The Astrophysical Journal, 917:37 (23pp), 2021 August 10 https://doi.org/10.3847/1538-4357/ac0252

TauREx 3: A Fast, Dynamic, and Extendable Framework for Retrievals

A. F. Al-Refaie, Q. Changeat, I. P. Waldmann, and G. Tinetti
Department of Physics & Astronomy, University College London, Gower Street, WC1E 6BT, London, UK; ahmed.al-refaie.12@ucl.ac.uk
Received 2019 December 16; revised 2021 May 13; accepted 2021 May 14; published 2021 August 12

Abstract

TauREx 3 is the next generation of the `TauREx` exoplanet atmospheric retrieval framework for Windows, Mac, and Linux. It is a complete rewrite with a full Python stack that makes it easy-to-use, high-performance, dynamic, and flexible. The new main `TauREx` program is built with modularity in mind, allowing the user to augment its functionalities with custom code and efficiently perform retrievals on custom parameters. We achieve this result by dynamic determination of fitting parameters, whereby TauREx 3 can detect new parameters for retrieval from user code through a simple interface. TauREx 3 can act as a library with a simple `import taurex` command, providing a rich set of classes and functions related to atmospheric modeling. A 10× speedup in forward model computations is achieved as compared to the previous version with a sixfold reduction in retrieval times while maintaining robust results. TauREx 3 is intended as a standalone, all-in-one package for retrievals while the TauREx 3 Python library can build or augment a user's custom data pipeline easily.

Unified Astronomy Thesaurus concepts: Open source software (1866); Astronomy software (1855); Exoplanet atmospheres (487); Radiative transfer (1335); Bayesian statistics (1900); Planetary atmospheres (1244); Planetary science (1255)

1. Introduction

Characterization of exoplanet atmospheres through spectroscopic methods has become a well-established and rapidly growing field. Many retrieval codes now exist to solve the inverse forward model problem utilizing varying methods such as optimal estimation and Bayesian analysis with Markov Chain Monte Carlo (MCMC; Jolliffe 2007) and nested sampling (Skilling 2012). A non-exhaustive list of such codes is as follows: NEMESIS (Irwin et al. 2008), CHIMERA (Line et al. 2013), ARCiS (Ormel & Min 2019), BART (Harrington 2021), petitRADTRANS (Mollière et al. 2019), Helios (Kitzmann et al. 2020; Lavie et al. 2017), POSEIDON (MacDonald & Madhusudhan 2017), Madhusudhan & Seager (Madhusudhan & Seager 2009), HyDRA (Gandhi & Madhusudhan 2018), SCARLET (Benneke 2015), PLATON (Zhang et al. 2019), and Pyrat-Bay (Cubillos & Blecic 2021).

Over the last few years, these methods have been successfully applied for a large number of cases. Initially designed for the analysis of hot Jupiters (e.g., Tsiaras et al. 2018), TauREx has been successfully applied to colder and smaller planets (e.g., Tsiaras et al. 2016, 2019; Edwards et al. 2021) as well as provided the basis for theoretical performance studies of current and future instruments (e.g., Rocchetto et al. 2016; Venot et al. 2018; Changeat et al. 2019; Venot et al. 2020; Changeat et al. 2020a; Chubb et al. 2020).

To date, the majority of atmospheric spectra of exoplanets have been observed with the Hubble Space Telescope (HST). The G102 and G141 grisms provide high signal-to-noise ratio (S/N) observations in the near-infrared covering two dominant water signatures (e.g., Madhusudhan & Seager 2009; Swain et al. 2009, 2014; Kreidberg et al. 2014; Stevenson et al. 2014, 2017; Sing et al. 2015; Line et al. 2016; MacDonald & Madhusudhan 2017; Tsiaras et al. 2018). Observations from the Spitzer Space Telescope are sometimes combined to provide a more extensive wavelength coverage and some additional constraints on the carbon-bearing species: CH_4 at 3.6 μm and CO and CO_2 at 4.5 μm (e.g., Sing et al. 2015; Evans et al. 2017; Sheppard et al. 2017; Stevenson et al. 2017). Combining the limited wavelength range covered by HST with Spitzer photometry allows for more precise temperature constraints in emission spectroscopy but comes at the risk of potentially introducing systematic errors (Hou Yip et al. 2020). Additional observations from the ground can be used in atmospheric retrieval studies, providing better insight into cloud properties, atomic lines, and some of the metal oxide species (MacDonald & Madhusudhan 2017; Sedaghati et al. 2017; Nikolov et al. 2018). Similarly, combining high-resolution ground-based spectroscopy with low-resolution space-based observations allows us to better leverage information contained in both during Bayesian retrievals (e.g., Brogi & Line 2019). For now, the focus of this article will be the retrieval of low-resolution spectroscopy only.

Recently, TauREx 2 forward model and retrieval results have been verified against the NEMESIS and CHIMERA retrieval codes by Barstow et al. (2020). The study benchmarked retrievals from each code on mock forward models generated by the other codes. This cross-validation showed good agreement down to an observational noise floor of 30 ppm.

As the field of exoplanet atmosphere sounding matures, the complexity of atmospheric forward models begins to outpace our ability to implement these models in existing "static" retrieval suites. Similarly, the growing complexity of state-of-the-art atmospheric forward models places a computational upper limit on what can realistically be included in a computationally intensive retrieval. This is particularly true for data obtained with the next generation of space telescopes: the European Space Agency (ESA) and NASA's James Webb Space Telescope (JWST; Gardner et al. 2006; Bean et al. 2018) and Ariel mission (Tinetti et al. 2018), to be launched in 2021 and 2029, respectively. Here, forward models will have to evolve in order to cope with the new information content of these spectra. This puts more constraints on computing resources. In this context, it is necessary to develop the next generation of atmospheric retrieval frameworks able to cope effectively and efficiently with the increasing complexity of atmospheric modeling.

TauREx 3 is a new atmospheric retrieval code for Windows, Mac, and Linux written with a full Python 3 stack. It is a complete rewrite of TauREx 2 and it aims to improve upon its predecessor in three main areas: 1) performance in the computation of forward models, 2) flexibility in implementing and building new forward models, and 3) dynamic retrievals of any or all possible parameters.

We divide this paper into the following sections: 2) the initial setup, where we explain the installation and basic run command; 3) the framework structure, which discusses the architecture in more detail; 4) a description of the forward models; 5) the available atmospheric opacities; 6) the dynamic parameters and retrieval setups; 7) instrument simulation modes; 8) benchmarking the code against TauREx 2; and 9) future works and discussions.

2. Initial Setup

The minimum requirements for TauREx 3 are a Python 3 installation and `numpy`; all other dependencies required are automatically downloaded and installed. Full functionality (equilibrium chemistry and specific Mie scattering methods) will require additional FORTRAN and C++ compilers. To take advantage of message passing interface (MPI) nested samplers, an MPI library and compiler is required.

Installation of TauREx 3 has been significantly simplified and requires only a single command:

```
$pip install taurex
```

or, if compiling from a source,

```
$cd TauREx3/
$pip install.
```

This gives the user access to a new program that can be run from anywhere in the operating system:

```
$taurex −−help
usage: taurex [−h] −i INPUT_FILE [−R]
↪[−p] [−g] [−c] [−C] [−−light]
[−−lighter] [−o OUTPUT_FILE]
[−S SAVE_SPECTRUM]
```

Running an input file `input.par`, storing, and plotting a forward model can be done as follows:

```
$ taurex −i input.par −−plot −o
myoutput.h5
```

where "myoutput.h5" is an HDF5 file containing all generated data products such as spectra, contribution functions, and molecular profiles. Performing a retrieval requires only the `retrieval` argument:

```
$ taurex −i input.par −−plot
−o myretrieval.h5 −−retrieval
```

The structure of the input file format has been reworked with each component of the atmosphere given an input header. For instance, the temperature profile is defined under `[Temperature]` and the chemical profile under `[Chemistry]`. These changes aim to improve readability significantly, allowing the user to easily infer the type of atmosphere being computed and the nature of the retrieval conducted. Figure 1 shows the input parameter file setup

```
[Global]
xsec_path=path/to/xsec/
cia_path=path/to/cia/

[Chemistry]
chemistry_type = free
fill_gases = H2,He
ratio = 0.17
    [[H2O]]
    gas_type = constant
    mix_ratio = 1e-3

[Temperature]
profile_type = isothermal
T=1300.0

[Pressure]
profile_type = simple
atm_max_pressure = 1e6
nlayers = 100

[Planet]
planet_type = simple
planet_mass = 1.0
planet_radius = 1.0

[Star]
star_type = blackbody
radius = 1.0

[Model]
model_type = transmission
    [[Absorption]]
    [[CIA]]
    cia_pairs=H2-H2,H2-He
    [[Rayleigh]]

[Observation]
observed_spectrum = hd209458b.txt

[Optimizer]
optimizer = nestle

[Fitting]
planet_radius:fit = True
planet_radius:bounds = 0.5,1.5
H2O:fit = True
H2O:bounds = 1e-12,1e-1
H2O:mode = log
```

Figure 1. An example input file for TauREx 3.

to perform retrievals for an example atmosphere with a free chemistry model[1] and an isothermal temperature profile. The modular nature of the input parameter file allows for easy addition and customization of parameters.

One of the most powerful features within TauREx 3 is it allows users to extend the pipeline by injecting their own external atmospheric codes without modifying the main codebase. Extensions can be as simple as a new temperature profile to something more complex like a new radiative transfer model or an entirely new statistical sampling library for retrievals. Within the input file, a user can point to a custom Python file and define key variables. At runtime TauREx 3 will compile, determine new keywords, match them to the input, place them into the pipeline, and retrieve any new fitting parameters the user has designed. Taking a reasonable scenario, if a user creates some new chemistry model for TauREx 3 in a separate `mychemistry.py` file, it will have the following form:

```
class MyCustomChemistry(Chemistry):
  def__init__(self, param_one = 10,
        param_two = 'H2CO'):
#Implement other features
#and methods
```

We can now use this new chemistry model under the `[Chemistry]` header by pointing to the Python script and TauREx 3 will automatically create new input keywords based on the custom initialization keywords. An example of the now available input parameters is given below:

```
[Chemistry]
chemistry_type = custom
python_file = /path/to/mychemistry.py
param_one = 20
param_two = 'H2O'
```

This simple step fully integrates the custom model into the `TauREx` pipeline. A user can exploit automatic input keyword generation to simplify the inclusion of external libraries such as statistical samplers or equilibrium chemistry models by defining the required arguments for a library as class initialization keywords.

In a sense, all of the atmospheric parameters, models, and optimizers defined in this paper are simply the "batteries included."

3. Framework Structure

TauREx 3 provides flexibility and expandability by representing atmospheric parameters and contributions in the form of building blocks. These can be mixed and matched to form a complete forward model. The form of these building blocks is based on abstract skeleton classes defined within `TauREx`. These classes (defined in Table 1) provide a set of interfaces, in essence a guarantee on what functions are provided for other parts of the code to use. With this framework, we can interconnect them knowing each object's responsibility.

[1] A heuristic chemical model where the mixing ratios of molecules are freely chosen.

Table 1
The Base Classes in TauREx 3

Base Class	Main Purpose
`TemperatureProfile`	Computes temperature profile
`Chemistry`	Computes chemistry model
`Gas`	Computes single-species mixing ratio for free-type chemistry model
`PressureProfile`	Computes pressure profile
`Planet`	Computes planet properties
`Star`	Computes stellar properties and flux
`Contribution`	Performs a calculation on optical depth
`ForwardModel`	Builds and computes a forward model
`Spectrum`	Provides some form of spectral data to fit against
`Optimizer`	Performs retrievals
`Binner`	Bins spectra to given grid
`Output`	Handles file writes
`Instrument`	Bins and generates noise

For example, when interpolating cross sections we require temperatures for each layer of the atmosphere; an interpolator does not require knowledge of the processes to build the profile, only that it can be built and that a temperature profile is a result. This logic can be (and is) applied to almost every aspect of the TauREx framework, from the chemistry and stellar profiles to the forward models, binning, and optimizers used in retrievals. The framework makes very few assumptions about what is passed into the system, which increases the code's flexibility.

Table 1 describes each of the available base classes within TauREx 3. Each of them is created independently of the other. Figure 2 describes the structure of TauREx 3 and the interactions between different classes. TauREx 3 consists of two separate frameworks: the forward model framework and the retrieval framework. The responsibility of the forward model framework is to construct a `ForwardModel` class. The responsibility of the retrieval framework is to fit a `ForwardModel` against an observation (a `Spectrum`). The `ForwardModel` class acts as a bridge between the two frameworks since at its most irreducible representation, it can be initialized, parameters can be read and set through a common interface (see Section 6), and a spectrum can be produced. The benefit of this is that the retrieval framework does not require a forward model produced by the forward model framework. A user can disregard it and pass in a custom `ForwardModel` class. The retrieval framework as a whole supports any arbitrary forward model. The only requirement is that the forward model output and the observation to fit against match in their overall shape. Fitting for a single spectrum will only function with a forward model that outputs a single spectrum. Fitting multiple spectra simultaneously requires a forward model that returns multiple spectra. What the forward model computes internally is irrelevant to the optimizer.

With regard to the forward model framework, its purpose is to build a forward model from replaceable atmospheric components dynamically. Some atmospheric properties have a natural dependency on other aspects of the atmosphere. These dependencies occur through interface methods for each class that provide their own data. A chemistry profile may require temperature–pressure (TP) points, and a contribution function may require mixing profiles for each species. It is the responsibility of the `ForwardModel` class to aggregate all of these objects and interconnect them appropriately to resolve

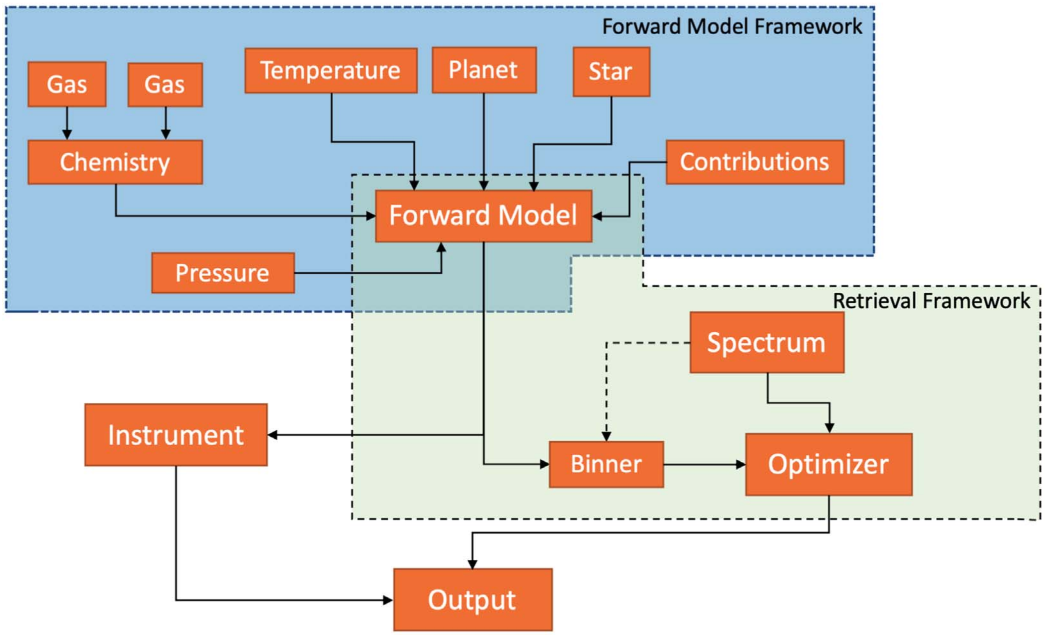


Figure 2. The overall structure of TauREx 3. Highlighted is the two-framework structure of the complete framework. Each box describes a class from Table 1, solid arrows flowing out describe outputs, and solid arrows flowing in describe inputs. A dashed arrow describes the creation of an object.

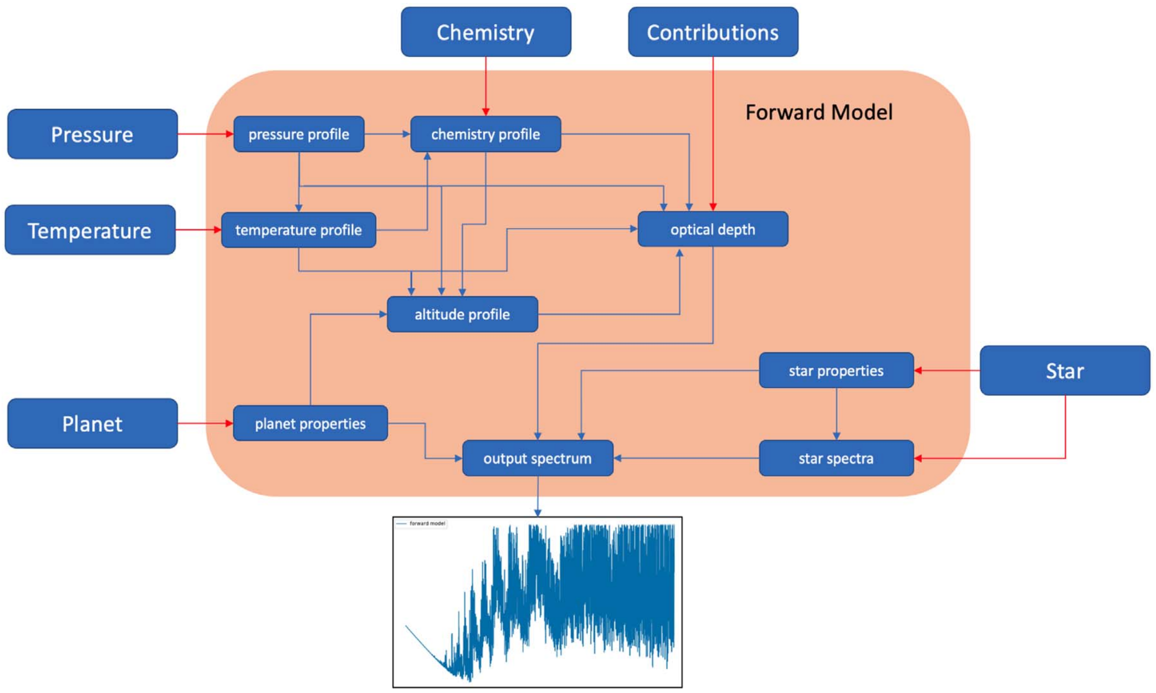


Figure 3. A flow graph describing each atmospheric property class, its dependencies, and the resolution of these dependencies within the forward model. The large blue boxes describe a class from Table 1 and the red arrows describe the property it provides to the forward model. Smaller blue boxes describe atmospheric properties produced within the forward model. Arrows flowing inward display inputs required by a particular property, and arrows flowing out display output properties.

their dependencies. A basic implementation is provided by the higher-level abstract class `SimpleForwardModel`. Figure 3 details how `SimpleForwardModel` moves data between each atmospheric parameter.

Generally, base and abstract classes do not assume the form of the coordinate system of the output data for each atmospheric component. The documentation for each of these base classes only describes what output data type is expected from each interface method, e.g., an array of values, a scalar, etc. There are specific rules for concrete implementations. Between atmospheric properties, like-for-like dimensionality guarantees their ability to function with each other (e.g., any 1D TP profile can interconnect with any 1D chemistry profile). Mixing outputs from different coordinate systems does not always guarantee compatibility. The currently implemented free chemistry profile only supports 1D TP profiles as an input and outputs 1D molecular profiles. The isothermal profile is coordinate-agnostic and will output a 1D/2D or 3D profile based on the input pressure grid. Ultimately, the forward model dictates how each component will function with the other. A 1D transmission forward model will only work with components that output 1D data. The only concrete implementations available are 1D in this release. The forward model framework standardizes all units to SI. It is up to the developer of new custom classes to ensure that inputs are converted to the required units and outputs to the TauREx 3 required SI units.

The consequence of this highly modular structure is that TauREx 3 can also act as a library providing ready-to-use classes related to atmospheric modeling, such as cross-section interpolators, temperature profiles, chemistry models, contribution functions, and optimizers, to name a few. In essence, TauREx can be run as a standalone program requiring no additional coding or in a more interactive way via library imports for advanced and personalized usage. This is evident as once installed, TauREx 3 can be imported as a library in any Python notebook, editor, or shell:

```
>>> import taurex
```

Creating and computing a 2000 K isothermal temperature profile is simply done:

```
>>> from taurex.temperature import
↪Isothermal
>>> iso = Isothermal(T = 2000)
>>> iso.initialize_profile(nlayers = 100)
>>> iso.profile
 array([1000., 1000., 1000., ..., 1000.,
↪1000., 1000.])
```

Loading and computing cross sections for arbitrary temperatures and pressures is quickly done in three lines:

```
>>> from taurex.opacity import
↪PickleOpacity
>>> h2o = PickleOpacity(
↪'xsecpath/H2O.pickle')
>>> h2o.opacity(2000.0, 1e1)
↪#2000 K10Paarray([4.67879319e − 29,
4.63121388e − 28,
↪3.97618363e − 28,...,
    6.84398649e − 34, 3.06316350e − 34,
    5.57666065e − 34])
```

These are just a small number of examples of individual parts of TauREx that can be exploited for various purposes. The primary purpose is to combine each of these parts to form our atmospheric model. We can set up the rest of the planetary and stellar parameters for a Jupiter-like planet around a Sun-like star,

```
>>> from taurex.planets import Planet
>>> from taurex.steller import
↪BlackbodyStar
>>> planet = Planet(planet_mass = 1.0,
↪planet_radius = 1.0)
>>> star = BlackbodyStar(temperature=
↪5700, radius = 1.0)
```

generate an H_2–He atmosphere with constant H_2O,

```
>>> from taurex.chemistry import
↪TaurexChemistry
>>> from taurex.chemistry import
↪ConstantGas
>>> chemistry = TaurexChemistry(
↪fill_gases = ['H2', 'He'])
>>> chemistry.addGas( ConstantGas('H2O',
↪mix_ratio = 1e − 4))
```

build a transmission model with our atmosphere parameters,

```
>>> from taurex.model import
↪TransmissionModel
>>> tm = TranmissionModel(temperature=
↪iso, chemistry = chemistry,
          planet = planet
          ,star=
          star,
          nlayers=
          100)
```

add molecular absorption, collision-induced absorption (CIA), and Rayleigh scattering,

```
>>> from taurex.contributionsimport*
>>> tm.add_contribution(
↪AbsorptionContribution())
>>> tm.add_contribution(CIAContribution
↪(cia_pairs = ['H2 − H2', 'H2 − He']))
>>> tm.add_contribution(
↪RayleighContribution())
```

finally construct the model,

```
>>> tm.build()
```

and then run it,

```
>>> fromtaurex.contributionsimport*
>>> wngrid, rprs, tau, extra = tm.model
↪()
>>> rprs
array([0.01061007, 0.01073071,
↪0.01065356,..., 0.01065933])
```

Once built, our model can now be altered at will, including the temperature and mix ratios:

```
>>> tm['T'] = 1500.0
>>> tm['H2O'] = 1e − 3
>>> wngrid, rprs, tau, extra = tm.model
↪()
>>> rprs
array([0.01063282, 0.01074269,
↪0.01066373,..., 0.01052198])
```

This library works with more interactive flavors of Python such as IPython (Perez & Granger 2007) and Jupyter Notebook, where forward models can be created, dynamically altered in real time, and used in retrievals. Taking the library at a lower level, TauREx 3 can be exploited simply for its dynamic fitting classes, allowing a developer to take advantage of the Bayesian retrieval framework for custom codes and models.

TauREx 3 also aims to conform to strict coding standards with full PEP-8 compliance. Full documentation is included with a suite of unit tests used for debugging and maintaining stability in the codebase during feature development. Internally git-flow is used to manage contributions from multiple developers while maintaining compatibility. Strict adherence to coding standards and source control is essential as often in the previous version new features became isolated versions of the main code. With TauREx, we aim for continuous and compatible integration of new features into the main codebase. For external developers, we will use the fork-and-pull model. Included is a developers guide, which highlights the coding standards and rules for those wishing to contribute to the development and provides templates and examples.

4. Forward Models

All forward models derive from the abstract base class `ForwardModel`. This defines a simple skeleton, with an abstract `model` method that must return a native wavenumber grid, the result of the forward model, the optical depth at each layer, and any other information.

For this current release, only transmission and emission models are available but single- and multiple-scattering models will be included in future releases with additions to the framework to allow for more rapid implementation of these types of models from external libraries.

One higher-level abstraction of the forward model is the `SimpleForwardModel`, which handles the majority of the setup for a 1D forward model by initializing and connecting each atmospheric component. It also computes the altitude profile z at each layer l using the expressions

$$\begin{aligned} z_l &= z_{l-1} + \Delta z_l \\ \Delta z_l &= -H_{l-1}\log\left(\frac{P_l}{P_{l-1}}\right) \\ H_l &= \frac{k_B T_l}{\mu_l g_l} \\ z_0 &= 0 \end{aligned} \tag{1}$$

where P_l, T_l, μ_l, g_l, and H_l are the pressure, temperature, mean molecular weight, acceleration due to gravity, and scale height at layer l, respectively; Δz_l is the change in altitude from layer $l-1$ to l; and $l = 0$ is considered the bottom of the atmosphere, where TauREx 3 defines the planetary radius. The only method that must be defined is `path_integral`. This design streamlined our transmission and emission implementation into a few lines of code.

For application to exoplanet retrievals, we provide the basic forward models described from previous versions (Waldmann et al. 2015b, 2015a) for primary transits and secondary eclipses.

For the transit case, we model a 1D atmosphere where the altitude is parameterized by layers (default 100 layers). The total transit depth at wavelength λ is given by

$$\Delta_\lambda = \frac{R_p^2 + a_\lambda}{R_s^2}, \tag{2}$$

where R_p is the planet radius and R_s is the parent star radius. We define a_λ as the wavelength-dependent atmospheric depth with the form

$$a_\lambda = 2\int_0^{z_{\max}} (R_p + z)(1 - e^{-\tau_\lambda(z)})dz, \tag{3}$$

where $z_{\max}$ is defined as the altitude at the top of the atmosphere. We define the wavelength-dependent global optical depth $\tau_\lambda(z)$ as

$$\tau_\lambda(z) = \sum_i \tau_{\lambda,i}(z) \tag{4}$$

where $\tau_{\lambda,i}$ denotes the optical depth for each absorber i, given by

$$\tau_{\lambda,i}(z) = \int_z^{z_{\max}} \zeta_{i,\lambda}(z')\chi_i(z')\rho(z')dz'. \tag{5}$$

Here $\zeta_{m,\lambda}$ is the cross section of a single absorbing species i, χ_i is the column density of the species i, and ρ is the number density of the atmosphere. Opacity from CIA depends on a pair of molecular species; the integral instead has the form

$$\tau_{\lambda,i}(z) = \int_z^{z_{\max}} \zeta_{i,\lambda}(z')\chi_i(z')\chi_i'(z')\rho(z')^2dz', \tag{6}$$

where χ_i and χ_i' are the column densities for their respective species in pair i. The emission model describes a plane-parallel atmosphere in which we integrate the emission from each layer to produce the final spectrum. The wavelength-dependent intensity at the top of the atmosphere from a viewing angle θ is

$$\begin{aligned} I(\tau = 0, \mu) &= B_\lambda(T_s)e^{-\frac{\tau_s}{\mu}} \\ &\quad + \int_0^1\int_0^{\tau_s} B_\lambda(T_\tau)e^{-\frac{\tau}{\mu}}d\tau d\mu, \end{aligned} \tag{7}$$

where we define $\mu = \cos(\theta)$; $B_\lambda(T)$ as the Planck function at a given temperature T, with T_s denoting the temperature at the maximum atmospheric pressure; and τ_s as the total optical depth from the planetary surface to the top of the atmosphere. We integrate for the cosine viewing angle μ using an N-point Gauss–Legendre quadrature scheme:

$$I(\tau = 0) = \sum_i^N I(\tau = 0, x_i)x_i w_i \tag{8}$$

where w_i and x_i are our weights and abscissas, respectively. The final emission spectrum is expressed as

$$\frac{F_p}{F_s} = \frac{I(\tau = 0)}{I_s} \times \left(\frac{R_p}{R_s}\right)^2. \tag{9}$$

The user builds both transmission and emission models by providing a temperature profile, pressure parameters, a chemistry model, and contribution functions to the optical depth.

Table 2
Available Contributions in TauREx 3

Class	Description
`TemperatureProfile`	Base class
`Isothermal`	Isothermal temperature profile
`Guillot2010`	Radiative equilibrium (Guillot 2010)
`Rodgers2000`	Layer-by-layer (Rodgers 2000)
`Npoint`	*N*-point temperature profile (Waldmann et al. 2015a)
`TemperatureFile`	Profile loaded from file

4.1. Pressure Profiles

Pressure profiles are represented as `PressureProfile` objects. Currently, only the `SimplePressureProfile` class is implemented, which for a given maximum pressure $P_{\max}$, minimum pressure $P_{\min}$, and number of layers N_l computes an equally spaced logarithmic grid of $N_l + 1$ pressures at each layer boundary. From this, we compute the central geometric pressure P_l for each layer l as

$$P_l = p_l \sqrt{\frac{p_{l+1}}{p_l}} \tag{10}$$

where p is the pressure at each layer boundary.

4.2. Temperature Profiles

TauREx 3 adopts the layer-by-layer approach for the currently supported temperature profiles. Included are a basic isothermal profile, a radiative two-stream approximation (Guillot 2010), a profile loaded from a file, and a multipoint temperature profile. Their classes are given in Table 2.

The previous version, TauREx 2, included three-point and four-point temperature profiles that defined temperature points at different atmospheric pressures. Smoothing was applied using a moving-average kernel with a user-definable window size. These profiles are deprecated for a more general *N*-point profile that supports an arbitrary number of pressure and temperature points. Each point defined by the user dynamically generates new fitting parameters for the retrieval. This aspect will be discussed further in Section 6.

4.3. Chemistry

TauREx 3 supports equilibrium chemistry using the ACE FORTRAN chemistry code (Agúndez et al. 2012) using the thermochemical data by Venot et al. (2012) and is installed if a suitable FORTRAN compiler is detected. Here the C/O ratio and stellar metallicity can be retrieved.

For free chemistry models, TauREx 3 can define different vertical mixing profiles for each molecule. For now, only three profiles are implemented: a constant mixing profile along the entirety of the atmosphere, a two-layer profile (Changeat et al. 2019), and a profile read from a file. However, a custom profile can be used by implementing a `Gas` class and either adding it into the chemistry model through the `addGas` method or defining the molecule with the `gas_type=custom` field and passing it in the Python file. All molecules included will generate their own set of fitting parameters (see Section 6), with each individual parameter for each molecular species capable of being retrieved. The free chemistry model has been updated from TauREx 2, which only supported H_2–He atmospheres, to allow for more massive atmospheres with any number of molecules through the `fill_gases` and `ratio` options. The ratio term determines the portion of the remaining atmospheric volume relative to the first fill molecule. We can, for example, arbitrarily define a heavy CO_2–CO–He atmosphere with a 4:3:1 ratio, a constant H_2O, and two-layer CH_4 mixing profiles in the input file as follows:

```
[Chemistry]
chemistry_type = free
fill_gases = CO2, CO, He
ratio = 0.375, 0.25
    [[H2O]]
    gas_type = constant
    mix_ratio = 1e − 4
    [[CH4]]
    gas_type = twolayer
    mix_ratio_surface = 1e − 6
    mix_ratio_top = 1e − 8
```

Custom chemistry models can also be used by setting `chemistry_type=custom` and pointing to an appropriate Python file. For all chemistry models, each molecule is considered either active or inactive. This is automatically determined at runtime by the opacity caching system (discussed in Section 5). Discovery of absorption cross sections for the particular molecule will label it active; otherwise, it will be designated inactive. Active molecules will have a direct influence on the final spectrum and molecular weight while inactive molecules will only affect the molecular weight.

4.4. Contributions

We can broadly generalize the optical depth τ from Equation (4) by considering it as a combination of multiple functions C:

$$\tau(\lambda, z) = \sum_i C_i(\lambda, z). \tag{11}$$

The most basic contribution function is pure absorption from Equation (5):

$$C_i(\lambda, z) = \int \zeta_i(\lambda, z) w_i(z) \rho(z) dz \tag{12}$$

where ζ_i is some cross section i weighted by some altitude-dependent function w_i and atmospheric density ρ. For molecular absorption, Mie scattering, and Rayleigh scattering, w_i is the column density χ_i of the absorbing species. We are not limited to this form though; we can redefine Equation (6) for CIA as

$$C_i(\lambda, z) = \int \zeta_i(\lambda, z) w_i(z) \rho(z)^2 dz \tag{13}$$

where w_i is the product of the column densities $w_i = \chi_i \chi_i'$. We are free to use more exotic functions—for example, flat-opacity

Table 3
Available Contributions in TauREx 3

Class	Type of Contribution
`Contribution`	Base class
`AbsorptionContribution`	Molecular absorption
`CIAContribution`	CIA
`RayleighContribution`	Rayleigh scattering
`SimpleCloudsContribution`	Gray clouds
`BHMieContribution`	Mie scattering (Bohren & Huffman 2007)
`LeeMieContribution`	Mie scattering (Lee et al. 2013)
`FlatMieContribution`	Constant-value opacity

clouds:

$$C_{\mathrm{clouds}}(\lambda, z) = \begin{cases} \sigma & \text{if } P_{\mathrm{t}} \geqslant P(z) \geqslant P_{\mathrm{b}} \\ 0 & \text{if } P(z) < P_{\mathrm{b}} \\ 0 & \text{if } P(z) > P_{\mathrm{t}} \end{cases} \tag{14}$$

where σ is a user-defined opacity value in square meters, $P(z)$ is the pressure at altitude z, P_{t} is the pressure at the top of the cloud deck, and P_{b} is the pressure at the bottom of the cloud deck. We note that the cloud model given by Equation (14) is discretized along $P(z)$. The choice of P_{t} and P_{b} will produce identical results when placed between two atmospheric layers. Discrete models are generally sufficient for currently available spectroscopic data, but next-generation telescopes will require a comprehensive description of scattering from clouds. Future versions of TauREx 3 will include more complex cloud models; however, a user is free to build on the current contribution framework to include these effects.

The calculation of the optical depth reduces to an array of these functions. We are free to dynamically create, add, and remove them from the forward model according to our demands. Implementing new scattering processes requires no modification of the underlying transmission and emission code.

These contribution functions are encapsulated in the `Contribution` class. The `prepare` method is called before each path integral and should perform initialization of any required data (e.g., loading and interpolating cross sections for molecular absorption). Our contribution function, C, is then implemented in the `contribute` method. A user can choose what type of contribution to add by inserting it into a forward model using the `add_contribution` method or by defining it in the input file. Each contribution can have its own set of parameters that can be optimized during retrieval. A list of available contributions is listed in Table 3.

Supplementing the pipeline with custom contributions in TauREx 3 is only possible when it is used in the library form. There is no custom option in the input file at the moment as conveying the option in the current input file design would risk cluttering its structure. A possible future option may be a user-defined folder containing the user's set of custom Python files, which could then be automatically collected and parsed accordingly.

4.5. Binning

The `Binner` classes handle resampling spectra. TauREx 3 provides the `FluxBinner` implementation, which bins both flux and uncertainties to a given grid with the corresponding widths defined at class creation. The implementation takes into account the relation between the spectral grid defined by the central bin λ_i and bin widths $\Delta\lambda_i$ and the resampling target grid defined by λ_j and its corresponding bin widths $\Delta\lambda_j$. For clarity we define the minimum span of a spectral bin as $\lambda^- = \lambda - \frac{\Delta\lambda}{2}$ and the maximum as $\lambda^+ = \lambda + \frac{\Delta\lambda}{2}$. For each spectral bin λ_i that satisfies either of the rules

$$\begin{aligned} \lambda_j^- < \lambda_i^- < \lambda_j^+ \\ \lambda_j^- < \lambda_i^+ < \lambda_j^+ \end{aligned} \tag{15}$$

we compute a corresponding weight w_{ij} dependent on its occupancy within the resampling bin:

$$w_{ij} = \frac{\min(\lambda_j^+, \lambda_i^+) - \max(\lambda_j^-, \lambda_i^-)}{\Delta\lambda_i} \tag{16}$$

where max and min are functions that select the largest and smallest values, respectively. For bins that fully lie within the resampling bin, this reduces to $w_{ij} = 1$. The weighted mean of corresponding spectral fluxes F_i is computed to produce the resampled flux F_j:

$$F_j = \frac{\sum_i F_i w_{ij}}{\sum_i w_{ij}}. \tag{17}$$

If uncertainties σ are included then the resampling takes the form of weighted propagation of uncertainties:

$$\sigma_j = \sqrt{\frac{\sum_i w_{ij}^2 \sigma_i^2}{\left(\sum_i w_{ij}\right)^2}}. \tag{18}$$

The algorithm can take into account overlapping bins and nonuniform grids. The class can be used outside of TauREx 3 to bin spectra with and without uncertainties. Defining the binning in the input file will constrain the final spectrum to the given region.

5. Opacities

The previous version of TauREx 2 utilized both k-tables and absorption cross sections; TauREx 3 makes exclusive use of absorption cross sections. The major optimizations within TauREx 3 (discussed in Section 8) have meant that k-tables perform no better computationally and are in fact slower when taking multiple species into account.

The absorption cross sections come in the form of temperature–pressure–wavelength grids. TauREx 3 includes two interpolation schemes. A faster linear interpolation for temperature and pressure is as follows:

$$\sigma_i(T) = \sigma_i(T_1) + m(T - T_1), \tag{19}$$

$$\text{where} \quad m = \frac{\sigma_i(T_2) - \sigma_i(T_1)}{T_2 - T_1}, \tag{20}$$

$$\sigma_i(P) = \sigma_i(P_1) + m(P - P_1), \tag{21}$$

$$\text{where} \quad m = \frac{\sigma_i(P_2) - \sigma_i(P_1)}{P_2 - P_1}. \tag{22}$$

Here σ_i is the absorption coefficient at wavelength λ_i; P and T are our chosen pressure and temperature, respectively; and P_1, P_2, T_1, and T_2 are our pressure and temperature points on the grid chosen so that $P_1 < P < P_2$ and $T_1 < T < T_2$. A second, more accurate scheme (Hill et al. 2013) for temperature interpolation employs the form

$$\sigma_i(T) = a_i e^{-b_i/T}$$
$$b_i = \left(\frac{1}{T_2} - \frac{1}{T_1}\right)^{-1} \ln \frac{\sigma_i(T_1)}{\sigma_i(T_2)}$$
$$a_i = \sigma_i(T_1) e^{b_i/T_1}.$$

Analyses by Hill et al. (2013) and Barton et al. (2017) have demonstrated interpolation residuals of less than 1.64% for H_2O with a temperature sampling grid of $\Delta T = 100$. Testing on temperature grids with $\Delta T = 200$ gives interpolation residuals of ≈3.4% compared to those of the linear scheme of ≈11.2%.

The `sympy` library (Meurer et al. 2017) was used to generate the most computationally efficient form of both interpolation schemes. Either scheme can be activated using the `interpolation_mode` keyword in the input file:

```
[Global]
#Activate linear scheme
interpolation_mode = linear
#Activate exp scheme
interpolation_mode = exp
```

The exponential interpolation time is about three times longer than the linear scheme due to the inclusion of the `exp` and `log` transcendental functions. The linear scheme is the default. There is no standard or agreed-upon method for handling cross-section interpolation outside of its applicable temperature ranges. If the upper/lower value lies outside of the cross section's temperature or precalculated pressure range, we fix it to either the maximum or the minimum temperature/pressure value. For the case where both upper and lower bounds are below the minimum applicable range, we return zero. When both pressure and temperature are above the maximum, the cross section at the maximum temperature and pressure is returned. We do not extrapolate wavelengths and all values outside the applicable wavelength range of the cross section will return zero.

The wavenumber grid of the forward model is selected at runtime from whichever loaded opacity has the highest resolution. Every other opacity is then resampled to the chosen opacity's grid before any computations begin. This grid then becomes the "native" grid of the forward model.

5.1. Formats

TauREx 3 supports the pickle format (based on Python object serialization) used in TauREx 2 for the absorption cross sections but also includes support for the new HDF5 format from Chubb et al. (2021). The format comes with the option of streaming the coefficients used in the path integral directly from the HDF5 file, saving memory at the cost of approximately a 5–10× (dependent on the performance of the storage medium) degradation in performance from a significant increase in input/output reads. Reducing memory cost is advantageous when dealing with very high resolution opacities. For example, opacities at $R = 100{,}000$ covering a wavelength range of 0.3–15 μm with 20 temperature points and 20 pressure points require roughly 2.6 GB of memory. For an 8 GB workstation, this limits us to about three to four molecules. Streaming the opacities offloads this memory requirement to local storage. By default, all cross sections are loaded into memory, but the streaming option can be activated using the `in_memory=False` tag in the input file. The `.dat` Exo-Transmit format (Kempton et al. 2017) is also supported. We recommend the ExoMol[2] project (Tennyson & Yurchenko 2012; Tennyson et al. 2016) as a go-to source for cross sections. It provides the molecular line lists calculated by ExoMol as well as links to the latest available line lists from third-party sources. It further provides molecular broadening parameters and the ExoCross code (Yurchenko et al. 2018) used to build cross sections for many of the molecular species in this study. The ExoMolOP library (Chubb et al. 2021) provides precomputed cross sections in HDF5 format ready for use with TauREx 3 and can be downloaded from the ExoMol website.

For CIA, the HITRAN (Richard et al. 2012; Rothman et al. 2013; Gordon et al. 2017; Karman et al. 2019) `.cia` files are now supported and can be used directly rather than having to be converted to pickle format. CIAs that contain different wavelength grids for different temperatures are also supported.

5.2. Cache

TauREx 3 employs a lazy-loading scheme for absorption coefficients facilitated by the `OpacityCache` class. This is a singleton that is globally accessible to the entire program and will search a user-defined folder and load absorption cross sections when used. The cache requires a path to be set either through the `xsec_path` option in the input file or using the `set_opacity_path` method. At this point a molecular cross-section object can be loaded into memory as follows:

```
>>> from taurex.cache import
↪OpacityCache
>>> OpacityCache().set_opacity_path(
↪'path/to/xsec')
>>> h2o = Opacity()['H2O']
<taurex.opacity.pickleopacity.
↪PickleOpacity at 0x106b54c50>
>>> h2o.opacity(temperature = 2000.0,
↪pressure = 1e0)
array([8.73239546e - 29, 2.57633453e - 28,
↪8.40033984e - 29,...,
       6.84213835e - 34, 3.05461786e - 34,
↪        5.57537933e - 34])
```

In the initial run (depending on whether streaming is used) accessing the molecule using the square-bracket operator can take a few seconds. Subsequent calls will very quickly retrieve

[2] exomol.com

the cross sections from the cache:

```
#First load of H2O cross − sections
>>> %timeit − r 1 − n 1 OpacityCache()[
↪'H2O']
1.24 s#Second load of cross − sections
>>> %timeit OpacityCache()['H2O']
669 ns
```

A user can expect the first run of the forward model to be delayed by up to a few seconds depending on the format and number of active molecules included. When loading cross sections from a path with multiple formats, loading priority is given to the HDF5 files before other formats are considered. `CIACache` follows the same structure but for CIA files.

6. Dynamic Parameters and Retrievals

In the previous version of TauREx 2, fitting new physical parameters required explicitly hard-coding them into the retrieval. This approach is common to almost all retrieval codes that are currently available. There are significant limitations associated with this type of implementation. For instance, it does not scale well when adding new parameters, as it significantly increases code complexity. This issue becomes much more apparent when attempting to merge features from multiple developers. Apart from code complexity, a common issue is the discovery and determination of "fittable" parameters. Taking chemistry as an example, equilibrium chemistry models simplify implementation as they compute complex chemistry from a small and fixed number of parameters. However, when dealing with free chemistry models, the large number of free parameters becomes problematic. Furthermore, different molecules may have different mixing profiles, and we may wish to fit different types of chemistry profiles, implying different fittable parameters and different prior configurations. In the previous version of TauREx 2, parameter-specific priors were not supported, and all molecules had to be fit to the same prior. At most, a later implementation of the two-layer model allowed the use of two different mixing profiles at the same time.

Finally, the scaling of the parameter space is commonly fixed in most codes. Generally, this is predetermined by the expected magnitude range of the parameter. Parameters such as trace gas volume mixing ratios have an extensive range of values and have their priors transform into logarithmic space. However, when it comes to the main constituent gases of the atmosphere (in the case of secondary atmospheres), it could be more appropriate to fit in linear space or to fit for ratios of components (such as H_2–He or H_2–N_2 ratios). A choice in scaling often requires an explicit implementation for the specific parameter, which leads to more complexity in the codebase.

TauREx 3 aims to solve this issue by dynamically determining the fitting parameters in a forward model. Objects in TauREx 3, which have parameters to fit, inherit from the `Fittable` class. This includes `TemperatureProfile`, `Chemistry`, and `ForwardModel` to name a few. The main purpose of the class is to discover, generate, and advertise the fitting parameters in the form of a Python dictionary with each item containing the following:

1. The name of the parameter
2. The `latex` name of the parameter
3. How it is read (its `fget`)
4. How it is written (its `fset`)
5. The default fitting space
6. Whether to fit it by default
7. Its prior bounds

The `fget` and `fset` functions are vital to this system as they provide the retrieval with a means to sample parameters without knowledge of what the parameters are and where they originated. The last three items in the list above are used by the optimizer to control the nature of the prior transform for the parameter and can be altered at runtime.

Taking the `Guillot2010` temperature profile as an example, we can determine the fittable parameters by querying the object:

```
>>> guillot = Guillot2010(T_irr = 1200)
>>> params = guillot.fitting_parameters
↪()
params.keys()
dict_keys(['T_irr', 'kappa_irr',
↪'kappa_v1', 'kappa_v2', 'alpha'])
```

We can read the parameter name and LATEX form:

```
>>> params['T_irr'][0]
↪T_irr
>>> params['T_irr'][1]
'$T_\\mathrm{irr$'
```

We can show that these getters and setters have a direct influence on the temperature profile:

```
>>> guillot.equilTemperature
1200.0
>>> params['T_irr'][2]()
1200.0
>>> params['T_irr'][3](1300.0)
>>> params['Tirr'][2]()
1300.0
>>> guillot.equilTemperature
1300.0
```

Finally we can obtain the default fit space, whether it is enabled, and what its default prior bounds are:

```
>>> params['T_irr'][4] linear
>>> params['T_irr'][5]
True
>>> params['T_irr'][6]
[1300, 2500]
```

This approach gives the retrieval scheme all the necessary information required to sample without explicitly defining the parameters in the code. However, this form is relatively cumbersome to use. When placed inside a `ForwardModel` class, the parameters are collected into a unified parameter pool allowing

Table 4
Arguments for the `@fitparam` Decorator

Argument	Description	Values
`param_name`	Parameter key name	
`param_latex`	`latex` name	
`default_mode`	Default fitting space	Either "linear" or "log"
`default_fit`	Retrieve by default	`True` or `False`
`default_bounds`	Default prior bounds	[bound min, bound max]

all of them to be easily accessed:

```
>>> tm = TransmissionModel(
↪temperature_profile = guillot)
>>> tm.build()
>>> tm.fittingParameters.keys()
dict_keys(['planet_mass',
'planet_radius', 'planet_distance',
'atm_min_pressure', 'atm_max_pressure',
↪'T_irr', 'kappa_irr',
'kappa_v1', 'kappa_v2', 'alpha', 'H2O',
↪'CH4', 'He_H2'])
```

We see that our `Guillot2010` profile has been detected by the forward model and been added to the pool. The same occurs if we use `Isothermal` instead:

```
>>> tm_iso = TransmissionModel(
↪temperature_profile = Isothermal(T=
↪1000))
>>> tm_iso.build()
>>> tm_iso.fittingParameters.keys()
dict_keys(['planet_mass',
↪'planet_radius', 'planet_distance',
'atm_min_pressure', 'atm_max_pressure',
↪'T', 'H2O',
'CH4', 'He_H2'])
```

The other parameters arise from the default profiles used when nothing else is defined in the forward model. The `ForwardModel` classes provide a very simple method of accessing these parameters with the square-bracket operator:

```
>>> tm['T_irr']
1300.0
>>> tm['T_irr'] = 1400.0
>>> tm['T_irr']
1400.0
>>> guillot.equilTemperature
1400.0
```

The process of creating fitting parameters is simple. The `Fittable` class provides two ways of defining them. The first method is provided by the `@fitparam` decorator. This decorator behaves identically to the built-in Python `property` with extra arguments given in Table 4.

We can create a custom temperature profile `FoobarProfile` with parameter `Foobar_T` as seen in Figure 4:

```
class FoobarProfile(TemperatureProfile
↪ ):

    def __init__(self,foobar = 1000):
        super().__init__()

        self._foobar = foobar

    # Other code.....

    @fitparam(param_name='Foobar_T',
              default_mode = 'linear',
              default_bounds=[1200.0,
              ↪ 1500.0])
    def myFooBar(self):
        return self._foobar

    @myFoobar.setter
    def myFooBar(self,value):
        self._foobar = value

    # More code.....
```

Figure 4. Our custom temperature profile.

We can get and set the parameter like a normal Python `property`:

```
>>> foo = FoobarProfile()
>>> foo.myFooBar
1000.0
```

By adding it to a forward model we demonstrate that it is detected:

```
>>> tm = TransmissionModel(
↪temperature_profile = foo)
>>> tm.fittingParameters.keys()
dict_keys(['planet_mass',
↪'planet_radius', 'planet_distance',
'atm_min_pressure', 'atm_max_pressure',
↪'Foobar_T', 'H2O',
'CH4', 'He_H2'])
>>> tm['Foobar_T']
1000.0
>>> tm['Foobar_T'] = 1200.0
>>> foo.myFoobar
1200.0
```

Now, our custom class is ready for retrievals.

The second method is the `add_fittable_param` method. This has similar arguments to the `@fitparam` decorator but they must be explicitly provided with the getter and setter

Table 5
Supported Retrieval Libraries in TauREx 3

Method	TauREx 3 Class	Reference
MultiNest	`MultiNestOptimizer`	Feroz et al. (2009)
Nestle	`NestleOptimizer`	Barbary (2015)
PolyChord	`PolychordOptimizer`	Handley et al. (2015)
dyPolyChord	`dyPolychordOptimizer`	Higson (2018); Higson et al. (2019)

functions. This method allows for dynamic fitting parameters that can be altered depending on how they were created (such as profiles with arrays). `NPoint` demonstrates this dynamic nature, where the fitting parameters change depending on the number of temperature and pressure points set:

```
# Two point profile
>>> twop = NPoint()
>>>twop.fitting_parameters().keys()
dict_keys(['T_surface', 'T_top',
↪'P_surface', 'P_top'])
# Four point profile
>>> fourp = NPoint(temperature_points = [
↪1000.0, 2000.0],
                    pressure_points = [
↪                     1e2, 1e0])
>>> fourp.fitting_parameters().keys()
dict_keys(['T_surface', 'T_top',
↪'P_surface', 'P_top', 'P_point1',
↪'P_point2', 'T_point1', 'T_point2'
↪])
```

6.1. Optimization

For the retrievals, TauREx 3 comes with the methods given in Table 5 built in. Nestle (Barbary 2015) is a pure Python Bayesian nested sampler that is automatically downloaded and installed with TauREx 3 and can immediately be used for retrievals. MultiNest (Feroz et al. 2009), PolyChord (Handley et al. 2015), and dyPolyChord (Higson 2018; Higson et al. 2019) require their respective FORTRAN libraries to be built and Python wrappers to be installed before they can be used in TauREx 3. They will automatically be detected at runtime once they are installed. A user can include a sampler through the `optimizer=custom` flag in the input file.

The base `Optimizer` class is responsible for collecting the fitting parameters from the forward model, updating the model using the fitting parameters from the sampler, and computing the likelihood. The responsibility of optimization is handled by the `compute_fit` method, which must be defined in concrete classes. The user can inform the optimizer which fitting parameter to retrieve through its `enable_fit` and `disable_fit` methods. The optimizer also handles the parameter space conversion by wrapping `fget` and `fset` with an appropriate conversion function, determined by the `default_mode` attribute within each fitting parameter. This conversion can be altered by the user programmatically using the `set_mode` method. Currently, only linear space and log-10 space are supported. Fits can also be defined in the input file under the `[Fitting]` section. The input file is dynamic and is capable of setting user-defined fitting parameters as well. Take our profile from Figure 4 as an example—we can fit for `Foobar_T` in log scale with bounds 200.0/1000.0 as follows:

```
[Temperature]
profile_type = custom
python_file = foobar.py
foobar = 1500.0
[Fitting]
Foobar_T: fit = True
Foobar_T: mode = log
Foobar_T: bounds = 200.0, 1000.0
```

It should be noted that the bounds are always in linear space. The log conversion is automatically handled by the optimizer. Currently, only uniform priors are supported.

6.2. Model Rejection

During sampling, there may be regions within the parameter space that are nonphysical. Nonphysical forward models can include atmospheres that have greater-than-unity mixing ratios or multipoint temperature profiles that have inverted pressure points. Sampling is wasted on these regions as they may create accidental modes in the solution if they inadvertently produce "correct" spectra. To combat this, TauREx 3 provides the `InvalidModelException` exception. During retrieval, any profile or chemistry model can trigger this exception, forcing the log-likelihood to the lowest possible value. For both nested sampling and classical MCMC sampling, this results in an overall avoidance of these regions, which should result in slightly faster sampling.

7. Instruments

One of the new features within the `TauREx` pipeline is the instrument model simulator. If defined, it passes the result of a forward model simulation into an instrument noise model and generates a new binned spectrum with instrument noise and systematics. The number of observations can also be passed in to simulate further the effect of stacking multiple observations. Currently, a generic S/N model is included which computes normally distributed noise $\mathcal{N}$ at all wavelengths for a given S/N based on the simple relation

$$\mathcal{N} = \frac{S}{\mathrm{S/N}} \tag{23}$$

with S representing the maximum value (generally the maximum transit/eclipse depth) in the spectrum. The final noise $\mathcal{N}'$ is computed based on the number of observations n passed in by the user:

$$\mathcal{N}' = \frac{1}{\sqrt{n}}\mathcal{N}. \tag{24}$$

Figure 5 shows the same forward model binned to $R = 50$ and applied with increasing values of S/N. The forward model is arbitrary, but the same for all of the plots. This instrument model produces only symmetric uncertainties. The user can provide custom noise models for instruments by passing in the `instrument=custom` flag as well as pointing to the correct Python file. An example is provided in Figure 6, where a

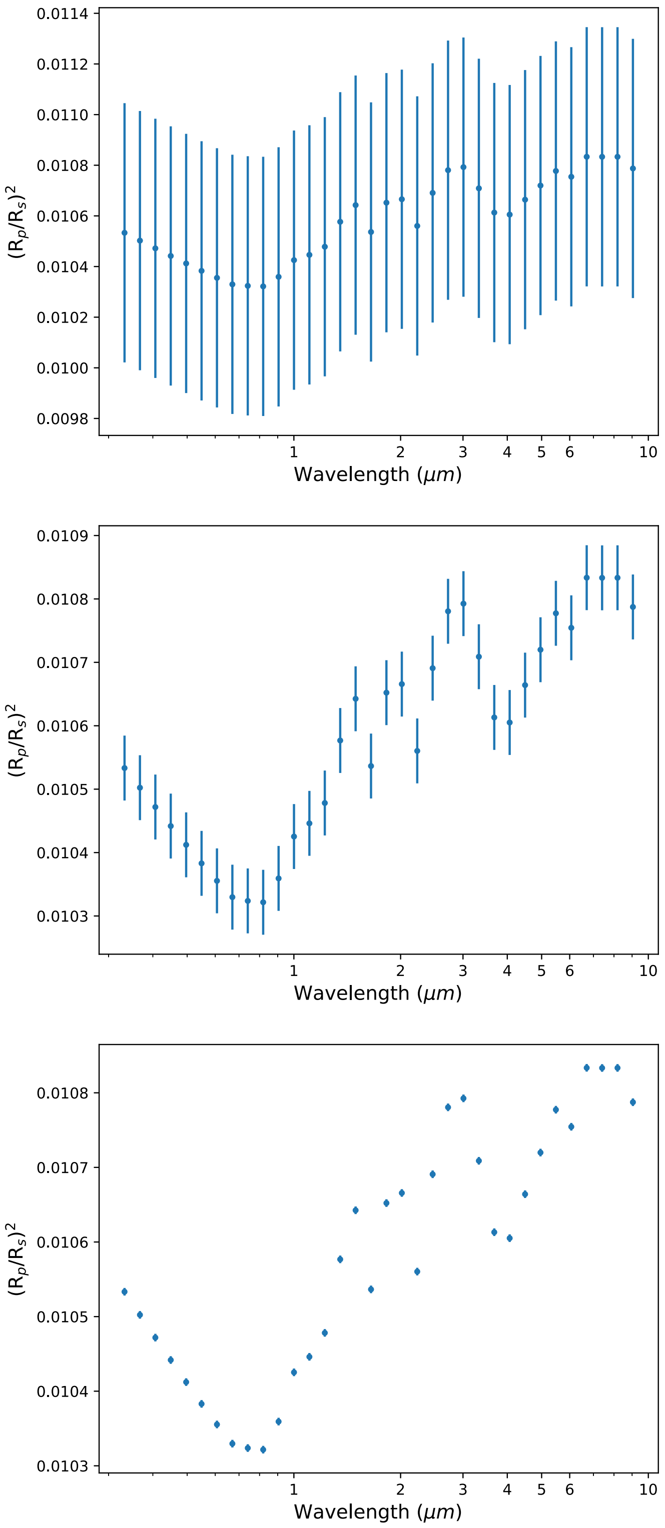


Figure 5. Plots of same forward model applied with the signal-to-noise systematic model binned to $R = 10$ for visual clarity. Points denote the spectrum and error bars denote the noise. The forward model is a clear isothermal atmosphere with $T = 2000$, 0.001% H_2O (Barton et al. 2017; Polyansky et al. 2018), and Rayleigh scattering (Cox 2015). Each plot shows an increasing value of S/N: top, S/N = 1; middle, S/N = 10; and bottom, S/N = 100.

custom instrument model is built that generates random Gaussian noise and then subsequently scatters the spectrum according to its uncertainties.

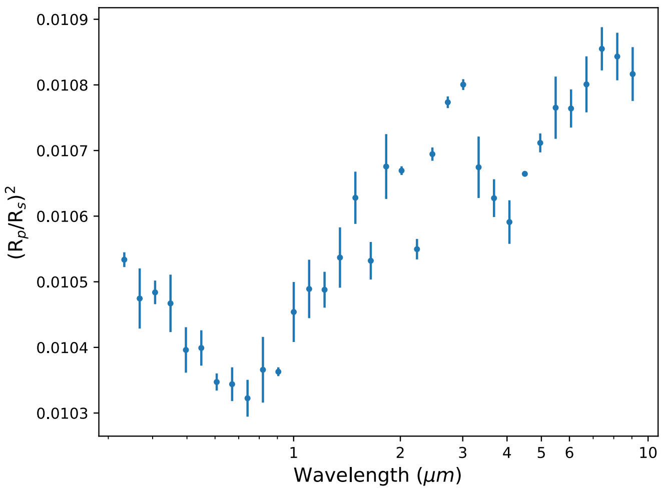


Figure 6. A custom instrument applied to the same forward model from Figure 5. This instrument model generates random Gaussian noise and scatters the spectra.

The instrument model in the `TauREx` pipeline can be used to bypass loading in an observation and to perform, instead, a retrieval directly on the simulated observation. These simulated observation retrievals provide the user with a convenient toolbox to estimate the retrievability of atmospheric parameters giving a range of telescope/instrument setups (e.g., optimizing future JWST/Ariel observations).

8. Benchmark

8.1. Computational

In the previous version of TauREx 2, the framework was written using a combination of Python for the general codebase and C++ for the heavy computational work. TauREx 3 has switched to a full Python stack, which includes the computation of the path integral. Common knowledge dictates that Python is slower than compiled languages such as C++ and FORTRAN. However, we mitigate slower computational speeds using a suite of available libraries to speed up and even match performance as compared to compiled languages, without sacrificing the flexibility of Python.

TauREx 3 fully leverages `numpy` (van der Walt et al. 2011) for its array vectorization capabilities and `numexpr` (McLeod et al. 2018) for faster NumPy operations. The brunt of the calculation exploits the `numba` (Lam et al. 2015) library, which just-in-time compiles the path integral code for even faster performance. An additional performance gain is achieved in the opacity calculation: once each opacity has been interpolated and weighted, they are fused into a single cross section from which the path integral can then be calculated. This method significantly improves the scaling of performance with the number of molecules. This optimization is also applied to Rayleigh scattering and CIA.

Multithreading was not used in TauREx 3 as it does not benefit retrievals. It is generally better to let an MPI sampler (such as MultiNest) have more cores to sample the forward models in parallel as one can generally get linear scaling with core counts. For both MultiNest and PolyChord, this holds as long as the sampling is the dominant computational bottleneck.

For our forward model benchmarks, we test on a MacBook Pro 2018 with a 2.3 GHz Intel Core i5. The absorption opacities are computed from the ExoMol line lists (Tennyson & Yurchenko 2012; Tennyson et al. 2016) using

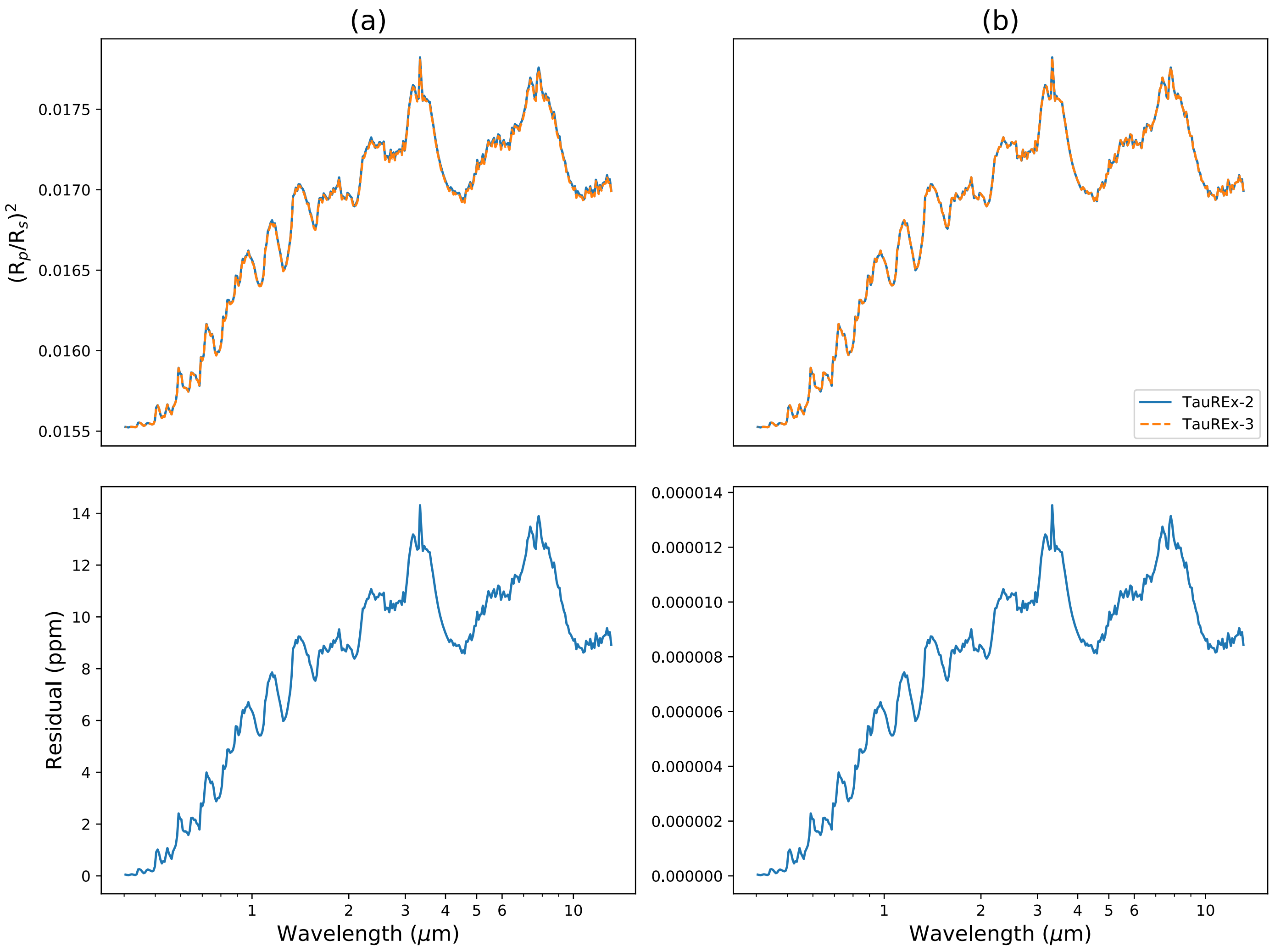


Figure 7. Plots comparing transmission spectra computed at $R = 10{,}000$ and binned to $R = 100$ for clarity with their corresponding residuals in parts per million. Both model the same planet and atmosphere and only include molecular absorption from H_2O and CH_4. Plot (a) shows the outputs given by the respective codes. Plot (b) compares TauREx 3 with a modified TauREx 2 that includes higher-precision atomic and molecular mass values.

Table 6
List of Opacities Used for Our Benchmarks

Opacity	Type	Reference
H_2–H_2	CIA	Abel et al. (2011); Fletcher et al. (2018)
H_2–He	CIA	Abel et al. (2012)
H_2O	Abs.	Barton et al. (2017); Polyansky et al. (2018)
CH_4	Abs.	Hill et al. (2013); Yurchenko & Tennyson (2014)
CO	Abs.	Li et al. (2015)
CO_2	Abs.	Rothman et al. (2010)
NH_3	Abs.	Yurchenko et al. (2011)

Table 7
List of the Atomic and Molecular Mass Values Used in Figure 7 between TauREx 2 and TauREx 3

Atom/Molecule	TauREx 2 Mass (amu)	TauREx 3 Mass (amu)
H	1	1.007940
He	4	4.002602
H_2	2	2.01588
H_2O	18	18.01528
CH_4	16	16.04276

the ExoCross (Yurchenko et al. 2018) FORTRAN code at a wavelength range of 0.3–15 μm with resolution of $R = 10{,}000$. The k-tables are also generated from the same line lists at $R = 100$ using 20 Gaussian quadrature points. Linear interpolation is used in all benchmarks. The particular sources for the absorption line lists and CIA opacities are listed in Table 6. The timings for the forward model are conducted using the `timeit` module.

For the first benchmark, we verify the accuracy of spectra produced by TauREx 3 as compared to TauREx 2 for both the transit and eclipse cases. An arbitrary atmosphere is built which includes molecular absorption from H_2O and CH_4. Figure 7(a) compares both spectra, and residuals of an about 15 ppm difference are observed.

The majority of the residuals can be explained by the varying numerical precision of the atomic and molecular masses implemented in either code. TauREx 3 implements more precise values for the atomic and molecular masses given in Table 7. This has the effect of compressing the atmosphere through the slightly heavier molecules, in turn reducing the optical path length. We can observe this effect in Figure 7(b)

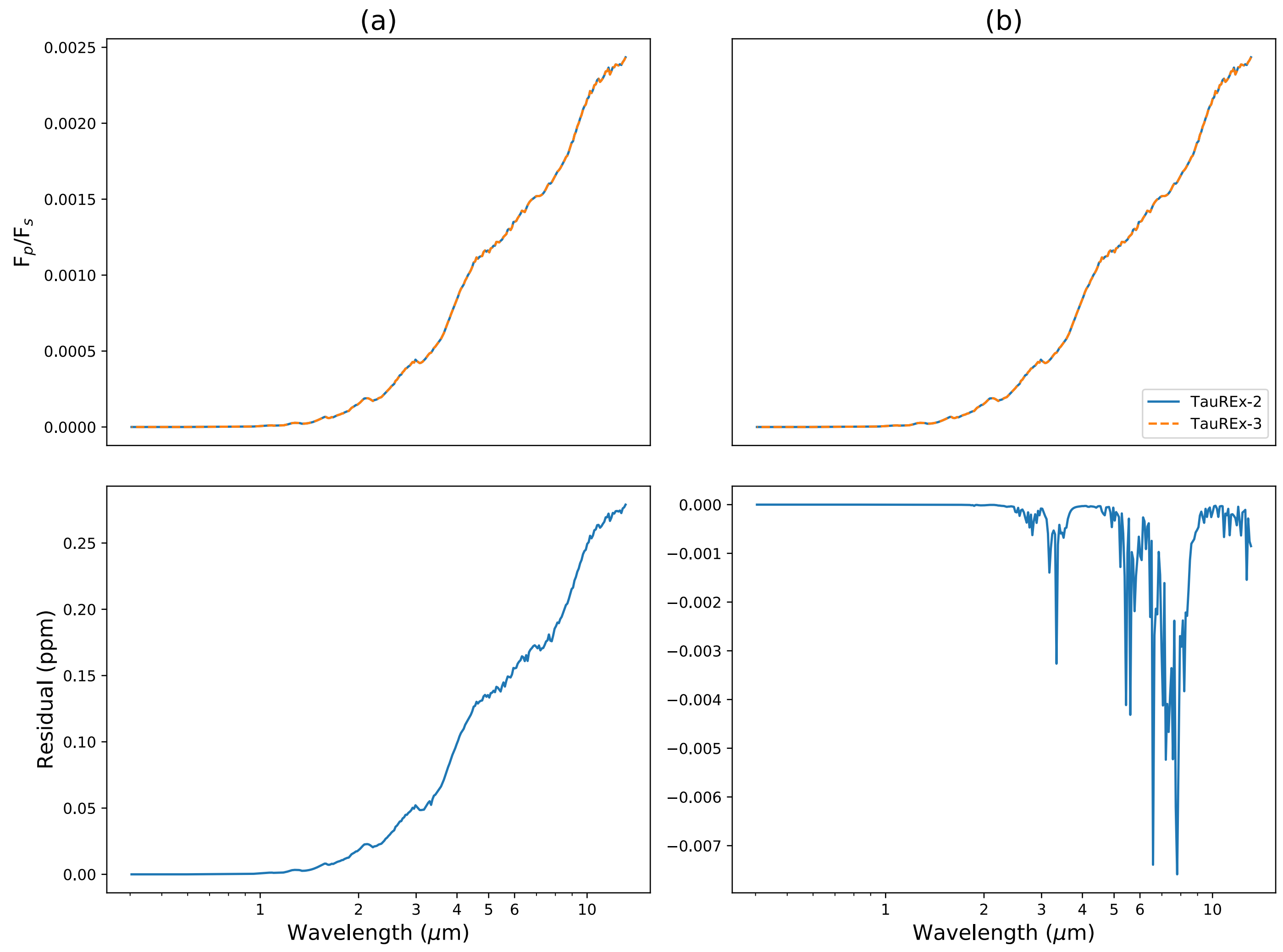


Figure 8. Plots comparing four-point quadrature emission spectra computed at $R = 10{,}000$ and binned to $R = 100$ for clarity with their corresponding residuals in parts per million. Both model the same planet and atmosphere and only include molecular absorption from H_2O and CH_4. Plot (a) compares TauREx 3 with a modified TauREx 2 that includes higher-precision atomic and molecular mass values. Plot (b) compares TauREx 3 with TauREx 2 that has been further modified with higher-precision quadrature points.

when modifying the masses in TauREx 2 to match those in TauREx 3. In this case, the residual differences become negligible ($\approx 10^{-5}$ ppm).

A similar exercise can be performed on the eclipse case. Figure 8(a) compares both emission spectra from TauREx 3 against the modified TauREx 2 from Figure 7(b) as a baseline. The spectra match well with a residual of about 0.3 ppm after removing the mass discrepancy between the codes. In the emission case, there are other sources of errors that arise from numerical precision. One example is the quadrature points: increasing the number of significant figures of the abscissa and weights in TauREx 2 to match those in TauREx 3 further reduces the residuals by two orders of magnitude, as shown in Figure 8(b).

In summary, the more efficient path integral code has not degraded the quality of the output spectrum and the majority of differences between the two spectra stem from the higher-precision constants used in TauREx 3.

The next set of benchmarks assess the performance scaling with the number of atmospheric layers for the full wavelength range. This step helps us to gauge the performance of the path integral code with minimal influence from interpolation. Three methods are used: the first two are conducted using the previous version of TauREx 2 using the cross sections and *k*-tables method, and the last is TauREx 3 using cross sections only. Each builds an atmospheric model with two active molecules with constant chemical profiles and an isothermal temperature profile. CIA with H_2–H_2 and H_2–He and Rayleigh scattering are included in the calculation.

Table 8 demonstrates the significant performance upgrade from the previous version's cross-section code with an around 10× performance boost. For the 50-layer test, the interpolation time is the dominant computational bottleneck, which gives the *k*-table method the advantage with its smaller opacity array. After this step, the path integral becomes the dominant computation bottleneck and TauREx 3 matches the *k*-table method in performance at 100 layers. After this point, using TauREx 3 with cross sections is about 1.1–2× faster than TauREx 2 with *k*-tables and around 54× faster than the older cross-section code.

Another essential comparison is how computation time scales with the number of molecules. The same test is conducted but is instead fixed at 100 atmospheric layers. Pseudo-molecules are generated by replicating the available

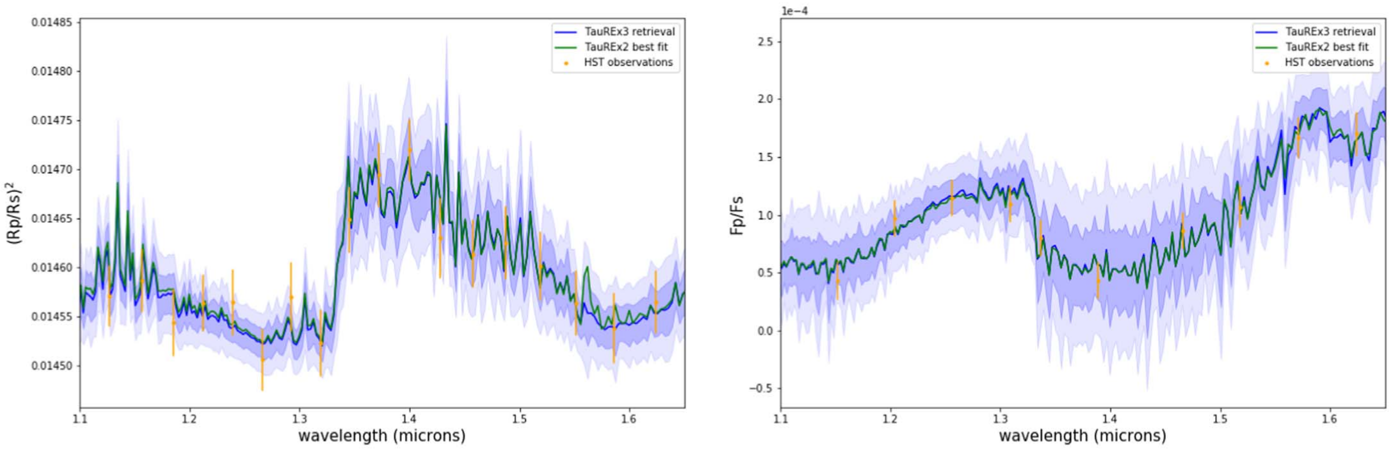


Figure 9. Best-fit spectra (denoted by solid lines) for the HD 209458b retrievals with TauREx 2 and TauREx 3. The 1σ (dark shaded region) and 2σ (light shaded region) spectra are also plotted for the TauREx 3 retrieval. Left: Retrieval of the transmission spectrum from Tsiaras et al. (2018). Right: Retrieval of the emission spectrum from Line et al. (2016). The TauREx 2 retrieval (green) and TauREx 3 retrieval (blue) give very similar spectra for both cases.

Table 8
A Comparison of the Forward Model Computation Time between TauREx 2 Using Cross Sections, TauREx 2 Using k-Tables, and TauREx 3 Using Cross Sections for the Same Atmospheric Parameters but with Increasing Number of Atmospheric Layers

Layers	TauREx 2 xsec (s)	TauREx 2 k-tables (s)	TauREx 3 xsec (s)
50	2.24	0.20	0.24
100	8.60	0.79	0.62
150	19.29	1.81	1.53
200	35.53	3.04	2.29
600	876.24	28.90	15.35

Table 9
A Comparison of the Forward Model Computation Time between TauREx 2 Using Cross Sections, TauREx 2 Using k-Tables, and TauREx 3 Using Cross Sections for the Same Atmospheric Parameters but with Increasing Number of Molecules

Molecules	TauREx 2 xsec (s)	TauREx 2 k-tables (s)	TauREx 3 xsec (s)
1	7.23	0.45	0.61
2	8.90	0.78	0.74
4	12.42	1.49	0.92
7	19.02	2.63	1.23
15	263.56	8.21	2.34

cross sections multiple times as different molecules. The results of Table 9 again show that k-tables perform best using a single molecule. With an increasing number of molecules, TauREx 3 performs significantly better than both k-tables and the older cross-section code with a 2–8× and 8–100× performance gain, respectively.

8.2. Retrieval Benchmark: HD 209458b

For our retrieval benchmark, we will study HD 209458b. Our first test will benchmark the current HST/WFC3 data, and the second will retrieve a simulated observation from the ESA Ariel mission (Tinetti et al. 2018). The aim is to assess both the consistency of the results and the computational performance of TauREx 3 against TauREx 2. We do not intend to perform a comprehensive study or reinterpretation of the available HD 209458b data as this task would be beyond the scope of this paper. We highlight that the previous version of TauREx 2 was cross-compared with other retrieval codes in Barstow et al. (2020), showing agreement between the TauREx 2, NEMESIS (Irwin et al. 2008), and CHIMERA (Line et al. 2013) retrieval codes. For the optimizer, we use MultiNest (Feroz et al. 2009; Buchner et al. 2014) compiled with an MPI. We utilize 1500 live points and an evidence tolerance of 0.5. The choice of hyperparameters ensures the adequate sampling of the retrieval's likelihood surface. Each retrieval is done on a single node of the University College London Cobweb cluster, which has a 24-core Xeon E5-2697 v2 clocked at 2.70 GHz. The timings are only for sampling and do not account for any startup time or post-processing.

For the first test, we compare the results of TauREx 3 with the ones from TauREx 2 in a real scenario for transmission and

Table 10
List of Parameters Retrieved in the Transmission and Emission Retrieval along with Their Uniform Bound Priors and the Retrieval Mode

Retrieved Parameter	Transmission Priors	Emission Priors	Mode
H_2O	−12, −1	−12, −1	log
CH_4	−12, −1	−12, −1	log
CO	−12, −1	−12, −1	log
CO_2	−12, −1	−12, −1	log
NH_3	−12, −1	−12, −1	log
$T_{\rm isothermal}$ (K)	400, 2000	none	linear
$T_{\rm surface}$ (K)	none	500, 2500	linear
$T_{\rm point1}$ (K)	none	500, 2500	linear
$T_{\rm top}$ (K)	none	500, 2500	linear
radius ($R_{\rm J}$)	1.2, 1.5	1.2, 1.5	linear
cloud pressure (bar)	1, −8	none	log

Table 11
A Comparison of the Retrieval Model Computation Time between TauREx 2 and TauREx 3 Using Cross Sections for the Same Atmospheric Priors

Time	TauREx 2 (s)	TauREx 3 (s)	No. of Samples	Speedup (x)
Transit	6140	837	110,000	7.3
Eclipse	3569	780	66,000	4.5

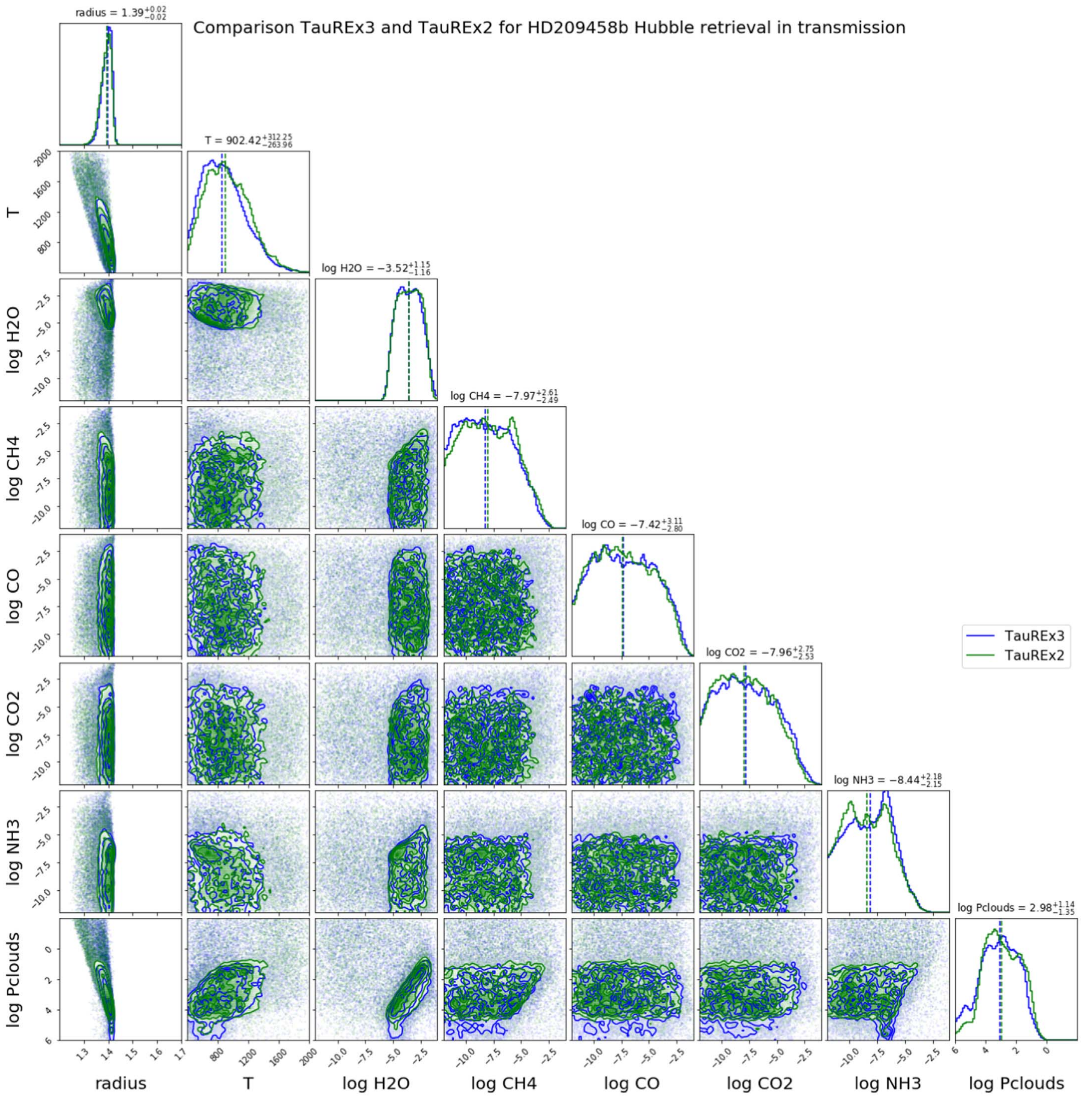


Figure 10. Posterior distribution for the HD 209458b retrievals of the transmission spectrum from Tsiaras et al. (2018) with TauREx 2 and TauREx 3. The temperatures are in kelvin, the pressures in pascals, the radii in Jupiter radii, and the molecule abundance in volume mixing ratios. The dashed lines are the median values of the posterior. The values quoted above are the median from TauREx 2 with 16% and 84% quantiles relative to it.

emission spectroscopy. We use the HST/WFC3 spectrum of HD 209458b in Tsiaras et al. (2018) for our transmission scenario and the HST/WFC3 spectrum from Line et al. (2016) for the emission case. For the latter case, we choose not to include the Spitzer points for our retrievals as combining instruments may lead to biases (Hou Yip et al. 2020).

In our comparison retrieval, we attempt to constrain isocompositions for five molecules (H_2O, CH_4, CO, CO_2, and NH_3), using cross sections at a resolution of 10,000, given in Table 6.

Along with the chemistry, we retrieve a temperature profile and the planet radius. In the transmission case, the temperature profile is isothermal and parameterized by a single parameter. We also retrieve the cloud-top pressure of a fully opaque cloud deck in the transmission case.

In the emission case, we do not consider clouds as analysis by Line et al. (2016) suggests their inclusion has a minimal impact on the dayside spectrum. We provide flexibility to the temperature by using an *N*-point profile and retrieving three distinct values (the temperature at the surface (10 bar),

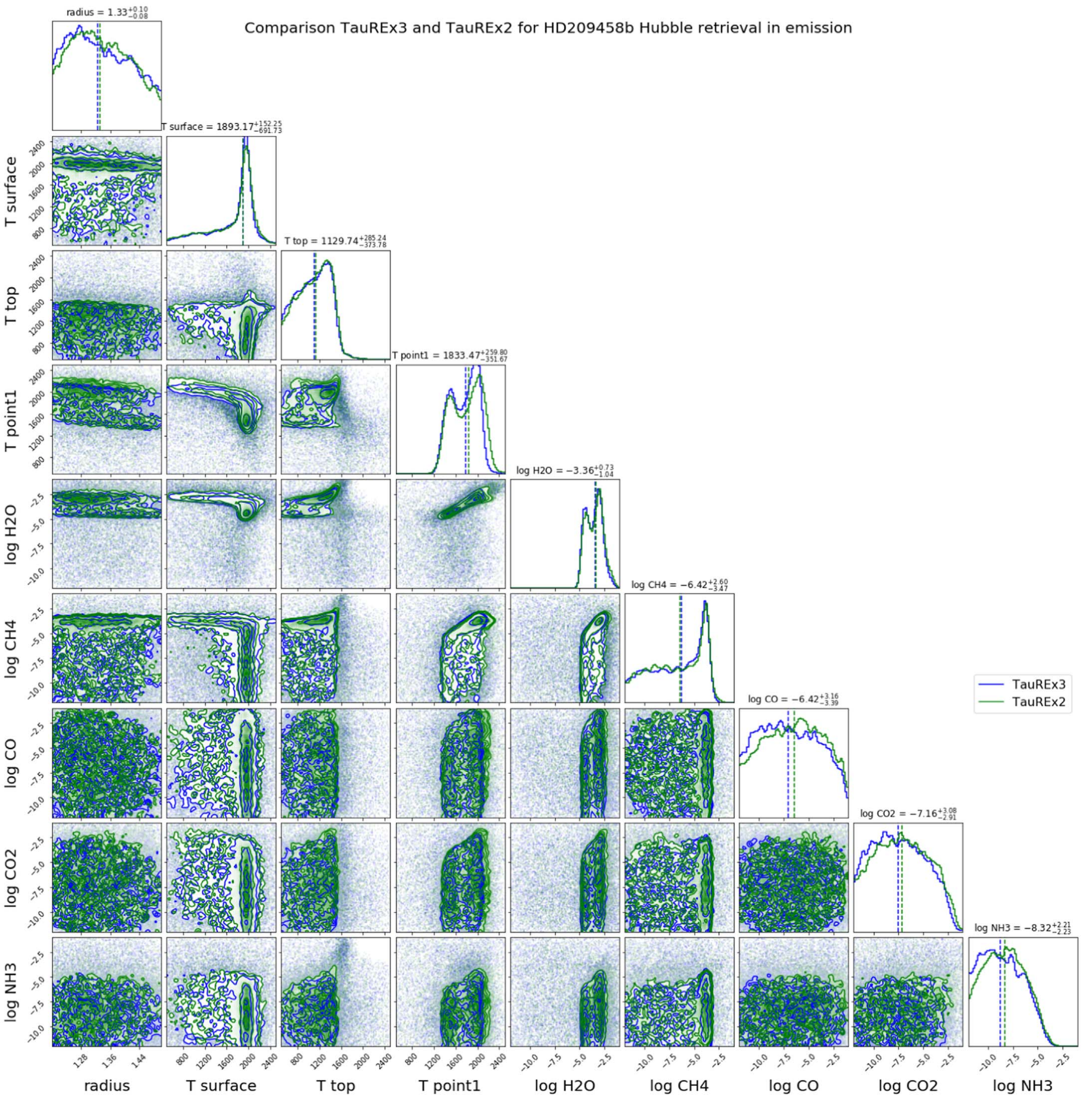


Figure 11. Posterior distribution for the HD 209458b retrievals of the emission spectrum from Line et al. (2016) with TauREx 2 and TauREx 3. The temperatures are in kelvin, the pressures in pascals, the radii in Jupiter radii, and the molecule abundance in volume mixing ratios. The dashed lines are the median values of the posterior. The values quoted above are the median from TauREx 2 with 16% and 84% quantiles relative to it.

1.5×10^{-1} bar, and 2×10^{-3} bar). We initially consider retrieving the pressure for these temperature points, but we find large degeneracies and decide to keep them fixed. The priors for the planet radius are kept narrow as they have some degree of degeneracy with temperature in the eclipse case due to the smaller wavelength coverage. In this scenario, preliminary knowledge (e.g., the value obtained in the transit case) constrains the radius bounds as these observations are more sensitive to this parameter. By constraining the bounds, we can break the degeneracy and benchmark the retrieval of the *N*-point temperature profile against TauREx 2.

For all these parameters, we use uniform priors, which are listed in Table 10.

The planet mass and star radius are fixed to the literature values, respectively 0.73 M_J and 1.19 $R_\odot$ from Stassun et al. (2017). On top of the five mentioned chemical species, we fill the rest of the atmosphere with hydrogen and helium at a ratio $H_2/He = 0.17$. Rayleigh scattering is calculated for all molecules (Cox 2015),

Table 12
Retrieved Parameters and Their Associated Uncertainties for Both Emission and Transmission of the HD 209458b HST/WFC3 Data for Both TauREx 2 and TauREx 3

	Transmission		Emission	
Parameter	TauREx 2	TauREx 3	TauREx 2	TauREx 3
log(H_2O)	$-3.52^{+1.15}_{-1.16}$	$-3.57^{+1.20}_{-1.21}$	$-3.35^{+1.03}_{-0.72}$	$-3.47^{+0.75}_{-0.98}$
log(CH_4)	$-7.97^{+2.61}_{-2.49}$	$-7.87^{+2.52}_{-2.59}$	$-6.42^{+2.60}_{-2.47}$	$-6.41^{+2.48}_{-3.64}$
log(CO)	$-7.41^{+3.10}_{-2.79}$	$-7.66^{+3.22}_{-2.70}$	$-6.42^{+3.16}_{-3.39}$	$-6.88^{+3.54}_{-3.32}$
log(CO_2)	$-7.95^{+2.74}_{-2.52}$	$-7.76^{+2.73}_{-2.60}$	$-7.15^{+3.07}_{-2.90}$	$-7.39^{+3.03}_{-2.95}$
log(NH_3)	$-8.43^{+2.18}_{-2.15}$	$-8.47^{+2.19}_{-2.24}$	$-8.31^{+2.21}_{-2.23}$	$-8.46^{+2.26}_{-2.16}$
$T_{\rm isothermal}$ (K)	$902.40^{+312.22}_{-283.98}$	$885.88^{+352.88}_{-262.00}$	...	...
$T_{\rm surface}$ (K)	...	...	$1893.13^{+152.27}_{-283.98}$	$1904.30^{+128.93}_{-606.58}$
$T_{\rm point1}$ (K)	...	...	$1833.35^{+259.90}_{-351.62}$	$1773.91^{+232.89}_{-304.51}$
$T_{\rm top}$ (K)	...	...	$1129.62^{+285.30}_{-373.69}$	$1129.56^{+286.15}_{-382.78}$
radius (R_J)	$1.39^{+0.01}_{-0.02}$	$1.39^{+0.01}_{-0.02}$	$1.33^{+0.09}_{-0.08}$	$1.33^{+0.09}_{-0.08}$
log(cloud pressure) (Pa)	$2.97^{+1.13}_{-1.34}$	$3.01^{+1.28}_{-1.27}$	...	...
μ (derived)	$2.29^{+0.09}_{-0.01}$	$2.31^{+0.07}_{-0.01}$	$2.30^{+0.13}_{-0.01}$	$2.31^{+0.09}_{-0.01}$

Note. μ is defined as the mean molecular weight of the atmosphere at the surface in atomic mass units. The Jupiter radius (R_J) is defined as 6.9911×10^7 m.

Table 13
A Comparison of the Retrieval Model Computation Time between TauREx 2 and TauREx 3 Using Cross Sections for the Same Atmospheric Priors Computed on Ariel Simulated Spectra

Time	TauREx 2 (s)	TauREx 3 (s)	No. of Samples	Speedup (x)
Transit	42,145	6885	180,000	6.2
Eclipse	72,607	10,559	150,000	6.87

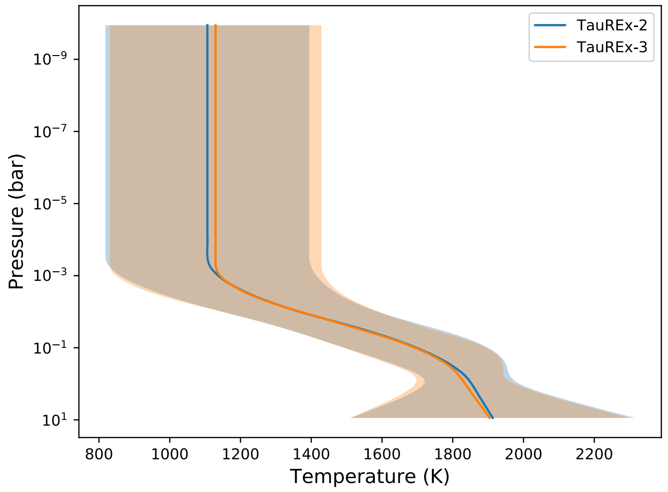
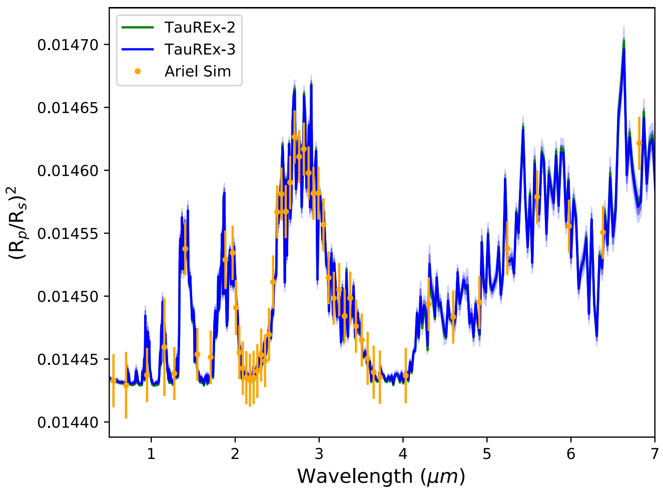


Figure 12. Temperature profiles of dayside emission retrievals of HD 209458b HST/WFC3 spectra (Line et al. 2016) by TauREx 2 and TauREx 3. Solid lines denote the best-fit values. Shaded areas denote the 1σ span.

while we limit CIA to the pairs H_2–H_2 and H_2–He. Finally, our model is computed in a grid of 100 layers with pressure ranging from 10 bar at the surface to 10^{-10} bar at the top of the considered atmosphere. Figure 9 shows the best-fit spectra for TauREx 2 and TauREx 3 in the transmission and emission cases.

The posterior distributions for the transmission and emission cases are displayed in Figures 10 and 11. Retrieval times from Table 11 show a 4.5 – 7× speed up in sampling using TauREx 3 compared to TauREx 2.

From the spectra, the posterior distributions, and the retrieved parameters in Table 12, one can see that the results given by



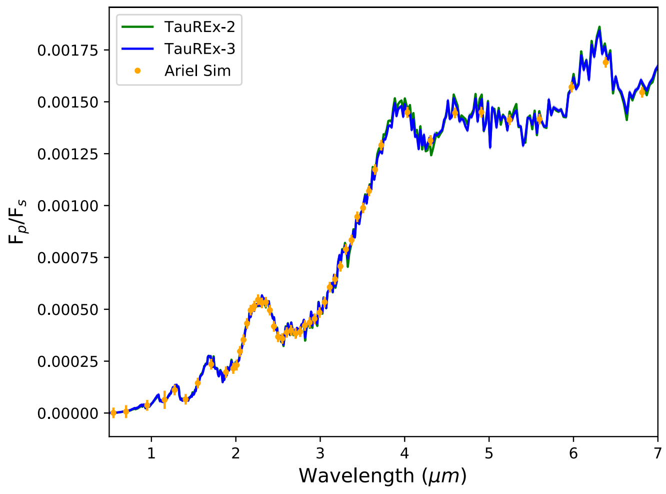


Figure 13. Best-fit spectra for the HD 209458b retrievals with TauREx 2 and TauREx 3. The 1σ and 2σ spectra are also plotted for the TauREx 3 retrieval. Top: Retrieval of the transmission spectrum from the simulated Ariel spectrum. Bottom: Retrieval of the emission spectrum from the simulated Ariel spectrum. The simulated spectrum for each plot is generated from the best-fit values of TauREx 2 from Table 12 with the noise simulated by ArielRad (Mugnai et al. 2020).

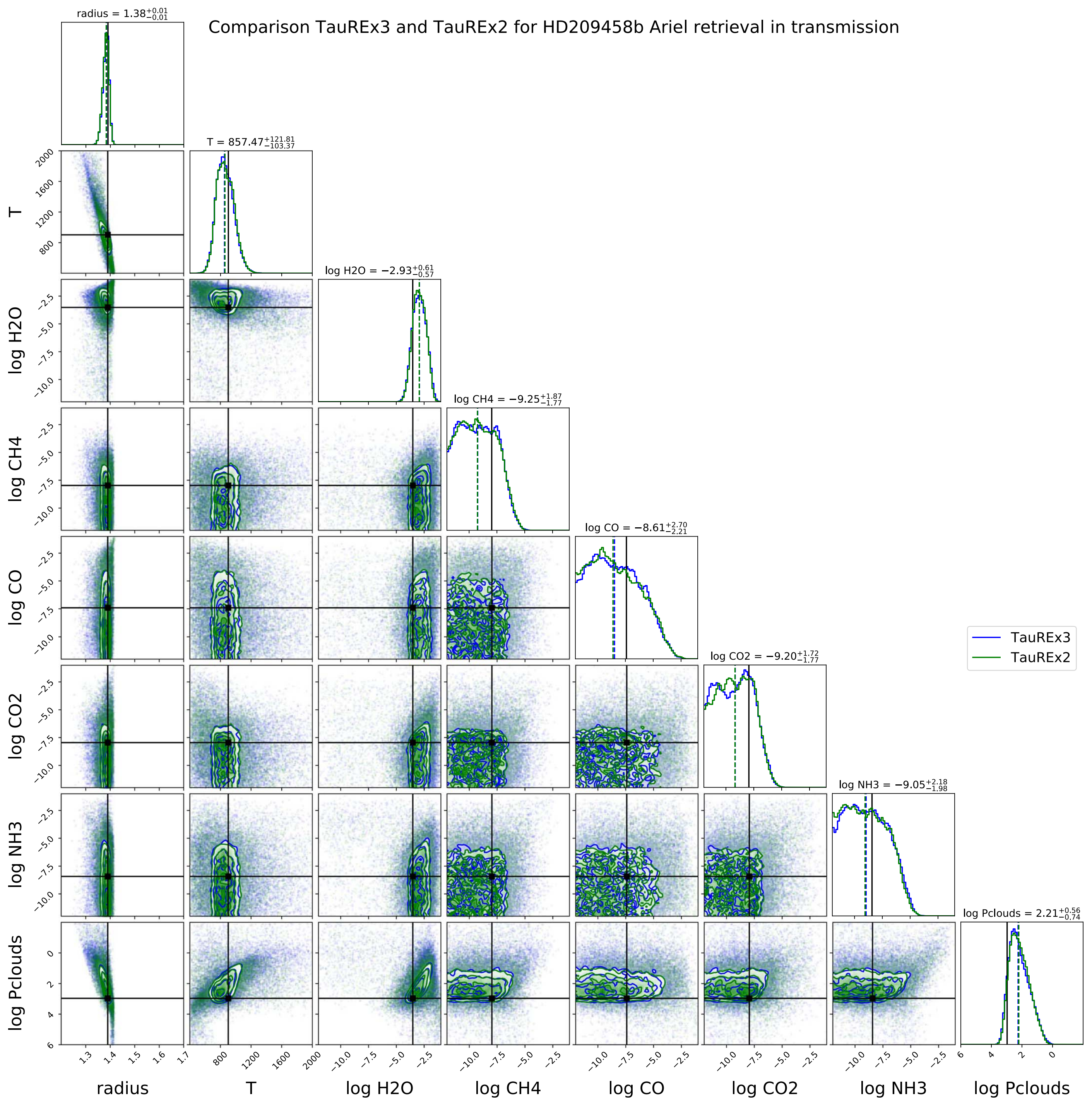


Figure 14. Posterior distribution for the HD 209458b retrievals of the transmission spectrum on the HD 209458b simulated Ariel observation with TauREx 2 and TauREx 3. The simulated spectrum uses the best-fit values of TauREx 2 from Table 12 (black solid lines). The dashed lines are the median values of the posterior. The values quoted above are the median from TauREx 2 with 16% and 84% quantiles relative to it.

TauREx 2 and TauREx 3 are almost equivalent. The best-fit spectra are within 1σ and the posterior distributions present the same shapes. Examining the parameters, we also see that all lie within 1σ of each other's best-fit values with variations arising only from the random sampling.

The two retrievals are also consistent with the main literature results. We retrieve H_2O in the dayside and terminator of the planet, respectively $\log(H_2O) = 3.36$ and $\log(H_2O) = 3.52$ in mixing ratios. In the terminator, we do not find significant evidence of additional molecules, but we retrieve a cloud pressure of about 10^{-2} bar. On the dayside, however, we find that the posterior distribution for CH_4 peaks around 10^{-4}. This molecule seems to be degenerated with the temperature, which presents a double-peak correlation. This result contrasts with both of the findings from Line et al. (2016), who found evidence for CO.

When conducting a retrieval using the two-stream approximation temperature profile described, e.g., in Parmentier & Guillot (2014), we find similar constraints on H_2O but no constraints on CO, CO_2, or CH_4. We speculate that the

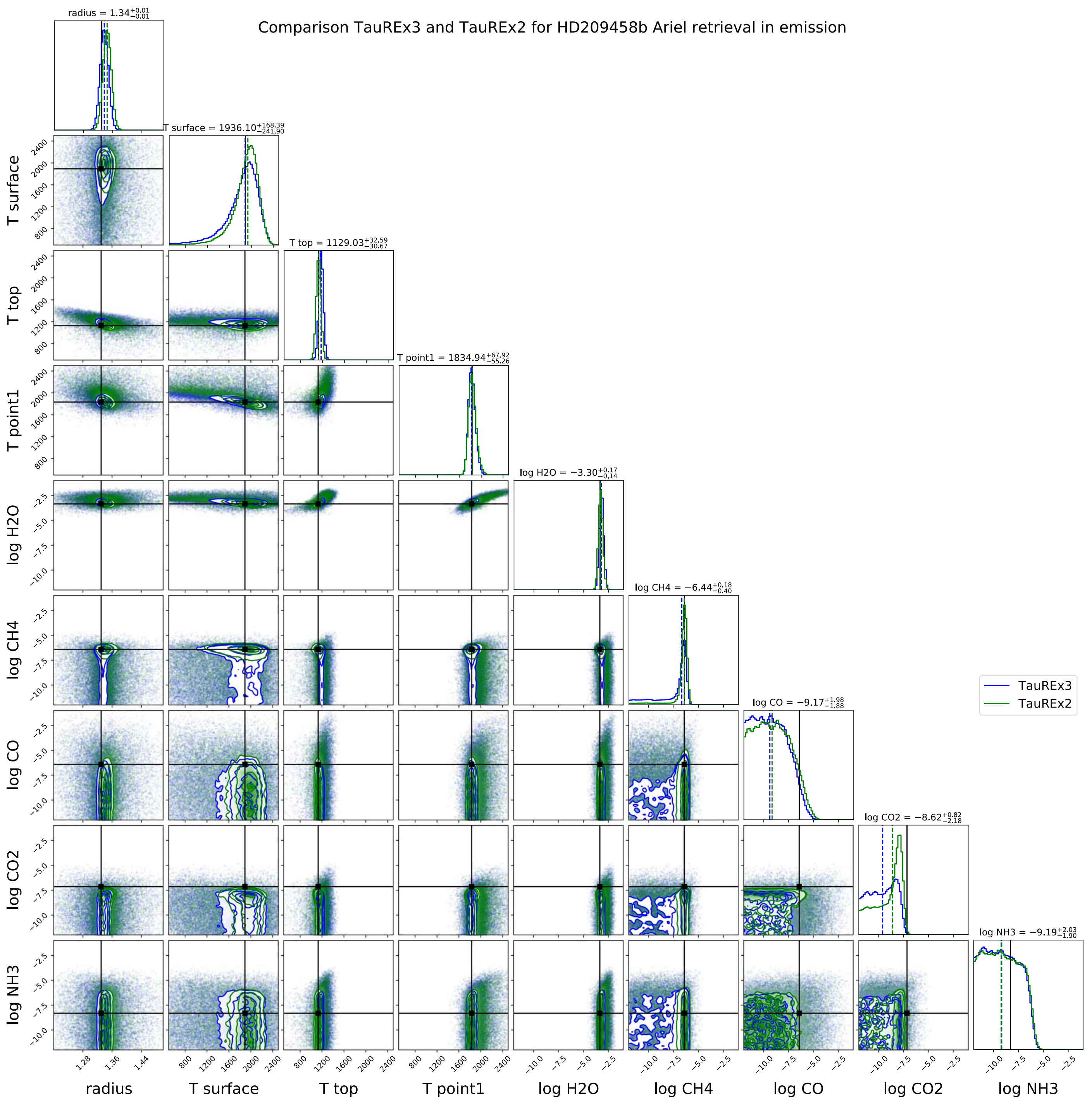


Figure 15. Posterior distribution for the HD 209458b retrievals of the emission spectrum on the HD 209458b simulated Ariel observation with TauREx 2 and TauREx 3. The simulated spectrum uses the best-fit values of TauREx 2 from Table 12 (black solid lines). The dashed lines are the median values of the posterior. The values quoted above are the median from TauREx 2 with 16% and 84% quantiles relative to the median.

differences from the results of Line et al. (2016) come from their including Spitzer data in their analysis, which are more sensitive to carbon-bearing species. Assessing the dayside temperature profile (Figure 12) both retrievals give similar profiles and uncertainties with differences of about 0.1–60 K between temperature points.

For the second retrieval benchmark, we will use the TauREx 2 best-fit values in Table 12 and simulate a spectrum as seen from Ariel using ArielRad (Mugnai et al. 2020). Our wavelength coverage ranges from 0.5 to 7.8 μm, effectively increasing the number of wavelength bins sixfold as compared to the HST/WFC3 case. The increase in information will impact the computational cost in calculating the forward model at each iteration and the number of samples required to achieve adequate convergence during retrievals.

Comparing the retrievals in Table 13, we indeed see a significant increase in the number of samples required and the time taken to complete. Highlighted is the significantly reduced retrieval time with TauREx 3 for both transit and eclipse times requiring only 2 and 3 hr, respectively, as compared to the

TauREx 2 retrievals taking 11 and 20 hr, respectively. This does not include the fact that the original code takes almost 30 minutes to set up before starting a retrieval compared to seconds for the current version. Scaling up to Ariel resolution introduces a sixfold increase in single sampling times for TauREx 3, which is in line with the increase in resolution and in contrast to the 20× increase in runtime for TauREx 2.

Examining the spectra in Figure 13 we see that both are essentially identical. This result is expected since the greater resolution and lower S/N increase the information available to the retrieval and greatly help to break degeneracies. This behavior is evident in the posterior distributions given in Figures 14 and 15, where most parameters are well constrained and within 1σ of the truth values. The emission posteriors for the radius, T_{point1} and T_{top}, are now well defined. As an aside, this highlights how dedicated missions such as Ariel present a significant improvement to our ability to resolve spectral features in exoplanets. Some of the truth values, namely P_{clouds}, log(CO), and log(CO_2), lie outside the 1σ retrieved values. This result is expected as the retrieval traces the information content and degeneracies. For example, CO does not produce visible features in our simulated spectrum, meaning the retrieval only recovers an upper limit. The calculated mean and 1σ for these parameters become prior-dependent. We refer the interested reader to literature discussing retrieval correlation in further detail (e.g., Feng et al. 2016; Rocchetto et al. 2016; Changeat et al. 2019, 2020a, 2020b).

In summary, benchmarking shows the results of TauREx 3 are consistent with those of TauREx 2 (which was benchmarked against both NEMESIS and CHIMERA; Barstow et al. 2020), while showing a sixfold improvement in retrieval runtime. This result demonstrates that retrievals of higher-resolution spectra from missions such as JWST and Ariel are feasible within a couple of hours.

9. Future Work

The flexibility afforded in TauREx 3 will allow for a wide range of novel applications to be developed on top of its core functionality. In future publications, currently in preparation, we will present new applications using this library. The list includes a better treatment of scattering processes with two-stream (Goody & Yung 1989) and multistream approximations (Laszlo et al. 2016) for the emission model, a new forward model generation and retrieval pipeline for large-scale studies of planetary populations for next-generation telescopes (Changeat et al. 2020a), applications to solar system bodies with solar occultation (Vago et al. 2015), nadir models and integration of the Mars Climate Database chemistry model (Forget et al. 1999), and a new phase curve forward model for exoplanetary applications.

In general, the design of TauREx 3 aims to reduce significantly the time and effort for groups to include their own chemistry, cloud, forward, or temperature schemes. The framework is fully open-source, under a BSD license, and we hope to provide a community-wide tool for retrieval code development in the future.

10. Summary

In this publication, we present our new retrieval framework for TauREx. The code can act as a library providing ready-to-use functions for atmospheric modeling. These components can combine into a full pipeline for atmospheric retrievals. TauREx 3 is flexible, seamlessly utilizing new codes defined by the user. We have demonstrated its dynamic nature, responding to changes in the forward model and generating new and appropriate fitting parameters for retrievals. TauREx 3 is designed to adapt to the rapid development of atmospheric theory and include cutting-edge methods with minimal effort. It allows for the rapid prototyping of new methods and retrieval regimes. This includes more complex retrievals such as observation geometry and model hyperparameters. Our benchmarks demonstrate significant improvement in performance at high resolution ($R = 10{,}000$), which reaches 10× the level of the previous version at large wavelength ranges. Retrieval times are significantly reduced with simulated Ariel retrievals being completed in a couple of hours. Our benchmarks also demonstrate the code's robustness and show that the results of TauREx 3 match precisely those of the previous version. The code is open-source, licensed under a BSD license, and available at Github[3] and through the Python Package Index.[4]

This project has received funding from the European Research Council under the European Union's Horizon 2020 research and innovation program (grant agreement No. 758892, ExoAI) and the European Union's Horizon 2020 COMPET program (grant agreement No. 776403, ExoplANETS A). Furthermore, we acknowledge funding by the UK Space Agency and Science and Technology Funding Council grants ST/K502406/1, ST/P000282/1, ST/P002153/1, ST/S002634/1, ST/T001836/1, and ST/V003380/1.

ORCID iDs

A. F. Al-Refaie https://orcid.org/0000-0003-2241-5330
Q. Changeat https://orcid.org/0000-0001-6516-4493
I. P. Waldmann https://orcid.org/0000-0002-4205-5267
G. Tinetti https://orcid.org/0000-0001-6058-6654

References

Abel, M., Frommhold, L., Li, X., & Hunt, K. L. 2011, JPCA, 115, 6805
Abel, M., Frommhold, L., Li, X., & Hunt, K. L. 2012, JChPh, 136, 044319
Agúndez, M., Venot, O., Iro, N., et al. 2012, A&A, 548, A73
Barbary, K. 2015, Nestle Sampling Library, https://github.com/kbarbary/nestle
Barstow, J. K., Changeat, Q., Garland, R., et al. 2020, MNRAS, 493, 4884
Barton, E. J., Hill, C., Yurchenko, S. N., et al. 2017, JQSRT, 187, 453
Bean, J. L., Stevenson, K. B., Batalha, N. M., et al. 2018, PASP, 130, 114402
Benneke, B. 2015, arXiv:1504.07655
Bohren, C. F., & Huffman, D. R. 2007, Appendix A: Homogeneous Sphere (Hoboken, NJ: John Wiley & Sons, Ltd), 477
Brogi, M., & Line, M. R. 2019, AJ, 157, 114
Buchner, J., Georgakakis, A., Nandra, K., et al. 2014, A&A, 564, A125
Changeat, Q., Al-Refaie, A., Mugnai, L. V., et al. 2020a, AJ, 160, 80
Changeat, Q., Edwards, B., Waldmann, I., & Tinetti, G. 2019, ApJ, 886, 39
Changeat, Q., Keyte, L., Waldmann, I. P., & Tinetti, G. 2020b, ApJ, 896, 107
Chubb, K. L., Min, M., Kawashima, Y., Helling, C., & Waldmann, I. 2020, A&A, 639, A3
Chubb, K. L., Rocchetto, M., Yurchenko, S. N., et al. 2021, A&A, 646, A21
Cox, A. N. 2015, Allen's Astrophysical Quantities (Berlin: Springer)
Cubillos, P. E., & Blecic, J. 2021, MNRAS, 505, 2675
Edwards, B., Changeat, Q., Mori, M., et al. 2021, AJ, 161, 44
Evans, T. M., Sing, D. K., Kataria, T., et al. 2017, Natur, 548, 58
Feng, Y. K., Line, M. R., Fortney, J. J., et al. 2016, ApJ, 829, 52
Feroz, F., Hobson, M. P., & Bridges, M. 2009, MNRAS, 398, 1601
Fletcher, L. N., Gustafsson, M., & Orton, G. S. 2018, ApJS, 235, 24

[3] http://github.com/ucl-exoplanets/TauREx3_public
[4] https://pypi.org/project/taurex/

Forget, F., Hourdin, F., Fournier, R., et al. 1999, JGR, 104, 24155
Gandhi, S., & Madhusudhan, N. 2018, MNRAS, 474, 271
Gardner, J. P., Mather, J. C., Clampin, M., et al. 2006, SSRv, 123, 485
Goody, R., & Yung, Y. 1989, Atmospheric Radiation: Theoretical Basis (New York/Oxford: Oxford University Press)
Gordon, I., Rothman, L., Hill, C., et al. 2017, JQSRT, 203, 3
Guillot, T. 2010, A&A, 520, A27
Handley, W. J., Hobson, M. P., & Lasenby, A. N. 2015, MNRAS, 453, 4384
Harrington, J. 2021, arXiv:2104.12522
Higson, E. 2018, JOSS, 3, 916
Higson, E., Handley, W., Hobson, M., & Lasenby, A. 2019, S&C, 29, 891
Hill, C., Yurchenko, S. N., & Tennyson, J. 2013, Icar, 226, 1673
Hou Yip, K., Waldmann, I. P., Tsiaras, A., & Tinetti, G. 2020, AJ, 160, 171
Irwin, P. G. J., Teanby, N. A., de Kok, R., et al. 2008, J. Quant. Spec. Radiat. Transf., 109, 1136
Jolliffe, I. T. 2007, Principal Component Analysis (2nd edn.; New York: Springer Verlag)
Karman, T., Gordon, I. E., van der Avoird, A., et al. 2019, Icar, 328, 160
Kempton, E. M.-R., Lupu, R., Owusu-Asare, A., Slough, P., & Cale, B. 2017, PASP, 129, 044402
Kitzmann, D., Heng, K., Oreshenko, M., et al. 2020, ApJ, 890, 174
Kreidberg, L., Bean, J. L., Désert, J.-M., et al. 2014, ApJ, 793, L27
Lam, S. K., Pitrou, A., & Seibert, S. 2015, in Proc. of the 2nd Workshop on the LLVM Compiler Infrastructure in HPC, LLVM '15, Ser. 7 (New York: ACM), 1
Laszlo, I., Stamnes, K., Wiscombe, W., & Tsay, S. 2016, The Discrete Ordinate Algorithm, DISORT for Radiative Transfer (Berlin: Springer Praxis Books)
Lavie, B., Mendonça, J. M., Mordasini, C., et al. 2017, AJ, 154, 91
Lee, J.-M., Heng, K., & Irwin, P. G. J. 2013, ApJ, 778, 97
Li, G., Gordon, I. E., Rothman, L. S., et al. 2015, ApJS, 216, 15
Line, M. R., Stevenson, K. B., Bean, J., et al. 2016, AJ, 152, 203
Line, M. R., Wolf, A. S., Zhang, X., et al. 2013, ApJ, 775, 137
MacDonald, R. J., & Madhusudhan, N. 2017, MNRAS, 469, 1979
Madhusudhan, N., & Seager, S. 2009, ApJ, 707, 24
McLeod, R., Alted, F., Valentino, A., et al. 2018, Pydata/numexpr: NumExpr v2.6.9, Zenodo, doi:10.5281/zenodo.2483274
Meurer, A., Smith, C. P., Paprocki, M., et al. 2017, PeerJ Comp. Sci., 3, e103
Mollière, P., Wardenier, J. P., van Boekel, R., et al. 2019, A&A, 627, A67
Mugnai, L. V., Pascale, E., Edwards, B., Papageorgiou, A., & Sarkar, S. 2020, ExA, 50, 303
Nikolov, N., Sing, D. K., Fortney, J. J., et al. 2018, Natur, 557, 526
Ormel, C. W., & Min, M. 2019, A&A, 622, A121
Parmentier, V., & Guillot, T. 2014, A&A, 562, A133
Perez, F., & Granger, B. E. 2007, CSE, 9, 21
Polyansky, O. L., Kyuberis, A. A., Zobov, N. F., et al. 2018, MNRAS, 480, 2597
Richard, C., Gordon, I., Rothman, L., et al. 2012, JQSRT, 113, 1276
Rocchetto, M., Waldmann, I. P., Venot, O., Lagage, P.-O., & Tinetti, G. 2016, ApJ, 833, 120
Rodgers, C. D. 2000, Inverse Methods for Atmospheric Sounding—Theory and Practice (Singapore: World Scientific)
Rothman, L., Gordon, I., Babikov, Y., et al. 2013, JQSRT, 130, 4
Rothman, L., Gordon, I., Barber, R., et al. 2010, JQSRT, 111, 2139
Sedaghati, E., Boffin, H. M. J., MacDonald, R. J., et al. 2017, Natur, 549, 238
Sheppard, K. B., Mandell, A. M., Tamburo, P., et al. 2017, ApJ, 850, L32
Sing, D. K., Fortney, J. J., Nikolov, N., et al. 2015, Natur, 529, 59
Skilling, J. 2012, in AIP Conf. Proc. 1443, Bayesian Inference and Maximum Entropy Methods in Science and Engineering, ed. P. Goyal et al., 145
Stassun, K. G., Collins, K. A., & Gaudi, B. S. 2017, AJ, 153, 136
Stevenson, K. B., Desert, J.-M., Line, M. R., et al. 2014, Sci, 346, 838
Stevenson, K. B., Line, M. R., Bean, J. L., et al. 2017, AJ, 153, 68
Swain, M. R., Line, M. R., & Deroo, P. 2014, ApJ, 784, 133
Swain, M. R., Tinetti, G., Vasisht, G., et al. 2009, ApJ, 704, 1616
Tennyson, J., & Yurchenko, S. N. 2012, MNRAS, 425, 21
Tennyson, J., Yurchenko, S. N., Al-Refaie, A. F., et al. 2016, JMoSp, 327, 73
Tinetti, G., Drossart, P., Eccleston, P., et al. 2018, ExA, 46, 135
Tsiaras, A., Rocchetto, M., Waldmann, I. P., et al. 2016, ApJ, 820, 99
Tsiaras, A., Waldmann, I. P., Tinetti, G., Tennyson, J., & Yurchenko, S. N. 2019, NatAs, 3, 1086
Tsiaras, A., Waldmann, I. P., Zingales, T., et al. 2018, AJ, 155, 156
Vago, J., Witasse, O., Svedhem, H., et al. 2015, SoSyR, 49, 518
van der Walt, S., Colbert, S. C., & Varoquaux, G. 2011, CSE, 13, 22
Venot, O., Drummond, B., Miguel, Y., et al. 2018, ExA, 46, 101
Venot, O., Hébrard, E., Agúndez, M., et al. 2012, A&A, 546, A43
Venot, O., Parmentier, V., Blecic, J., et al. 2020, ApJ, 890, 176
Waldmann, I. P., Rocchetto, M., Tinetti, G., et al. 2015a, ApJ, 813, 13
Waldmann, I. P., Tinetti, G., Rocchetto, M., et al. 2015b, ApJ, 802, 107
Yurchenko, S. N., Al-Refaie, A. F., & Tennyson, J. 2018, A&A, 614, A131
Yurchenko, S. N., Barber, R. J., & Tennyson, J. 2011, MNRAS, 413, 1828
Yurchenko, S. N., & Tennyson, J. 2014, MNRAS, 440, 1649
Zhang, M., Chachan, Y., Kempton, E. M. R., & Knutson, H. A. 2019, PASP, 131, 034501