shown here.

First, apart from the general trend of increasing time required for increasing resolution, the rate of increase in computational time is larger when doubling the resolution in x-direction than when doubling it in y-direction (`360×90` to `360×180` and `720×180` to `720×360`) This is likely due to the anisotropy of the grid cells for the torus, with the x-spacing much larger than the y-spacing. Therefore, when the ny is increased, the y-spacing is further reduced and leads to a more stringent CFL criterion and smaller timesteps than those in the case of increasing nx.

Moreover, there are clearly two groups of curves, with `numpy` and `gt4py-numpy` having a much sharper increase in time required compared to `gt4py-gt:cpu_ifirst`, `gt4py-CUDA` and `gt4py-gt:GPU`. The longer time required for `gt4py-numpy` over `numpy` are likely the result of the overhead created when invoking `gt4py` structures and function calls. However, once the backend is switched to `C++` based `GridTools`, no matter which computer architecture is used, a huge gain in performance is observed. The time required for the simulation at `720×360` resolution is reduced by around 8 times. This demonstrates the advantage of using a domain-specific language, as the optimization problem is hidden and domain experts can focus on development and answering scientific questions with minimal performance concern. The increased performance also allows for higher resolution simulations or longer simulation periods.

To further investigate the potential performance gain of using GPUs, the `gt4py` solver using different backends are used to simulate the test case at further enhanced grid resolution (see Figure 13). The resolution of `1440×360` and `1440×720` are added to the experiment and the backend `gt4py-gt:cpu_kfirst` is also included. At higher resolutions, the differences between CPU-based and GPU-based architecture become significant. Around 15% of time is saved if the backend is switched from CPU-based to GPU-based at the highest resolution tested. This also illustrates another important advantage of using a domain specific language, which allows the same set of

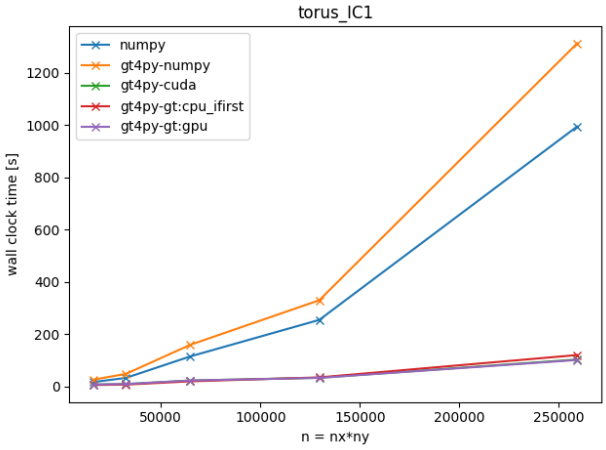



Figure 12: Required simulation time (s) for a test case using `numpy` (blue), `gt4py` with different backends, including `numpy` (orange), `CUDA` (green), `CPU i-first` (red) and `GPU` (purple) at 5 different resolutions.

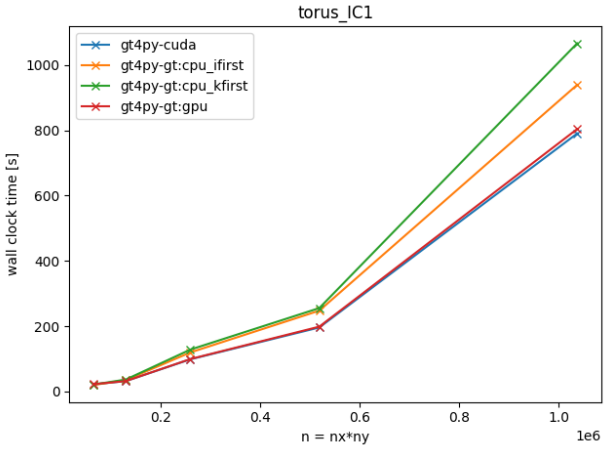



Figure 13: Required simulation time (s) for a test case using `gt4py` with different backends, including `CUDA` (blue), `CPU i-first` (orange), `CPU k-first` (green) and `GPU` (red) at 5 different increased resolutions.

codes to be employed on various systems with different computer architectures. For systems with GPUs available, the solver can make use of the more efficient calculation on GPUs and speed up the simulation without any change in the source code.

# 4 Summary

In this work, we have shown how to implement shallow water equations on a toroidal planet. We find two persistent modes of circulation on a rotating torus, namely (i) zonal jets at the equators, aligned with the planetary rotation at the outer equator and opposed at the the inner equator with global ridges "orbiting" around the minor radius, and (ii) quasi-stationary flow with wave-number 2 at the outer equator, each with counter rotating flow revolving around the two equatorial height minima.

The model was then converted to `gt4py`. Since the Lax-Wendroff scheme uses a mix of staggered and unstaggered grids, it was decided to split the model into different functions to simplify the correct indexing of variables in the functions. However, this came at the cost of transferring many variables between the different stencils, which likely reduced the performance of the model. Still, the present configuration in `gt4py` led to a clear speed-up of the computations compared to the `numpy` implementation when different backends were used. The best performance was found for model runs using the GPU compared to runs using CPU, especially at high resolutions. Further improvements could likely be made by integrating the different stencil operations into one function.

Finally, this project was also a good exercise of application-oriented mode of operation, as two of us were initially working on implementing the toroidal planetary physics, while the other two were porting the existing spherical model to GPU. After both sub-projects concluded, it was relatively easy to port the toroidal model to `GT4Py` using the spherical models as a template.

# Code avaliability

This project is made available on github[3]. The code within `sw_gt4py/` allows to run the SWE both on the torus and on the sphere, as well as with `numpy` and differente `GT4Py` backends. The folders `visualization/` and `toroidal_physics/` contain tests and plots concerning the toroidal setup.

[3] https://github.com/dana-grund/HPC4WC/tree/main/projects/2023/project06_shallow_water

# References


Anders, S. (2014). *Torus–Earth*. URL: http://www.aleph.se/andart/archives/2014/02/torusearth.html.

Ansorg, M., A. Kleinwachter, and R. Meinel (2003). Uniformly rotating axisymmetric fluid configurations bifurcating from highly flattened Maclaurin spheroids. *Monthly Notices of the Royal Astronomical Society* 339.2, 515–523. ISSN: 0035-8711, 1365-2966. DOI: 10.1046/j.1365-8711.2003.06190.x.

Dyson, F. W. and J. J. Thomson (1997). II. The potential of an anchor ring. *Philosophical Transactions of the Royal Society of London. (A.)* 184. Publisher: Royal Society, 43–95. DOI: 10.1098/rsta.1893.0002.

Irigi (2010). *Toroidal World*. URL: http://irigi.blogspot.com/2010/11/toroidal-world.html.

Jenkins, J. M., J. D. Twicken, N. M. Batalha, D. A. Caldwell, W. D. Cochran, M. Endl, D. W. Latham, G. A. Esquerdo, S. Seader, A. Bieryla, E. Petigura, D. R. Ciardi, G. W. Marcy, H. Isaacson, D. Huber, J. F. Rowe, G. Torres, S. T. Bryson, L. Buchhave, I. Ramirez, A. Wolfgang, J. Li, J. R. Campbell, P. Tenenbaum, D. Sanderfer, C. E. Henze, J. H. Catanzarite, R. L. Gilliland, and W. J. Borucki (2015). Discovery and validation of Kepler-452b: A 1.6 $R$ super Earth exoplanet in the habitable zone of a G2 star. *The Astronomical Journal* 150.2, 56. ISSN: 1538-3881. DOI: 10.1088/0004-6256/150/2/56.

Kumar, K. (2022). *Real-Time Fluid Simulation for Toroidal Planets: Visualizing the Impact of Rotational Speeds on Heat Transfer*. SSRN Scholarly Paper. Rochester, NY. DOI: 10.2139/ssrn.4224436.

LeVeque, R. J. (1992). *Numerical methods for conservation laws*. 2nd ed. Lectures in mathematics ETH Zürich. Basel ; Boston: Birkhäuser Verlag. ISBN: 978-3-7643-2723-1 978-0-8176-2723-2.

Pages, M. (2023). *Gravity of a Torus*. URL: https://www.mathpages.com/home/kmath402/kmath402.htm.

Seager, S. and D. Deming (2010). Exoplanet Atmospheres. *Annual Review of Astronomy and Astrophysics* 48.1, 631–672. DOI: 10.1146/annurev-astro-081309-130837.

Showman, A. P., R. D. Wordsworth, T. M. Merlis, and Y. Kaspi (2013). Atmospheric Circulation of Terrestrial Exoplanets. *arXiv:1306.2418 [astro-ph]*. arXiv: 1306.2418. DOI: 10.2458/azu_uapress_9780816530595-ch12.

Williamson, D. L. (1992). A standard test set for numerical approximations to the shallow water equations in spherical geometry. *Journal of Computational Physics* 102.1. Publisher: Academic Press, 211–224. ISSN: 0021-9991. DOI: 10.1016/S0021-9991(05)80016-6.