The Astrophysical Journal, 932:123 (24pp), 2022 June 20 https://doi.org/10.3847/1538-4357/ac6dcd

A Comparison of Chemical Models of Exoplanet Atmospheres Enabled by TauREx 3.1

A. F. Al-Refaie[1], Q. Changeat[1], O. Venot[2], I. P. Waldmann[1], and G. Tinetti[1]

[1] Dept. Physics & Astronomy, University College London, Gower Street, London WC1E 6BT, UK; ahmed.al-refaie.12@ucl.ac.uk

[2] Université de Paris and Univ Paris Est Creteil, CNRS, LISA, F-75013 Paris, France

Received 2021 August 19; revised 2022 April 25; accepted 2022 May 5; published 2022 June 23

Abstract

Thermochemical equilibrium is one of the most commonly used assumptions in current exoplanet retrievals. As science operations with the James Webb Space Telescope (JWST) draw near and with the planned launch of Ariel, it is crucial to assess the underlying biases and assumptions made when applying self-consistent chemistry to spectral retrievals. Here we use the flexibility of TauREx 3.1 to cross-compare three state-of-the-art chemical equilibrium codes: ACE, FastChem, and GGchem. We simulate JWST spectra for ACE, FastChem, GGchem, and GGchem+condensation containing only the elements C, H, O, and N and spectra for FastChem, GGchem, and GGchem+condensation with a more extensive range of elements, giving seven simulated JWST spectra in total, and then cross-retrieve, giving a total of 56 retrievals. Our analysis demonstrates that, like-for-like, all chemical codes retrieve the correct parameters to within 1% of the truth. However, in retrievals, where the contained elements do not match the truth, parameters such as metallicity deviate by 20% while maintaining extremely low uncertainties <1%, giving false confidence. This point is of major importance for future analyses on JWST and Ariel, highlighting that self-consistent chemical schemes that do not employ the proper assumptions (missing species, fixed elemental ratios, condensation) are at risk of confidently biasing interpretations. Free chemistry retrievals employing parametric descriptions of the chemical profiles can provide alternative unbiased explorations.

Unified Astronomy Thesaurus concepts: Open source software (1866); Publicly available software (1864); Chemical abundances (224); Bayesian statistics (1900); Exoplanet atmospheres (487); Exoplanet astronomy (486); Exoplanet atmospheric composition (2021); Exoplanets (498); Radiative transfer (1335)

1. Introduction

In the past 15 years an increasing number of exoplanetary atmospheres have been observed using space- and ground-based facilities (Charbonneau et al. 2002; Grillmair et al. 2008; Deming et al. 2013; Sing et al. 2016; Tsiaras et al. 2018; Pinhas et al. 2019; Welbanks et al. 2019). Current data are often sparse and with low signal-to-noise ratio, leading to degeneracies that require the use of sophisticated modeling tools to be interpreted correctly. Despite these limitations, the atmospheres of large ultrahot Jupiters (Haynes et al. 2015; Mikal-Evans et al. 2019, 2020; Edwards et al. 2020; Gandhi et al. 2020; Changeat & Edwards 2021), hot Jupiters (Tinetti et al. 2007; Swain et al. 2008, 2009a, 2009b; Line et al. 2014; Stevenson et al. 2014, 2017; MacDonald & Madhusudhan 2017; Changeat et al. 2020b, 2021; Pluriel et al. 2020), and hot Saturns (Nikolov et al. 2018; Skaf et al. 2020; Anisman et al. 2020; Yip et al. 2021) are regularly characterized using the transit, eclipse, and phase-curve techniques. Mini-Neptunes and super-Earths are notoriously more challenging, but the Hubble Space Telescope has allowed us to infer some constraints for a few of these worlds (Kreidberg et al. 2014; de Wit et al. 2016, 2018; Tsiaras et al. 2016, 2019; Benneke et al. 2019; Madhusudhan et al. 2020; Guilluy et al. 2021; Edwards et al. 2021; Mugnai et al. 2021; Swain et al. 2021). Successful characterization of atmospheres must take into account many processes outside of radiative transfer. Incorporating phenomena such as chemical reactions (Lodders & Fegley 2002; Moses et al. 2011), fluid dynamics (Cho et al. 2008; Showman et al. 2009; Skinner & Cho 2021),clouds (Marley et al. 2013), and nuclear motion (Tennyson & Yurchenko 2012; Tennyson et al. 2016) is paramount to understanding these extrasolar objects. An influx of spectral retrieval models that solve the inverse problem has left modellers spoilt for choice. Retrieval codes include: Madhusudhan & Seager (2009), Lee et al. (2012), TauRex 3 (Al-Refaie et al. 2021), NEMESIS (Irwin et al. 2008), CHIMERA (Line et al. 2013), ARCiS (Ormel & Min 2019; Min et al. 2020), BART (Harrington 2016), petitRADTRANS (Mollière et al. 2019), HELIOS (Kitzmann et al. 2020; Lavie et al. 2017), POSEIDON (MacDonald & Madhusudhan 2017), HyDRA (Gandhi & Madhusudhan 2018), SCARLET (Benneke 2015), PLATON II (Zhang et al. 2019), and Pyrat-Bay (Cubillos & Blecic 2021). Ideally, a retrieval code would aim to infer the abundance profiles of the atmospheric species. Sampling molecular abundances at each layer of the atmosphere exponentially inflates the sampling dimension to hundreds of parameters for each molecule. Pursuing this method of abundance retrieval will be riddled with parameter degeneracy and huge computation time. To address this issue, the molecular abundances are often parameterized through either heuristic or self-consistent means.

Heuristic parameterization extracts abundance profiles directly from the data (Madhusudhan & Seager 2009). These types of parameterizations usually make straightforward assumptions about the chemistry in the atmosphere. An example is isoabundance profiles, where the molecular mixing ratios are invariant for all atmospheric depths. Profiles such as isoabundance help study the limits of what species can be identified and constrain their abundances based on the shape and uncertainty of the input spectrum. They are also advantageous because they represent species using a few parameters, reducing the sampling space. Furthermore, they are fast computationally. The dimensionality of the problem generally grows linearly with the number of molecules.

Self-consistent parameterization (Line et al. 2014) relies on an underlying chemical model to produce molecular abundances. Such models significantly reduce the number of parameters required to generate abundance profiles and have a physical basis behind them. These models can be slower than heuristic, requiring some simulation of physical processes to generate the final chemical profile. The majority of these self-consistent models follow the same basic principle, i.e., they set the initial abundance of elements in the atmosphere and iterate until convergence. The approach taken by each model can differ significantly, including the thermochemical data used, the iteration scheme, and the chemical network employed.

One of the most commonly used self-consistent chemical models is the *thermochemical equilibrium* approximation (White et al. 1958; Eriksson et al. 1971). This approximation assumes chemical reaction timescales are faster than the dynamical timescale and ignores photochemical processes. Chemical reaction rates are generally proportional to temperature and pressure, so the approximation holds in high-temperature, high-pressure environments such as those found in stars and ultrahot Jupiters. For nonequilibrium or "disequilibrium" environments (Allen et al. 1981; Moses et al. 2011), the approximation is often used as the initial condition to facilitate faster convergence of chemical kinetics. Each retrieval code attempts to solve the problem differently: e.g., HELIOS exploits GPU acceleration and FastChem (Stock et al. 2018) chemistry for self-consistent retrievals using Multinest (Feroz et al. 2009; Buchner et al. 2014). PLATON utilizes interpolation of precomputed GGchem (Woitke et al. 2018) chemical tables to speed up retrievals using dynesty. petitRADTRANS takes a "no-frills" approach and only solves the radiative transfer problem, leaving it up to the user to provide the chemistry and sampler. TauREx 3 includes ACE (Agúndez et al. 2012, 2020) as the chemical equilibrium model but allows a degree of flexibility in incorporating new chemical codes, forward models, samplers, and retrieval parameters.

In its most general form, computing the chemical composition of an atmosphere in equilibrium requires minimizing the Gibbs free energy of the system given an initial atomic abundance. However, the nonlinear dependence of the Gibbs free energy on the number density of species in the system and linear constraints such as mass and charge conservation make the computation highly demanding and nontrivial. Various algorithms have been developed to solve for chemical equilibrium. In particular, exploiting the law of mass action (Berline & Bricker 1969) and thermochemical equilibrium constants K transforms the problem into a series of N unknowns by eliminating molecular number densities. The system is solvable using a root-finding algorithm such as the Newton–Raphson method. High temperatures ($T \geqslant 1000$ K) converge quickly and are numerically stable, but lower temperatures become more untenable as $K > 10^{1000}$, which cannot be represented in standard double-precision floating-point calculations (Woitke et al. 2018).

Equilibrium constants K for species i can be derived from the Gibbs free energy of formation:

$$\ln K_i = -\frac{\Delta G_f^\circ}{RT} \tag{1}$$

Table 1
Planetary and Parent Star Parameters from Stassun et al. (2017) Used to Generate the Simulated JWST Transit Spectra of HD 209458b

Parameter	Value
HD 209458	
R_s	1.19 $R_\odot$
T_s	6091.0 K
$K_{\rm mag}$	6.308
D_s	48.37 pc
Z_s	0.01 $Z_\odot$
M_s	1.23 $M_\odot$
HD 209458b	
R	1.39 R_J
M	0.73 M_J
Semimajor axis	0.0486 au
$t_{\rm period}$	3.5247 days
$t_{\rm transit}$	11037.18 s
T	1500 K
Z	$Z_\odot$
C/O	0.5

where ΔG_f° is the Gibbs free energy of formation, R is the universal gas constant, and T is the temperature. Sources of $\ln K_i$ include the NIST-JANAF database (Chase 1986) but must be fitted as a function of temperature before they can be used. Various fitting functions have been proposed in the literature including the form by Tsuji (1973):

$$\ln K_i = a_0 + a_1\theta + a_2\theta^2 + a_3\theta^3 + a_4\theta^4 \tag{2}$$

where a_n are fitting coefficients and $\theta = 5040/T$ where T is temperature in kelvin. Another form used is the NASA polynomials formalism given by McBride et al. (1993):

$$\begin{aligned} H_i &= a_0 + a_1\frac{T}{2} + a_2\frac{T^2}{3} + a_3\frac{T^3}{4} + a_4\frac{T^4}{5} + \frac{a_5}{T} \\ S_i &= a_0 \ln T + a_1 T + a_2\frac{T^2}{2} + a_3\frac{T^3}{3} + a_4\frac{T^4}{4} + a_6 \\ \ln K_i &= H_i - S_i \end{aligned} \tag{3}$$

where H_i and S_i are the enthalpy and entropy for species i respectively.

To summarize, building a chemical equilibrium code will require three choices: (1) the choice in an algorithm for minimizing the Gibbs free energy. In addition, if low-temperature applicability is required, the algorithm must also deal with numerical stability issues. (2) The source of the equilibrium constants K and (3) the choice of fitting function for ln K.

The underlying biases associated with the Gibbs minimization strategy, thermochemical data source, fitting function, and choice of elements and their initial abundances are also not well known and may significantly affect the direction of an atmospheric retrieval. Studying the effects of each chemical code on atmospheric retrieval is nontrivial. Each retrieval code is essentially married to a particular chemical model. Switching between different retrieval codes can introduce variability from the implementation of radiative transfer (such as k-table and cross-section opacities) and the choice of Bayesian sampler. TauREx 3.1 (Appendix A) remedies this issue through the introduction of a plugin system. This system allows for light

Table 2
List of Opacities used

Opacities	Type	Ref.
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)
K	Abs.	Allard et al. (2016)
Na	Abs.	Allard et al. (2019)
C_2H_2	Abs.	Chubb et al. (2020)
SO_2	Abs.	Underwood et al. (2016)
TiO	Abs.	McKemmish et al. (2019)
VO	Abs.	McKemmish et al. (2016)

Table 3
Cases Modeled for Each Chemical Code

Model	Cases			
	C/H/O/N	C/H/O/N + cond.	Heavy	Heavy + cond.
ACE	Yes	No	No	No
FastChem	Yes	No	Yes	No
GGchem	Yes	Yes	Yes	Yes

Note. The C/H/O/N cases limits the available elements to H, He, C, N, and O. The *Heavy* case includes the elements H, He, C, O, N, Mg, S, Ti, V, K, and Na. Cases with *cond.* introduce sequestration from condensation.

packages that can be built, distributed through PyPI and installed. These packages are then detected by TauREx and seamlessly "plugged in" to the framework (see Appendix B for more details). Plugins can extend and enhance every aspect of the framework, including but not limited to: GPU acceleration (Appendix B.1), forward models, opacities, prior functions, samplers, and, importantly, chemical models (Appendix C).

We exploit TauREx 3.1 and its plugin feature to compare and cross-validate three chemical codes: ACE (Agúndez et al. 2012, 2020), FastChem (Stock et al. 2018), and GGchem (Woitke et al. 2018) under the TauREx retrieval framework.

2. Methods

2.1. Description of Chemical Codes

2.1.1. ACE

The ACE chemical FORTRAN code (Agúndez et al. 2012, 2020) was developed with a primary focus on studying chemical compositions in exoplanet atmospheres and is the equilibrium model in the Ariel target retrieval study by Changeat et al. (2020a). ACE minimizes the Gibbs free energy using the algorithm outlined by Gordon & McBride (1994) and the included thermochemical data contain 105 neutral C/H/O/N-bearing chemical species sourced from Gordon & McBride (1994), Atkinson et al. (2006), and Bounaceur et al. (2010), and with some constants computed using THERGAS (Muller et al. 1995). The equilibrium constants are fit with the NASA polynomial formalism (Equation (3)) applicable to the 300–5000 K temperature range.

The ACE code supports a range of species and ions but its provided thermochemical data are heavily focused toward neutral

Table 4
Uniform Priors used by ACE, FastChem, and GGchem in the Retrieval

Parameter	Prior Bounds
R_p	[0.5–2.0] R_J
T	[500–2000] K
Z	[0.1–4.0] $Z_\odot$
C/O	[0.1–4.0]

Table 5
Uniform Priors Used by the Free Chemistry Retrieval

Parameter	Prior Bounds
R_p	[0.5–2.0] R_J
T	[500–2000] K
log H_2O	[−12–−2]
log CH_4	[−12–−2]
log NH_3	[−12–−2]
log C_2H_2	[−12–−2]
log CO	[−12–−2]
log CO_2	[−12–−2]
log TiO	[−12–−2]
log VO	[−12–−2]

Table 6
Average Retrieval Times for Each Chemical Code and Species Used

Code	Species	Retrieval Time (min)
ACE	C/H/O/N	4.40
FastChem	C/H/O/N	4.45
FastChem	heavy	10.55
GGchem	C/H/O/N	10.52
GGchem	C/H/O/N+cond.	52.2
GGchem	Heavy	30.2
GGchem	Heavy+cond.	540.43

Note. Benchmarks were conducted on the OzSTAR Supercomputing @ Swinbourne cluster on five nodes. Each node has two Intel Gold 6140 18-core processors and two nVidia P100 12 GB PCIe GPUs.

C/H/O/N species because it is commonly used in conjunction with chemical kinetic models by Venot et al. (2012, 2020). Many of the reactions and species in the respective networks derive a portion of their rates from combustion mechanisms that deal with C/H/O/N-bearing molecules at temperatures up to 3000 K and pressures up to 100 bar. Such conditions are similar to those found in ultrahot Jupiters; ACE provides almost twice as many C/H/O/N-bearing species as other thermochemical data sets such as Chase (1986).

2.1.2. FastChem

FastChem (Stock et al. 2018) is a C++ chemical equilibrium code from the Exoclimes simulation platform that implements a new semianalytical method to minimize the Gibbs energy. It decomposes the system into a set of singular variable-coupled nonlinear equations. FastChem also exploits quadruple precision and the Nelder–Mead optimization algorithm to converge quickly at temperatures as low as 100 K. Thermochemical data for the 396 neutral and 114 charged species are sourced from the NIST-JANAF database (Chase 1986) and fit the expression

$$\ln K = \frac{a_0}{T} + a_1 \ln T + a_2 + a_3 T + a_4 T^2. \qquad (4)$$

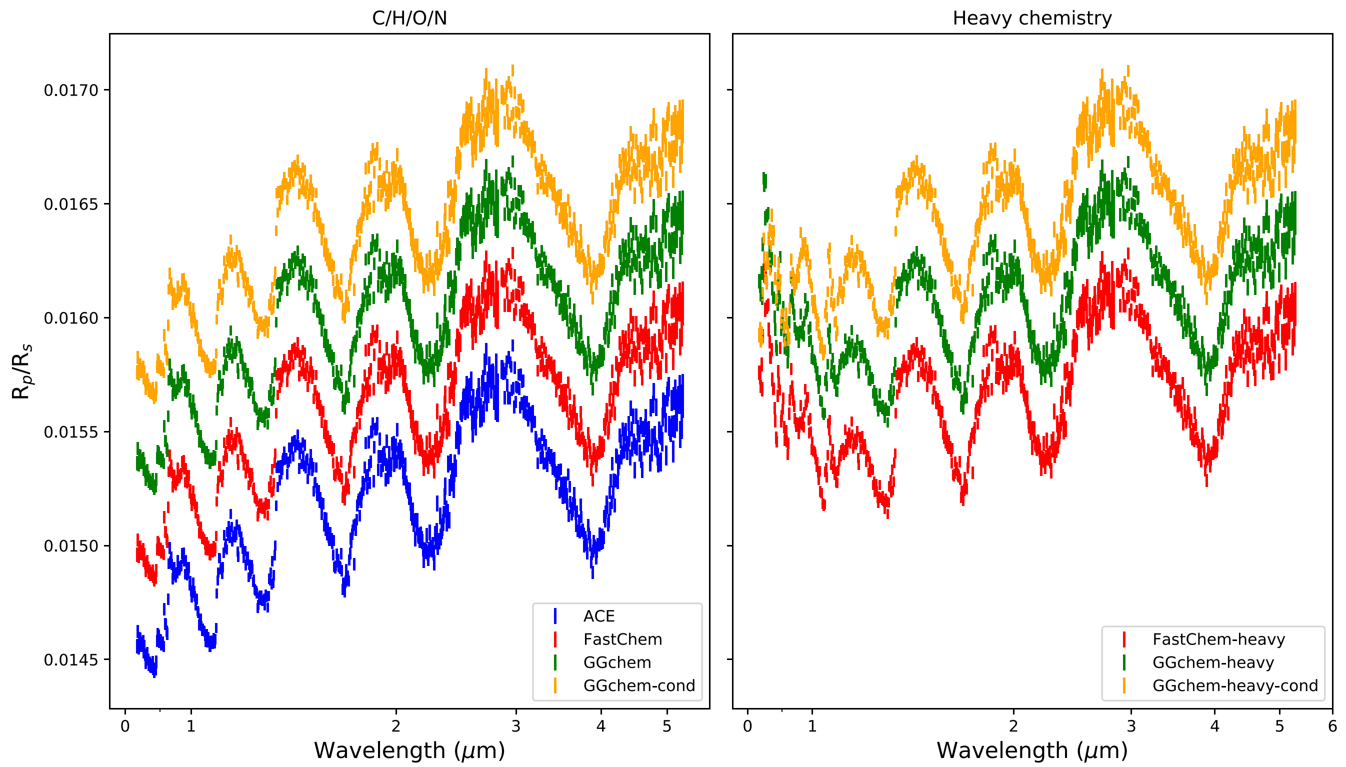

Figure 1. Plots of simulated JWST transit spectra from different chemistry models for the test planet given in Table 1. Each spectrum above ACE has been artificially offset for clarity. C/H/O/N chemistry restricts each equilibrium model to the elements H, He, C, N, and O. *Heavy chemistry* includes H, He, C, O, N, Mg, S, K, and Na and is suffixed with *heavy*. The *cond* suffix denotes condensation.

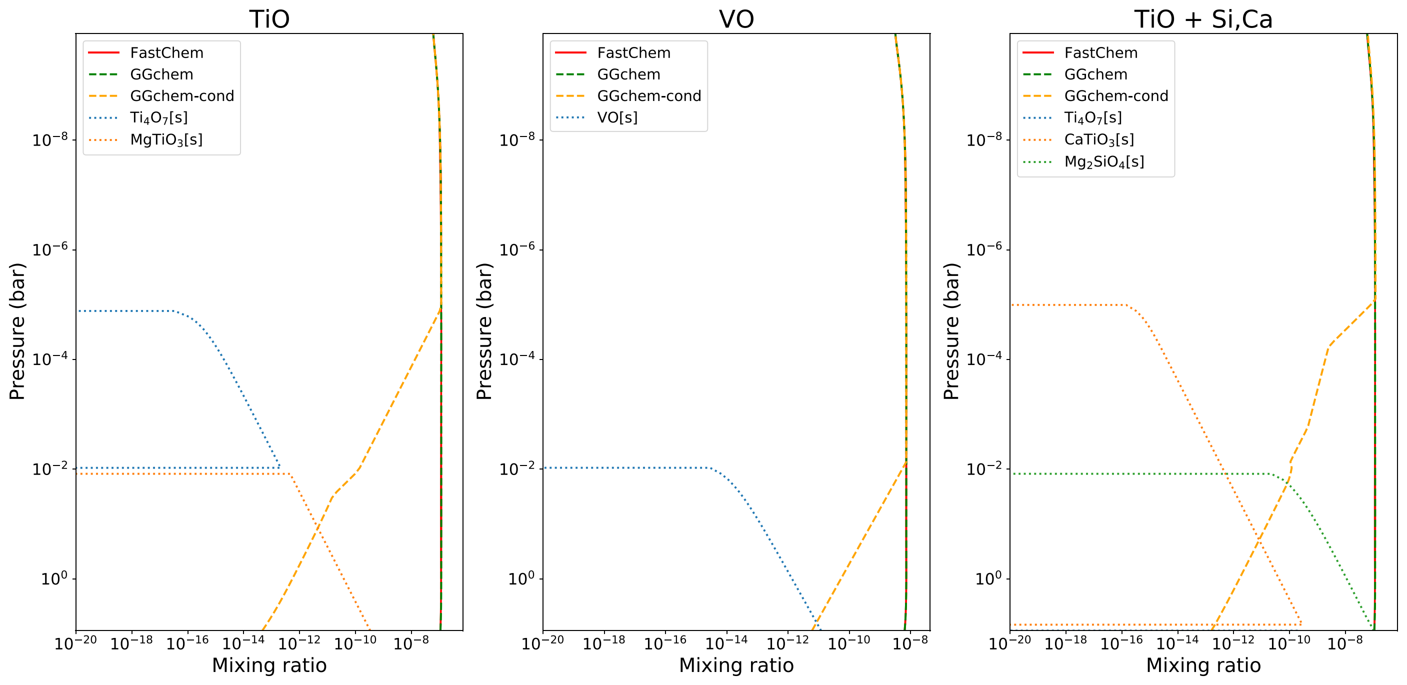

Figure 2. Plots of volume mixing ratios of TiO and VO using the heavy chemistry for FastChem, GGchem, and GGchem with condensation. The rightmost plot is the heavy chemistry where silicon and calcium are included.

FastChem is the main chemical model used in the HELIOS radiative transfer code (Kitzmann et al. 2020; Lavie et al. 2017) and has seen successful use in retrievals of the brown dwarf companions HD 1160B and HD 19467B (Mesa et al. 2020) and secondary eclipses of WASP-121b (Bourrier et al. 2020).

2.1.3. GGchem

GGchem (Gleich-Gewichts-Chemie) (Woitke et al. 2018) is a FORTRAN-90 rewrite of the code that was originally used to study condensation processes in late M-type stars. The rewrite updates the iteration scheme, extends numerical accuracy to

Table 7
A Summary of All Retrieved Values and Uncertainties for Each Combination of Simulated Spectra and Input Model

Model	C/H/O/N	C/H/O/N + cond.	Heavy	Heavy + cond.
Radius (R_J)				
ACE	$1.39^{+0.00}_{-0.00}$	$1.39^{+0.00}_{-0.00}$	$1.42^{+0.00}_{-0.00}$	$1.41^{+0.00}_{-0.00}$
FastChem	$1.39^{+0.00}_{-0.00}$	$1.39^{+0.00}_{-0.00}$	$1.42^{+0.00}_{-0.00}$	$1.41^{+0.00}_{-0.00}$
GGchem	$1.39^{+0.00}_{-0.00}$	$1.39^{+0.00}_{-0.00}$	$1.42^{+0.00}_{-0.00}$	$1.41^{+0.00}_{-0.00}$
GGchem+cond.	$1.39^{+0.00}_{-0.00}$	$1.39^{+0.00}_{-0.00}$	$1.42^{+0.00}_{-0.00}$	$1.41^{+0.00}_{-0.00}$
FastChem (Heavy)	$1.39^{+0.00}_{-0.00}$	$1.39^{+0.00}_{-0.00}$	$1.39^{+0.00}_{-0.00}$	$1.38^{+0.00}_{-0.00}$
GGchem (Heavy)	$1.39^{+0.00}_{-0.00}$	$1.39^{+0.00}_{-0.00}$	$1.39^{+0.00}_{-0.00}$	$1.38^{+0.00}_{-0.00}$
GGchem (Heavy) + cond.	$1.39^{+0.00}_{-0.00}$	$1.39^{+0.00}_{-0.00}$	$1.39^{+0.00}_{-0.00}$	$1.39^{+0.00}_{-0.00}$
Free	$1.39^{+0.00}_{-0.00}$	$1.39^{+0.00}_{-0.00}$	$1.39^{+0.00}_{-0.00}$	$1.39^{+0.00}_{-0.00}$
Temperature (K)				
ACE	$1498.57^{+3.85}_{-4.18}$	$1498.54^{+3.53}_{-4.12}$	$1167.21^{+5.22}_{-5.18}$	$1343.73^{+4.40}_{-4.55}$
FastChem	$1495.27^{+4.26}_{-4.46}$	$1495.29^{+4.27}_{-4.30}$	$1166.49^{+5.24}_{-5.27}$	$1341.80^{+4.34}_{-4.51}$
GGchem	$1496.28^{+4.02}_{-4.21}$	$1496.56^{+3.85}_{-4.33}$	$1166.47^{+5.20}_{-5.03}$	$1342.52^{+4.49}_{-4.66}$
GGchem+cond.	$1496.50^{+3.85}_{-4.19}$	$1496.57^{+3.86}_{-4.29}$	$1166.70^{+5.08}_{-5.48}$	$1342.41^{+4.73}_{-4.66}$
FastChem (Heavy)	$1329.74^{+2.56}_{-2.45}$	$1329.93^{+2.52}_{-2.44}$	$1498.85^{+4.81}_{-5.28}$	$1450.00^{+2.14}_{-2.83}$
GGchem (Heavy)	$1329.70^{+2.61}_{-2.55}$	$1330.04^{+2.44}_{-2.45}$	$1499.06^{+4.70}_{-5.31}$	$1448.77^{+4.03}_{-3.71}$
GGchem (Heavy) + cond.	$1373.13^{+1.85}_{-2.05}$	$1372.16^{+2.07}_{-2.40}$	$1620.19^{+3.11}_{-2.96}$	$1499.70^{+2.25}_{-2.32}$
Free	$1497.75^{+4.05}_{-4.46}$	$1497.85^{+4.02}_{-4.75}$	$1499.39^{+4.84}_{-5.28}$	$1507.68^{+4.59}_{-4.45}$
Metallicity ($Z_\odot$)				
ACE	$0.99^{+0.08}_{-0.08}$	$0.99^{+0.08}_{-0.07}$	$0.27^{+0.02}_{-0.02}$	$0.28^{+0.02}_{-0.02}$
FastChem	$0.98^{+0.08}_{-0.08}$	$0.97^{+0.07}_{-0.07}$	$0.27^{+0.02}_{-0.02}$	$0.27^{+0.02}_{-0.02}$
GGchem	$0.98^{+0.08}_{-0.07}$	$0.98^{+0.07}_{-0.07}$	$0.27^{+0.02}_{-0.02}$	$0.28^{+0.02}_{-0.02}$
GGchem+cond.	$0.98^{+0.08}_{-0.07}$	$0.98^{+0.07}_{-0.07}$	$0.27^{+0.02}_{-0.02}$	$0.28^{+0.02}_{-0.02}$
FastChem (Heavy)	$4.00^{+0.00}_{-0.00}$	$4.00^{+0.00}_{-0.00}$	$1.00^{+0.08}_{-0.07}$	$3.99^{+0.01}_{-0.01}$
GGchem (Heavy)	$4.00^{+0.00}_{-0.00}$	$4.00^{+0.00}_{-0.00}$	$1.01^{+0.08}_{-0.08}$	$3.99^{+0.01}_{-0.01}$
GGchem (Heavy) + cond.	$2.10^{+0.13}_{-0.13}$	$2.05^{+0.12}_{-0.12}$	$0.35^{+0.03}_{-0.02}$	$0.98^{+0.08}_{-0.07}$
Free	$0.69^{+0.15}_{-0.12}$	$0.70^{+0.16}_{-0.12}$	$0.66^{+0.17}_{-0.13}$	$0.85^{+0.20}_{-0.16}$
C/O				
ACE	$0.50^{+0.02}_{-0.02}$	$0.55^{+0.02}_{-0.02}$	$0.36^{+0.02}_{-0.02}$	$0.63^{+0.02}_{-0.02}$
FastChem	$0.50^{+0.02}_{-0.02}$	$0.55^{+0.02}_{-0.02}$	$0.36^{+0.02}_{-0.02}$	$0.62^{+0.02}_{-0.02}$
GGchem	$0.50^{+0.02}_{-0.02}$	$0.55^{+0.02}_{-0.02}$	$0.35^{+0.02}_{-0.02}$	$0.63^{+0.02}_{-0.02}$
GGchem+cond.	$0.50^{+0.02}_{-0.02}$	$0.55^{+0.02}_{-0.02}$	$0.35^{+0.02}_{-0.02}$	$0.63^{+0.02}_{-0.02}$
FastChem (Heavy)	$0.10^{+0.00}_{-0.00}$	$0.10^{+0.00}_{-0.00}$	$0.50^{+0.01}_{-0.01}$	$0.13^{+0.00}_{-0.00}$
GGchem (Heavy)	$0.10^{+0.00}_{-0.00}$	$0.10^{+0.00}_{-0.00}$	$0.50^{+0.01}_{-0.01}$	$0.12^{+0.01}_{-0.01}$
GGchem (Heavy) + cond.	$0.30^{+0.02}_{-0.02}$	$0.34^{+0.02}_{-0.02}$	$0.59^{+0.01}_{-0.01}$	$0.55^{+0.02}_{-0.02}$
Free	$0.51^{+0.04}_{-0.04}$	$0.56^{+0.04}_{-0.04}$	$0.50^{+0.04}_{-0.04}$	$0.55^{+0.04}_{-0.04}$

Note. C/H/O/N and C/H/O/N + cond. are ACE and GGchem+condensation with only the elements C, H, O, and N, and simulated as JWST spectra respectively. Heavy and Heavy + cond. are FastChem and GGchem+condensation with elements H, He, C, O, N, Mg, S, Ti, V, K, and Na and simulated as JWST spectra respectively.

quadruple precision, improves the range of applicability down to 100 K, and includes equilibrium condensation. GGchem supports the elements from hydrogen to zirconium plus tungsten and includes ion chemistry. GGchem supports the widest range of fitting function, which includes forms given by Gail & Sedlmayr (1986), Sharp & Huebner (1990), and Equations (2), (3), and (4). GGchem can exploit multiple sources of thermochemical data. GGchem will continuously add new species from each file; any species already defined will be overwritten with the newer data. By default, GGchem will load $\ln K$ for diatomics and atoms from Barklem & Collet (2016) fit to Equation (4), NIST-JANAF (Chase 1986) fit to Equation (4), and some additional refits of the Tsuji (1973) fit to Equation (4), giving in total 448 neutral and 117 charged species. GGchem is extensive in the atmospheric retrieval literature; it was recently used to analyze the atmospheres of the hot Jupiters KELT-9b (Changeat & Edwards 2021) and WASP-103b (Kirk et al. 2021).

2.1.4. Free Chemistry

In this paper, we also compare our results with the three chemical equilibrium codes to the baseline TauREx 3.1 free chemistry. The assumption of free chemistry is a conservative approach as it is only informed by the spectral features in the observed spectrum and does not assume any particular physical or chemical processes. While the free chemistry module in TauREx 3.1 allows for any parametric law to describe the chemical profiles of each species, we only consider the simplest assumption of constant profiles with altitude. In Changeat et al. (2019), a detailed example of a more complex profile is presented.

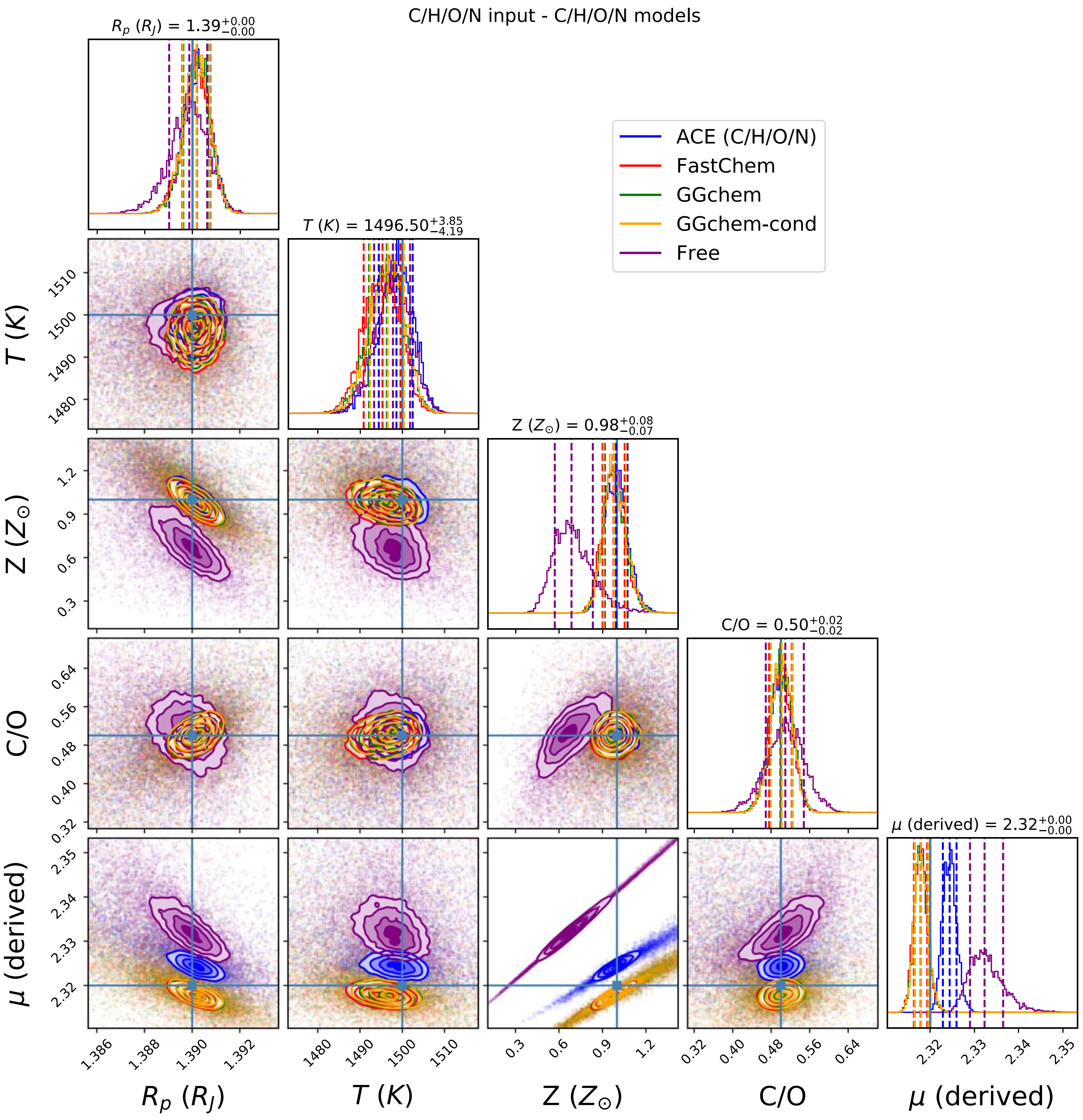

Figure 3. Retrieval posteriors of ACE (blue), FastChem (red), GGchem (green), GGchem with condensation (orange), and free chemistry (purple). Each equilibrium chemistry code utilizes the C/H/O/N element list and is fit against a JWST simulated transit spectrum of HD 209458b. Simulation (Truth) parameters are given in Table 1 and chemical abundances are computed using ACE (Agúndez et al. 2012, 2020). Priors used are given in Table 4.

2.2. Forward Model Comparison

For each chemical equilibrium code we model an HD 209458b-like planet as our test system with stellar and planetary parameters from Stassun et al. (2017) given in Table 1. The atmosphere is modeled as a cloud-free isothermal transit spectrum that includes opacities given in Table 2 from molecular absorptions, collisionally induced absorptions (CIAs) from H_2–H_2 and H_2–He, and Rayleigh scattering calculated for all molecules given by Cox (2015). We then simulate observed spectra as recorded with the JWST NIRISS instrument with GR700XD grism and the NIRSPEC instrument with an G395M/F290LP filter. We generated the uncertainties using `taurex_jwst` plugin, which uses the Exowebb library as its noise model backend.

For FastChem and GGchem, we model two cases: The C/H/O/N case where we limit the available elements to H, He, C, N, and O to allow for direct comparison with the ACE chemistry model. The second case loosens the limitation on elements by including H, He, C, O, N, Mg, S, Ti, V, K, and Na in the chemistry models. We defined this case as heavy chemistry.

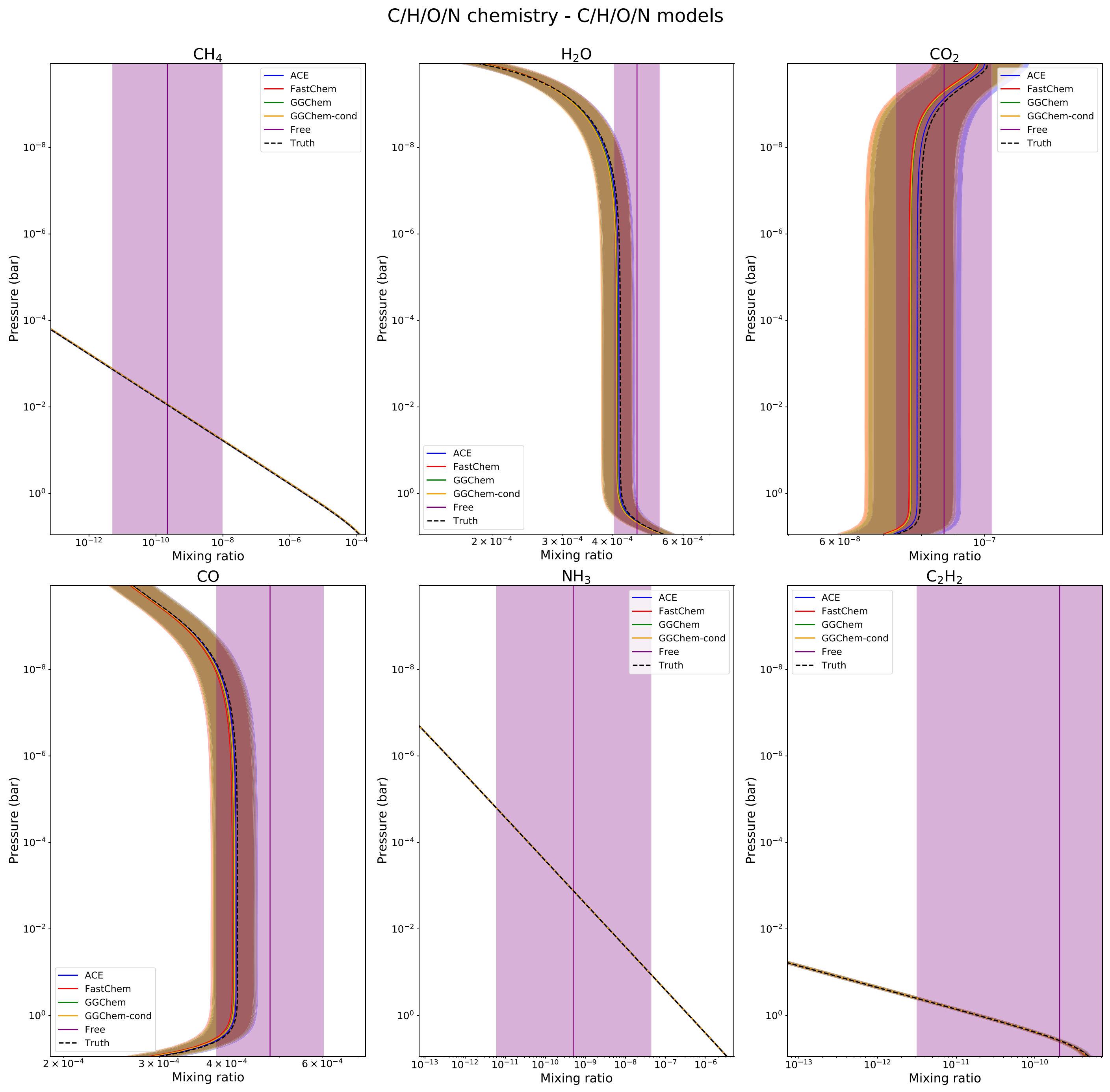

Figure 4. Retrieved molecular profiles for ACE (blue), FastChem (red), GGchem (green), GGchem with condensation (orange), and free chemistry (purple). Each equilibrium chemistry code utilizes the C/H/O/N element list. Shaded regions denote 1σ span; dashed lines are truth values computed from the ACE chemical code using Table 1.

For GGchem, we also include a particular case for both C/H/O/N and heavy chemistries, where we introduce sequestration from condensation. Table 3 describes the codes and the cases modeled. In total, there are seven forward models produced.

2.3. Retrieval Cross-validation

For each of the seven forward models produced, we retrieve them against each code and case described in Table 3, requiring 49 retrievals. We also retrieve each code and case using a free chemistry model that includes an isoabundance profile for H_2O, CH_4, NH_3, C_2H_2, CO, CO_2, TiO, and VO. In total, 56 retrievals were conducted. For this exercise, we do not scatter the spectra because it has been repeatedly demonstrated (Feng et al. 2018; Changeat et al. 2019, 2020a) that for normally distributed noise and a large number of observations, scattered and unscattered spectra are equivalent due to the central limit theorem.

In the discussion of results we only highlight four retrieval cases: C/H/O/N chemistry and Heavy chemistry models fit against an ACE C/H/O/N input spectrum, Heavy chemistry and ACE fits against a FastChem heavy input spectrum, and Heavy chemistry and ACE fit against a GGchem input spectrum using the heavy condensation model. The cases where the elemental composition and processes of the chemical code match the input

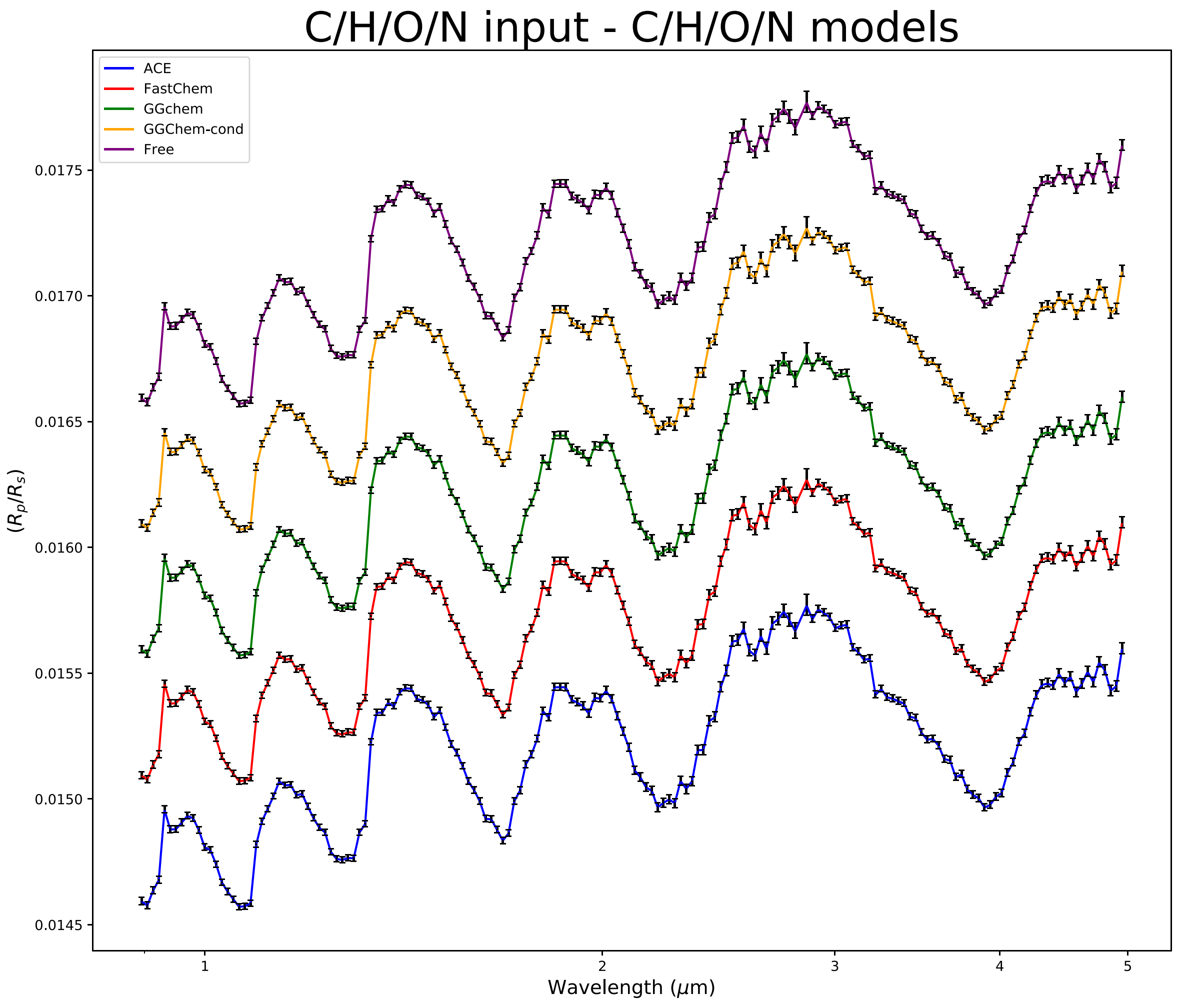

Figure 5. Retrieved spectra for ACE (blue), FastChem (red), GGchem (green), GGchem with condensation (orange), and free chemistry (purple). Each equilibrium chemistry code utilizes the C/H/O/N element list and is retrieved against simulated JWST spectra (black) computed with ACE using parameters from Table 1. Spectra have been binned to a lower resolution and offset for clarity.

spectrum will assess the underlying biases associated with each code's methodology, thermochemical data source, and choice of equilibrium constant function. Conversely, the cases where the elemental compositions and processes do not match will assess the biases associated with our initial assumption of the atmosphere's chemistry. Our retrieval setup utilizes TauREx-CUDA contributions (Appendix B.1) to accelerate the forward modeling computation and the MultiNest Bayesian nested sampling code (Feroz et al. 2009; Buchner et al. 2014) with message passing interface (MPI). The nested sampling utilizes 750 live points, with an evidence tolerance of 0.5 and uniform priors given by Table 4. For the free retrieval, we use priors given by Table 5 and derive Z and C/O after the retrieval through post-processing (Lee et al. 2013; MacDonald & Madhusudhan 2019). We perform the retrievals on the OzSTAR Supercomputing @ Swinbourne cluster. Each node has two Intel Gold 6140 18-core processors and two nVidia P100 12 GB PCIe GPUs. For every retrieval, we use five nodes on the cluster. On each node, we only allocate four of the available cores and spawn four TauREx MPI processes. Each nVidia P100 is assigned two processes in order to maximize the GPU resources. In total, each retrieval is run on 20 cores exploiting 10 GPUs.

Examining the computational performance of the retrievals, we required ≈40,000 samples for completion. Table 6 displays the mean sampling times for each code. It is evident that the CUDA plugin greatly accelerates the retrieval compared to the previous version (Al-Refaie et al. 2021), with retrieval times ranging from 4 to 52 minutes for most of the chemical codes and element list. The exception is the heavy condensation case, which required over 9 hours to complete. The calculation of equilibrium condensation on the CPU is a significant proportion of the forward modeling time; therefore, GPU acceleration of the optical depth is of little benefit in this particular case.

For all the plots, we will maintain the same color scheme, i.e., blue for ACE, red for FastChem, green for GGchem, and orange for GGchem with condensation.

3. Results

3.1. Forward Model Comparison

Figure 1 shows simulated JWST transit spectra obtained from different chemistry models for the test planet given in Table 1. Qualitatively the C/H/O/N case (left panel of Figure 1) presents highly similar spectral features since water absorption dominates most of the spectrum. There is no condensation apparent when selecting only C/H/O/N species. Assessing the heavy case (right panel of Figure 1), we observe strong absorption features from TiO (0.3 μm–1 μm) and an absorption feature from VO at 1.1 μm in FastChem and GGchem that is not accounted for in ACE.

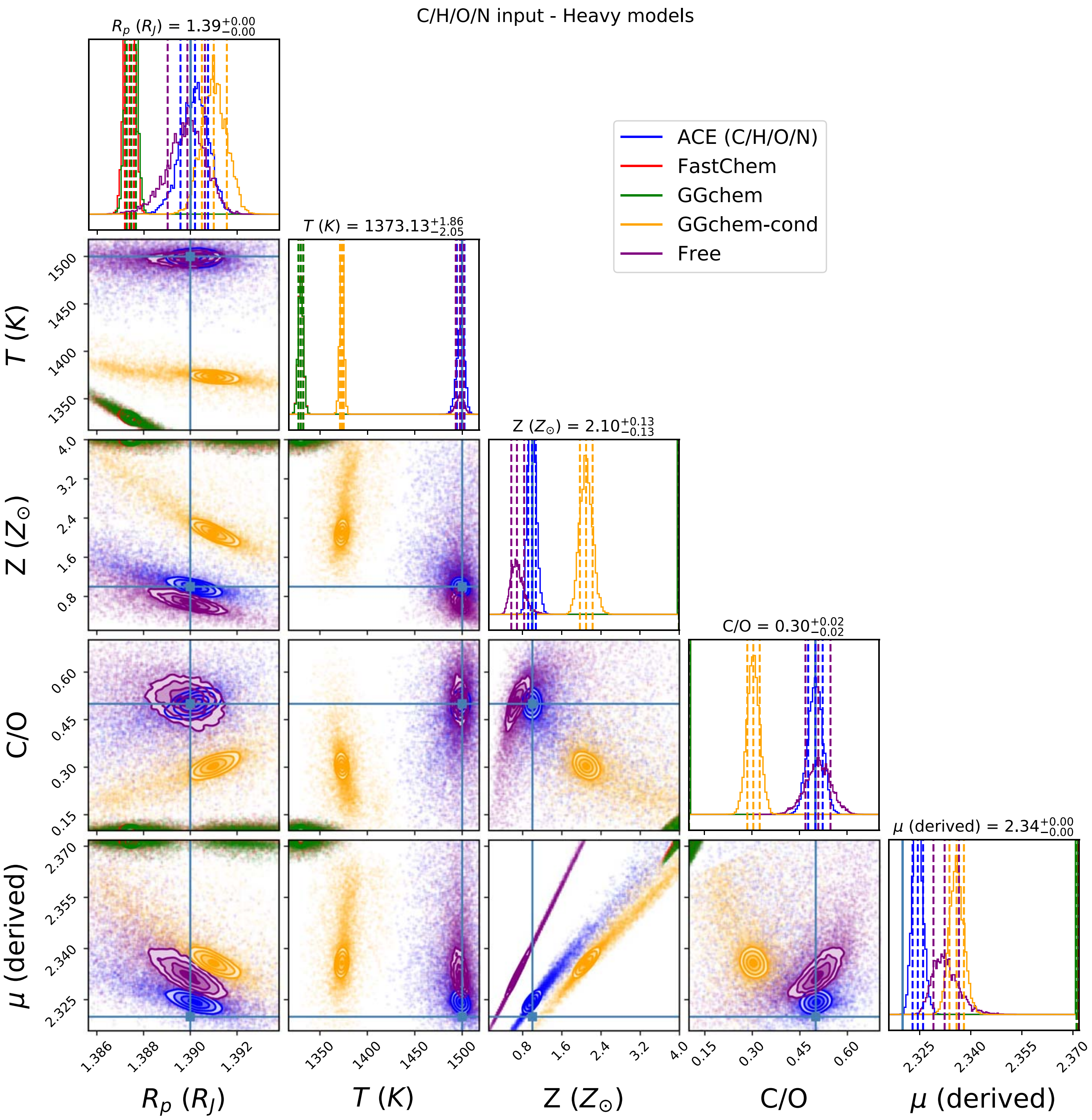

Figure 6. Retrieval posteriors of ACE (blue), FastChem (red), GGchem (green), and GGchem with condensation (orange). Each equilibrium chemical code utilizes the heavy element list (except ACE and free) and is fit against a JWST simulated transit spectrum of HD 209458b using parameters given by Table 1 and abundances computed using ACE (Agúndez et al. 2012, 2020). Priors used are given in Table 4.

For the case where GGchem includes condensation, we observe a reduction in the spectral features in TiO and a minor reduction in VO, which are due to the decrease in their abundance in the atmosphere. Figure 2 shows clearly that these molecules condense in the low atmosphere. The main contributor to TiO loss comes from the formation of solid-phase $MgTiO_3$ in the troposphere, and Ti_4O_7 in the stratosphere (see, e.g., Lodders 2002). However, the overabundance of $MgTiO_3$ is a consequence of our choice of elements lacking silicon, which sequesters magnesium through forsterite (Mg_2SiO_4) as seen in Figure 2. Additionally, the calcium-containing condensate $CaTiO_2$ is stable in the atmospheric region below 10^{-5} bar and is the principal constituent of Ti-bearing condensates. $CaTiO_2$ is more efficient at removing TiO from the troposphere and hinders the production of Ti_4O_7. The reduction in Ti_4O_7 allows for a higher concentration of TiO in the stratosphere.

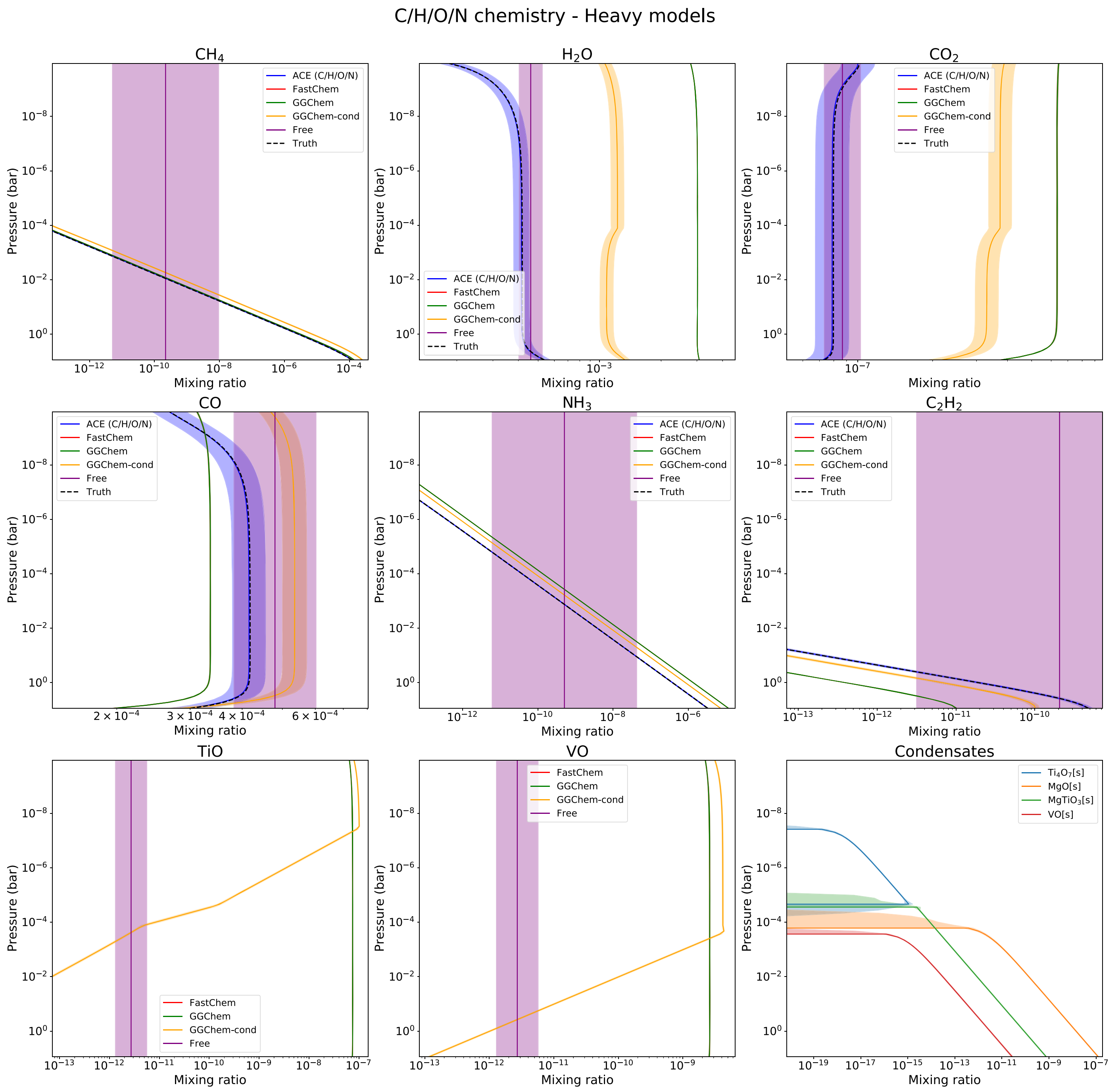

Figure 7. Retrieved molecular profiles for ACE (blue), FastChem (red), GGchem (green), GGchem with condensation (orange), and free chemistry (purple). Each equilibrium chemical code utilizes the heavy element list (except ACE and free). Shaded regions denote 1σ span; dashed lines are truth values computed from the ACE chemical code (C/H/O/N) using Table 1.

For vanadium, the first and most stable condensate is vanadium monoxide in the solid phase (Burrows & Sharp 1999; Allard et al. 2001), which removes gas-phase VO from the lower atmosphere. The introduction of calcium to form vanadium analogues to the calcium titanates introduces no new condensates as these are not stable (Lodders 2002).

Overall, all equilibrium chemical models arrive at very similar conclusions when presented with the same elements. Differences only appear based on the choice of the elemental composition and condensation of the atmosphere rather than the choice of chemical code.

3.2. Retrieval Cross-validation

Table 7 summarizes all retrieved values and uncertainties for 32 combinations of simulated spectra and chemical model. We choose to omit 24 specific combinations of simulated spectra (C/H/O/N produced by FastChem and GGchem and Heavy produced by GGchem) as they produced identical posteriors. We find the same posterior distributions for all parameters as expected by matching the initial chemical elements with the simulated spectra. The actual implementation of equilibrium chemistry and the thermochemical tables and fitting functions used have no significant effect on the retrieval, and all give

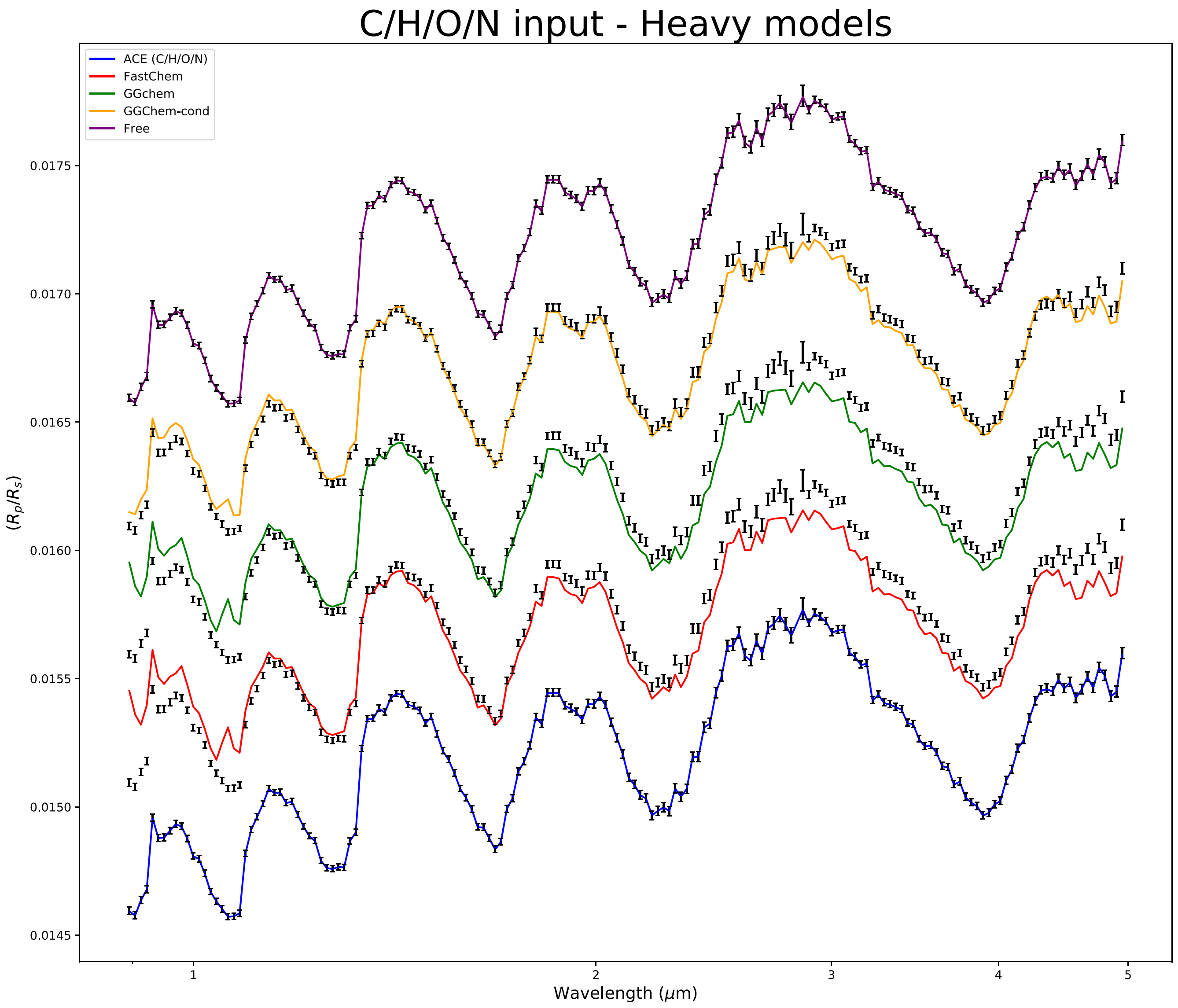

Figure 8. Retrieved spectra for ACE (blue), FastChem (red), GGchem (green), GGchem with condensation (orange), and free chemistry (purple). Each equilibrium chemical code utilizes the heavy element list (except ACE and free) and is retrieved against simulated JWST spectra (black) computed with ACE using parameters from Table 1. Spectra have been binned to a lower resolution and offset for clarity.

consistent results. For example, retrieving against ACE, FastChem with C/H/O/N and GGchem with C/H/O/N spectra give the same posteriors, as expected and described in the previous section. This result is important because the means of solving thermochemical equilibrium introduces no visible bias in the retrieval. No single code produces the "best" retrieval. Its choice is entirely a user preference based on needs such as ease-of-use, speed, thermochemical completeness, applicability, and whether to include condensation.

By contrast, when the model species and input spectrum species do not match, we find a significant deviation between true and retrieved values. We discuss these results in more detail in the following sections.

Regarding the free chemistry case, our retrievals manage to recover the input abundances simulated from the chemical equilibrium models. In all cases, C/H/O/N, Heavy, and Heavy (condensation), the input spectra are well fit and the true abundances are within the retrieved uncertainties. This provides confidence that a free approach can be used to interpret exoplanet spectra, while avoiding the main user-dependent choices implied in chemical equilibrium codes.

3.2.1. C/H/O/N Input Spectrum versus C/H/O/N Models

In Figure 3, we compare C/H/O/N species for all codes against the ACE simulated spectra. The posteriors of FastChem, GGchem, and GGchem with condensation are within 1σ of ACE's posterior volume. The mean molecular weight is slightly higher for ACE as it has almost twice as many molecules (105) as FastChem (57) and GGchem (58) when selecting only C/H/O/N species. The retrieved atmospheric molecular profiles in Figure 4 reveal that all codes recover well the correct molecular profile for all significant molecules present.

Looking at the retrieved chemical profiles, the free chemistry manages to recover the abundances of individual species very precisely. In turn, posteriors recover well the radius, temperature, and C/O ratio but underestimate the metallicity since

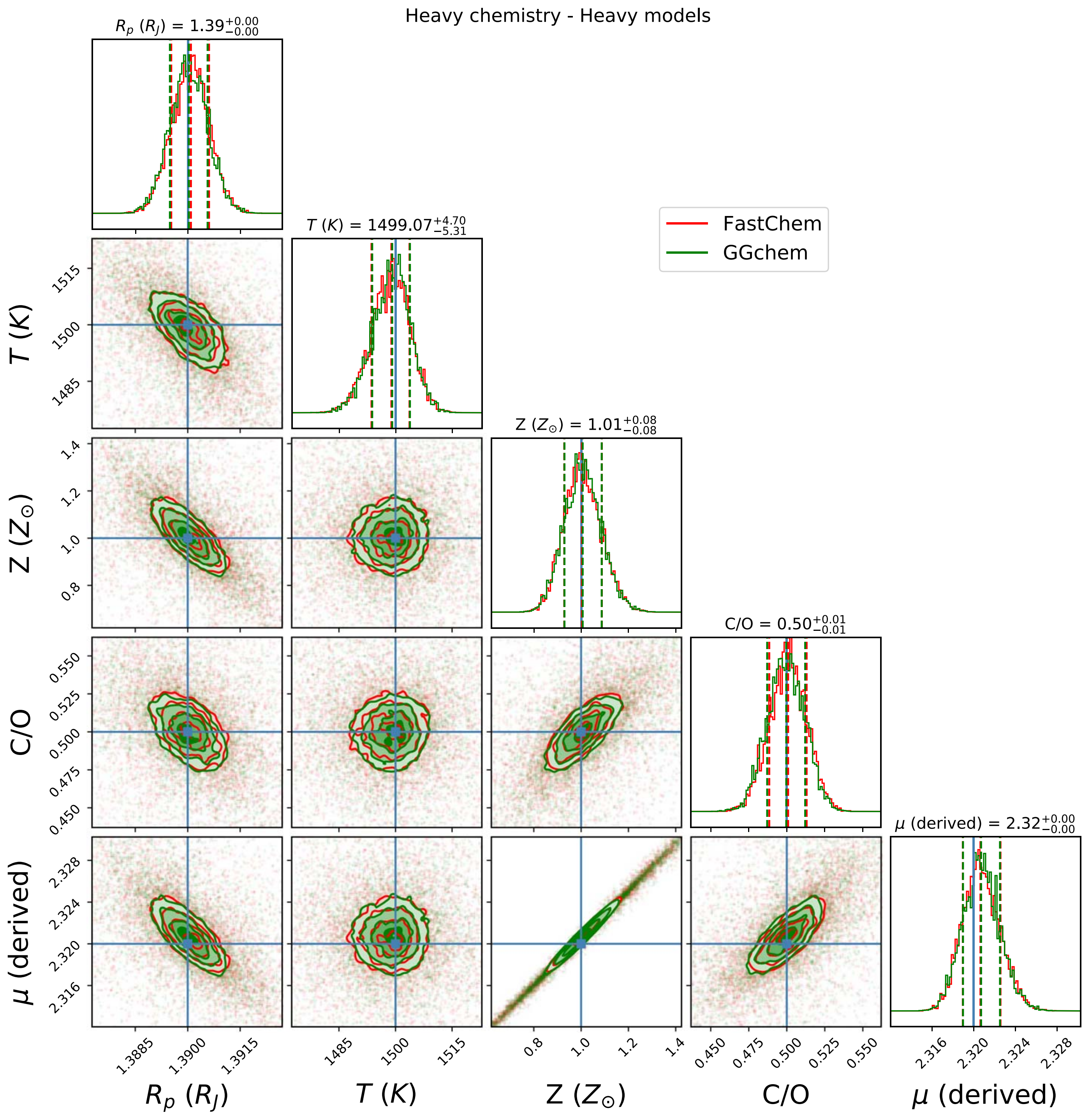

Figure 9. Retrieval posteriors for FastChem (red) and GGchem (green) only. Each chemical code utilizes the heavy element list and is fit against a JWST simulated transit spectrum of HD 209458b using parameters given in Table 1 and abundances computed using FastChem (Stock et al. 2018) with heavy species. Priors used are given in Table 4.

many inactive molecules cannot be retrieved in the free chemistry model.

Finally, the resultant spectra in Figure 5 all lie within the uncertainties of the simulated JWST observations.

3.2.2. C/H/O/N Input Spectrum versus Heavy Models

We observed significant differences in the posteriors (Figure 6) when applying the heavy element list on FastChem and GGchem to the C/H/O/N spectrum. The heavy-only element list has no direct mechanism to remove TiO and VO from the atmosphere due to our choice of priors, so the sampler attempts to mask their features underneath H_2O for both GGchem and FastChem by increasing its abundance (Figure 7) through the metallicity parameter up to the prior boundaries at 4.0 $Z_\odot$. The increased metallicity introduces more strong carbon-bearing features. However, these are muted by a significant reduction in the C/O ratio to 0.15. The increase in depth caused by H_2O is compensated for through a reduction in the planetary radius. The resulting heavy spectra in Figure 8 are almost entirely water with a reduction in

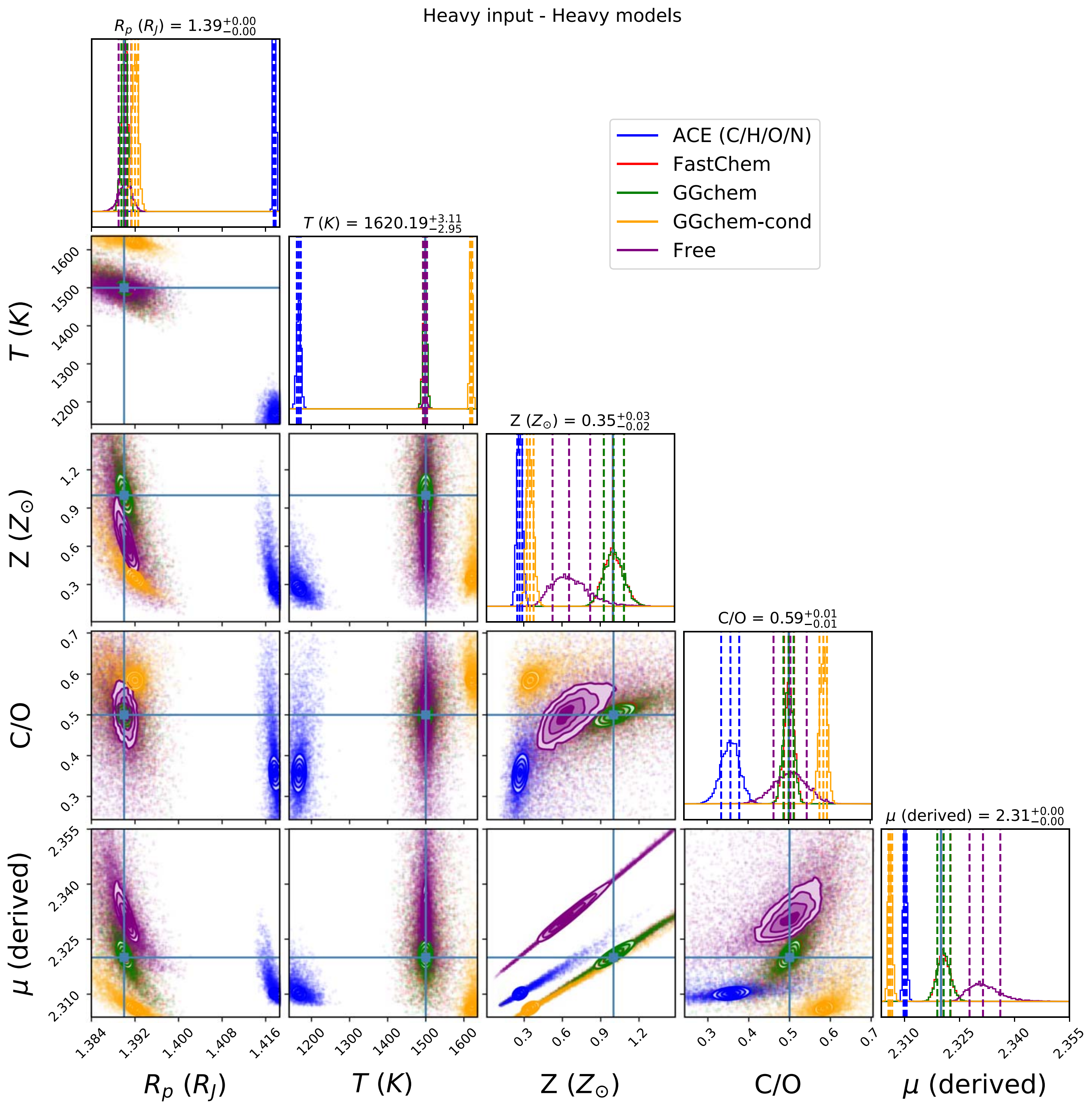

Figure 10. Posteriors of ACE (blue), FastChem (red), GGchem (green), GGchem with condensation (orange), and free chemistry (purple). Each equilibrium chemical code utilizes the heavy element list (except ACE and free) and is fit against a JWST simulated transit spectrum of HD 209458b using parameters given in Table 1 and abundances computed using FastChem (Stock et al. 2018) with heavy species. Priors used are given in Table 4.

depth at 2 μm and 3–5 μm caused by the loss of CO and CH_4. The heavy element list with condensation also employs a similar tactic but not as aggressively because it has the means to reduce gas-phase TiO and VO through condensation. The condensate mixing profiles in Figure 7 show a higher altitude of stability with an increase to 10^{-5} bar for $MgTiO_3$ and 10^{-8} bar for Ti_4O_7.

The resultant spectrum in Figure 8 for this case retains many of the spectral features seen in the C/H/O/N chemistry with a significant reduction in TiO and VO absorption features in the optical.

3.2.3. Heavy Input Spectrum

We examine the case where the retrievals are fit against a heavy species spectrum. When examining only FastChem and GGchem, the posteriors in Figure 9 display a high degree of overlap and shape similarity. Both achieve almost the same correlations between parameters and tightly constrain their posteriors.

Looking at the retrieval in its full context, the posteriors in Figure 10 include both ACE and GGchem with heavy condensation, and we see a significant deviation in their

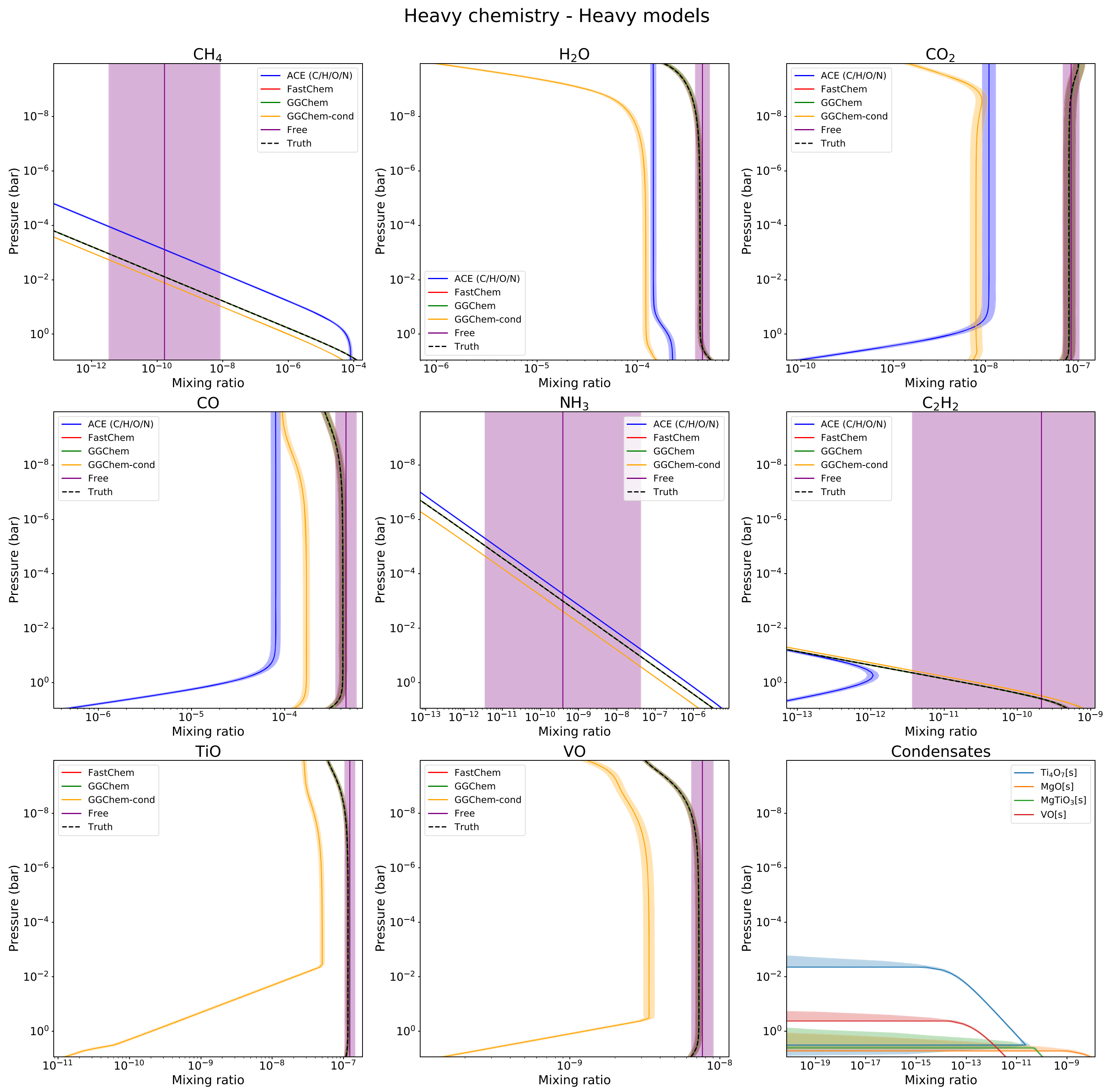

Figure 11. Retrieved molecular profiles for ACE (blue), FastChem (red), GGchem (green), GGchem with condensation (orange), and free chemistry (purple). Each equilibrium chemical code utilizes the heavy element list (except ACE and free). Shaded regions denote 1σ span; dashed lines are truth values computed from the GGchem chemical code with heavy species using Table 1.

resultant posteriors. The resultant molecular profiles in Figure 11 and spectra in Figure 12 reveal the different approaches taken by both ACE and GGchem with condensation to reach their retrievals. For the ACE case (and C/H/O/N chemistry in general), the sampler attempts to raise the features at 1 μm by increasing the amount of CH_4 in the atmosphere while lowering the contribution from CO.

This effect manifests by significantly reducing the amount of oxygen (i.e., the metallicity) as well as slightly reducing the C/O ratio. The knock-on effect of the reduction of oxygen lowers the overall transit depth as the abundance of H_2O drops significantly. The sampler counteracts this change by increasing the planetary radius to about 1.41 Jupiter radii. By contrast, the condensation retrieval attempts to minimize the condensates present to maximize the available TiO and VO. Looking at the condensation posteriors in Figure 10, the higher temperature and lower metallicity decrease the stability and abundance of the condensates to pressures around 10^{-3} bar with some condensates such as $MgTiO_3$ and MgO only appearing at pressures near the surface. The loss of oxygen is counteracted by a slight increase in the C/O ratio to maintain CH_4 abundance and in planet radius to maintain relative depths.

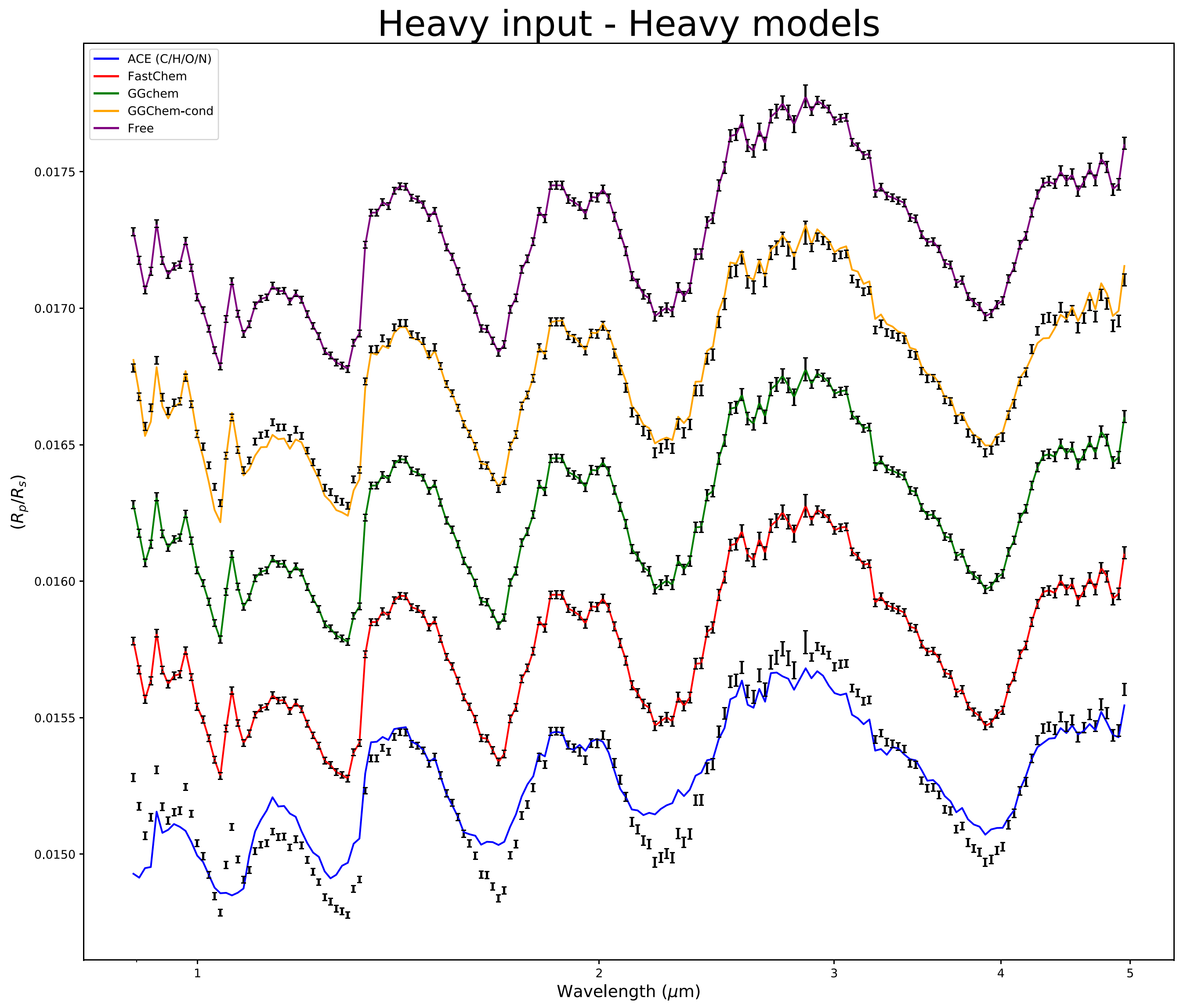

Figure 12. Retrieved spectra for ACE (blue), FastChem (red), GGchem (green), GGchem with condensation (orange), and free chemistry (purple). Each equilibrium chemical code utilizes the heavy element list (except ACE and free) and is retrieved against a simulated JWST spectrum (black) computed with FastChem with heavy chemistry using parameters from Table 1. Spectra have been binned to a lower resolution and offset for clarity.

The resultant spectrum in Figure 12 restores most of the spectral features present in the heavy-only species with only slight losses in VO opacity near 1.2 μm and CO at 4 μm. Both FastChem and GGchem retrieve almost exactly the same abundances for all molecules present.

Concerning the free chemistry models, the retrieved chemical profiles represent well the inputs (from the heavy chemical models) for H_2O, CO, CO_2, VO, and TiO, which are the most abundant molecules in the input spectrum. For CH_4, NH_3, and C_2H_2, the retrieval only provides an upper limit of about $10^{-7.5}$ due to the lack of features of these molecules in the spectrum.

3.2.4. Heavy with Condensation Input Spectrum

We finally consider the case where all chemical codes retrieve against a heavy condensation truth. We will not consider C/H/O/N with condensation, as it is evident in Figure 3 that it is equivalent to C/H/O/N with no condensation for temperatures considered. For the C/H/O/N-only species, we see similar behavior to the previous comparison in Figure 13 as the CH_4 abundance is increased to raise the overall baseline spectrum. However, the lower TiO and VO signature results in the temperature dropping only to 1350 K compared to 1120 K in the previous case, and the reduction in metallicity is only to 0.7 $Z_\odot$. Similarly, the heavy models mask TiO and VO by raising the water abundance, as seen in Figure 14. Since there is still TiO and VO present, the temperature difference from truth is only 60 K compared to the 150 K drop in the C/H/O/N case. The results of the free retrieval in the heavy with condensation case are similar to those from the heavy case, except for TiO and VO. Due to condensation, TiO and VO are sequestered from the bottom of the atmosphere, which leads the retrieval to average these abundances. We believe a more complex parametric description of those profiles, such as the one presented in Changeat et al. (2019), would avoid those biases. Due to the biases arising from the constant profile assumption on TiO and VO, the H_2O, CO, and CO_2 are more abundant in the atmosphere, causing a higher metallicity in the posteriors.

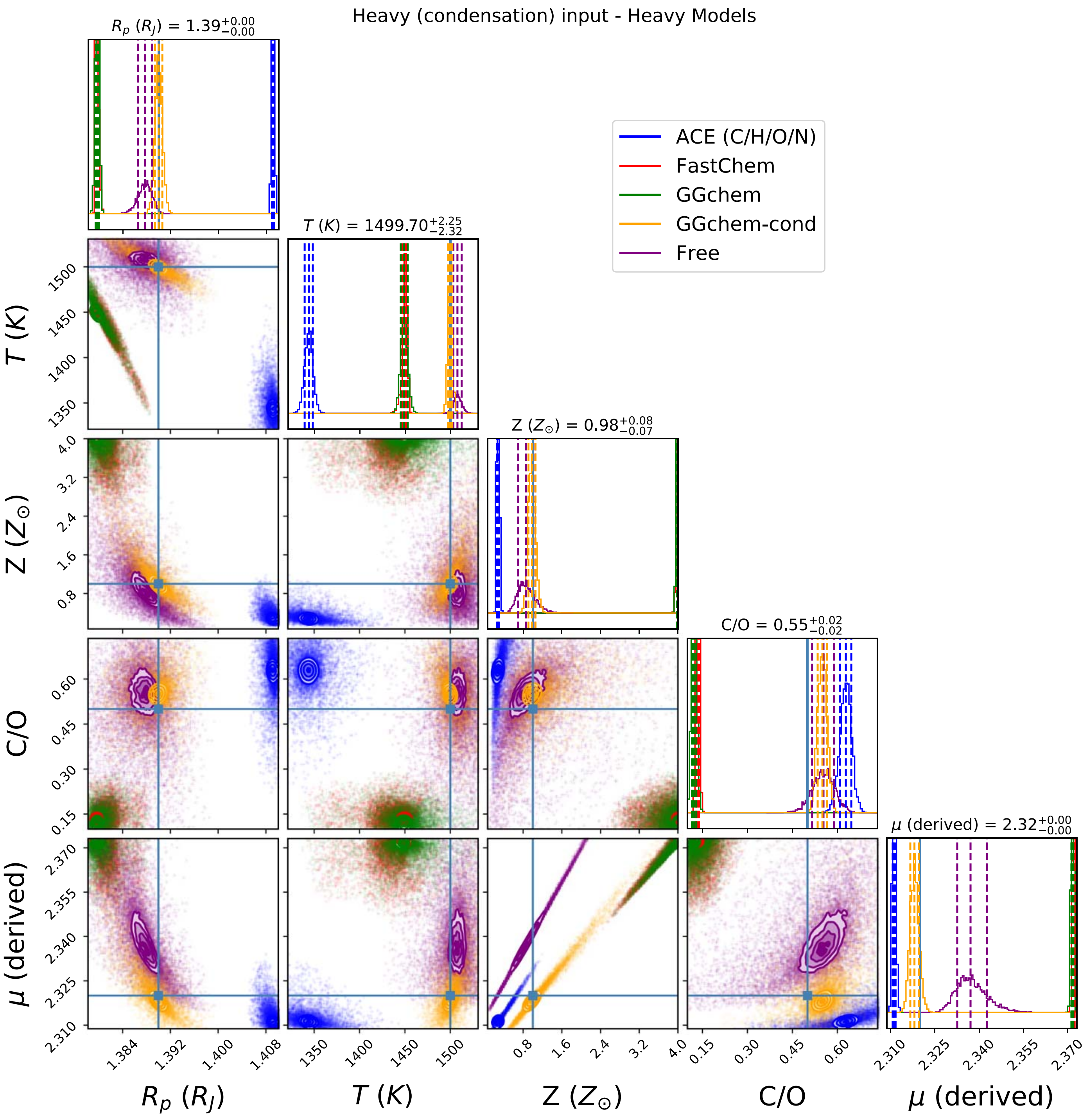

Figure 13. ACE (blue), FastChem (red), GGchem (green), GGchem with condensation (orange), and free chemistry (purple). Each equilibrium chemical code utilizes the heavy element list (except ACE and free) and is fit against a JWST simulated transit spectrum of HD 209458b using parameters given in Table 1 and abundances computed using GGchem (Stock et al. 2018) with heavy species and condensation. Priors used are given in Table 4.

Overall, the final spectra in Figure 15 show that the C/H/O/N and heavy cases agree qualitatively better than their respective counterpart retrievals (see Figure 13).

4. Discussion

When the model species and input spectrum species do not match, we find a significant deviation between true and retrieved values. For example, the retrieved uncertainties for temperature are less than 1% even when the deviation from the truth is greater than 8%. Far worse are the chemistry parameter values; When comparing C/H/O/N models with heavy input spectra, the metallicity posteriors have a smaller deviation (±0.02) than models with the same element list (±0.08) even though the retrieved value deviates by over 20%. Comparably the uncertainties for the retrieved C/O ratio also exhibit the same behavior while deviating from the truth by 10%. Similarly, introducing condensation into the input spectrum appears to retrieve metallicity posteriors with significantly smaller variances and can erroneously attribute subsolar

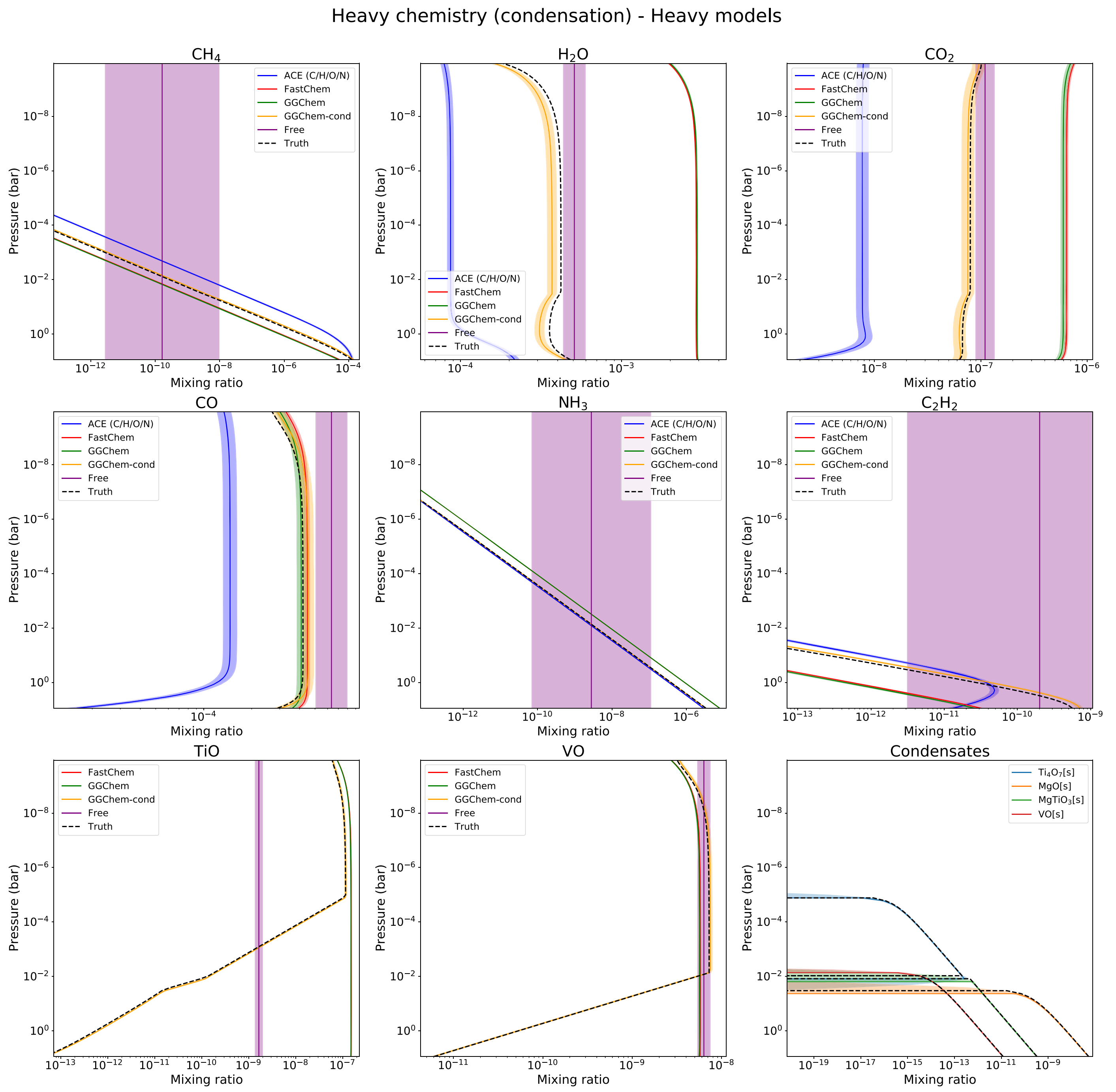

Figure 14. Retrieved molecular profiles for ACE (blue), FastChem (red), GGchem (green), GGchem with condensation (orange), and free chemistry (purple). Each equilibrium chemical code utilizes the heavy element list (except ACE and free). Shaded regions denote 1σ span; dashed lines are truth values computed from the GGchem chemical code with heavy species and condensation using Table 1.

composition to the planet. Finally, comparing heavy chemistry with C/H/O/N atmospheres, we find the retrieval is incapable of building posteriors within the tight prior bounds for metallicity and C/O ratio, with the retrieval moving toward a high metallicity value. These retrievals demonstrate that the assumptions we make on the composition and processes (i.e., choice of elements, condensation, etc.) occurring in the atmosphere are a significant source of bias. Unfortunately, we have no prior knowledge of the underlying physics within exoplanetary atmospheres in a realistic setting, so immediately assuming a self-consistent model can lead to confident and incorrect conclusions for the atmospheric composition.

In the context of the next-generation extensive surveys (Ariel: Tinetti et al. 2021) and dedicated studies (JWST: Greene et al. 2016) of exoplanets, the higher information content of spectra observed means that biases from our assumptions become significantly more pronounced than in HST data and greater care must be taken in analyzing exoplanetary atmospheres. Using less complex models and assumptions such as isoabundance initially will allow for careful exploration of the information content of the spectra and identification of species and processes. Following this approach would allow us to place an informed constraint on priors when introducing more complexity and self-consistency.

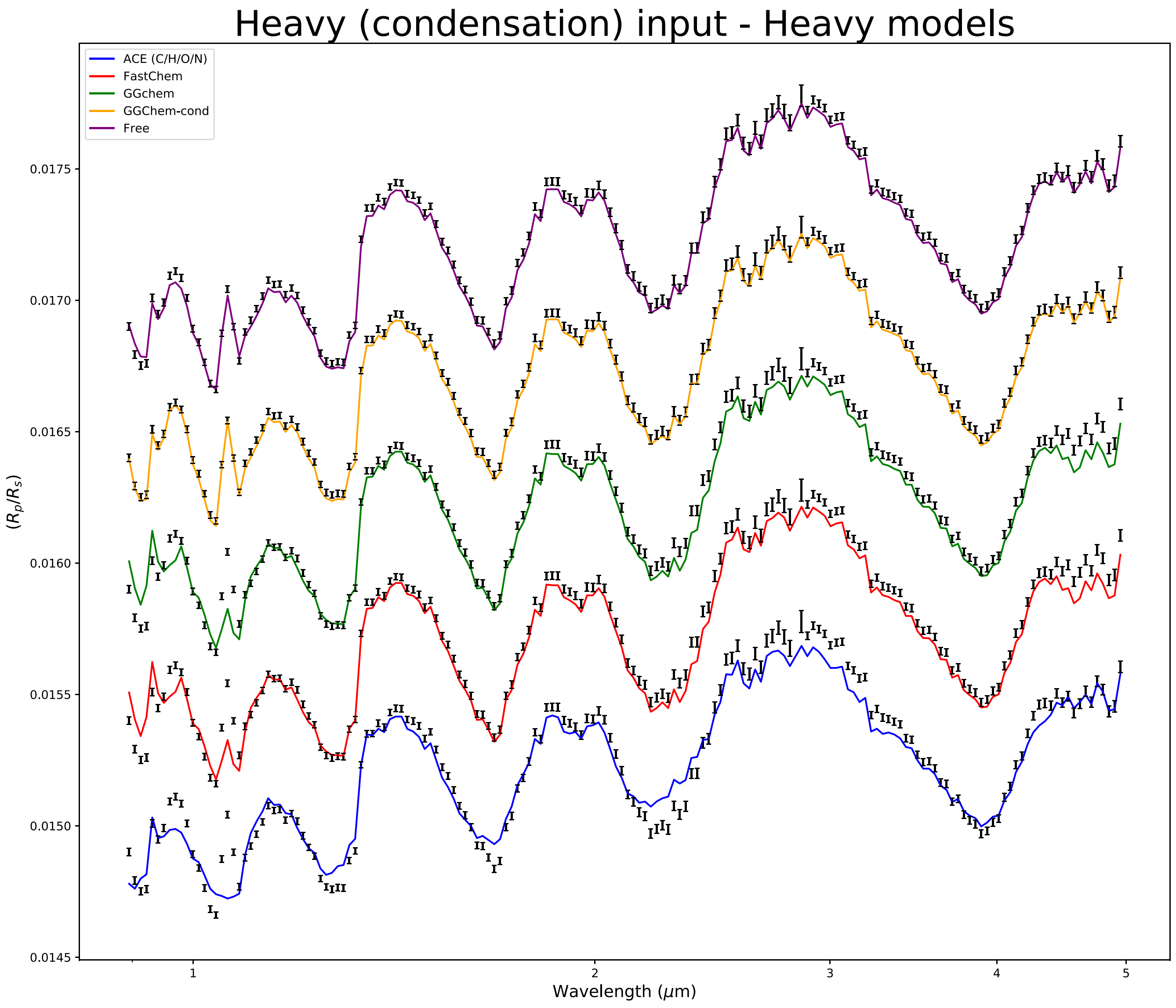

Figure 15. Retrieved spectra for ACE (blue), FastChem (red), GGchem (green), GGchem with condensation (orange), and free chemistry (purple). Each equilibrium chemical code utilizes the heavy element list (except ACE and free) and is retrieved against a simulated JWST spectrum (black) computed with GGchem with heavy elements and condensation using parameters from Table 1. Spectra have been binned to a lower resolution and offset for clarity

Next-generation retrievals must have the flexibility to include a wide range of processes and increase/decrease the model complexity to interpret spectra successfully.

5. Conclusions

We have presented here one of the first systematic comparisons of equilibrium chemical codes in the context of exoplanet retrievals. This comparison was made possible by the newest version of TauREx 3.1, allowing each chemical code to be "plugged in" to the framework. TauREx 3.1 and its available plugins are available on PyPI as source distributions and binaries for Windows, MacOS, and Linux. The sources for TauREX 3.1 and its plugins are also available on the ucl-exoplanet GitHub[3] with permissible licenses on most plugins. We hope that providing a framework with a plugin interface will allow for more collaboration and interconnectivity between different fields and future comparative work. Regarding the chemical comparisons, we demonstrate that ACE, GGchem, and FastChem reach the same conclusions given the same assumptions. The implementation, source of thermochemical data, and fitting of equilibrium functions introduce little to no bias in retrievals. The most significant source of bias comes from assumptions on the chemical processes in the atmosphere and that these retrievals arrive at confident but incorrect solutions. To overcome such biases, free retrievals of the chemistry should be considered as a first step. This is particularly relevant for the next generation of telescopes, such as JWST and Ariel, which will be sensitive to thermal, chemical, and dynamical processes that are not yet fully understood. Until our understanding of chemistry in exoplanets can evolve, simpler approaches relying on the information content of the observations have to be favored.

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 758892, ExoAI) and the European Union's Horizon 2020 COMPET

[3] https://github.com/ucl-exoplanets

program (grant agreement No. 776403, ExoplANETS A). Furthermore, we acknowledge funding by the UK Space Agency and Science and Technology Facilities Council (STFC) grants: ST/K502406/1, ST/P000282/1, ST/P002153/1, ST/S002634/1, ST/T001836/1, and ST/V003380/1. The authors would like to acknowledge P. Woitke, C. Helling, D. Kitzmann, and J. Stock for providing their codes under permissive open-source. A.F.A. would also like to thank Quratulain Jahangir for her support. O.V. acknowledges the CNRS/INSU Programme National de Planétologie (PNP) and CNES for funding support.

This work utilized the OzSTAR national facility at Swinburne University of Technology. The OzSTAR program receives funding in part from the Astronomy National Collaborative Research Infrastructure Strategy (NCRIS) allocation provided by the Australian Government. This work utilized the Cambridge Service for Data Driven Discovery (CSD3), part of which is operated by the University of Cambridge Research Computing on behalf of the STFC DiRAC HPC Facility (www.dirac.ac.uk. The DiRAC component of CSD3 was funded by BEIS capital funding via STFC capital grants ST/P002307/1 and ST/R002452/1 and STFC operations grant ST/R00689X/1. DiRAC is part of the National e-Infrastructure.

Appendix A
TauREx 3.1

TauREx 3.1 is the next version of the TauREx 3 library, backward-compatible with the previous version's input files but offering a swathe of improvements and optimizations to the overall architecture. The goal of this version was to expand the dynamic architecture and provide significantly more flexibility to the TauREx framework.

A.1. Nonuniform Priors

We include a new type of class in the retrieval framework that handles the prior transform in Bayesian retrievals. The `Prior` class allows for the inclusion of custom functions to be used as the prior transform and removes the uniform-only limitation of the previous version. A new *prior* flag can be used when in the input file definition of the fitting parameters:

```
[Fitting]
planet_radius:fit = True
planet_radius:prior = 'LogUniform(bounds = [−2,2])''
```

This is equivalent to the previous version's definition of fitting priors:

```
[Fitting]
planet_radius:fit = True
planet_radius:mode = log
planet_radius:bounds = 1e-2,1e2
```

It must be noted that the older input definition is still compatible with TauREx 3.1; however, internally they are converted into the new `Prior` form. On installation there are only four functions available: `Uniform`, `LogUniform`, `Gaussian`, and `LogGaussian`. Installing the `taurex_scipypriors` plugins increases this to over 50 different functions.

A.2. H^- opacity

A new `HydrogenIon` contribution class can be included in both transmission and emission forward models that provide opacities from continuous absorption of H^-. We compute the absorption opacity using Equations (3)–(6) from John (1988) for free–free and bound–free transitions and require the chemistry model to provide atmospheric abundances of gas-phase H and e^-. The algorithm is optimized heavily, requiring only 500 μs to run for $R = 15{,}000$.

Appendix B
Plugins

A key feature in the first release of TauREx 3 allowed the user to inject their code into the retrieval pipeline from the input file. This feature allowed for the inclusion of new atmospheric models and parameters in the retrievals without extensive knowledge of the underlying systems within TauREx 3. However, these codes are considered exceptions from the standard TauREx 3 pipeline and required specific input keywords and files in order to function. This route also prevents the accumulation of enhancements to the framework provided from the feature because they must now obtain a specific code file. The distribution of such custom enhancements was cumbersome, especially for FORTRAN and C++ codes, as they would require additional manual stages before they could be used. In most cases, developers directly placed their code into their copy of the TauREx 3 codebase. This approach, while valid, runs the risk of splintering TauREx 3 into multiple, incompatible versions. The latest version of TauREx 3 (3.1+) remedies this problem through the use of *plugins*. Plugins are, in essence, installable enhancements to TauREx 3. They allow new features to be included in the pipeline natively without modifying the main TauREx 3 codebase. A plugin author can build new chemistries, profiles, models, and optimizers and distribute them to other users through the Python packaging system.

To give an example, when TauREx 3.1 is installed (or upgraded from version 3.0), only `free` chemistry is available. An arbitrary input file for that may look like this:

```
[Chemistry]
chemistry_type = free
fill_gases = H2,He
ratio = 0.17
 &H2O;
 gas_type = constant
 mix_ratio = 1e-3
 &NH3;
 gas_type = constant
 mix_ratio = 1e-5
 [Fitting]
H2O:fit = True
H2O:priors = 'LogGaussian(mean = -1,std = 0.25)''
```

No other chemical model is available. However, if an equilibrium model such as GGchem (Woitke et al. 2018) is desired, the user can run

```
pip install taurex_ggchem
```

At which point, the plugin is automatically downloaded, compiled if needed (or prebuilt binaries used) and installed. Now GGchem becomes available and can be utilized in the input file with its own set of input keywords and retrieval parameters:

```
[Chemistry]
chemistry_type = ggchem
equilibrium_condensation = True
metallicity = 1.0
selected_elements = H, He, C, N, O
new_back_it = 6
include_charge = False
[Fitting]
C_O_ratio:fit = True
C_O_ratio:priors = 'Uniform(bounds = [1e-2, 2.0])''
```

On initialization, TauREx 3 searches for modules in the Python environment for these plugins and adds any new keywords to be parsed in the input file. TauREx 3 does not assume the inclusion of its inbuilt classes and parameters and will perform the same search of parameters and models on itself. The plugins function, like any other Python library, can be installed from PyPi and git through `pip install`, which significantly improves the ability to distribute plugins to other users. This ability to be installed also allows for the compilation stages of FORTRAN and C++ codes to be completely automated and transparent to the user, significantly improving their usability. Importantly, the original author has full control of development, and is able to host and develop their own plugins independently of TauREx 3. The plugins for our source control forks of the FastChem (Stock et al. 2018) and GGchem equilibrium chemistries introduce minor or no changes to the original code. Instead, the installation stage references and compiles against the chemistry author's original FORTRAN/C++ code. Once the TauREx Python wrappers and plugin setup files have been written, an author can continue development in their language of choice; leaving their original makefiles and overall compilation pipeline intact and allowing the plugin to benefit from their newest improvements. Development of these wrappers is also accessible. A developer can focus solely on converting TauREx inputs and units to their desired input, performing the calculation, and converting them into the TauREx expected outputs and units. As TauREx standardizes each atmospheric component, there is a guarantee that the implementation will fully function in the retrieval pipeline.

Plugins also solve another problem of distribution of codes. Some of these codes may have restrictive licenses that prevent them from being fully open-source. These plugins may instead be installed directly from private repositories or distributed as compiled Python code (i.e `.pyc` instead of `.py`) and binaries with licenses. A list of the available plugins is given in Table 8.

B.1. TauREx-CUDA

One of the first plugins developed provides GPU acceleration to the forward models and retrievals using the `PyCUDA` library. It is installed by executing `pip install taurex_cuda` and provides replacement models and contributions functions that take advantage of nVidia GPU cards. Once installed, utilizing the GPU requires replacing the forward models and contributions with `cuda` suffix versions in the input file. For example, a transmission model with Rayleigh scattering

```
[Model]
model_type = transmission
 &Absorption;
 [[Rayleigh]
```

can be GPU-enabled by replacing with TauREx-CUDA versions of the model and contributions:

```
[Model]
model_type = transmission-cuda
 &AbsorptionCUDA;
 [[RayleighCUDA]
```

The plugin also contains a cross section caching system that will move and reuse absorption and CIA cross sections within GPU memory. The GPU opacities work slightly differently to their CPU counterpart as they now perform interpolation and weighting of all layers in the atmosphere in parallel. Similarly the integration of the atmospheric layer works the same way, where rather than calculating layer by layer, the contributions now compute in parallel all wavelengths and layers in the calculation. All CUDA kernels within the plugin are generated on the fly, compiled and cached for later reuse. This approach allows for the generation of optimized CUDA code for a particular calculation. For example, on the first run, the calculation of the eclipse depth unwraps and inserts the values of the Gaussian quadrature weights and abscissa into the kernel source code. This completely eliminates global reads, significantly boosting performance. Table 9 highlights this approach as 50-point quadrature integration results in only a 40% increase in modeling time against a four-point quadrature on a V100 GPU compared to a 4× increase for the CPU version.

This is important to consider as previous studies (Changeat & Al-Refaie 2020) have shown that a minimum quadrature of 10 points is needed to achieve relatively good convergence.

Assessing performance, Table 10 shows the runtime of both transmission and emission models for a range of resolutions. The benchmarks were conducted on an Intel(R) Xeon(R) Gold 6130 CPU at 2.10 GHz with a single 16 GB nVidia V100 GPU. The performance gain from the CUDA-accelerated contribution functions is significant with up to 150× reduction in modeling time compared to the CPU implementation. For the line-by-line case ($R = 1{,}000{,}000$), memory pressure required limiting the benchmark to a single absorbing molecule. However, comparing like-for-like with the CPU version demonstrates a degree of viability in using line-by-line in retrievals in the transmission case. The current version of the CUDA plugin uses a lazy but straightforward memory management scheme. Cross sections are cached in their entirety in GPU memory. The results of their interpolation and the optical depth itself are also generated in GPU memory. In the line-by-line scheme, each of these arrays occupies roughly 4 GB of memory. The transmission case only requires 12 GB to complete, which just about fits in a 16 GB V100 GPU. The emission case requires two additional arrays, which bring the memory cost to 20 GB. An easy solution would be to use the 32 GB variant of the V100 or exploit the newer 40 GB A100 cards. Future versions of the plugin will aim to introduce a smarter memory management system that would significantly reduce memory usage through the exploitation of asynchronous memory transfers. With either the introduction of higher-memory GPUs or smarter memory algorithms, we hope to transition to line-by-line retrievals shortly.

All contribution functions are supported, including non-CUDA-enabled ones. TauREx-CUDA will perform any non-CUDA calculations first before transferring the results to the GPU and completing any remaining calculations. Finally, the TauREx-CUDA plugin can be used to build new CUDA-enabled contribution functions and models and also provides a

Table 8
List of Currently Available Plugins of TauREx 3.1

Plugin	Description	Availability
`taurex_cuda`	CUDA-acceleration of forward models	PyPI
`taurex_hip`	HIP-acceleration of forward models	PyPI
`taurex_ace`	Equilibrium chemistry using ACE	PyPI
`taurex_ggchem`	Equilibrium chemistry using GGchem	PyPI
`taurex_fastchem`	Equilibrium chemistry using FastChem	PyPI
`taurex_ultranest`	Ultranest sampler support for TauREx	PyPI
`taurex_dynesty`	Dynesty sampler support for TauREx	PyPI
`taurex_scipypriors`	`scipy.stat` continuous functions as priors	PyPI
`taurex_petitrad`	petitRADTRANS forward models and opacity formats	PyPi
`taurex_catalog`	Set planetary and stellar parameters from name	On publication
`taurex_uv`	UV stellar spectra slicing	On publication
`taurex_phasecurve`	1.5D phase-curve forward models	On publication
`taurex_jwst`	JWST instrument noise simulator	On publication

Note.
Availability describes where a user may acquire the plugin.
PyPI availability means that the plugin can be acquired using pip install.
On publication refers to plugins that will be available after their relevant publication.

Table 9
CPU and GPU Computation Times for an Emission Model Using a Varying Number of Gaussian Quadrature Points

N_{quads}	CPU (ms)	GPU (ms)	Speed-up
4	1170	21.2	55
8	1280	23.6	54
16	1500	24.1	62
32	2140	33.4	64
50	4560	37.2	122

Note. The atmosphere contains 100 layers with five actively absorbing molecules, CIA, and Rayleigh scattering.

Table 10
CPU and GPU Computation Times for Transmission and a Four-point Quadrature Emission Model for a Range of Resolutions over the Full Wavelength Range of 0.3 μm–15 μm

	Transmission		Emission	
R	CPU (ms)	GPU (ms)	CPU (ms)	GPU (ms)
7,000	1319	10	1777	18
10,000	1864	12	2758	12
15,000	3055	16	4501	15
1,000,000[a]	121,580	406	63,040	n/a

Note. The atmosphere contains 100 layers with five actively absorbing molecules, CIA, and Rayleigh scattering. The benchmarks were conducted on an Intel(R) Xeon(R) Gold 6130 CPU at 2.10 GHz with a single 16 GB nVidia V100 GPU. Timing was conducted using the `timeit` Python module.
[a] Due to memory constraints of the GPU the $R = 1{,}000{,}000$ benchmarks were conducted without CIA and Rayleigh scattering and with only a single molecule. The source of the superhigh-resolution opacity is the line-by-line data from petitRADTRANS (Mollière et al. 2019) loaded using the taurex_petitrad plugin. It has a wavelength range of 0.1 μm–28 μm

GPU opacity caching system for accelerated interpolation of molecular cross sections.

Appendix C
Chemistry Plugins

The plugin system has now expanded the scope of chemistry modeling to include three equilibrium chemistries: ACE (Agúndez et al. 2012, 2020), FastChem (Stock et al. 2018), and GGchem (Woitke et al. 2018). All plugins can be installed through PyPi on Windows, Linux, and MacOS and all are capable of being used in optimizations to retrieve elemental abundances. ACE and GGchem were originally written in FORTRAN. To facilitate their inclusion, the numpy f2py module was used to automatically generate efficient and convenient Python wrappers as well as handle their compilation during installation. FastChem is a C++ code, so light wrappers were written using Cython. For all of these codes, a dedicated TauREx 3 chemistry interface was written and exposed to the plugin system. When installed through PyPI, the Python packaging system handles all of the wrapping and compilation in the background. For Windows and MacOS, PyPI will bypass compilation and instead download prebuilt binaries.

All chemistries present can be controlled and fit based on metallicity relative to their initial abundance (usually solar) and the ratios of each metal element relative to oxygen. For the ACE plugin, only C/O and N/O ratios can be fit. The FastChem and GGchem plugins will dynamically generate oxygen ratio fitting parameters (Al-Refaie et al. 2021) for each metal element selected by the user. For example, if the user selects C, N, S, and Ti then C/O, N/O, S/O, and Ti/O ratios can be fit during retrievals.

C.1. GGchem

The plugin compiles against the original GGchem FORTRAN source code during installation as well as introducing additional FORTRAN-90 glue code that directly interfaces with TauREx 3. The GGchem plugin allows for the user selection of elements. Hydrogen, helium, and oxygen must be present. If the user does not specify these then they are automatically included. The initial abundances of these elements can be chosen from one of four default profiles: `solar`, `earth`, `ocean`, or `meteorite`. Modification of abundances after initialization (and during retrievals) is controlled by the `metallicity` parameter, which determines the amount of oxygen relative to the initial profile and an `O_ratio` parameter that determines the ratio for each metal element relative to oxygen. The oxygen ratio fitting parameters are dynamically generated (Al-Refaie et al. 2021). The plugin also allows for equilibrium condensation, enabled at initialization by setting `equilibrium_condensation = True`, and ions enabled by setting `include_charge = True`. As ion chemistry defines electron abundance, it can be seamlessly used with the H^{-} opacity component (Appendix A.2).

The `taurex_ggchem` plugin, to the authors' knowledge, presents the only true Python wrapper to GGchem. All functionality of GGchem is present and keywords in the TauREx 3.1 input file match (or closely match) a GGchem input file. The plugin can be used outside of TauREx 3 to generate chemistry profiles through a simple Python interface:

```
>>> from taurex_ggchem import GGchem
>>>gg = GGchem(metallicity = 1.0,
 selected_elements = ['H','He','C','O','N','K'],
 abundance_profile = 'earthcrust',
 equilibrium_condensation = True)
>>> nlayers = 10
 >>> temperature = np.linspace(300,100,nlayers)
 >>> pressure = np.logspace(5,-3, nlayers) # Pa
 >>> gg.initialize_chemistry(nlayers,temperature,
  pressure)
 >>> gg.gases
['H', 'He', 'C', 'O', 'N',…, 'N3', 'O3', 'C3H']
>>>] gg.mixProfile
array([[4.75989782e-04, 4.93144149e-04, 5.10561665e-
  04, ...,
 2.89575385e-05, 2.47386006e-05, 2.10241059e-05],
 ...,
 [2.49670621e-16, 1.44224904e-16, 8.29805526e-17, ...,
 9.48249338e-42, 4.75884162e-42, 2.37999459e-42]])
 >>> gg.condensates
['C[s]', 'H2O[s]', 'H2O[l]', 'NH3[s]', 'CH4[s]', 'CO
  [s]', 'CO2[s]']
>>> gg.condensateMixProfile
array([[0.00000000e+00, 0.00000000e+00, 0.00000000e
  +00,…,
 0.00000000e+00, 0.00000000e+00],
 [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
 0.00000000e+00, 9.82922802e-10, 1.88551848e-10,
  2.88471985e-11,
 4.40651877e-12, 6.95597887e-13],
 …,
 [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
 0.00000000e+00, 0.00000000e+00]])
```

GGchem presents certain features of FORTRAN that make integration with Python more difficult. First it possesses a global state, meaning only one instance of GGchem can execute in a single Python script. Attempting to create two separate GGchem instances will either crash the code or generate incorrect results because the global state is being shared and modified between instances. An example of this would be if a user is attempting to generate forward models for multiple planets. The second issue comes from FORTRAN's `STOP` command, which at its lowest level is a system call that kills the running process.

Retrievals that encounter this halt and crash TauREx immediately. For GGchem, this is triggered when it cannot converge to a result. Rather than stopping completely, it would be more desirable to continue a retrieval while avoiding these regions. The simplest solution would be to refactor code to remove these issues but this can be a major undertaking for authors of these codes and inherently goes against the philosophy of the plugin system. An alternative solution is to use the `SafeFortranCaller` class introduced in TauREx 3.1, which circumvents these issues. The class utilizes the Python `multiprocessing` module to spawn a child Python process that will execute the FORTRAN code. User requests for reading and writing variables or subroutine calls are passed as messages to the child process, which will return the results to the parent process. With this method, the global state is now local to the child process. If another GGchem instance is required, a new child process is spawned. During a get/set/call request, the parent process will monitor the child process; if at some point the child process has died, it is assumed that a `STOP` command was called. The parent process will perform cleanup of hanging threads and raise a `FortranStopException`. On the next get/set/call request, a new process is spun up. An additional benefit to this class is that all writes to standard output in the FORTRAN code are also redirected back to TauREx 3 and logged. This is useful as FORTRAN outputs can now be hidden during retrieval.

The GGchem plugin utilizes this class extensively. Multiple instances can now be created and retrievals freely run. Anytime a convergence issue is encountered, the instance is destroyed, GGchem reinitialized, and an exception raised. The retrieval is informed of this and will avoid regions in the parameter space that cause it.

C.2. ACE

Previously the ACE equilibrium chemistry was built into the original TauREx 3. For the newest release it has been removed and turned into a plugin. This was to simplify the installation process of the main TauREx 3 code and removed the need for a FORTRAN compiler to be present in the user's computer. With this, TauREx 3 has a full Python stack. To restore the previous functionality, a user need only install the `taurex_ace` plugin in order utilize ACE as before.

C.3. FastChem

The FastChem plugin, like the GGchem plugin, allows for the calculation of ion chemistry and can be used outside of TauREx 3 to generate chemistry profiles using Python:

```
>>> from taurex_fastchem import FastChem
>>> fc = FastChem
  (selected_elements = ['H','He','C','O','N','K','e-
  '],
 with_ions = True, metallicity = 1.0)
>>> nlayers = 10
>>> temperature = np.linspace(300,100,nlayers)
>>>] pressure = np.logspace(5,-3, nlayers) # Pa
>>> fc.initialize_chemistry(nlayers,temperature,
  pressure)
>>> fc.gases
['H', 'He', 'O', 'C', 'K', 'N', 'e-', ..., 'O+', 'O-', 'O2
  +', 'O2-']
>>> fc.mixProfile
array([[3.87435866e-036, 9.95149979e-039,
  7.62616463e-042,
 1.23490910e-045, 2.58839801e-050, 3.41640407e-056,
 9.40930967e-064, 9.08433703e-074, 1.41255491e-087,
 1.38065040e-167],
 …,
 [1.42400626e-001, 1.42400626e-001, 1.42400626e-001,
 1.42400626e-001, 1.42400626e-001, 1.42400791e-001,
 1.42398731e-001, 1.42398284e-001, 1.42367067e-001,
 9.96186945e-001]])
```

Under the hood, the FastChem plugin generates parameter and custom elemental abundance files in a temporary directory that is then passed into the FastChem library. Doing this allows us to vary the elemental abundances through the oxygen ratio and metallicity parameters. Like GGchem, the retrieval parameters include metallicity, which controls the amount of oxygen relative to the initial abundance, and an `O_ratio` parameter that determines the ratio for each selected metal element relative to oxygen. The initial abundance can be modified by passing-in FastChem abundance files through the `elements_abundance_file` keyword. Once more, enabling ions, FastChem computes electron abundances that can be used by the H^- opacity code (Appendix A.2).

ORCID iDs

A. F. Al-Refaie https://orcid.org/0000-0003-2241-5330
Q. Changeat https://orcid.org/0000-0001-6516-4493
O. Venot https://orcid.org/0000-0003-2854-765X
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. C. 2011, JPCA, 115, 6805
Abel, M., Frommhold, L., Li, X., & Hunt, K. L. C. 2012, JChPh, 136, 044319
Agúndez, M., Martínez, J. I., de Andres, P. L., Cernicharo, J., & Martín-Gago, J. A. 2020, A&A, 637, A59
Agúndez, M., Venot, O., Iro, N., et al. 2012, A&A, 548, A73
Al-Refaie, A. F., Changeat, Q., Waldmann, I. P., & Tinetti, G. 2021, ApJ, 917, 37
Allard, F., Hauschildt, P. H., Alexander, D. R., Tamanai, A., & Schweitzer, A. 2001, ApJ, 556, 357
Allard, N. F., Spiegelman, F., & Kielkopf, J. F. 2016, A&A, 589, A21
Allard, N. F., Spiegelman, F., Leininger, T., & Molliere, P. 2019, A&A, 628, A120
Allen, M., Yung, Y. L., & Waters, J. W. 1981, JGR, 86, 3617
Anisman, L. O., Edwards, B., Changeat, Q., et al. 2020, AJ, 160, 233
Atkinson, R., Baulch, D. L., Cox, R. A., et al. 2006, ACP, 6, 3625
Barklem, P. S., & Collet, R. 2016, A&A, 588, A96
Barton, E. J., Hill, C., Yurchenko, S. N., et al. 2017, JQSRT, 187, 453
Benneke, B. 2015, arXiv:1504.07655
Benneke, B., Wong, I., Piaulet, C., et al. 2019, ApJL, 887, L14
Berline, S., & Bricker, C. 1969, JChEd, 46, 499
Bounaceur, R., Herbinet, O., Fournet, R., et al. 2010, SAE 2010 World Congress Exhibition (SAE International), 2010, 2010-01-0546
Bourrier, V., Kitzmann, D., Kuntzer, T., et al. 2020, A&A, 637, A36
Buchner, J., Georgakakis, A., Nandra, K., et al. 2014, A&A, 564, A125
Burrows, A., & Sharp, C. M. 1999, ApJ, 512, 843
Changeat, Q., & Al-Refaie, A. 2020, ApJ, 898, 155
Changeat, Q., Al-Refaie, A., Mugnai, L. V., et al. 2020a, AJ, 160, 80
Changeat, Q., Edwards, B., Al-Refaie, A. F., et al. 2020b, AJ, 160, 260
Changeat, Q., Al-Refaie, A. F., Edwards, B., Waldmann, I. P., & Tinetti, G. 2021, ApJ, 913, 73
Changeat, Q., & Edwards, B. 2021, ApJL, 907, L22
Changeat, Q., Edwards, B., Waldmann, I. P., & Tinetti, G. 2019, ApJ, 886, 39
Charbonneau, D., Brown, T. M., Noyes, R. W., & Gilliland, R. L. 2002, ApJ, 568, 377
Chase, M. W. 1986, JANAF Thermochemical Tables (3rd; Washington, D.C.: American Chemical Society)
Cho, J. Y. K., Menou, K., Hansen, B. M. S., & Seager, S. 2008, ApJ, 675, 817
Chubb, K. L., Tennyson, J., & Yurchenko, S. N. 2020, MNRAS, 493, 1531
Cox, A. N. 2015, Allen's Astrophysical Quantities (Berlin: Springer)
Cubillos, P. E., & Blecic, J. 2021, MNRAS,
de Wit, J., Wakeford, H. R., Gillon, M., et al. 2016, Natur, 537, 69
de Wit, J., Wakeford, H. R., Lewis, N. K., et al. 2018, NatAs, 2, 214
Deming, D., Wilkins, A., McCullough, P., et al. 2013, ApJ, 774, 95
Edwards, B., Changeat, Q., Baeyens, R., et al. 2020, AJ, 160, 8
Edwards, B., Changeat, Q., Mori, M., et al. 2021, AJ, 161, 44
Eriksson, G., Holm, J. L., Welch, B. J., et al. 1971, Acta Chem. Scand., 25, 2651
Feng, Y. K., Robinson, T. D., Fortney, J. J., et al. 2018, AJ, 155, 200
Feroz, F., Hobson, M. P., & Bridges, M. 2009, MNRAS, 398, 1601
Fletcher, L. N., Gustafsson, M., & Orton, G. S. 2018, ApJS, 235, 24
Gail, H.-P., & Sedlmayr, E. 1986, A&A, 166, 225
Gandhi, S., & Madhusudhan, N. 2018, MNRAS, 474, 271
Gandhi, S., Madhusudhan, N., & Mandell, A. 2020, AJ, 159, 232
Gordon, S., & McBride, B. J. 1994, Computer Program for Calculation of Complex Chemical Equilibrium Compositions and Applications. Part 1: Analysis, NASA Reference Publication, NASA Technical Report 1311
Greene, T. P., Line, M. R., Montero, C., et al. 2016, ApJ, 817, 17
Grillmair, C. J., Burrows, A., Charbonneau, D., et al. 2008, Natur, 456, 767
Guilluy, G., Gressier, A., Wright, S., et al. 2021, AJ, 161, 19
Harrington, J. 2016, Atmospheric Retrievals from Exoplanet Observations and Simulations with BART, NASA proposal 16-XPR16-10
Haynes, K., Mandell, A. M., Madhusudhan, N., Deming, D., & Knutson, H. 2015, ApJ, 806, 146
Hill, C., Yurchenko, S. N., & Tennyson, J. 2013, Icar, 226, 1673
Irwin, P. G. J., Teanby, N. A., de Kok, R., et al. 2008, JQSRT, 109, 1136
John, T. L. 1988, A&A, 193, 189
Kirk, J., Rackham, B. V., MacDonald, R. J., et al. 2021, AJ, 162, 34
Kitzmann, D., Heng, K., Oreshenko, M., et al. 2020, ApJ, 890, 174
Kreidberg, L., Bean, J. L., Désert, J.-M., et al. 2014, Natur, 505, 69
Lavie, B., Mendonça, J. M., Mordasini, C., et al. 2017, AJ, 154, 91
Lee, J. M., Fletcher, L. N., & Irwin, P. G. J. 2012, MNRAS, 420, 170
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., Knutson, H., Wolf, A. S., & Yung, Y. L. 2014, ApJ, 783, 70
Line, M. R., Wolf, A. S., Zhang, X., et al. 2013, ApJ, 775, 137
Lodders, K. 2002, ApJ, 577, 974
Lodders, K., & Fegley, B. 2002, Icar, 155, 393
MacDonald, R. J., & Madhusudhan, N. 2017, MNRAS, 469, 1979
MacDonald, R. J., & Madhusudhan, N. 2019, MNRAS, 486, 1292
Madhusudhan, N., Nixon, M. C., Welbanks, L., Piette, A. A. A., & Booth, R. A. 2020, ApJL, 891, L7
Madhusudhan, N., & Seager, S. 2009, ApJ, 707, 24
Marley, M. S., Ackerman, A. S., Cuzzi, J. N., & Kitzmann, D. 2013, in Comparative Climatology of Terrestrial Planets, Vol. 15, ed. S. J. Mackwell et al. (Tucson, AZ: Univ. Arizona Press), 367
McBride, B. J., Gordon, S., & Reno, M. A. 1993, Coefficients for calculating thermodynamic and transport properties of individual species, NASA Technical Memorandum, 4513
McKemmish, L. K., Masseron, T., Hoeijmakers, H. J., et al. 2019, MNRAS, 488, 2836
McKemmish, L. K., Yurchenko, S. N., & Tennyson, J. 2016, MNRAS, 463, 771
Mesa, D., D'Orazi, V., Vigan, A., et al. 2020, MNRAS, 495, 4279
Mikal-Evans, T., Sing, D. K., Kataria, T., et al. 2020, MNRAS, 496, 1638
Mikal-Evans, T., Sing, D. K., Goyal, J. M., et al. 2019, MNRAS, 488, 2222
Min, M., Ormel, C. W., Chubb, K., Helling, C., & Kawashima, Y. 2020, A&A, 642, A28
Mollière, P., Wardenier, J. P., van Boekel, R., et al. 2019, A&A, 627, A67
Moses, J. I., Visscher, C., Fortney, J. J., et al. 2011, ApJ, 737, 15
Mugnai, L. V., Modirrousta-Galian, D., Edwards, B., et al. 2021, AJ, 161, 284
Muller, C., Michel, V., Scacchi, G., & Côme, G. 1995, JCP, 92, 1154
Nikolov, N., Sing, D. K., Fortney, J. J., et al. 2018, Natur, 557, 526
Ormel, C. W., & Min, M. 2019, A&A, 622, A121
Pinhas, A., Madhusudhan, N., Gandhi, S., & MacDonald, R. 2019, MNRAS, 482, 1485
Pluriel, W., Whiteford, N., Edwards, B., et al. 2020, AJ, 160, 112
Polyansky, O. L., Kyuberis, A. A., Zobov, N. F., et al. 2018, MNRAS, 480, 2597
Rothman, L., Gordon, I., Barber, R., et al. 2010, JQSRT, 111, 2139
Sharp, C. M., & Huebner, W. F. 1990, ApJS, 72, 417
Showman, A. P., Fortney, J. J., Lian, Y., et al. 2009, ApJ, 699, 564
Sing, D. K., Fortney, J. J., Nikolov, N., et al. 2016, Natur, 529, 59
Skaf, N., Bieger, M. F., Edwards, B., et al. 2020, AJ, 160, 109
Skinner, J. W., & Cho, J. Y. K. 2021, MNRAS, 504, 5172
Stassun, K. G., Collins, K. A., & Gaudi, B. S. 2017, AJ, 153, 136
Stevenson, K. B., Désert, 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
Stock, J. W., Kitzmann, D., Patzer, A. B. C., & Sedlmayr, E. 2018, MNRAS, 479, 865
Swain, M. R., Vasisht, G., & Tinetti, G. 2008, Natur, 452, 329
Swain, M. R., Vasisht, G., Tinetti, G., et al. 2009a, ApJL, 690, L114
Swain, M. R., Tinetti, G., Vasisht, G., et al. 2009b, ApJ, 704, 1616
Swain, M. R., Estrela, R., Roudier, G. M., et al. 2021, AJ, 161, 213

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., Vidal-Madjar, A., Liang, M.-C., et al. 2007, Natur, 448, 169
Tinetti, G., Eccleston, P., Haswell, C., et al. 2021, arXiv:2104.04824
Tsiaras, A., Waldmann, I. P., Tinetti, G., Tennyson, J., & Yurchenko, S. N. 2019, NatAs, 3, 1086
Tsiaras, A., Rocchetto, M., Waldmann, I. P., et al. 2016, ApJ, 820, 99
Tsiaras, A., Waldmann, I. P., Zingales, T., et al. 2018, AJ, 155, 156
Tsuji, T. 1973, A&A, 23, 411
Underwood, D. S., Tennyson, J., Yurchenko, S. N., et al. 2016, MNRAS, 459, 3890
Venot, O., Cavalié, T., Bounaceur, R., et al. 2020, A&A, 634, A78
Venot, O., Hébrard, E., Agúndez, M., et al. 2012, A&A, 546, A43
Welbanks, L., Madhusudhan, N., Allard, N. F., et al. 2019, ApJL, 887, L20
White, W. B., Johnson, S. M., & Dantzig, G. B. 1958, JChPh, 28, 751
Woitke, P., Helling, C., Hunter, G. H., et al. 2018, A&A, 614, A1
Yip, K. H., Changeat, Q., Edwards, B., et al. 2021, AJ, 161, 4
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