equilibrium-c: A Lightweight Modern Equilibrium Chemistry Calculator

Dr. Nick Gibbons

The University of Queensland

Monday, 7th July

About Me!

I am Nick:

- PhD in supersonic combustion, 2019
- Started as postdoctoral fellow @ UQ in 2020
- Senior Engineer (Eilmer) and Hypersonic CFD Researcher

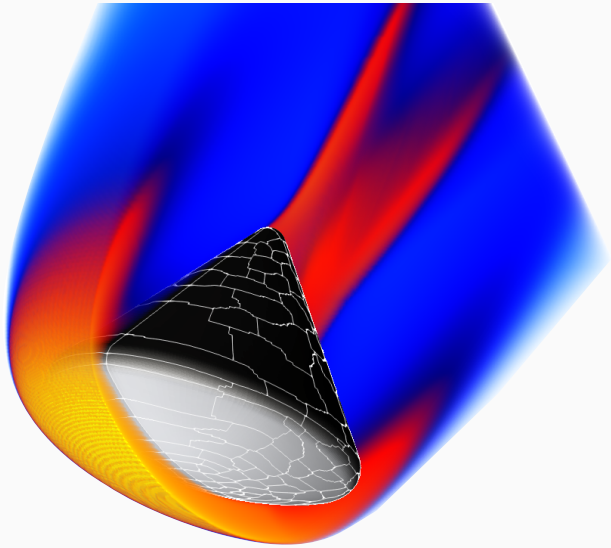
n.gibbons@uq.edu.au

nicholas-gibbons-67342555

About Me: The Gasdynamics Toolkit

GDTk is a collection of software tools for analysing hypersonic flow:

- Includes our flagship compressible flow code Eilmer
- Specialised tools for facility design and more
- Maintained at UQ and UniSQ
- Free and Open-Source



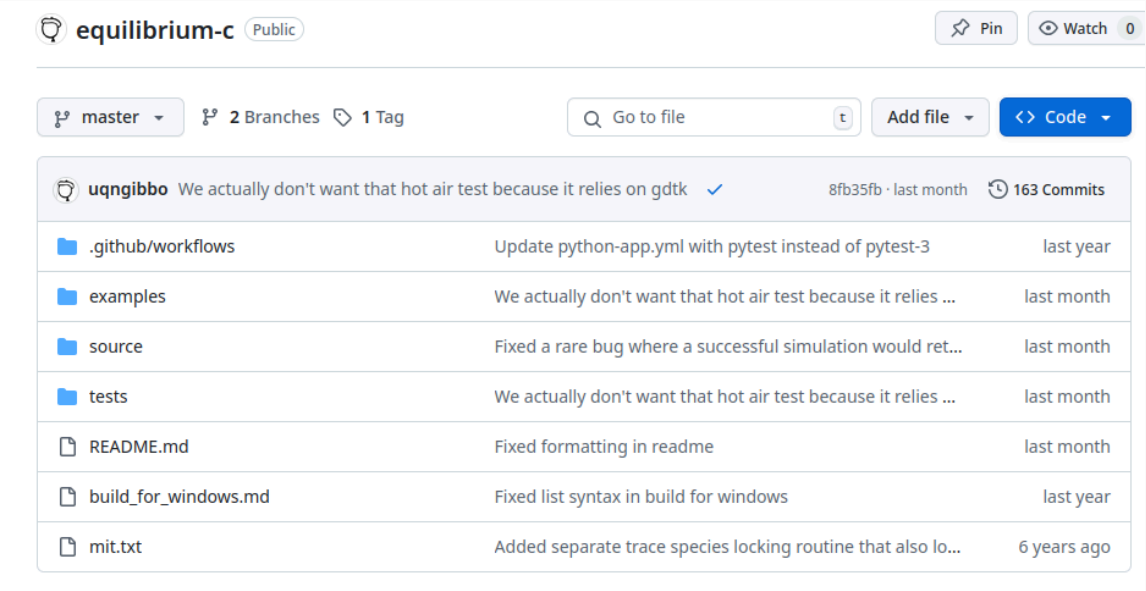
Apollo capsule at 18.6° angle of attack

Project Website:

Today's Talk: A Modernised Approach for Solving Chemical Equilibrium

I've written an equilibrium chemistry solver called equilibrium-c:

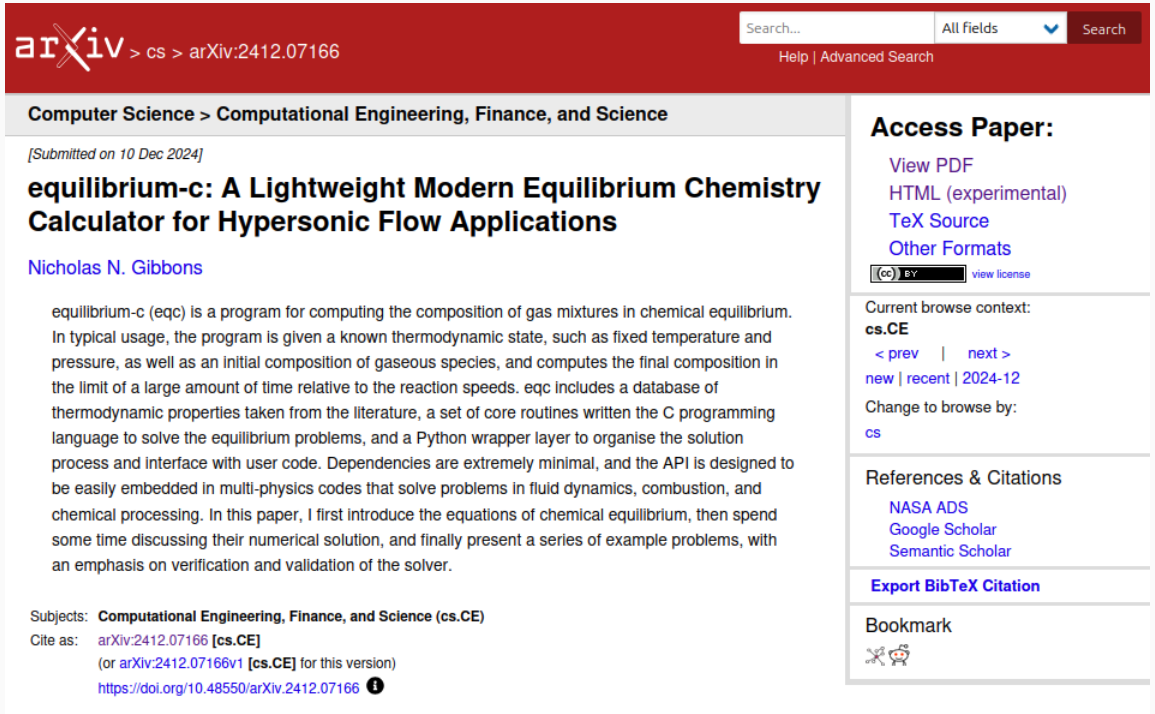
- Hybrid C/Python solution of the thermochemimcal equations [1]
- Improved numerical method for tighter convergence and more stability
- Automated tests, better performance, easier scripting etc.



Link:

Today's Talk: A Modernised Approach for Solving Chemical Equilibrium

There's also a paper (preprint) which has much more detail:



arXiv > cs > arXiv:2412.07166

Search... All fields Search

Help | Advanced Search

Computer Science > Computational Engineering, Finance, and Science

[Submitted on 10 Dec 2024]

equilibrium-c: A Lightweight Modern Equilibrium Chemistry Calculator for Hypersonic Flow Applications

Nicholas N. Gibbons

equilibrium-c (eqc) is a program for computing the composition of gas mixtures in chemical equilibrium. In typical usage, the program is given a known thermodynamic state, such as fixed temperature and pressure, as well as an initial composition of gaseous species, and computes the final composition in the limit of a large amount of time relative to the reaction speeds. eqc includes a database of thermodynamic properties taken from the literature, a set of core routines written the C programming language to solve the equilibrium problems, and a Python wrapper layer to organise the solution process and interface with user code. Dependencies are extremely minimal, and the API is designed to be easily embedded in multi-physics codes that solve problems in fluid dynamics, combustion, and chemical processing. In this paper, I first introduce the equations of chemical equilibrium, then spend some time discussing their numerical solution, and finally present a series of example problems, with an emphasis on verification and validation of the solver.

Subjects: Computational Engineering, Finance, and Science (cs.CE)
Cite as: arXiv:2412.07166 [cs.CE]
(or arXiv:2412.07166v1 [cs.CE] for this version)
https://doi.org/10.48550/arXiv.2412.07166

Access Paper:
View PDF
HTML (experimental)
TeX Source
Other Formats
view license

Current browse context:
cs.CE
< prev | next >
new | recent | 2024-12
Change to browse by:
cs

References & Citations
NASA ADS
Google Scholar
Semantic Scholar

Export BibTeX Citation

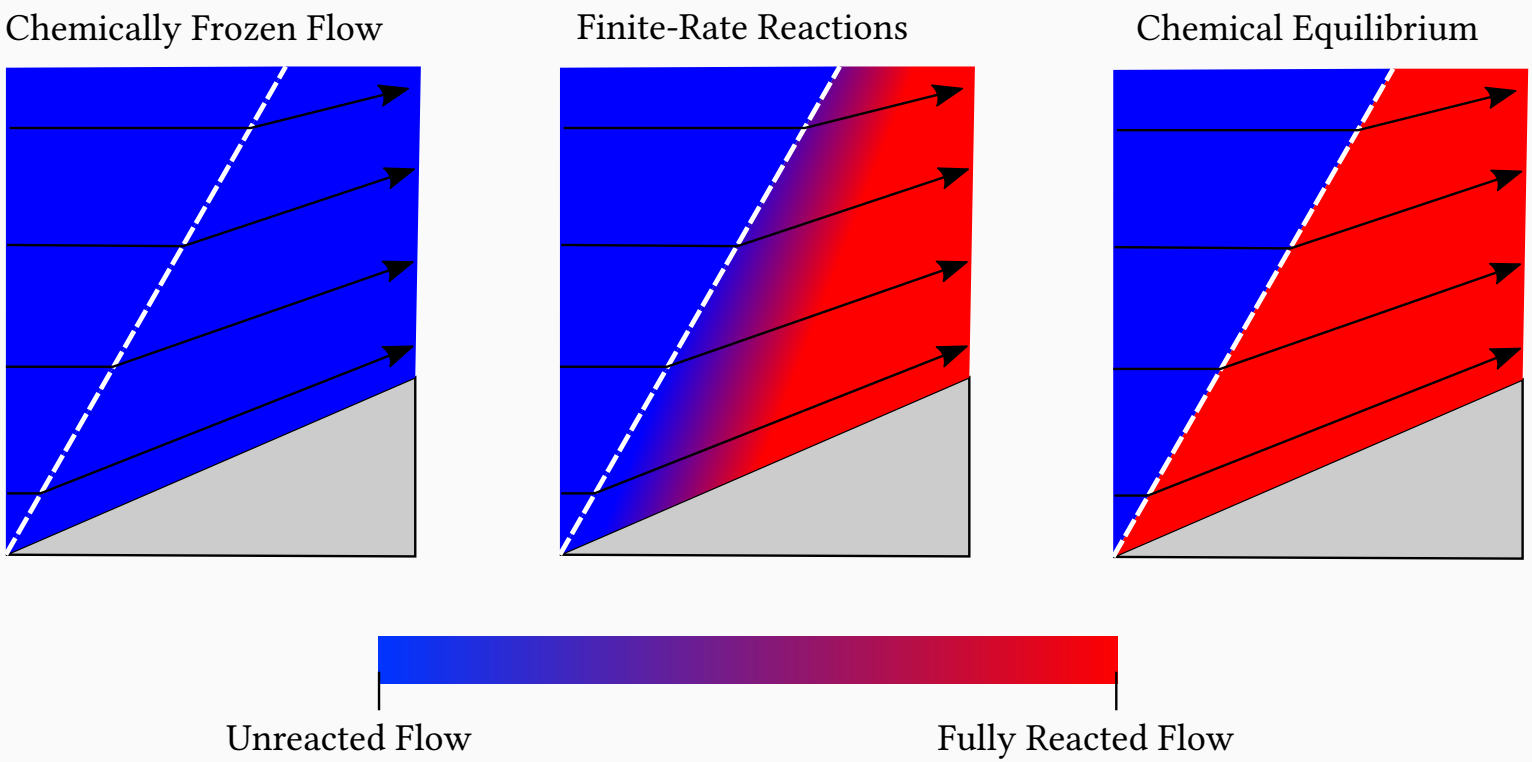
Bookmark

Link:

Why Equilibrium Chemistry?

Chemical Reactions are a hallmark of hypersonic flow:

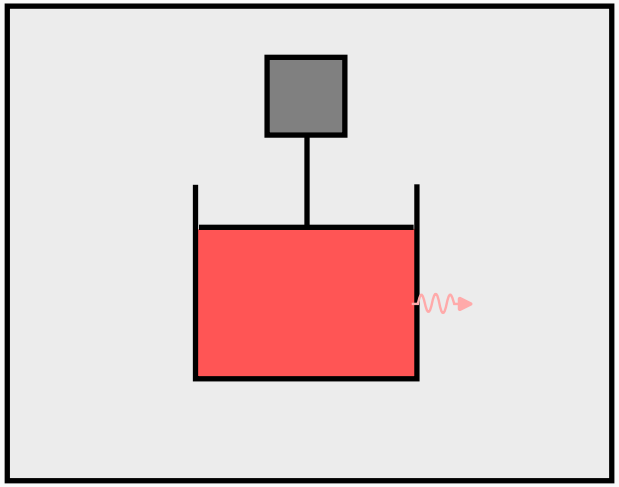
- When reactions are fast, much simplification is possible



How do we solve for Thermochemical Equilibrium?

For a gas in a fixed temperature and pressure scenario, minimise the Gibbs Energy, G:

$$G = U + pV - TS$$



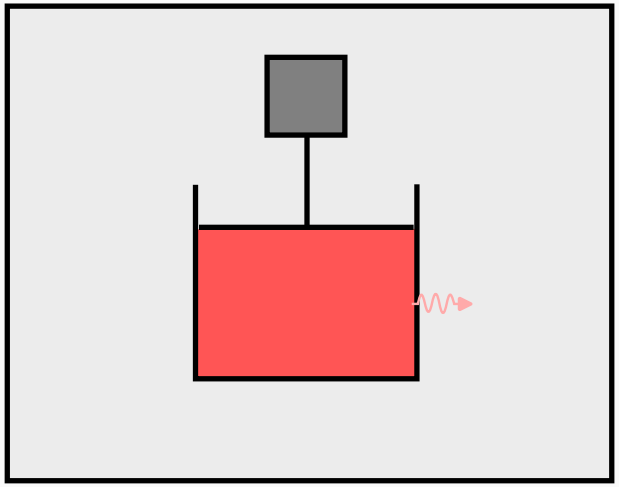
How do we solve for Thermochemical Equilibrium?

For a gas in a fixed temperature and pressure scenario, minimise the Gibbs Energy, G:

$$G = U + pV - TS$$

Introduce the unusual composition quantity n_s:

$$n_s = \frac{N_s}{\rho V}$$



How do we solve for Thermochemical Equilibrium?

For a gas in a fixed temperature and pressure scenario, minimise the Gibbs Energy, G:

$$G = U + pV - TS$$

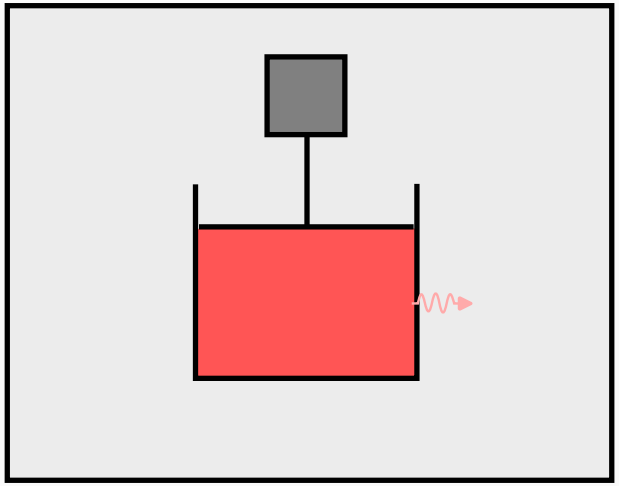
Introduce the unusual composition quantity n_s:

$$n_s = \frac{N_s}{\rho V}$$

Solve for the minima of the specific Gibbs energy g:

$$g = \sum_s n_s G_s(p, T, n_0, n_1, n_2, \ldots)$$

$$\frac{\partial g}{\partial n_s} = 0 \qquad s = \left(0, 1, \ldots N_{\text{species}}\right)$$



A Constrained Minimisation Problem

We have some constraints, usually a fixed initial composition: n_s^0

- Atomic nucleai are conserved in chemical reactions, so the constraint can be written:

$$\sum_j a_{js} n_s - a_{js} n_s^0 = 0$$

- Matrix a keeps track of which elements are in which species. For example:

$$CO_2 \leftrightharpoons CO + \frac{1}{2} O_2$$

$$\begin{array}{cc} & \\ a = & \begin{bmatrix} 1 & 1 & 0 \\ 2 & 1 & 2 \end{bmatrix} \begin{array}{l} \leftarrow C \\ \leftarrow O \end{array} \\ & \begin{array}{ccc} CO_2 & CO & O_2 \end{array} \end{array}$$

A Constrained Minimisation Problem

Using the Method of Lagrange Multipliers

$$\mathcal{L} = \sum_s n_s G_s(p, T, n_s) + \sum_j \lambda_j \left(\sum_j a_{js} n_s - a_{js} n_s^0 \right)$$

Solve for $\frac{\partial \mathcal{L}}{\partial n_s} = 0$ and $\frac{\partial \mathcal{L}}{\partial \lambda_j} = 0$:

A Constrained Minimisation Problem

Using the Method of Lagrange Multipliers

$$\mathcal{L} = \sum_s n_s G_s(p, T, n_s) + \sum_j \lambda_j \left(\sum_j a_{js} n_s - a_{js} n_s^0 \right)$$

Solve for $\frac{\partial \mathcal{L}}{\partial n_s} = 0$ and $\frac{\partial \mathcal{L}}{\partial \lambda_j} = 0$:

$$\frac{H_s - TS_s^\circ}{R_u T} + \ln\left(\frac{n_s}{n}\right) + \ln\left(\frac{p}{p^\circ}\right) + \sum_j \lambda_j \frac{a_{js}}{R_u T} = 0 \qquad (s = 0, \ldots, N_{\text{species}})$$

$$\sum_s a_{js} n_s - a_{js} n_s^0 = 0 \qquad (j = 0, \ldots, N_{\text{elem}})$$

$$\sum_s n_s - n = 0$$

Numerical Method: Custom Multidimensional Newton Solver

Start with a guess and iteratively solve a matrix expression for corrections:

$$J\Delta\tilde{x} = -F(\tilde{x})$$

Numerical Method: Custom Multidimensional Newton Solver

Start with a guess and iteratively solve a matrix expression for corrections:

$$J\Delta\tilde{x} = -F(\tilde{x})$$

A couple of complicating factors:

- We actually want to solve for $\ln(n_s)$
- Most of the rows of J have only a single entry in them
- We can borrow an analytic substitution trick from [1]

Numerical Method: Custom Multidimensional Newton Solver

Each step solves a linear system for equations:

$$\sum_i \sum_s a_{js} a_{is} n_s \pi_i + \sum_s a_{js} n_s \Delta \ln(n) = \sum_s a_{js} n_s^0 - a_{js} n_s + \sum_s \frac{a_{js} n_s G_s}{R_u T}$$

$$(j = 0, \ldots, N_{\text{elem}})$$

$$\sum_i \sum_s a_{is} n_s \pi_i + \left(\sum_s n_s - n \right) \Delta \ln(n) = n - \sum_s n_s + \sum_s \frac{n_s G_s}{R_u T}$$

Then get the change in composition directly:

$$\Delta \ln(n_s) = \Delta \ln(n) + \sum_j a_{sj} \pi_i - \frac{G_s}{R_u T}$$

Problems

Of course, it doesn't always work:

- Underrelaxation factor Λ based on $\sum_s n$

$$\Lambda = \min\left(1, \frac{1}{2}\frac{|\ln(n)|}{|\Delta \ln(n_s)|}\right)$$

$$\ln(n_s)^{\text{new}} = \ln(n_s)^{\text{old}} + \Lambda \Delta \ln(n_s)$$

Problems

Of course, it doesn't always work:

- Underrelaxation factor Λ based on $\sum_s n$

$$\Lambda = \min\left(1, \frac{1}{2}\frac{|\ln(n)|}{|\Delta \ln(n_s)|}\right)$$

$$\ln(n_s)^{\text{new}} = \ln(n_s)^{\text{old}} + \Lambda\Delta \ln(n_s)$$

- Solve for $\ln(n_s)$ and compute n_s, NEVER the other way around

$$n_s \times \ln(n_s) \to 0$$

Problems

Of course, it doesn't always work:

- Underrelaxation factor Λ based on $\sum_s n$

$$\Lambda = \min\left(1, \frac{1}{2}\frac{|\ln(n)|}{|\Delta \ln(n_s)|}\right)$$

$$\ln(n_s)^{\text{new}} = \ln(n_s)^{\text{old}} + \Lambda \Delta \ln(n_s)$$

- Solve for $\ln(n_s)$ and compute n_s, NEVER the other way around

$$n_s \times \ln(n_s) \to 0$$

- Check for convergence using a modified residual:

$$\varepsilon = \sqrt{\sum_s n_s F_s^2 + \sum_j F_j^2 + F_n^2}$$

Problems

Of course, it doesn't always work:

- Check for convergence using a modified residual:

$$\varepsilon = \sqrt{\sum_s n_s F_s^2 + \sum_j F_j^2 + F_n^2}$$

$$F_s = \frac{H_s - TS_s^\circ}{R_u T} + \ln\left(\frac{n_s}{n}\right) + \ln\left(\frac{p}{p^\circ}\right) + \sum_j \lambda_j \frac{a_{js}}{R_u T} \qquad (s = 0, \ldots, N_{\text{species}})$$

$$F_j = \sum_s a_{js} n_s - a_{js} n_s^0 = 0 \qquad (j = 0, \ldots, N_{\text{elem}})$$

$$F_n = \sum_s n_s - n$$

Example Calculation: 5 Species Air

Lots of facility calculations use a mixture of N_2, O_2, NO, N, O:

```
from numpy import array
import eqc

spnames = ['N2', 'O2', 'N', 'O', 'NO']
T = 2500.0
p = 0.1*101.35e3
Xs0 = array([0.767, 0.233, 0.0, 0.0, 0.0])

eq = eqc.EqCalculator(spnames)
Xs1 = eq.pt(p, T, Xs0, 1)

print("Xs1: ", Xs1)
```

Example Calculation: 5 Species Air

Lots of facility calculations use a mixture of N_2, O_2, NO, N, O:

- Let's compare against CEA [1]:

	N_2	O_2	NO	N	O
eqc	0.747849	0.209004	7.93101e-07	0.0207964	0.0223493
CEA	0.74785	0.20900	7.93200e-07	0.020799	0.022349
error	0.0%	0.0%	0.01%	0.01%	0.0

Example Calculation: 5 Species Air

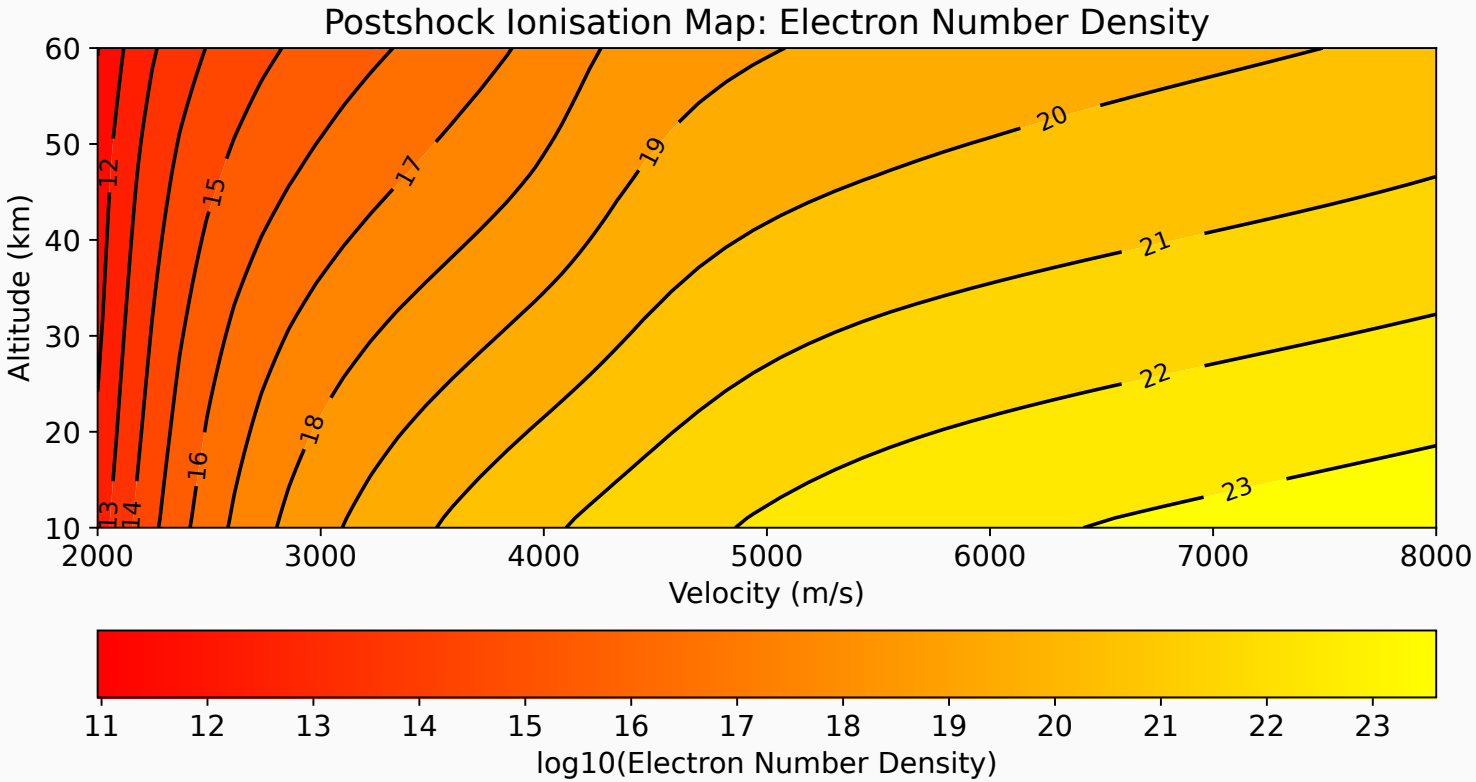
Lots of facility calculations use a mixture of N_2, O_2, NO, N, O:

- Let's compare against CEA [1]:
- We can verify by finite differencing the Lagrangian $\mathcal{L}$

	N_2	O_2	NO	N	O
eqc	0.747849	0.209004	7.93101e-07	0.0207964	0.0223493
CEA	0.74785	0.20900	7.93200e-07	0.020799	0.022349
error	0.0%	0.0%	0.01%	0.01%	0.0
$\frac{\frac{d\mathcal{L}}{dn_s}}{\mathcal{L}}$	0.0	−2.29e-10	0.0	0.0	−2.14e-9

Thanks!

- The GDTk Team: Rowan, PJ, Kyle, Reece, and Rob
- Vince Wheatley
- Typst



Bibliography

[1] S. Gordon and B. J. McBride, “Computer Program for Calculation of Complex Chemical Equilibrium Compositions and Applications,” 1994.