frontiers | Frontiers in Physics

TYPE Methods
PUBLISHED 20 October 2022
DOI 10.3389/fphy.2022.1005621

OPEN ACCESS

EDITED BY
Sophie A Murray,
Dublin Institute for Advanced Studies (DIAS), Ireland

REVIEWED BY
Stefano Markidis,
KTH Royal Institute of Technology, Sweden
Camilla Scolini,
University of New Hampshire, United States
Ravindra Desai,
Imperial College London, United Kingdom

*CORRESPONDENCE
Luke Barnard,
l.a.barnard@reading.ac.uk
Mathew Owens,
m.j.owens@reading.ac.uk

SPECIALTY SECTION
This article was submitted to Space Physics, a section of the journal Frontiers in Physics

RECEIVED 28 July 2022
ACCEPTED 05 October 2022
PUBLISHED 20 October 2022

CITATION
Barnard L and Owens M (2022), HUXt—An open source, computationally efficient reduced-physics solar wind model, written in Python.
Front. Phys. 10:1005621.
doi: 10.3389/fphy.2022.1005621

HUXt—An open source, computationally efficient reduced-physics solar wind model, written in Python

Luke Barnard* and Mathew Owens*

Department of Meteorology, University of Reading, Reading, United Kingdom

HUXt is an open source numerical model of the solar wind written in Python. It is based on the solution of the 1D inviscid Burger's equation. This reduced-physics approach produces solar wind flow simulations that closely emulate the flow produced by 3-D magnetohydrodynamic (MHD) solar wind models at a small fraction of the computational expense. While not intended as a replacement for 3-D MHD, the simplicity and computational efficiency of HUXt offers several key advantages that enable experiments and the use of techniques that would otherwise be cost prohibitive. For example, large ensembles of 10^2–10^5 members can easily be run with modest computing resources, which are useful for exploring and quantifying the uncertainty in space weather predictions, as well as for the application of some data assimilation methods. In this article we present the developments in the latest version of HUXt, v4.0, and discuss our plans for future developments and applications of the model. The three key developments in v4.0 are: 1) a restructuring of the models solver to enable fully time-dependent boundary conditions, such that HUXt can in principle be initialised with *in-situ* observations from any of the fleet of heliospheric monitors; 2) new functionality to trace streaklines through the HUXt flow solutions, which can be used to track features such as the Heliospheric Current Sheet; 3) introduction of a small test-suite so that we can better ensure the reliability and reproducibility of HUXt simulations for all users across future versions. Other more minor developments are discussed in the article. Future applications of HUXt are discussed, including the development of both sequential and variational data assimilation schemes for assimilation of both remote sensing and *in-situ* plasma measures. Finally, we briefly discuss the progress of transitioning HUXt into an operational model at the UK's Met Office Space Weather Operations Center as part of the UK governments SWIMMR programme.

KEYWORDS

space weather, heliophysics, simulations, Python, open source

1 Introduction

Variability in the near-Earth space environment gives rise to space weather, which can have adverse effects on space- and ground-based technologies [1]. The largest disturbances to the terrestrial magnetospheric system are the result of coronal mass ejections (CMEs) arriving in near-Earth space [2]. Thus, advanced forecasting of the arrival time and properties of CMEs at Earth is highly desirable [3]. While near-Sun CME properties—notably speed and direction – can be estimated from coronagraph observations [4], forecasting at Earth also requires accurate knowledge of the variable solar wind conditions through which CMEs propagate [5].

The dynamic interaction between a CME and the ambient solar wind is typically modelled using time-dependent magnetohydrodynamic (MHD) simulations of the solar wind (e.g., [6–9]). These models capture the large-scale MHD fluid behaviour which governs much of the physics of CME propagation and evolution to Earth, and are a vital tool that enable both heliophysics research and space weather forecasting (e.g., [3,10,11]). More recently, it has been demonstrated that similar forecasting results, at least to first order, can be obtained with greatly simplified models [12,13]. Such models are not intended to replace the more sophisticated simulations, but open up complementary capabilities *via* greatly reduced computational overhead.

This paper is intended to briefly review the reduced-physics model, HUXt (heliospheric upwind extrapolation with time-dependence), and then highlight some of the functionality of the HUXt Python implementation.

HUXt has already proved useful in a number of contexts. Barnard et al. [14] demonstrated how an ensemble of HUXt runs could be weighted by comparison with STEREO Heliospheric Imager (HI) time-elongation profiles of CME fronts to return improved ensemble hindcasts of CME arrival times. Chi et al. [15] used HUXt simulations to examine the evolving structure of two interacting CMEs, finding that the HUXt simulations were consistent with the STEREO HI observations of these CMEs.

Barnard and Owens [16] used HUXt simulations of Cone CMEs to examine the validity of the assumptions of CME geometric models and the ability of these models to reconstruct a CMEs kinematic profile from time-elongation profiles of a CMEs flank, such as those derived from HI observations. Similarly, Hinterreiter et al. [17] used HUXt simulations as part of their work to introduce time-dependent structure to their ElEvoHI geometric model. These works both highlighted the importance of including time-dependent structure and solar-wind CME interactions for increasing the real-world representivity of CME modelling.

Additionally, separate from these CME focused studies, Macneil et al. [18] demonstrated how HUXt may be used to backmap *in-situ* solar wind observation to the source regions in the low corona, which is an important technique for estimating the origins of solar wind plasma structures. Furthermore, Bunting and Morgan [19] used HUXt simulations in their calibration and long-term validation experiments into using coronal tomography to generate inner boundary conditions for solar wind numerical models.

We now proceed to review both the background and the current status of our Python implementation of HUXt. Section 2 describes the theoretical background to HUXt, whilst Section 3 discusses the numerical scheme used to solve the discretised model equations. Section 3.1 presents some new results of testing the performance of the numerical scheme. Section 5 describes some of they key functionality included in our Python implementation of HUXt. Finally, some developing applications of HUXt are described in Section 6.

2 The HUXt reduced-physics solar wind model

HUXt takes a reduced-physics approach to modelling the solar wind flow, employing approximations to greatly reduce the complexity of the MHD momentum equation. A full derivation is provided in Owens et al. [13] and so we provide only a summary here. Magnetic, gravitational, and pressure gradient forces are neglected, as in the solar wind these terms are typically small compared to the flow momentum [12]. Additionally, the solar wind is assumed to be purely radial, and so non radial terms are neglected. In this limit, the solar wind is modelled by the inviscid Burger's equation.

$$\frac{\partial V_r}{\partial t} + V_r \frac{\partial V_r}{\partial r} = 0, \tag{1}$$

Where V_r is the radial solar wind speed. Pizzo [20] and Riley and Lionello [12] outlined this simplified description of the solar wind and made the further assumption of time-stationary flows to give the Heliospheric Upwind eXtrapolation (HUX) model (see also [21]). HUXt, however, maintains explicit time dependence, allowing structures such as coronal mass ejections (CMEs) and time-dependent ambient solar wind flow to be modelled.

The solar wind continues to accelerate throughout the inner heliosphere and in HUXt this residual acceleration, $\Delta V(r)$, is represented by an empirical parameterisation, which for a uniform solar wind is represented as

$$V_r(r) = V_0 + \Delta V(r), \tag{2}$$

Where V_0 is the speed at a reference height and

$$\Delta V(r) = \alpha V_0 \left[1 - e^{\left(\frac{(r_0 - r)}{r_h}\right)}\right], \tag{3}$$

Where α is the acceleration factor, set to be 0.15, and r_h is the scale height over which it applies, set to be 50 $r_\odot$, whilst r_0 is the

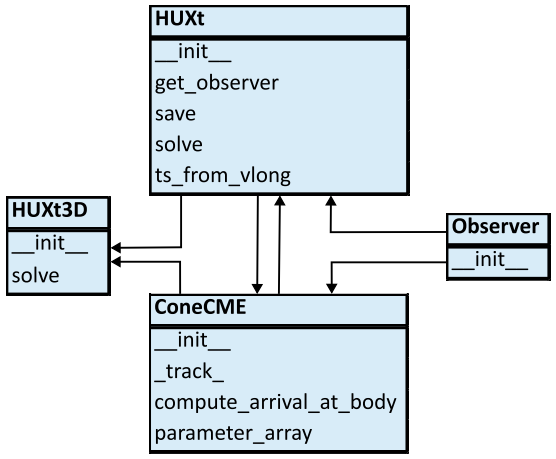

FIGURE 1
This diagram highlights the classes used within HUXt, and their associated methods, and the relationships between them.

radial distance at the reference height corresponding to V_0, taken to be 30 $r_\odot$. This form is designed to mimic that arising from the energy equations in MHD solar wind models [12].

Owens et al. [13] analysed the performance of HUXt relative to the HelioMAS MHD model of the solar wind, comparing each models representation of phenomena such as CIRs and Cone CMEs. Figure 1 of Owens et al. [13] shows that HUXt and HelioMAS both return very similar CIR structures. However, there are small but systematic discrepancies. For example, in HUXt, CIRs tend to have sharper gradients and higher maximum speeds relative to HelioMAS. Two factors likely affect this; firstly, this could be due to non-radial dynamics in HelioMAS allowing flow deflections that HUXt does not model; secondly, the upwind numerical scheme used in HUXt is known to permit sharper gradients than the numerical schemes employed by MHD models. For a more complete review of the scientific justification of HUXt, and its comparison to MHD simulations, see Owens et al. [13].

3 Numerical scheme

The HUXt model equations are solved numerically using a first order upwind scheme [22], and the discretised equations are given in Owens et al. [13].

The default radial grid of HUXt has an inner boundary at 30 $r_\odot$, outer boundary at 240 $r_\odot$, with a radial grid step of 1.5 $r_\odot$. However, HUXt has built-in functionality to work with different radial grids and boundary heights, as discussed in Sections 5.2.3 and Section 5.2.1.

Fundamentally, HUXt is a 1-D radial model of the solar wind. Single 1D solutions are useful in their own right, due to the rapid computation time (much less than a second on a modest CPU for a 5-day simulation out to Earth orbit). Thus, one use-case is to run HUXt in a synodic frame of reference and simulate the Earth-Sun line. This can be done in large ensembles of O (10^3–10^5), enabling case-specific estimates of forecast uncertainty (e.g., [23]). Examples of idealised and data-driven 1-D solutions are provided in the *HUXt_examples.ipynb* notebook, discussed in Section 4. Here, the examples shown are for the more generalised case of 2-D and 3-D solutions that incorporate a range of longitudes and/or latitudes by assembling a set of the 1-D radial HUXt solutions. Such 2-D and 3-D solutions would generally be performed in the sidereal reference frame.

The default longitudinal grid spans 0 to 2π in 128 bins. The default latitudinal grid has 45 equally spaced cells in sine latitude. On initialisation, a single longitude or range of longitudes must be specified, whilst, unless otherwise specified, the latitude is assumed to be 0°. Finer or coarser grids can be used as required. The model grid sets zero longitude to that of Earth at the time of the start of the run. In the synodic reference frame, this is maintained throughout the run. For the sidereal reference frame, Earth increases in longitude by around 1° per day. Note that because the 2-D and 3-D solutions are collections of independent 1-D radial solutions, any subset of the longitude and latitude grid can be considered, without any impact on the solution due to longitude or latitude boundary conditions. (Figure 2 shows an example of this functionality for a 3-D solution, discussed in Section 5.2.4.).

The model time-step, Δt is set by the Courant–Friedrichs–Lewy (CFL) condition, $\Delta t \leq \frac{\Delta r}{v_{max}}$, with the default value of $v_{\max} = 2000\ km\ s^{-1}$ resulting in $\Delta t = 8.70$ min. The default $v_{\max}$ is chosen as a compromise that is suitable for all plausible ambient solar winds and the large majority of plausible Cone CME scenarios, whilst also not being set so high to be an inefficient use of computing resources.

To initialise HUXt, a uniform solar wind speed of 400 $km\ s^{-1}$ is set at all model coordinates. The model is then spun up for a time period that depends on the minimum solar wind speed in the time-dependent boundary conditions. The spin up time is set to 1.5 times the travel time it would take the slowest speed to traverse the model domain. The results of the spin up are discarded, meaning the user is presented with 'usable' solar wind conditions from the start of the requested simulation time.

3.1 Testing the numerical scheme

HUXt was primarily developed as a surrogate for 3-D MHD simulations for situations where 3-D MHD simulations would be too computationally expensive or complex. Consequently, Owens et al. [13] presented a thorough comparison between HUXt and HelioMAS simulations. Over a 40^+ year period of 578 Carrington rotations, the mean absolute error in the ambient solar wind solutions of HUXt and HelioMAS was 25.6 $km\ s^{-1}$ or

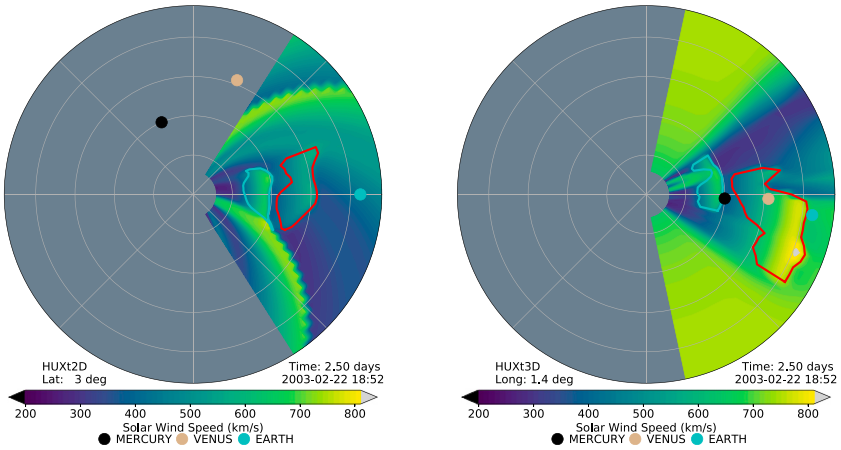

FIGURE 2
A snapshot of a HUXt3D simulation from February 2003. (left) A radius-longitude cut at Earth latitude and (right) a radius-latitude cut at Earth longitude. The boundaries of two cone CMEs are shown in red and cyan.

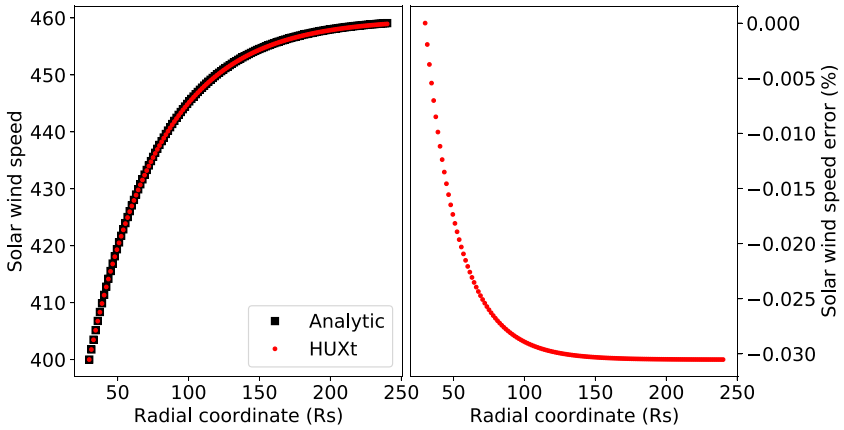

FIGURE 3
Comparison of the analytic and numerical solutions to the HUXt equations for a simplified scenario of uniform and stationary inner boundary conditions. (Left) The solar wind speed as a function of radius for the analytic (black) and HUXt solutions (red). (Right) The percentage error of the HUXt solution relative to the analytical solution as a function of radius.

6.4%. In this sense, the HUXt numerical scheme and default parameters were considered fit-for-purpose in serving as an surrogate for 3-D MHD solar wind simulations. Here we present new analysis that examines the consistency and convergence of the discretised HUXt numerical scheme in approximating the continuous solutions to the model equations.

To assess the consistency of HUXt with a solution to the continuous model equations, we first compare the numerical solution with the analytical solution for the simple scenario of a uniform and stationary inner boundary condition. Figure 3 compares the solutions for a constant inner boundary condition of 400 $km\ s^{-1}$. There is excellent agreement between the HUXt and analytical solution, with a very small negative bias that approaches an asymptotic value of approximately −0.03%. Further experiments show that the magnitude of this error is a function of the radial grid step and so we conclude that this error is related to the discretisation of the HUXt residual acceleration equation. However, the magnitude of this error is insignificant compared to the uncertainties relating to observationally derived boundary conditions (e.g., from coronal models) and from the simplifying physical assumptions used to derive the HUXt equations.

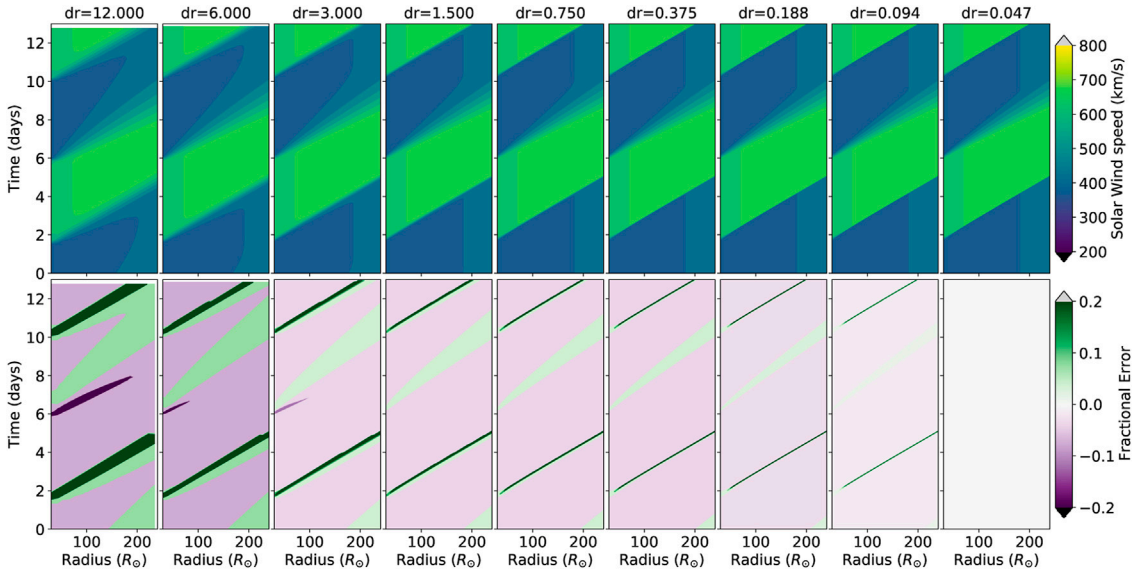

FIGURE 4
(Top) Hovmöller (distance-time) diagrams of the HUXt solar wind speed solutions for the set of different radial grid steps, with radial grid step decreasing from right to left. (Bottom) Hovmöller diagrams of the fractional error in the HUXt solution at a given radial grid step relative the HUXt solution at the finest radial grid step of dr = 0.047 $R_\odot$.

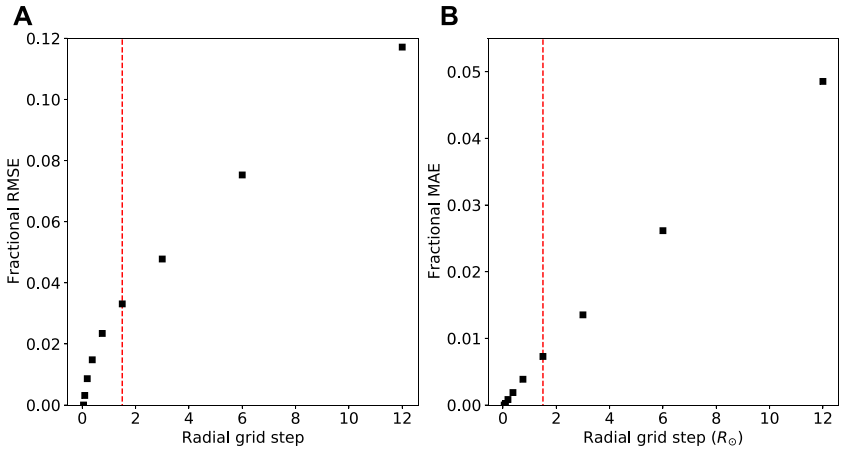

FIGURE 5
(A) The fractional root mean square error (RMSE) of the convergence tests in Figure 4 as a function of radial grid step. **(B)** The corresponding fractional mean absolute error (MAE) as a function of radial grid step. The red dashed lines mark the default HUXt radial grid step of 1.5 $R_\odot$.

The second test is convergence testing, which seeks to assess if and how the numerical solutions are converging towards an exact continuous solution as the discretised grid steps are reduced in size. We ran HUXt for a time dependent scenario of a bimodal solar wind, with slow and fast streams of 350 $km\ s^{-1}$ and 600 $km\ s^{-1}$ at the inner boundary. The model ran for 13.5 days, along a single longitude. The simulations of this scenario were generated using a range of decreasing radial grid steps, with dr being in the set {12, 6, 3, 1.5, 0.75, 0.375, 0.188, 0.094, 0.047}, in units of $R_\odot$. The finest radial grid step, dr = $0.047R_\odot$, was used as a reference standard to compare the other grid steps against.

Figure 4 presents the results of these simulations. For the largest values of dr, there are large regions of significant disagreement with the reference standard. But these errors rapidly decrease with decreasing grid step, which indicates

that the model is converging. As the grid step decreases the errors become more localised to the shock regions between fast and slow streams.

Assessment of the domain over which the model can be said to have converged is subjective, and depends on the acceptable tolerance of error for a particular application. For each of these radial grid steps, we computed both the mean absolute error (MAE) and root mean square error (RMSE). Figure 5 presents the fractional MAE and RMSE data as a function of radial grid step. This summarises what could be inferred from Figure 4, that the errors do decrease with grid step. The red dashed line marks the 1.5 $R_\odot$ grid step, which is the default configuration for HUXt. At this radial grid step, the errors are modest, with MAE being 0.7% and RMSE being 3.3%. Therefore, we consider 1.5 $R_\odot$ to be a sensible upper limit on the radial grid step, which was chosen as the default *dr* to balance the requirement that HUXt run efficiently against the discretisation errors associated with larger radial grid steps. It is simple to specify other radial grid steps in HUXt by modifying the **huxt_constants** function in *huxt.py*.

4 Python implementation

A Jupyter notebook of examples, *HUXt_examples.ipynb*, is provided with HUXt, which shows in detail how HUXt can be used in different scenarios. Here we provide a brief overview of this Python implementation of HUXt. The core codes of HUXt are contained in *huxt.py*, and use of HUXt depends on three classes; **HUXt**, **ConeCME**, and **Observer**. Figure 1 presents a diagram of these classes and their methods. In this diagram, arrows indicate the passing of information from instances of one class to another.

Instantiating an instance of the **HUXt** class configures and initialises the model. There is a minimum required input of specifying the inner boundary condition with an array of solar wind speeds. Different examples of how these can be derived are presented below. Three main methods are attached to the **HUXt** class, **HUXt.solve**, **HUXt.save**, and **HUXt.get_observer**. The **HUXt.solve** method sets the model running with optional inputs such as CMEs and streakline tracing, after which the results will be stored as **HUXt** attributes. In this way it is possible and practical to work with HUXt both programatically and interactively. The **HUXt.save** method writes all the data stored in attributes to an HDF5 file. The **HUXt.ts_from_vlong** method is used to convert a Carrington longitude profile of solar wind speeds into an time series of solar wind speeds at the HUXt inner boundary, under the assumption of synodic or sidereal rotation of the inner boundary. Whilst the **HUXt.get_observer** links to the **Observer** class to provide ephemeris data on a range of solar system bodies interpolated onto the model time grid.

The **Observer** class provides access to ephemeris data for mercury, Venus, Earth, Mars, Jupiter, and Saturn as well as STEREO-A and STEREO-B, for the period spanning 1963-01-01 to 2029-01-01 at 4 h resolution. The **Observer** class requires as input the name of the body and an array of times to output the ephemeris data. The ephemeris are linearly interpolated onto the output times from the 4 h resolution data, and positions are provided in the Heliocentric Earth Equatorial (HEEQ), Heliocentric Aries Ecliptic (HAE), and Carrington coordinate systems.

Cone CMEs are represented by their own class, **ConeCME**. This class requires as input the 6 Cone CME parameters of source longitude, source latitude, full angular width, CME initial speed, CME radial thickness, and launch time relative to the model initiation time. Including Cone CMEs in a HUXt simulation is performed by passing a list of **ConeCME** objects to the **HUXt.solve** method. Attributes of the **ConeCME** class describe all of the CME's properties and its coordinates throughout the HUXt solution. There is also a method attached to the **ConeCME** class to compute the CME arrival time at any of the solar system bodies included in the HUXt ephemeris data. The **ConeCME.parameter_array** method returns a numpy array of the Cone CME parameters, which is primarily intended to pass the CME parameters to the Numba optimised numerical core. This is required as the **HUXt** and **ConeCME** attributes relating to physical parameters are stored as Astropy quantities with the associated units, which are not interoperable with Numba optimised functions. More details on the updated implementation and use of Cone CMEs are discussed Section 5.1.2.

Basic support for 3-D simulations is provided as part of the **HUXt3d** class. In essence, this class is a wrapper around a collection of **HUXt** classes; one for each latitude simulated. Cone CMEs can be included using the same syntax as with the **HUXt** class. At present time-dependent boundary conditions are not fully supported for the **HUXt3d** container class, but all standard data and methods can be accessed for the individual **HUXt** inner classes in the standard manner.

4.1 Test suite

In the latest release of HUXt, v4.0.0, a small test suite has been included in *test_huxt.py*. The aim of the test suite is to help ensure consistency and robustness of HUXt simulations for different users, as well as across future versions of HUXt. The test suite uses the *pytest* testing framework and, at present, includes four tests. The first test is based on the comparison of the HUXt numerical solution and the analytical solution for the simple scenario of a constant inner boundary condition, as discussed in 3.1 and Figure 3. Therefore, this test helps ensure that the numerical scheme is working as expected.

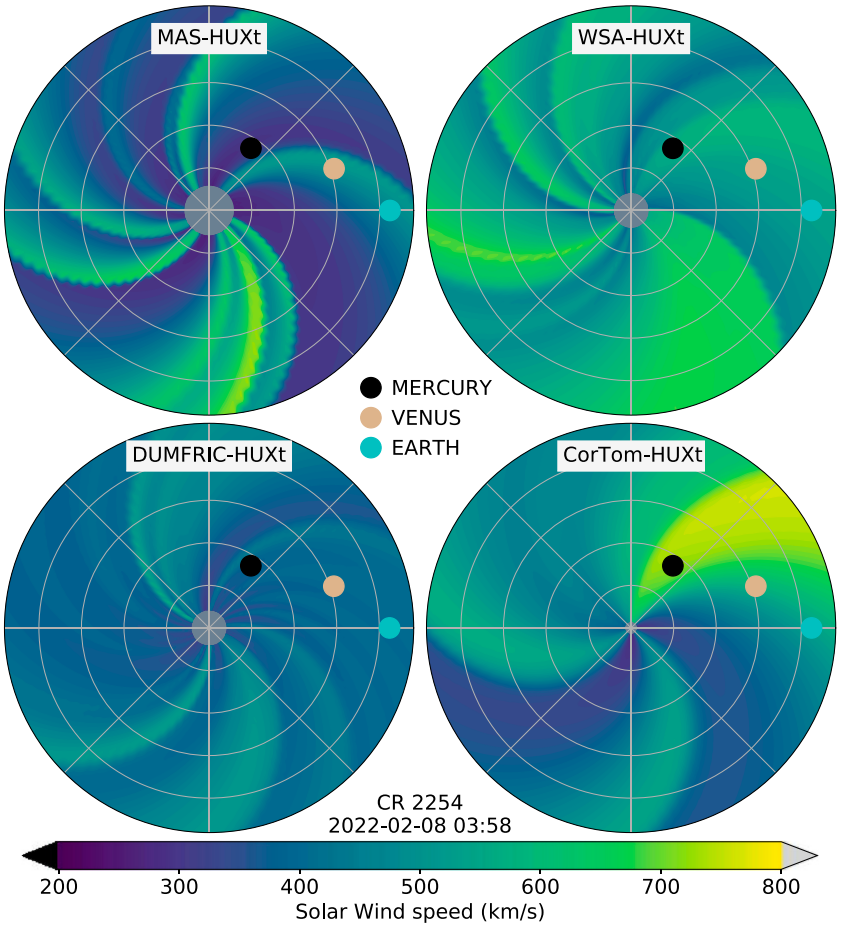

FIGURE 6
HUXt solutions for Carrington rotation 2254, using inner boundary conditions taken from the MAS, WSA, DUMFRIC and CorTom models at the heliographic equator.

The second test compares a HUXt simulation of a Cone CME erupting into structured solar wind with a reference simulation of the same scenario, where these reference data are included as part of HUXt. The solar wind solution, as well as the Cone CME properties and arrival time at Earth are checked for consistency. This test helps to ensure that the core functionality of HUXt is producing consistent results for different users, on different systems, and across different future versions.

The third test similarly compares output with reference data, but to test the reproduction of streakline tracing, described in Section 5.2.2.

Finally, the fourth test compares solutions using an inner boundary at 30 $R_\odot$ with the same boundary conditions backmapped to 10 $R_\odot$, as described in Section 5.2.3. While solutions are not expected to be identical, we define the acceptable tolerance for resulting conditions at 1 AU.

Whilst the current test suite goes some way to ensuring the core functionality of HUXt is performing reliably, at present not all of the HUXt functionality is supported by tests. Therefore it is a development priority to expand the test suite in future versions of HUXt.

5 Functionality

5.1 Updated functionality

5.1.1 Heliospheric extrapolations of coronal model output

The primary function of HUXt is to provide a computationally efficient heliospheric extrapolation of the radial solar wind speed estimated by coronal models, such as, for example, Wang-Sheeley-Arge (WSA) [24], Magnetohydrodyanmics-About-A-Sphere (MAS) Riley et al. [7] and the Durham Magnetofrictional (DUMFRIC) model [25]. HUXt is agnostic concerning the input data series; it is not tuned or intended to be used with

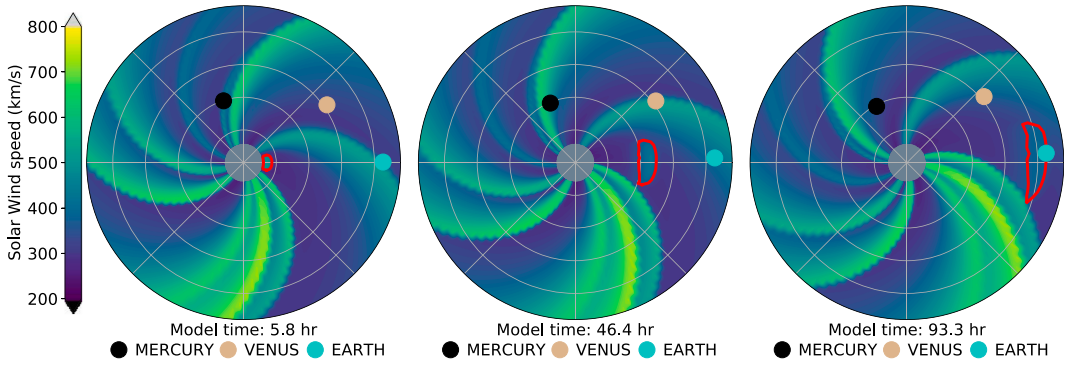

FIGURE 7
Snapshots a HUXt solution including a cone CME. The ambient solar wind boundary condition was taken from MAS for Carrington rotation 2254, as in Figure 6, while the Cone CME parameters represent a climatologically average CME. From left to right, the panels show the Cone CME (in red) shortly after initiation, halfway through its transit to Earth, an on arrival at Earth.

any particular source of boundary conditions. Figure 6 shows the HUXt solutions for Carrington rotation 2254 (beginning 2022-02-08) using boundary conditions taken from the MAS, WSA, and DUMFRIC coronal models, as well as boundary conditions derived from Coronal Tomography (CorTom)[26,27]. MAS provides conditions at 30 $R_\odot$, WSA and DUMFRIC produce conditions at 21.5 $R_\odot$, while CorTom output is at 8 $R_\odot$. HUXt is used with inner boundaries at these heights, without need to map the speeds to other radial distances. Whilst there is some agreement between the stream structure in these solutions, the absolute values and fine scale structure are significantly different. This serves to highlight the impact different model assumptions and architectures can have on the resulting estimate of the state of the corona, and consequently the state of the heliosphere. A detailed analysis of which factors determine the differences between these boundary conditions is outside the scope of this article. But we note that these models each make different approximations regarding the physics governing the structure of the coronal magnetic field; WSA approximates the coronal magnetic field as a potential magnetic field; DUMFRIC approximates the coronal magnetic field as a non-potential field; MAS approximates the coronal state using MHD. Furthermore, for WSA, DUMFRIC, and MHD, differences can arise due to the source and processing of the required magnetogram data [28]. CorTom is more fundamentally distinct from MAS, WSA and DUMFRIC, being based on a tomographic reconstruction of the coronal mass density derived from coronagraph data. Gonzi et al. [28] examined the sensitivity of ENLIL simulations to WSA and DUMFRIC inner boundary conditions. We think a future study that examines the differences between WSA, DUMFRIC, MAS and CorTom derived boundary conditions and their impact on heliospheric simulations would be a valuable addition to the literature.

We also recognise the possibility of using HUXt – which can investigate a large parameter space rapidly—to calibrate the coronal model solar wind speeds [29] in a manner which accounts for heliospheric acceleration and stream interactions. For example, Bunting and Morgan [19] used HUXt in the calibrating an empirical relationship that converts the CorTom coronal densities into solar wind speeds.

Regarding the ambient solar wind boundary conditions, several convenience functions are provided in *huxt_inputs.py* for loading and processing solar wind speed data from the MAS, WSA, PFSS, and DUMFRIC models and CorTom outputs.

5.1.2 Cone coronal mass ejections

CMEs are incorporated in HUXt *via* the Cone CME parameterisation, wherein CMEs are treated as a velocity perturbation at the model inner boundary. Cone CMEs are purely hydrodyanmic structures, and have no magnetic field structure. In Cartesian space the shape of Cone CME is formed by two hemispheres connected by a cylindrical section, akin to a short sausage, with a limiting case of a sphere. A detailed description of the Cone CME geometry is presented in Na et al. [30]. This structure is directed radially away from the Sun and advects through the model inner boundary at the CME speed. Any location on the model inner boundary that intersects the Cone CME volume is assigned the CME speed. A Cone CME is fully specified by 6 parameters; longitude and latitude in HEEQ coordinates, full angular width, speed, thickness (radial length of the cylindrical section), and the launch time relative to the model initialisation time. Conversion of CME coordinates for use with synodic and sidereal frames is handled automatically. Functionality exists for importing a standard 'Cone CME' input file, such as produced by the UK Met Office CAT tool.

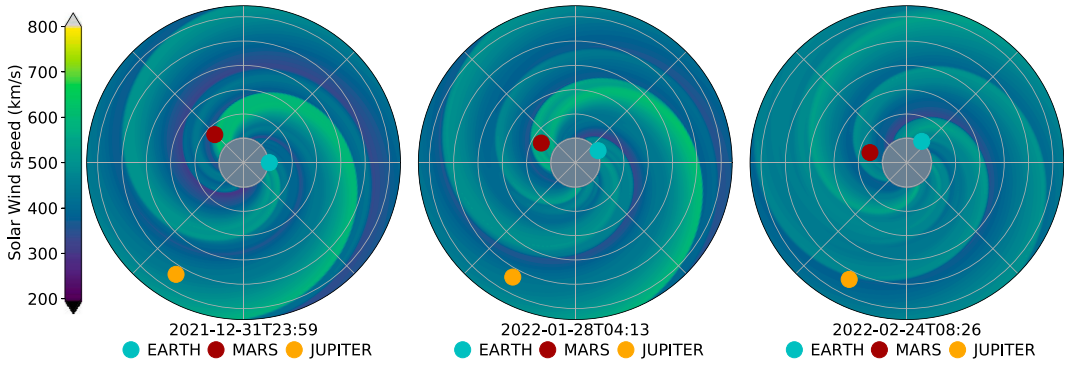

FIGURE 8
Snapshots from a HUXt simulation of a radial domain spanning 1 AU to 6 AU, initialised with the OMNI observations of *in-situ* solar wind conditions near-Earth. The snapshots are taken approximately 27 days apart.

In HUXt v1.0, the position of a Cone CME was tracked by comparing simulations with and without the Cone CME and extracting the boundaries of regions where the simulations differed by more than 20 $km\ s^{-1}$. This approach was generally successful but could struggle to identify the boundary of Cone CMEs with speeds similar to the ambient solar wind speed. To improve upon this, we experimented with tracking the Cone CMEs using a discretised tracer field, but this was found to be too diffusive in practice and consequently required arbitrary thresholds to determine the CME boundaries. Therefore, from v2.0 onwards, Cone CMEs are tracked through the HUXt solution using individual tracer particles that follow the leading and trailing edge of the CME. These tracer particles are inserted into the flow onto the CME boundary as it advects through the model inner boundary and the tracer particles then passively advect through the flow solution. The tracer particles are injected onto every longitudinal and latitudinal coordinate that intersects the ConeCME. An outline of the tracer particle advection algorithm is given in Algorithm 1 The CME boundary is computed automatically at each time-step from the locations of the tracer particles. This method of tracking the CME boundary performs well for all CME speeds, and so is a significant improvement over the tracker function in v1.0.

Algorithm 1 Tracer particle advection

for each model time, with time step Δt **do**
  Update the model solution.
  for each test particle **do**
    Interpolate the flow speed at the test particle's current location, V_{int}
    Advance the test particle's radial coordinate by $V_{int}\Delta t$.
    Update array of time history of test particle coordinates.
  end for
end for

Figure 7 shows snapshots from an example HUXt simulation including a Cone CME, with the red line marking the boundary of the CME. The Cone CME parameters were set to represent an Earth directed climatological average CME, with initial speed of 495 $km\ s^{-1}$ and full width of 37.4°, where these values were the median speed and width values from the KINCAT catalogue of CME parameters in the HELCATS database [31]. Using the **ConeCME.compute_arrival_at_body()** method, we calculated that the CME in the simulation in Figure 7 arrived at Earth after approximately 93.9 h, with an arrival speed of 360 $km\ s^{-1}$.

5.1.3 Analysis, figures, and animations

A range of basic analysis and plotting capabilities are provided in *huxt_analysis.py*. These capabilities are demonstrated throughout the figures in this paper. Support is provided for:

- Plotting time vs. speed at fixed spatial coordinate.
- Plotting radius vs. speed at fixed longitude and time.
- Polar plot of speed as function of radius and longitude at fixed time and latitude.
- Polar plot of speed as function of radius and latitude at fixed time and longitude.
- Extracting time series of model parameters at bodies included in the **Observer** class.
- Comparison of HUXt simulations with NASA's OMNI data.

Furthermore, both the latitudinal (e.g. Figure 6) and longitudinal (right-hand panel of Figure 2) cuts through the model can be animated over the model run duration using functions provided in *huxt_analysis.py*. Comparison of HUXt simulations with the NASA's OMNI data [32] is facilitated *via* the on-demand download of OMNI data using SunPy's FIDO functionality [33].

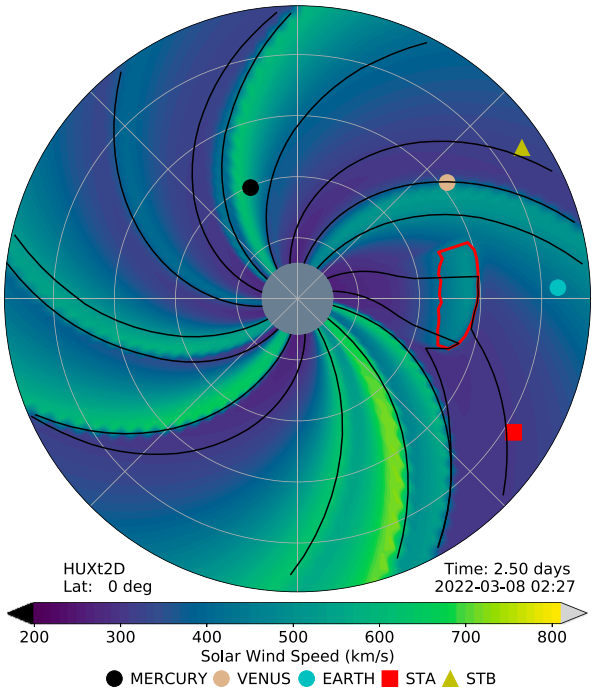

FIGURE 9
A snapshot of the HUXt simulation of Carrington rotation 2254 with streaklines (black lines) plotted for every 22.5° of Carrington longitude. The boundary of a cone CME is shown in red.

5.2 New functionality

5.2.1 *In situ* boundary conditions

HUXt accepts fully time-dependent boundary conditions by specifying solar wind speed (and magnetic field polarity, see Section 5.2.2) as a function of Carrington longitude and time at the inner boundary. In principle, this allows time-dependent coronal model output to be utilised, though this has yet to be fully tested. The second use case is initialisation of HUXt with *in-situ* solar wind measurements that are gridded into a Carrington longitude map. Functions for downloading the necessary OMNI data on-demand and generating the inner boundary condition, as well as setting up the HUXt model, are provided in **huxt_inputs.py**. Simple corotation smoothed back and forward in time (see [34], for more detail) is used to construct the boundary conditions. HUXt will accept inputs from more advanced methods, such as dynamic time warping [34], though such processing is not included as part of the HUXt codebase and can instead be accessed at https://github.com/University-of-Reading-Space-Science/SolarWindInputs_DTW.

Figure 8 shows an example of a HUXt simulation of the region from 1 AU to 6 AU, with the inner boundary condition derived from 4 months of near-Earth solar wind speed observations in the OMNI database [32]. These three snapshots are each approximately one solar rotation (27 days) apart, meaning for time-independent boundary conditions, they would be identical. Here, the time evolution of the ambient solar wind structure can clearly be seen, with the fast solar wind streams decreasing in strength from late 2021 into early 2022. Cone CMEs can be included in such time-dependent boundary condition runs, in the same manner as for 'standard' HUXt runs.

5.2.2 Streakline tracing and sector structure mapping

Functionality exists for tracing streaklines through HUXt flow fields. A streakline is the locus formed by connecting the locations of fluid parcels that originated or passed through a particular location, for example, the curve formed by smoke flowing from a chimney [35]. In HUXt, streaklines are computed by advecting test particles through the flow field from a fixed Carrington longitude. The algorithm for tracking a streakline from a fixed Carrington longitude is similar to tracking the Cone CME boundaries and hence depends primarily on Algorithm 1. The difference between the streakline tracing and the CME boundary tracing is in computing when and where the tracer particles are initialised. For the streaklines, this is done by computing the set of model time and longitude coordinates corresponding to when a specified Carrington longitude rotates into a model longitude bin. Because of the frozen-in-flux theorem, and under the assumption that the magnetic field is passive, the computed streakline will approximate a Parker spiral magnetic field line.

Figure 9 shows a snapshot from a HUXt simulation of CR2254 with streaklines plotted that originate from every 22.5° of Carrington longitude. As expected, the streaklines follow the expected Parker spiral pattern, with 'field lines' in faster flow regions being less tightly wound than in slower flow (c.f. Figure 4 of [36]). An Earth-directed cone CME has been inserted to show the disruption of the Parker spiral, with draping of the 'field' across the CME front.

Previous versions of HUXt used the radial magnetic field polarity at the inner boundary to track magnetic sector structure as a discretised passive tracer field. Unfortunately, this approach proved too diffusive, with narrow sectors being eroded away, particularly for long-duration, outer heliosphere simulations.

The streakline functionality can be used to more effectively track the position of Carrington longitudes of interest through the solar wind, such as the heliospheric current sheet (HCS). Changes in polarity of the radial magnetic field from a coronal model, such as MAS or WSA, can be traced out through the HUXt flow field. This is shown in the left-hand panel of Figure 10. The HCS locations which mark the transition from positive to negative radial field (with increasing radial distance) are shown by white lines, while the converse are shown by black lines. Then, the polarity map is found by associating regions of

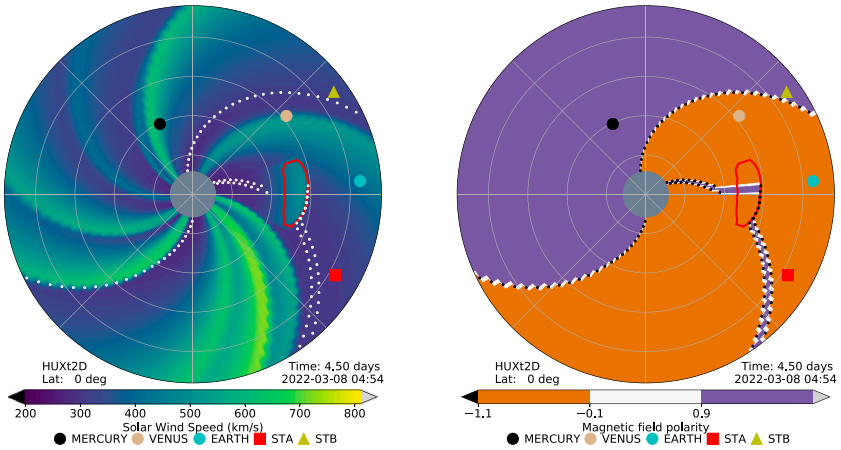

FIGURE 10
Snapshots of the HUXt solar wind speed (left) and magnetic field polarity (right) for Carrington rotation 2254. The boundary of a cone CME is shown in red. Black and white lines show heliospheric current sheet positions of positive to negative radial field transitions (with increasing radial distance) and negative to positive, respectively.

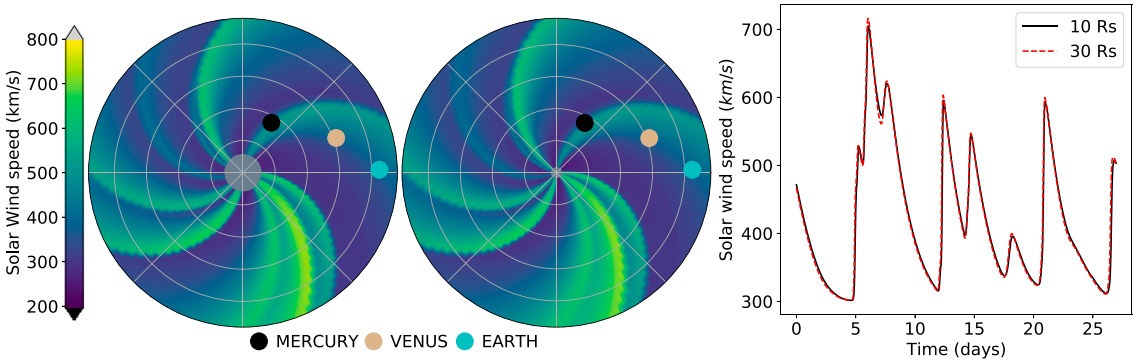

FIGURE 11
(Left) A snapshot of the HUXt simulation of Carrington rotation 2254 initialised at 30 $R_\odot$. (Middle) A snapshot of a HUXt simulation of Carrington rotation 2254 where the inner boundary condition has been backmapped to 10 $R_\odot$. (Right) Time series of the solar wind speed at Earth from the HUXt simulations initialised at 30 $R_\odot$ (red line) and 10 $R_\odot$ (black line).

the model domain between the streaklines of the HCS with the appropriate polarity, shown in the right-hand panel of Figure 10. While this period, CR2254, shows a predominantly two-sector magnetic structure, there is a very short pair of HCS crossings around Earth longitude. Despite their proximity, these are preserved by the streakline method. It can also be seen how the addition of a cone CME disrupts the normal Parker spiral pattern of the HCS.

5.2.3 Backmapping

Although the default inner boundary of HUXt is 30 $r_\odot$, it is easily configured to use different inner boundary heights. This facilitates easier comparisons with a range of observables, as well as models initialised at other radial distances—for example, the commonly used 21.5 $r_\odot$ inner boundary. In these circumstances, it can be necessary to map inner boundary conditions at one altitude to another. For example, mapping the input solar wind speed boundary condition from 30 $r_\odot$ down to 15 $r_\odot$.

This is a non-trivial calculation, because it is necessary to account for the expected solar wind acceleration between the original and desired altitudes. HUXt provides a function to compute this mapping. For a solar wind parcel on the initial boundary height, this function computes both the expected speed and source longitude of this parcel at the desired altitude, in a manner consistent with the HUXt model dynamics. The derivation of the equations used to compute this mapping are

provided in Appendix A. Note that stream interactions are ignored for backmapping, though this effect is expected to be very small close to the Sun.

Figure 11 shows an example of the results of the backmapping procedure. The left panel shows a HUXt simulation initialised at $30\,R_\odot$ with data from a MAS simulation of CR2254. The middle panel shows a HUXt simulation where the inner boundary was backmapped to $10\,R_\odot$. The right panel shows the resulting solar wind speed time series at Earth for 27 days. The time series are very similar, but the gradients and magnitudes are slightly smaller in the solution initialised at $10\,R_\odot$. This is an artefact of the backmapping procedure, which cannot distinguish between solar wind parcels of different speeds that have distinct source longitudes at a larger radii, but are estimated to have the same source longitude at a smaller radii. These overlapping parcels must necessarily be averaged and interpolated onto the regular HUXt longitude grid, which serves to smooth the inner boundary condition. It is important to be aware of this impact when backmapping an inner boundary condition to a different height.

Note that Cone CME parameters are defined relative to the model inner boundary. Thus moving the inner boundary closer to the Sun will require an earlier Cone CME insertion time and will likely need an increased initial speed, to counteract the addition deceleration of the increased propagation path. As the duration of the Cone CME speed perturbation at the inner boundary is set by the radius of the spherical disturbance, it may also be necessary to increase the Cone CME thickness in order to obtain the same speed perturbation far from the Sun.

5.2.4 3-D solutions

Figure 2 shows snapshots from a **HUXt3D** simulation of Carrington rotation 2000, from February 2003. The right-hand panel of Figure 2 shows a snapshot of the radial-latitudinal plane containing Earth. The left-hand panel of Figure 2 shows the ecliptic plane (i.e. the radial-longitude plane at a constant latitude closest to Earth's latitude). This simulation shows a somewhat typical declining/minimum solar cycle phase structure of fast wind at the poles and slower wind at mid-to-low latitudes. Two cone CMEs have been inserted, with their boundaries shown by the cyan and red lines.

6 Summary and future work

HUXt is an open source reduced physics numerical model of the solar wind, built in Python. The primary purpose of HUXt is to provide a solar wind model that is very computationally cheap and is of minimal complexity, so that it can serve as a surrogate for 3d-MHD solar wind models in context where such simulations are currently impractical, for example, large ensemble simulations and data asssimilative applications.

Version 4.0 includes two key improvements to the models functionality. Firstly, HUXt now accepts fully time-dependant boundary conditions, allowing HUXt to be run by time-series of solar wind speed observations from *in-situ* monitors, such as those provided by the OMNI dataset. Secondly, methods have been included to compute flow streaklines of the HUXt simulations. With these streaklines it is then possible to estimate the location of the Heliospheric Current Sheet, and produce maps radial heliospheric magnetic field polarity.

Additionally, this version is the first to incorporate a small test-suite, which tests the HUXt numerical scheme against a simple analytical solution, and checks a HUXt simulation for consistency with some reference simulation data. This is a first step to improving the reliability and reproducibility of HUXt. Expanding this test suite to cover all of HUXt's functionality is a development priority for future versions.

6.1 Data assimilation

Data assimilation (DA) is the process of combining observations and models, accounting for the uncertainty in both, to achieve an optimal estimate of the state of a system [37]. It is widely used in meteorology to dramatically improve forecasting skill [38], as well as in a number of areas of space physics [39–43]. There are two broad categories of DA methods, variational methods and sequential methods. Variational methodologies, such as 4DVar, aim to find the most probable system state by processing all the observations simultaneously; whereas sequential methodologies, such as the particle filters, aim to find the minimal-variance state by processing observations one-by-one, in a sequential fashion [43]. HUXt is of value to solar wind DA for two reasons.

Firstly, as it is computationally cheap, HUXt is amenable to ensemble DA methods, which require running 10s–1000s of instances of a model for each analysis, be this for research purposes or in a forecasting context [43]. For example, sequential DA methods, such as particle filters [37], use an ensemble of simulations to return an estimate of the state of a dynamical system. DA systems such as this are simple to implement and applicable in a broader range of contexts than some other DA methods (e.g., non-linear systems), but require an ensemble of particles that increases with the dimension of the state space of the model. Consequently, this can become very expensive to compute and becomes quickly impractical for models with large domains and/or numbers of parameters. HUXt is well suited to use with these methods, and we are developing a particle filter DA scheme for assimilating time-elongation profiles of CMEs observed by white-light imagers such as coronagraphs and heliospheric imagers.

Variational DA methods, such as 4DVar, often require the formulation of an 'adjoint' model, with which a models sensitivities to perturbations can be assessed. The relatively simplicity of the HUXt equations and code base means that the 'adjoint' model can be constructed with (relative) ease. This enables powerful variational DA methods to be employed Lang and Owens [44], which are proving very promising for assimilating *in situ* solar wind observations [45].

6.2 Space weather instrumentation, measurement, modelling and risk

The UK government is funding the improvement of its national space weather forecasting capability, particularly through the targeted Space Weather Instrumentation, Measurement, Modelling and Risk (SWIMMR) programme. Within SWIMMR, HUXt is being developed for operational use at the UK Met Office Space Weather Operations Center, as part of the Space Weather Empirical Ensembles Package (SWEEP). The motivation for SWEEP is that the dominant source of uncertainty in numerical model predictions and forecasts of heliosperhic solar wind speed are the uncertainties at the inner boundary of these models.

SWEEP aims to better quantify these uncertainties by producing a large ensemble of forecasts driven with boundary conditions derived from independent data sources (e.g., magnetograms and white light coronal observations) and different physical assumptions, e.g., potential and non-potential coronal magnetic fields. In doing so, SWEEP will produce ensemble forecasts with improved real-world representivity. This work depends critically on having a computationally cheap numerical model like HUXt, as such large ensembles across multiple inputs can not yet be practically generated with operationally used 3D MHD models. SWEEP is expected to be operational by 2024.

Data availability statement

The datasets presented in this study can be found in online repositories. The names of the repository/repositories and accession number(s) can be found in the article/supplementary material.

Author contributions

LB and MO contributed to the conception of this article. MO conceived the HUXt model, whilst both MO and LB co-lead the development of HUXt. Both authors contributed to the writing and revision of the submitted article.

Funding

This work was part-funded by Science and Technology Facilities Council (STFC) grant numbers ST/R000921/1 and ST/V000497/1, and NERC grant number NE/S010033/1.

Acknowledgments

This research made use of Astropy (http://www.astropy.org), a community-developed core Python package for Astronomy [46,47]. This research used version 4.0.0 [48] of the SunPy open source software package [33]. Figures for this article were made with version 3.3.4 of Matplotlib [49,50]. We have benefited greatly from the availability of the large archive of Predictive Science Inc's MAS coronal model solutions (https://www.predsci.com/mhdweb/home.php) for much of the HUXt testing and development and thank Pete Riley for useful discussions on solar wind modelling.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

References

1. Cannon P, Angling M, Barclay L, Curry C, Dyer C, Edwards R, et al. *Extreme space weather : Impacts on engineered Systems and infrastructure*. London: Royal Academy of Engineering (2013). Tech. rep.

2. Gosling JT. The solar flare myth. *J Geophys Res* (1993) 98:18937–49. doi:10.1029/93JA01896

3. Riley P, Mays ML, Andries J, Amerstorfer T, Biesecker D, Delouille V, et al. Forecasting the arrival time of coronal mass ejections: Analysis of the CCMC CME scoreboard. *Space Weather* (2018) 16:1245–60. doi:10.1029/2018SW001962

4. Gopalswamy N, Yashiro S, Michalek G, Stenborg G, Vourlidas A, Freeland S, et al. The SOHO/LASCO CME catalog. *Earth Moon Planets* (2009) 104:295–313. doi:10.1007/s11038-008-9282-7

5. Case AW, Spence HE, Owens MJ, Riley P, Odstrcil D. Ambient solar wind's effect on ICME transit times. *Geophys Res Lett* (2008) 35:L15105. doi:10.1029/2008GL034493

6. Merkin VG, Lyon JG, Lario D, Arge CN, Henney CJ. Time-dependent magnetohydrodynamic simulations of the inner heliosphere. *JGR Space Phys* (2016) 121:2866–90. doi:10.1002/2015JA022200

7. Riley P, Linker JA, Mikić Z. An empirically-driven global MHD model of the solar corona and inner heliosphere. *J Geophys Res* (2001) 106:15889–901. doi:10.1029/2000JA000121

8. Odstrcil D. Modeling 3-D solar wind structure. *Adv Space Res* (2003) 32:497–506. doi:10.1016/S0273-1177(03)00332-6

9. Narechania NM, Nikolić L, Freret L, Sterck HD, Groth CPT. An integrated data-driven solar wind – CME numerical framework for space weather forecasting. *J Space Weather Space Clim* (2021) 11:8. doi:10.1051/swsc/2020068

10. Mays ML, Taktakishvili A, Pulkkinen A, MacNeice PJ, Rastätter L, Odstrcil D, et al. Ensemble modeling of CMEs using the WSA–ENLIL+Cone model. *Sol Phys* (2015) 290:1775–814. doi:10.1007/s11207-015-0692-1

11. Odstrcil D, Mays ML, Hess P, Jones SI, Henney CJ, Arge CN. Operational modeling of heliospheric space weather for the parker solar probe. *Astrophys J Suppl Ser* (2020) 246:73. doi:10.3847/1538-4365/ab77cb

12. Riley P, Lionello R. Mapping solar wind streams from the Sun to 1 AU: A comparison of techniques. *Sol Phys* (2011) 270:575–92. doi:10.1007/s11207-011-9766-x

13. Owens MJ, Lang M, Barnard L, Riley P, Ben-Nun M, Scott CJ, et al. A computationally efficient, time-dependent model of the solar wind for use as a surrogate to three-dimensional numerical magnetohydrodynamic simulations. *Sol Phys* (2020) 295:43. doi:10.1007/s11207-020-01605-3

14. Barnard L, Owens MJ, Scott CJ, de Koning CA. Ensemble CME modeling constrained by heliospheric imager observations. *AGU Adv* (2020) 1:e2020AV000214. doi:10.1029/2020AV000214

15. Chi Y, Scott C, Shen C, Barnard L, Owens M, Xu M, et al. Modeling the observed distortion of multiple (ghost) CME fronts in STEREO heliospheric imagers. *Astrophys J Lett* (2021) 917:L16. doi:10.3847/2041-8213/ac1203

16. Barnard L, Owens M. University-of-Reading-Space-Science/HUXt. *Huxt Zenodo*. (2021). doi:10.5281/zenodo.4889327

17. Hinterreiter J, Amerstorfer T, Temmer M, Reiss MA, Weiss AJ, Möstl C, et al. Drag-based CME modeling with heliospheric images incorporating frontal deformation: ELEvoHI 2.0. *Space Weather* (2021) 19:e2021SW002836. doi:10.1029/2021SW002836

18. Macneil AR, Owens MJ, Finley AJ, Matt SP. A statistical evaluation of ballistic backmapping for the slow solar wind: The interplay of solar wind acceleration and corotation. *Mon Not R Astron Soc* (2022) 509:2390–403. doi:10.1093/mnras/stab2965

19. Bunting KA, Morgan H. An inner boundary condition for solar wind models based on coronal density. *J Space Weather Space Clim* (2022) 12:30. doi:10.1051/swsc/2022026

20. Pizzo V. A three-dimensional model of corotating streams in the solar wind, 1. Theoretical foundations. *J Geophys Res* (1978) 83:5563–72. doi:10.1029/JA083iA12p05563

21. Reiss MA, MacNeice PJ, Muglach K, Arge CN, Möstl C, Riley P, et al. Forecasting the ambient solar wind with numerical models. II. An adaptive prediction system for specifying solar wind speed near the Sun. *Astrophys J* (2020) 891:165. doi:10.3847/1538-4357/ab78a0

22. Press W, Teukolsky S, Vetterling W, Flannery B. *Numerical recipes - the art of scientific computing*. 3rd ed. Cambridge University Press (2007).

23. Owens MJ, Riley P. Probabilistic solar wind forecasting using large ensembles of near-sun conditions with a simple one-dimensional "upwind" scheme. *Space Weather* (2017) 15:1461–74. doi:10.1002/2017SW001679

24. Arge CN, Pizzo VJ. Improvement in the prediction of solar wind conditions using near-real time solar magnetic field updates. *J Geophys Res* (2000) 105:10465–79. doi:10.1029/1999JA000262

25. Yeates AR, Mackay DH, van Ballegooijen AA, Constable JA. A nonpotential model for the Sun's open magnetic flux. *J Geophys Res* (2010) 115. doi:10.1029/2010JA015611

26. Morgan H. An atlas of coronal electron density at $5R_{\odot}$. II. A spherical harmonic method for density reconstruction. *Astrophys J Suppl Ser* (2019) 242:3. doi:10.3847/1538-4365/ab125d

27. Morgan H, Cook AC. The width, density, and outflow of solar coronal streamers. *Astrophys J* (2020) 893:57. doi:10.3847/1538-4357/ab7e32

28. Gonzi S, Weinzierl M, Bocquet F-X, Bisi MM, Odstrcil D, Jackson BV, et al. Impact of inner heliospheric boundary conditions on solar wind predictions at Earth. *Space Weather* (2020) 19:1–40. doi:10.1029/2020SW002499

29. McGregor SL, Hughes WJ, Arge CN, Owens MJ, Odstrcil D. The distribution of solar wind speeds during solar minimum: Calibration for numerical solar wind modeling constraints on the source of the slow solar wind. *J Geophys Res* (2011) 116. doi:10.1029/2010JA015881

30. Na H, Moon Y-J, Lee H. Development of a full ice-cream cone model for halo coronal mass ejections. *Astrophys J* (2017) 839:82. doi:10.3847/1538-4357/aa697c

31. Barnes D, Davies JA, Harrison RA, Byrne JP, Perry CH, Bothmer V, et al. CMEs in the heliosphere: III. A statistical analysis of the kinematic properties derived from stereoscopic geometrical modelling techniques applied to CMEs detected in the heliosphere from 2008 to 2014 by STEREO/HI-1. *Sol Phys* (2020) 295:150. doi:10.1007/s11207-020-01717-w

32. King JH, Papitashvili NE. Solar wind spatial scales in and comparisons of hourly Wind and ACE plasma and magnetic field data. *J Geophys Res* (2005) 110:A02104. doi:10.1029/2004JA010649

33. Barnes WT, Bobra MG, Christe SD, Freij N, Hayes LA, Ireland J, et al. The SunPy project: Open source development and status of the version 1.0 core package. *Astrophys J* (2020) 890:68. doi:10.3847/1538-4357/ab4f7a

34. Owens MJ, Nichols JD. Using *in situ* solar-wind observations to generate inner-boundary conditions to outer-heliosphere simulations – I. Dynamic time warping applied to synthetic observations. *Mon Not R Astron Soc* (2021) 508:2575–82. doi:10.1093/mnras/stab2512

35. Batchelor GK. *An introduction to fluid dynamics*. Cambridge: Cambridge Mathematical LibraryCambridge University Press (2000). doi:10.1017/CBO9780511800955

36. Owens MJ, Forsyth RJ. The heliospheric magnetic field. *Living Rev Sol Phys* (2013) 10. doi:10.12942/lrsp-2013-5

37. Van Leeuwen PJ. Particle filtering in geophysical systems. *Mon Weather Rev* (2009) 137:4089–114. doi:10.1175/2009MWR2835.1

38. Kalnay E. *Atmospheric modeling, data assimilation and predictability — atmospheric science and meteorology*. Cambridge University Press (2005). doi:10.1017/CBO9780511802270

39. Brun AS. Towards using modern data assimilation and weather forecasting methods in solar physics. *Astron Nachr* (2007) 328:329–38. doi:10.1002/asna.200610739

40. Hickmann KS, Godinez HC, Henney CJ, Arge CN. Data assimilation in the ADAPT photospheric flux transport model. *Sol Phys* (2015) 290:1105–18. doi:10.1007/s11207-015-0666-3

41. Murray SA, Henley EM, Jackson DR, Bruinsma SL. Assessing the performance of thermospheric modeling with data assimilation throughout solar cycles 23 and 24. *Space Weather* (2015) 13(4): 220–232. doi:10.1002/2015SW001163

42. Chartier AT, Matsuo T, Anderson JL, Collins N, Hoar TJ, Lu G, et al. Ionospheric data assimilation and forecasting during storms. *J Geophys Res Space Phys* (2016) 121:764–78. doi:10.1002/2014JA020799

43. Lang M, Browne P, van Leeuwen PJ, Owens MJ. Data assimilation in the solar wind: Challenges and first results. *Space Weather* (2017) 15:1490–510. doi:10.1002/2017SW001681

44. Lang M, Owens MJ. A variational approach to data assimilation in the solar wind. *Space Weather* (2019) 17:59–83. doi:10.1029/2018SW001857

45. Lang M, Witherington J, Turner H, Owens MJ, Riley P. Improving solar wind forecasting using data assimilation. *Space Weather* (2021) 19:e2020SW002698. doi:10.1029/2020SW002698

46. Robitaille TP, Tollerud EJ, Greenfield P, Droettboom M, Bray E, Aldcroft T, et al. Astropy: A community Python package for astronomy. *Astron Astrophys* (2013) 558:A33. doi:10.1051/0004-6361/201322068

47. Price-Whelan AM, Sipőcz BM, Günther HM, Lim PL, Crawford SM, Conseil S, et al. The Astropy project: Building an open-science project and status of the v2.0 core package. *Astron J* (2018) 156:123. doi:10.3847/1538-3881/aabc4f

48. Mumford SJ, Freij N, Stansby D, Christe S, Ireland J, Mayer F, et al. SunPy. *Zenodo* (2022). doi:10.5281/zenodo.6524764

49. Caswell TA, Droettboom M, Lee A, de Andrade ES, Hunter J, Firing E, et al. Matplotlib/matplotlib: Rel: V3.3.4. *Zenodo* (2021). doi:10.5281/zenodo.4475376

50. Hunter JD. Matplotlib: A 2D graphics environment. *Comput Sci Eng* (2007) 9:90–5. doi:10.1109/MCSE.2007.55

Appendix A: Backmapping equations

Substituting Eq. 3 into Eq. 2, we obtain

$$V = V_0 + \alpha V_0 - \alpha V_0 e^{\left(\frac{r_0 - r}{r_h}\right)} \tag{4}$$

The travel time for a solar wind parcel to propagate between and inner radius, r_{in}, to an outer radius, r_{out}, is

$$T = \int_{r_{in}}^{r_{out}} \frac{1}{V} dr \tag{5}$$

Defining $A = V_0 + \alpha V_0$, this integral becomes

$$T = \int_{r_{in}}^{r_{out}} \frac{1}{A - \alpha V_0 e^{\left(\frac{r_0 - r}{r_h}\right)}} dr \tag{6}$$

For which the solution is

$$T = \left[\frac{r_h \, log\left(Ae^{\left(\frac{r}{r_h}\right)} - \alpha V_0 e^{\left(\frac{r_0}{r_h}\right)} \right)}{A} \right]_{r_{in}}^{r_{out}} \tag{7}$$

This travel time can be used in conjunction with the synodic solar rotation rate to estimate the source longitude of a solar wind parcel at an inner radius relative to its source longitude at the outer radius.