Numerical Modeling of Incomplete Combustion in a Liquid Chemical Rocket Engine

by

Anthony M. Hengst

Honors Thesis
Appalachian State University

Submitted to the Department of Mathematics
and The Honors College
in partial fulfillment of the requirements for the degree of
Bachelor of Science

December, 2020

Approved by:

Holly Hirst, Ph. D., Thesis Director

Jefferson Bates, Ph. D., Thesis Director

William J. Cook, Ph. D., Departmental Honors Director

Jefford Vahlbusch, Ph. D., Dean, Honors College

Abstract

Chemical rocket engines require very accurate modeling of chemical combustion to be optimized and safely developed. One may solve for chemical equilibrium by using the thermodynamic technique of minimizing the free energy of the system subject to mass-balance constraints. This is computationally accomplished by using Lagrange multipliers to constrain the system and using Newton's method for multivariate root finding to identify free-energy minima within the constraints.

In a manner suitable for a mathematical audience loosely familiar with chemistry, we will introduce the chemical problem which produces the constrained optimization problem Gordon and McBride seek to solve. We provide an overview of the concept of Lagrange multipliers and how they are used to transform constrained optimization problems into simpler unconstrained optimization problems, for both singly- and multiply-constrained systems. Further, we build upon the familiar Newton's method for single-variate root finding to explain how Newton's method for multivariate rootfinding works.

From these concepts and first thermodynamic principles, we derive a model of the chemical problem which may be solved by numerical methods. We present this algorithm which will solve for the equilibrium state of arbitrary incomplete combustion problems and demonstrate it as an operational Python program. We evaluate its accuracy and capability by comparing it to the algorithm of Gordon and McBride, which has been made publicly available as the program NASA Chemical Equilibrium with Applications (CEA).

Contents

List of Figures

List of Tables

Glossary of Thermodynamic Symbols

- a_{ij}: the number of atoms of element i in a molecule of species j, dimensionless
- b_i: specific number of mols of atoms of an element i, units of $\mathrm{mol\,kg^{-1}}$
- $b_{0,i}$: initial specific number of mols of atoms of an element i, units of $\mathrm{mol\,kg^{-1}}$
- G: Gibbs energy, units of J
- g: total specific Gibbs energy, units of $\mathrm{J\,kg^{-1}}$
- g_j: specific Gibbs energy of a species j, units of $\mathrm{J\,kg^{-1}}$
- H: enthalpy, units of J
- h: specific enthalpy, units of $\mathrm{J\,kg^{-1}}$
- h_j: specific enthalpy of a species j, units of $\mathrm{J\,kg^{-1}}$
- h_j°: specific enthalpy of a species j at standard pressure (1 bar), units of $\mathrm{J\,kg^{-1}}$
- i: an index, typically indicating a type of element
- j: an index, typically indicating a type of species
- N: number of mols of all gaseous species, units of mol
- N_j: number of mols of a species j, units of mol
- n: specific number of mols of all gaseous species, units of $\mathrm{mol\,kg^{-1}}$
- n_j: specific number of mols of a species j, units of $\mathrm{mol\,kg^{-1}}$
- P: pressure, units of $\mathrm{N\,m^{-2}}$
- P_j: partial pressure of a species j, equal to $P \cdot n_j/n$, units of $\mathrm{N\,m^{-2}}$
- P_0: initial pressure, units of $\mathrm{N\,m^{-2}}$
- Q: heat, units of J
- R: gas constant, $R \approx 8.314\,462\,618\,153\,24\,\mathrm{J\,K^{-1}\,mol^{-1}}$
- S: entropy, units of $\mathrm{J\,K^{-1}}$
- s: specific entropy, units of $\mathrm{J\,K^{-1}\,kg^{-1}}$
- s_j: specific entropy of a species j, units of $\mathrm{J\,K^{-1}\,kg^{-1}}$
- s_j°: specific entropy of a species j at standard pressure (1 bar), units of $\mathrm{J\,K^{-1}\,kg^{-1}}$
- T: temperature, units of K
- T_0: initial temperature, units of K
- U: internal energy, units of J
- u: specific internal energy, units of $\mathrm{J\,kg^{-1}}$
- u_j: specific internal energy of a species j, units of $\mathrm{J\,kg^{-1}}$
- V: volume, units of $\mathrm{m^3}$
- W: mechanical work, units of J

1 The Liquid-Fueled Chemical Rocket Engine

1.1 Overview

A rocket engine is a mechanical device operating on a very basic exploitation of Newton's first law. It expels a fluid propellant (the *reaction mass*) in more or less a direction opposite the intended motion, producing a force which propels the vehicle. In all rocket engines, from home-made water bottle rockets to the unflown nuclear-powered engines of the 1960s, this velocity is imparted to the propellant by generating a pressure gradient between the internals of the rocket and the ambient environment.

Nearly all rocket engines generate this pressure difference by increasing the temperature of gaseous propellant in a confined space. In a nuclear or arc-heated engine, the energy source responsible for heating the propellant is external to the propellant itself. In a chemical rocket, the combustion of a fuel and oxidizer produces both the reaction mass–that is, the products of the chemical reaction–and the energy required to accelerate the reaction mass (*Sutton and Biblarz*, 2016). Thus, the chemical rocket is both a mechanical and chemical device, and were it not for several characteristics of the design of a rocket engine fueled by fluid (not solid) propellants, the mechanical and chemical problems posed by modeling such a chemical rocket would be inextricably intertwined, as is the case in air-breathing jet engines. However, this is not the case. Let us quickly examine how a liquid-fueled chemical rocket operates.

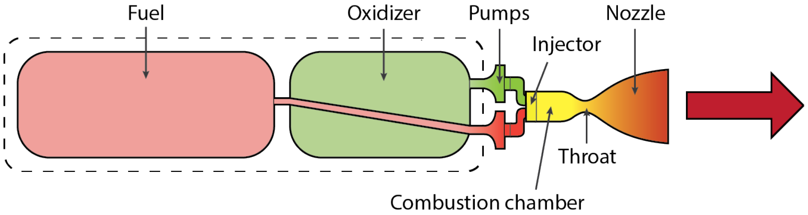

Figure 1: An idealized liquid rocket engine.

Fluid chemical reactants are mixed in the combustion chamber by being pumped through injector plate, an array of nozzles intended to atomize and combine the reacting fluids as effectively as possible (Figure 1). With proper injection, the resulting combustion is thorough and rapid. Combustion occurs to completion within the first few centimeters of the injector plate (*Sutton and Biblarz*, 2016; *Khan et al.*, 2013). For large rocket engines, it may be safely assumed that temperature, pressure, and chemical composition is homogeneous within the whole volume of the combustion chamber. While the injection process is a non-reversible expansion, and there may be a cooled layer near the boundary of regeneratively-cooled combustion chambers, the impact of these effects is marginal to theoretical performance calculations (*Sutton and Biblarz*, 2016).

The exhaust fluid is accelerated to the speed of sound (in the fluid) in the throat of the nozzle (Figure 1), and is supersonic leaving the nozzle. Thus, external conditions may not communicate information "upstream" into the combustion chamber. This isolates the combustion chamber entirely from the ambient environment and permits one to conceptualize the combustion chamber and nozzle as discrete systems. After startup oscillations, mass flux is identical flowing into the combustion chamber (through the injector plate) and out of the chamber (through the throat and nozzle). During operation, the rocket is at equilibrium, and pressure and temperature within the combustion chamber do not fluctuate (*Sutton and Biblarz*, 2016).

1.2 The Chemical Modeling Problem

Assuming the combustion chamber is in steady state and is mechanically isolated from the ambient environment, we may isolate the chemical problem from the mechanical problem and solve them separately. Modeling the fluid mechanics of exhaust expanding from a combustion chamber is a fairly simple and well-understood system (*Sutton and Biblarz*, 2016). However, it is very sensitive to the mechanical properties of the exhaust produced by the combustion chamber. The mechanics of the exhaust is a function of its composition, which is determined by the chemical reaction occurring within the chamber. At the high energies and temperatures required to make a useful rocket engine, combustion is less simple than the well-balanced reactions we learn in introductory chemistry, and disassociation of the products cannot be ignored. A precise and accurate answer to the chemical problem is required to model the engine well (*Sutton and Biblarz*, 2016).

These early chemical calculations were done by hand by tireless chemists, who, in some cases, found it more time- and cost-effective to build the engine and hope it did not blow up (as such engines often did), rather than perform the calculations to the requisite precision. With the advent of computing, combustion calculations were handed off to the machines (much to the chagrin of the chemists and the celebration of the test stand engineers), ushering in a less dramatic era of propulsion technology development (*Clark*, 1972). These calculations were still performed using the same methods from the era of manual computation, completed by solving a set of simultaneous equations describing the myriad of possible equilibria established in the combustion product mixture. This method was the most physically intuitive, but the work required increased immensely for more complex chemical systems–faster than a factorial function of the number of elements (*Reynolds*, 1981)!

W. C. Reynolds approached the problem differently, eschewing the equilibrium-reaction system of equations in favor of a new technique exploiting the properties of abstract partial derivatives he called the “elemental potentials”. His 1986 FORTRAN program, STANJAN, was the first modern numerical chemical equilibrium model (*Reynolds*, 1981; *Cengel and Boles*, 2007). The problem was further abstracted by legendary NASA chemist duo Sanford Gordon and Bonnie J. McBride, who abandoned Reynolds’ elemental potentials to solve the thermodynamic equations of state directly. Their seminal work, today known as CEA, Chemical Equilibrium with Applications, forms the basis of all modern chemical equilibrium programs (*Gordon and McBride*, 1994; *Heidman*, 2016) .

2 A Note on *Gordon and McBride* (1994)

We initially attacked the problem by going straight to the CEA paper authored by Sanford Gordon and Bonnie J. McBride. We spent the summer trying to parse this dense work. Their text is *very* brief, to the point of notoriety. One author comments that the works is “complete but terse... fine if you already understand the approach, but not very useful as a learning tool.” The author understates the significant of the mathematical leaps the CEA paper compacts into a single lines.

They produce a much more robust and extensive model than the limited one we develop here. Everything we were interested in was contained in fewer than 5 pages of the tome they produced–their work considers a host of secondary effects present within liquid rocket engines which we do not, being solely concerned with the product composition of the combustion.

The effort required to understand the enlightened work of Gordon and McBride gave me an appreciation for both the attention to detail required in such work as well as the monumental effort represented by the optimization of such calculations. Limited by the technology of the time, Gordon and McBride relentlessly linearize the system required by the model they develop, to the point of completely obscuring the relationship between their work and familiar thermodynamics equations.

Several months of picking through their work and trying to suss the derivation they left mostly implict provided us with a good understanding of the methods required to solve the system, but the motivation for several of the optimizing steps they take was still beyond our understanding and certainly not elaborated upon in their paper. Desiring to better understand why the CEA paper uses some of the techniques it does, and still to some extent confused about the assumptions made about the thermodynamics of the system, we decided to re-derive the problem completely from first principles.

Our efforts to re-derive their Gibbs iteration equations with additional explanation are presented in Appendix A.

3 An Introduction to the Chemical Problem

3.1 Internal Energy and the Equation of State

The internal energy, U, is a fundamental concept in the thermodynamic model of molecular interactions. This total energy of a molecular system incorporates translation, rotational, and vibrational motion of the nuclei, as well as the electronic binding energies for bonded nuclei (*Cengel and Boles*, 2007). This statement applies to *simple systems*, defined by *Callen* (1998) as those "that are macroscopically homogeneous, isotropic, and uncharged, that are large enough that surface effects can be neglected, and that are not acted on by electric, magnetic, or gravitational fields."

U of a homogeneous chemical system may be considered to be a function of entropy (S), volume (V), and the the molar composition, expressed as n_j for each chemical species j present in the system. For example, in a system of hydrogen gas, oxygen gas, and water vapor, $U = U(S, V, N_{\mathrm{H}_2}, N_{\mathrm{O}_2}, N_{\mathrm{H}_2\mathrm{O}})$. In general,

$$U = U(S, V, \{N_j\}).$$

This is a function of state, and knowing U and the arguments of U fully describes the thermodynamic state of a given homogeneous chemical system.

Taking the total derivative provides us with further insight into the internal energy function.

$$dU = \frac{\partial U}{\partial S}\ dS + \frac{\partial U}{\partial V}\ dV + \sum_j \frac{\partial U}{\partial N_j}\ dN_j$$

Each of these partial derivatives has a special, and occasionally, familiar definition (*Callen*, 1998). Each partial derivative implicitly holds all other variables constant, facilitating the following definitions of some familiar auxiliary quantities:

$$T = \frac{\partial U}{\partial S},\ -P = \frac{\partial U}{\partial V},\ \mu_j = \frac{\partial U}{\partial N_j}$$

Where T is temperature, P is pressure, and μ_j is a value known as the chemical potential of the species j. How delightful to find T and P, our friends from introductory physical sciences, in this treatment of thermodynamics! Now we can rewrite dU as:

$$dU = T\ dS - P\ dV + \sum_j \mu_j\ dN_j$$

In this formulation, each of these partial derivatives is itself a function of the state variables, so:

$$T = T(S, V, \{N_j\}),\ P = P(S, V, \{N_j\}),\ \mu_j = \mu_j(S, V, \{N_k\})$$

Particularly for temperature and pressure, it is unfamiliar to think of these everyday, intuitive values as functions of other variables, especially those as esoteric to the non-physical chemist as entropy and chemical potentials. This $U(S, V, \{N_j\})$ formulation of the equation of state also poses experimental difficulties, as there exist no instrumentation which can directly measure the internal energy or entropy of a system. We are in luck–other, more useful and intuitive formulations exist. We will examine those after discussing the variables of U and dU.

3.2 The State Variables

3.2.1 Entropy and Temperature

To directly quote from chapter 5 of Reynolds' text on thermodynamics, entropy can be viewed as a "quantitative measure of microscopic randomness" (*Cengel and Boles*, 2007). There are multiple approaches to defining entropy. The statistical approach is to define entropy, S, as a function of the probability of any given microscopic state that may be assumed by the macroscopic system in question for a constant energy. This perspective defines entropy as a measure of *uncertainty*, where higher entropy states are those states which exhibit more chaotic behavior, thus decreasing the probability of knowing any particle's exact behavior at a given time after the known initial state.

S, like U, is an absolute, extensive property of any system under consideration. Because entropy is a function of the probability of possible states of a system, entropy is a given, finite value which is proportional to the size of the system, provided spatial homogeneity.

This approach is particularly useful because from it, one may provide a conceptually elegant distinction between heat and work–heat includes those transfers of energy between systems which also produce a change in entropy, and work are those which do not. The question might arise: why bother with the entropy concept? It occurs that the entropy concept affords a particularly elegant expression of the second law of thermodynamics, and given its ties to such a fundamental axiom of the thermodynamic theory, S plays as much a role in thermodynamic analysis as U does.

Temperature, T, is an intuitive concept which becomes slightly obfuscated when trying to pin it down precisely for use in thermodynamic analysis. Fundamentally, temperature is a measure of the absolute energy of a system. Temperature differences determine the amount of heat that can flow in a given process. In a less abstract treatment of the thermodynamic model, we would note that heat processes are those processes which transfer kinetic energy through a disorganized mechanism. Thus, temperature is related to disorganized kinetic energy within a substance.

Without getting too deeply into the weeds yet, temperature of the types of materials with which we are concerned can be very closely approximately as a function of the average velocity of the particles composing it. This function is closely related to the Maxwell–Boltzmann distribution, a distribution describing the velocities within a substance at a given temperature. This statistical mechanical definition of temperature is similar to the definition we offered for S. It makes sense that they would be intimately related.

3.2.2 Volume and Pressure

We do not need to confuse our intuitive understanding of volume like we did temperature. V is simply the total amount of space the system is allowed to occupy. However, in practice, it may be difficult to know or control the volume. Consider a flame burning in the open air–what is the volume of this system? Especially in high-energy experiments, it becomes more difficult to experimentally control the volume, simply due to the fact that confined, high-energy chemical reactions tend to resemble munitions experiments more so than bench-top chemistry. As a result, it may be useful to reformulate the equation of state to be a function of pressure, not volume.

Pressure, P, is the amount of force exerted by the material on any unit area of surface. In energetic fluids, such as those we are working with, this force is almost perfectly isotopic. This is related to the kinetic understanding of temperature, as in energetic fluids, the pressure can be safely understood to be entirely the consequence of the elastic impacts between individual particles and the surrounding surfaces. Because we understand these particles to have velocities evenly distributed in every direction, the pressure exerted upon surfaces by these collisions is invariant with respect to the orientation of the surface in question.

Indeed, in ideal, non-reacting gaseous fluids, pressure can be modeled through the ideal gas law as a strict function of volume and temperature. As temperature increases and volume decreases, the kinetic energy of impact per particle will increase as well as the frequency of collisions. In cold, low-pressure environments, physical materials condense, and such models break down.

An effective approach to handling this distinction between gaseous and condensed phases is to consider the condensed phases separate species from their gaseous phases, "species" which do not contribute to pressure at all.

3.2.3 Composition and Chemical Potential

The variables of composition are very straightforward values–simply the number of each type of molecule. To makes these numbers more manageable, we divide them by Avogadro's number, 6.02217×10^{23}, to produce *mole numbers*, our variables N_j. This makes some things very simple–one mole of carbon-12 atoms (C, atomic mass 12 amu) is twelve grams, and one mole of oxygen gas (O_2, atomic mass 31.9988 amu) is correspondingly very nearly thirty-two grams.

μ_j, the chemical potentials, are less familiar than any of the state variables we have encountered so far. The best understanding of a chemical potential is to take its definition at face value:

$$\mu_j = \left(\frac{\partial U}{\partial N_j}\right)_{S,V,N_{k\neq j}}$$

So the chemical potential is the change in internal energy of a system, everything else held constant, for an infinitesimal change in the amount of some species j. Of course, such an infinitesimal change is not possible–one cannot add less than a single molecule of anything–but the numbers we are dealing with are astronomic and we cannot afford such attention to microscopic detail in the study of classical thermodynamics, which is concerned solely with "those few particular combinations of atomic coordinates that are essentially time-independent [and] are macroscopically observable" (*Callen*, 1998). The addition or absence of a single molecule is not macroscopically observable.

3.2.4 Specific Extensive Variables

Internal energy, entropy, volume, and composition are *extensive* variables. This means if two identical systems with state variables $U_0, S_0, V_0, \{N_{j,0}\}$ are brought together to form a new system, the new system has state variables exactly $2U_0, 2S_0, 2V_0, \{2N_{j,0}\}$.

This is in contrast to the variables found in the total derivative dU–no matter how many times we double or halve the amount of our homogeneous system, temperature, pressure, and the chemical potentials remain unchanged. These derived variables, $T, P, \{\mu_j\}$, are known as *intensive.*

It can be convenient to make the extensive variables intensive by normalizing them to the amount of mass in the system. These new, mass-specific variables use the same symbols, just in lowercase: $u, s, v, \{n_j\}$. Because we are not working in a relativistic model, we can safely assume mass is conserved, so this transformation to specific variables is a perfectly linear transformation, and all previous statements about the extensive forms still hold.

As we have hinted at repeatedly, the intensive variables tend to be much easier to gather experimental data on. We will now demonstrate a method for reformulating the equation of state, previously a function solely of extensive variables, as a function of the more-useful intensive ones.

3.3 On the First and Second Law

The first law of thermodynamics may be stated as:

$$dU_{\text{universe}} = 0$$

And the second law may be stated as:

$$dS_{\text{universe}} \geqslant 0$$

Note that these laws are expressed in terms of the entire universe. Unfortunately, we cannot observe the entire universe. As a result, were we unable to isolate parts of the universe small enough to observe from the rest of it, these laws would be of no use to us. Luckily, this is the case, and these laws also hold in such isolated systems. *Closed* systems are those with sufficient isolation such that the first and second laws hold.

In extremely broad strokes, the first law is concerned with which thermodynamic states are *possible* in a closed system, and the second law is concerned with which thermodynamic states of a closed system may be obtained from from prior thermodynamic states of the same system. Moving from one state to another may be possible, but its reverse not. Thus, the second law is concerned with *irreversibility* (*Callen*, 1998).

While the laws of thermodynamics will not be further discussed in this document–readers are assumed to be already loosely familiar with them–we will note they are axiomatic statements, assumptions upon which the model of thermodynamics is constructed, but which may not be proved or observed within that model (*Cengel and Boles*, 2007).

3.4 On Chemical Equilibrium

Chemical equilibrium in closed systems is the stable state obtained after all spontaneous processes have completed. Because of the second law, this state is the state of maximum entropy within the closed system. The increase of entropy without the loss of energy occurs through transformations of the internal energy–this may be by the disorganization of macroscopic motion, by de-excitement of excited particles (releasing light), by chemical reactions (which release bond energy as heat and light), by phase changes (typically, from macroscopically organized structures to disorganized arrangements), or even by nuclear reactions.

Chemical equilibrium is the equilibrium found through solely chemical processes. While all processes are at play simultaneously, the timescales on which they operate generally differ by orders of magnitude. At one extreme, many processes may be modeled as instantaneous (*Cengel and Boles*, 2007), and on the other, the heat death of the universe, that hypothetical state of ultimate and final equilibrium, is estimated will take up to $10^{10^{26}}$ years to be established (*Frautschi*, 1982). Clearly, we do not have time to account for all processes.

Even with the constraint that we seek chemical equilibrium and not perfect equilibrium, entirely closed systems are difficult to obtain in real life. While portions of a system may be analyzed as closed, rarely is an entire device closed, and never are such devices particularly useful–being unable, by definition, to affect their environment meaningfully. As a result, it may be useful to analyze systems where internal energy is allowed to vary. Where entropy is held constant and internal energy allowed to vary, it may be shown that equilibrium is where internal energy is minimized (*Callen*, 1998).

Yet constant-entropy (isentropic) systems are impossible to obtain in practice, and often difficult to approximate. We shall seek some reformulation of the internal energy in terms of different variables, such that we maintain the property that equilibrium is obtained when this reformulated energy is minimized, but subject to different constraints than the isentropic one.

3.5 Legendre Transformations

We have been hinting repeatedly that the equation of state, as we have presented it as a function of S and V, is unwieldy to work with in practical applications. A Legendre transformation is the tool needed to re-formulate the theory in terms of the intensive variables $T = \partial U/\partial S$ and $P = \partial U/\partial V$.

A Legendre transformation transforms a function $f : \mathbb{R}^n \to \mathbb{R}$ from a representation in the Cartesian coordinate set to one in the tangent coordinate set. This needs some explanation.

Note: we will be discussing exclusively functions of and spaces over $\mathbb{R}^n$, but our discussion generalizes to function of and spaces over $\mathbb{C}^n$ or any field.

3.5.1 The Cartesian Coordinate System

Points in $\mathbb{R}^n$ are described in the Cartesian coordinate system by a unique vector $(x_1, \ldots, x_n)$. It is familiar that smooth functions of the form $f : \mathbb{R}^n \to \mathbb{R}$ can always be expressed as hypersurfaces in $\mathbb{R}^{n+1}$. More generally, for any function $f : \mathbb{R}^n \to \mathbb{R}$, we can find a set $\mathcal{X} \subset \mathbb{R}^{n+1}$ given by:

$$\mathcal{X} = \mathcal{X}(f) = \{(x_1, \ldots, x_n, f(x_1, \ldots, x_n)) \mid \forall x_i \in \mathbb{R}\}$$

Which, again, is really quite an intuitive way to think about it.

3.5.2 Linear Space

What if we are not interested in points? What if we are interested in linear surfaces–lines, planes, and the like? The set of all linear functions we will call $\mathbb{L}$, linear space (because it is indeed, under standard addition and multiplication, a vector space). For example, linear space of dimension 3 contains all planes: that is, all the functions $\mathbb{R}^2 \to \mathbb{R}$ of the form $ax + by + c$. Additionally, we can uniquely refer to these planes with the coordinates (a, b, c).

In general, linear space of dimension n contains all linear functions $\ell : \mathbb{R}^{n-1} \to \mathbb{R}$:

$$\mathbb{L}^n = \{\ell(\vec{x}) = \varphi_1 x_1 + \cdots + \varphi_{n-1} x_{n-1} + \theta \mid \forall\, \varphi_1, \ldots, \varphi_{n-1}, \theta \in \mathbb{R}\}$$

When we use our linear coordinate system, given by the vector $(\varphi_1, \ldots, \varphi_{n-1}, \theta)$, $\mathbb{L}^n$ starts to look quite similar to $\mathbb{R}^n$–in fact, there exists a quite obvious continuous inverse mapping between them. This homeomorphism enables the Legendre transformation.

A Legendre transformation represents a change in our conception of a function $f : \mathbb{R}^n \to \mathbb{R}$. Rather than thinking of it as a subset of $\mathbb{R}^{n+1}$ satisfying $(x_1, \ldots, x_n, f(x_1, \ldots, x_n))$, we think of it as some subset of $\mathbb{L}^{n+1}$ containing all hyperplanes tangent to the function f.

3.5.3 The Tangent Set of f

Any function f can be decomposed into a linear function plus an intercept function, like so:

$$f(\vec{x}) = \sum_i \frac{\partial f}{\partial x_i} x_i + \theta(\vec{x})$$

Like f, θ is a function $\mathbb{R}^n \to \mathbb{R}$. The symmetries of the Legendre transformation should begin to make themselves apparent. If we perform such a decomposition at a point $\vec{p}$, we will generate new linear functions $\ell_{f,\vec{p}}$ defined such that they are tangent to f at $\vec{p}$:

$$\ell_{f,\vec{p}} : \mathbb{R}^n \to \mathbb{R}$$

$$\ell_{f,\vec{p}}(\vec{x}) = \sum_{i=1}^{n} \underbrace{\frac{\partial}{\partial x_i}\big[f(\vec{p})\big]}_{\varphi_i} \cdot x_i + \theta(\vec{p})$$

$$\ell_{f,\vec{p}} \in \mathbb{L}^{n+1}$$

Each $\ell_{f,\vec{p}}$ may be called the tangent space of f at the point $\vec{p}$, but we will just call them *tangent functions*. As they are linear, these tangent functions are elements of *linear space*, $\mathbb{L}^n$. Further, each $\ell_{f,\vec{p}}$ can be given a representation in the linear coordinate system:

$$(\varphi_1, \ldots, \varphi_n, \theta(\vec{p})), \text{ where } \varphi_i = \frac{\partial}{\partial x_i}\big[f(\vec{p})\big]$$

By finding every $\ell_{f,\vec{p}}$ for every $\vec{p}$ in the domain of f, we can generate the set of all hyperplanes tangent to the function, which we will denote $\mathcal{T} \subset \mathbb{L}^{n+1}$:

$$\mathcal{T} = \mathcal{T}(f) = \{\ell_{f,\vec{p}} \mid \forall \vec{p} \in \text{domain}\, f\}$$

$\mathcal{T}(f)$ we will call the *tangent set* of f.

3.5.4 An Example

This has all been fairly abstract. Let us consider an example. Suppose we have the function $f(x) = x^4 + x$. We can decompose it as:

$$x^4 + x = \underbrace{(4x^3 + 1)}_{\varphi} \cdot x + \theta(x)$$

To find the function $\ell_{f,p}$, we need to first find θ. The algebra is not difficult, and we obtain

$$\theta(x) = -3x^4.$$

And from this we get:

$$\ell_{f,p}(x) = (4p^3 + 1) \cdot x - 3p^4$$

In figure 2 we plot $f(x)$ against a couple of $\ell_{f,p}$ for a handful of different p.

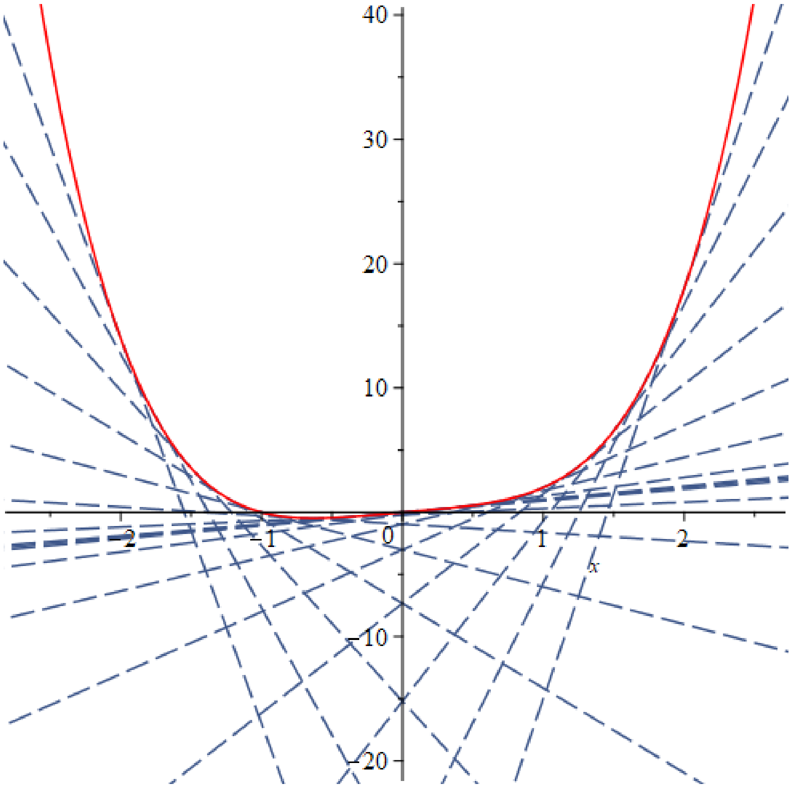

Figure 2: Tangent lines of $f(x) = x^4 + x$

3.5.5 The Legendre Transformation

So far, we have done a good amount of background work, but we have not developed anything particularly useful yet. Let us do a brief review.

For a function

$$f(\vec{x}) : \mathbb{R}^n \to \mathbb{R},\ \vec{x} = (x_1, \dots, x_n)$$

We can describe it using its representation in Cartesian coordinates:

$$\mathcal{X} = \{(x_1, \dots, x_n, f(x_1, \dots, x_n)) \mid \forall x_i \in \mathbb{R}\} \subset \mathbb{R}^{n+1}$$

Or its representation as a tangent set:

$$\mathcal{T} = \left\{ \sum_i \underbrace{\frac{\partial}{\partial x_i}\big[f(\vec{p})\big]}_{\varphi_i} \cdot x_i + \theta(\vec{p}) \mid \forall \vec{p} \in \text{domain}\, f \right\} \subset \mathbb{L}^{n+1}$$

From which we can find a representation in linear coordinates, given by the set:

$$\left\{ \Big(\underbrace{\frac{\partial}{\partial x_1}\big[f(\vec{p})\big]}_{\varphi_1}, \dots, \underbrace{\frac{\partial}{\partial x_n}\big[f(\vec{p})\big]}_{\varphi_n}, \theta(\vec{p}) \Big) \mid \forall \vec{p} \in \text{domain}\, f \right\}$$

Now, let us solve for θ as if it were some function of $\vec{\varphi}$. From earlier, we know that we can decompose f as the sum

$$f = \sum_i \varphi_i x_i + \theta(\vec{\varphi})$$

Giving us

$$\theta(\vec{\varphi}) = f - \sum_i \varphi_i x_i.$$

The function $\theta(\vec{\varphi})$ we call the *Legendre transformation* of $f(\vec{x})$. We can now think of it as just another function of $\mathbb{R}^n$, with its own set of Cartesian coordinates, which we will denote Φ:

$$\Phi = \left\{(\varphi_1, \ldots, \varphi_n, \theta(\varphi_1, \ldots, \varphi_n)) \mid \forall \varphi_i \in \text{range}\left(\frac{\partial f}{\partial x_i}\right)\right\} \subset \mathbb{R}^{n+1}$$

We now have two related functions: the original function f, in which x_i are the independent variables, and its Legendre transformation θ, in which $\varphi_i = \partial f/\partial x_i$ are the independent variables. However, for θ to be useful to us, we need to ensure that no information about f is lost in the transformation. If there exists a bijection between $\mathcal{X}$ and Φ, we will know we can go from f to θ without loss of information.

It turns out that we are limited by f for this bijection to work. If f is everywhere second-differentiable, and has monotonic partials with respect to each x_i, that is,

$$\frac{\partial^2 f}{\partial x_i^2} \geqslant 0 \text{ or } \frac{\partial^2 f}{\partial x_i^2} \leqslant 0$$

Then there exists an invertible mapping:

$$\zeta : \mathcal{X} \xrightarrow{\text{bij.}} \Phi,\ \zeta(a_1, \ldots, a_n, a_{n+1}) = \Big(\varphi_1, \ldots, \varphi_n, f(a_1, \ldots, a_n) - \sum_{i=0}^{n} \varphi_i a_i\Big)$$

Where, of course,

$$\varphi_i = \frac{\partial}{\partial x_i}\big[f(a_i, \ldots, a_n)\big].$$

Sadly, it is out of the scope of this thesis to prove this. Chemists and physicists take such bold statements on faith alone, and as we would like to spend our focus dabbling in their world, we must do the same. Another statement we must take on faith is that equally valid are *partial Legendre transformations*, where rather than finding $\theta(\varphi_1, \ldots, \varphi_2)$ we find some partially-transformed function $\theta(\varphi_1, \ldots, \varphi_i, x_{i+1}, x_n)$ which retains some of the original variables.

3.6 Transformations of the Equation of State

We have demonstrated that a function and its Legendre transformation describe homeomorphic spaces through an exchange of variables. Thus, we may transform the equation of state at will, selecting whichever set of variables suits our needs best, with the assurance that we have not lost information about the thermodynamic system and can draw meaningful conclusions about its behavior.

3.6.1 Enthalpy

Let us define some new surface defining the possible thermodynamic states of any system,

$$\mathcal{U} = \big\{(S, V, \{N_j\}, U(S, V, \{N_j\})\big\}$$

This is our fundamental equation of state, expressed as a surface in entropy-volume-composition space. We will now form a partial Legendre transformation of just the variable V. We will transform this surface into a new surface in entropy-pressure-composition space we will call $\mathcal{H}$, defined by

$$\mathcal{H} = \big\{\big(S, P, \{N_j\}, H(S, P, \{N_j\})\big)\big\}.$$

We seek the function H. Firstly, we know that

$$\frac{\partial U}{\partial V} = -P,$$

and we note that the negative sign on the pressure will not make things difficult. Secondly, we will assume that P fulfills the requirement of being monotonic with respect to U. This is, in fact, true, but we will not demonstrate that it is so (*Callen*, 1998).

Now let us find our Legendre transformation of U with respect to V, which we have chosen to call H. By our definition of the transformation above:

$$\theta(\vec{\varphi}) = f - \sum_i \varphi_i x_i$$

Substituting in H for θ, U for f, and only choosing V as our sole x_i (recall this is a *partial* transformation, so we leave S and N_j alone), we find

$$\varphi = \frac{\partial f}{\partial x} = \frac{\partial U}{\partial V} = -P$$

(Which is, of course, the plan), and from that we obtain

$$H = U + PV.$$

This value H is called *enthalphy*, and it is a function of the entropy, the *pressure*, and the chemical composition. Enthalpy is equivalent to the internal energy, but is much more easily measured, due to PV being exactly equal to work done by the system on it surroundings. As a result, data is easily available for enthalpy where no such data is present for internal energy.

3.6.2 Gibbs Energy

Let us do another partial Legendre transformation of U, this time, with respect to both S and V. Again, we begin with

$$\mathcal{U} = \big\{(S, V, \{N_j\}, U(S, V, \{N_j\})\big\}$$

And, defining

$$\mathcal{G} = \big\{\big(T, P, \{N_j\}, G(T, P, \{N_j\})\big)\big\}$$

We substitute into

$$\theta(\vec{\varphi}) = f - \sum_i \varphi_i x_i$$

Only choosing our x_i to be S and V; noting our φ_i will be $\partial U/\partial S = T$ and $\partial U/\partial V = -P$, and obtain:

$$G = U - TS + PV$$

This new function, G, is known as the *Gibbs energy.* As before, it is equivalent to the internal energy, but it is considered as a function of T, P, and $\{N_j\}$, not S, V, and $\{N_j\}$. In this way, it is much more intuitive to work with. Conceptually, the Gibbs energy is the amount of energy available to do work in a given constant-T, P thermodynamic process.

What is the significance of this? We have found a reformulation of the internal energy as we sought. Based upon our assessment of the Legendre method, we know we can uniquely describe some point in S-V space by their exchanged variables in T-P space. Note that the chemical composition variables N_j have remained untouched.

As a result, if we seek chemical equilibrium, we may rest assured that we will obtain the same chemical composition by minimization of U, subject to constant S and V, as by minimization of G subject to constant T and P.

4 Mathematical Tools

Before we seek chemical equilibrium by minimization of the Gibbs energy, we must furnish our mathematical toolbox. The problem before us: we will be minimizing a function subject to some constraints, and we will be solving systems of equations. As a result, we will need the method of Lagrange multipliers, to allow us to find constrained extrema, and we will need a multivariate rootfinding method.

4.1 Lagrange Multipliers

Given a function of n variables, represented in the vector $\vec{x} = \{x_1, \ldots, x_n\}$, which is locally differentiable and locally continuous near some extrema, the first derivative test states that the extrema will be located where the derivative of the function is zero. For multivariate functions, all first partials must be zero where there exists a local extrema (this is not a biconditional statement, consider saddle points). In other words,

$$\forall i, \ \frac{\partial f}{\partial x_i} = 0$$

Or, using gradient notation (more on that below):

$$\nabla f(\vec{x}) = \vec{0}$$

However, a local maximum *subject to some constraint* may not satisfy the first derivative test. A maximum is such because no matter in which direction you go, the function will have no greater value. Equality constraints limit the "directions" in which we "may go". To find an maximum subject to a constraint at a location which is not a maximum in the unconstrained function, we must find some location on the constraint such that the only place to increase the function value is to exit the constraint. Note that this is equivalent to saying that the gradient of f is perpendicular to the constraint.

Consider our equality constraints. Because they can be written as some function $c = 0$, they can be understood as the zero level surface of a function c. To use a geometric argument, recall that the gradient vector ∇c must always be normal to the level surfaces of c. Clearly, where there exists a maximum, ∇f is a multiple of ∇c. (All the prior reasoning holds for minima as well.) We introduce a new variable, λ, which we call a Lagrange multiplier, to form the system of expressions (recall that ∇ is a column vector):

$$\nabla f = \lambda \nabla c$$

Which, along with the constraint condition

$$c = 0$$

Forms a solvable system of equations for the variables of f along with the Lagrange variable λ.

Note that despite arriving at this system of expressions assuming we are seeking constrained maxima which are not also unconstrained maxima, unconstrained maxima also satisfy the Lagrange relationships, simply where $\lambda = 0$.

What about multiple constraints? We will use the same geometric reasoning. To find a maximum subject to multiple constraints, it must be that the only "direction to go" in which the value of f increases is normal to all of the the constraint surfaces. Therefore, ∇f must be a linear combination of the ∇c_is. The multiple-constraint Lagrange system is

$$\nabla f = \sum_i \lambda_i \nabla c_i$$

Along with the constraint conditions

$$c_i = 0, \ \forall i.$$

The same reasoning holds for minima.

4.1.1 Gradient Notation

∇f, the gradient of f, is a column vector of all the partials of f given by:

$$\nabla f = \begin{bmatrix} \dfrac{\partial f}{\partial x_1} \\ \vdots \\ \dfrac{\partial f}{\partial x_n} \end{bmatrix}, \text{ or at a point by } \nabla f(\vec{p}) = \begin{bmatrix} \left(\dfrac{\partial f}{\partial x_1}\right)_{\vec{p}} \\ \vdots \\ \left(\dfrac{\partial f}{\partial x_n}\right)_{\vec{p}} \end{bmatrix}$$

Not only does using gradient notation give us the chance to apply some of the tools we learned back in calc III, it is a vector, making it a touch more straightforward to manipulate than "all the partials," even if only in our heads.

Some things we recall from calc III:

- $||\nabla f||$, the magnitude of the vector is the max rate of change of f at a point.
- $\nabla f(\vec{p})$ is $\perp$ to the level curve of f at $\vec{p}$.
- The direction of steepest descent of f at a point $\vec{p}$ is in the direction of $\nabla f(\vec{p})$ at that point.

4.1.2 An Example of Lagrange Multipliers with a Single Constraint

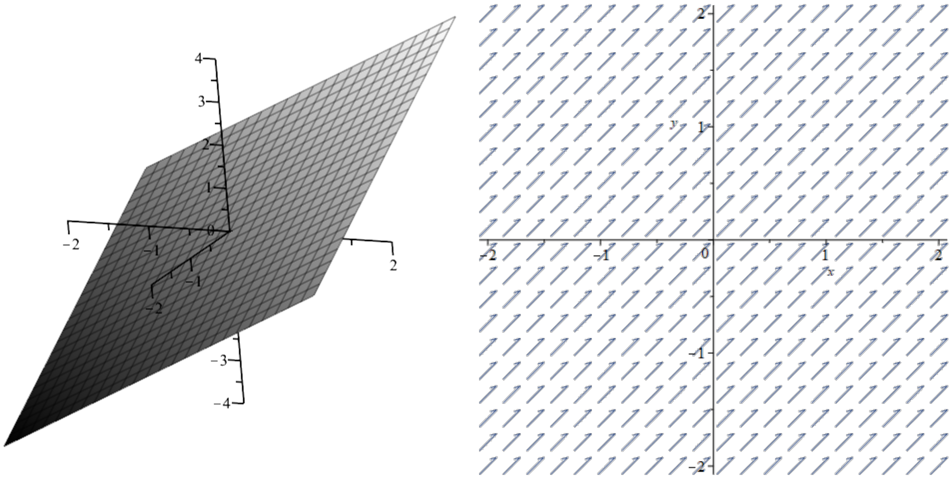

Figure 3: A simple example function and its gradient.

Suppose we have an $f : \mathbb{R}^2 \to \mathbb{R}$ defined by $f(x, y) = x + y$ (Figure 3). We seek to find extrema of f subject to an equality constraint, let's say $\sqrt{x^2 + y^2} = 1$ (a circle of radius one). With a little bit of algebra, we could express those constraints as functions equal to zero. That gives us

$$c(x, y) = \sqrt{x^2 + y^2} - 1 = 0$$

Of which the constraint is the level curve equal to 0. The level curve and the gradient of this function are plotted in figure 4.

Local maxima are to be found where the gradient of the function is equal to the gradient of the constraint. Recall that the gradient is always perpendicular to the level curves (by definition). Then, geometrically, the gradient of the function (the direction in which it increases) is perpendicular to the level curve of the constraint function–there is no way to increase the function value without leaving our constraint. (Without loss of generality, this applies to minima as well.) So there exists a field element λ such that:

$$\nabla f = \lambda \, \nabla c$$

This is equivalent to the system of equations:

$$\frac{\partial f}{\partial x} = \lambda \frac{\partial c}{\partial x} \text{ and } \frac{\partial f}{\partial y} = \lambda \frac{\partial c}{\partial y}$$

Calculating the partials and including the constraint equation, we obtain a system of three equations for the three variables x, y, λ.

$$\sqrt{x^2 + y^2} - 1 = 0 \qquad 1 = \lambda \frac{x}{\sqrt{x^2 + y^2}} \qquad 1 = \lambda \frac{y}{\sqrt{x^2 + y^2}}$$

Solving by whatever method gives us a maximum solution at the point $(\sqrt{2}, \sqrt{2})$ and a minimum solution at the point $(-\sqrt{2}, -\sqrt{2})$.

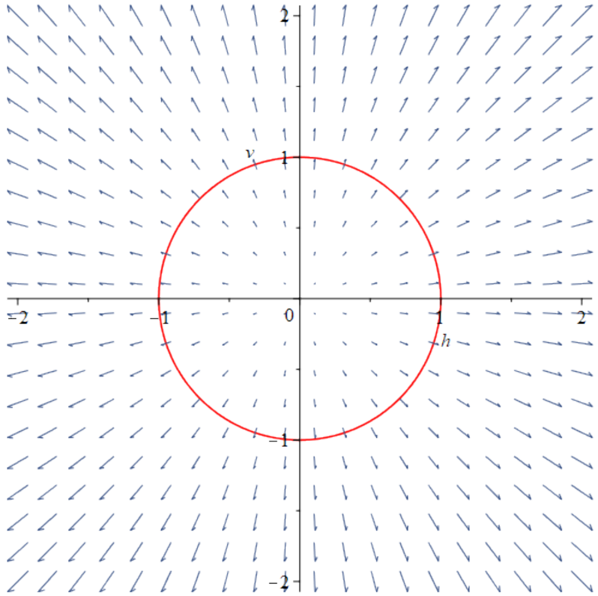

Figure 4: The constraint on our example function.

4.1.3 An Example of Lagrange Multipliers with Multiple Constraints

Due to the difficulty of plotting three-dimensional functions embedded in 4-dimensional space, we regretfully neglect illustrations in this example. Suppose we have an $f : \mathbb{R}^3 \to \mathbb{R}$ defined by $f(x, y, z) = x+y+z$. We seek to find extrema of f subject to some equality constraints, let's say $z = x^2 - y^2$ (a saddle-shaped paraboloid) and $\sqrt{x^2 + y^2 + z^2} = 1$ (a sphere of radius one). With a little bit of algebra, we could express those constraints as functions equal to zero. That gives us:

$$c_1(x, y, z) = x^2 - y^2 - z = 0 \text{ and } c_2(x, y, z) = \sqrt{x^2 + y^2 + z^2} - 1 = 0$$

Local maxima are to be found where the gradient of the function is some *linear combination* of the gradients of the constraints. Geometrically, the gradient of the function (the direction in which it increases) is in the plane perpendicular to the space described where all constraints are fulfilled–there is no way to increase the function value without violating one or more constraint. (Without loss of generality, this applies to minima as well.) So there exist some field elements $\lambda_1,\ \lambda_2$ such that:

$$\nabla f = \lambda_1 \nabla c_1 + \lambda_2 \nabla c_2$$

This is equivalent to the system of equations:

$$\frac{\partial f}{\partial x} = \lambda_1 \frac{\partial c_1}{\partial x} + \lambda_2 \frac{\partial c_1}{\partial x}$$

$$\frac{\partial f}{\partial y} = \lambda_1 \frac{\partial c_1}{\partial y} + \lambda_2 \frac{\partial c_1}{\partial y}$$

$$\frac{\partial f}{\partial z} = \lambda_1 \frac{\partial c_1}{\partial z} + \lambda_2 \frac{\partial c_1}{\partial z}$$

Calculating the partials and including the constraint equations, we obtain a system of five equations for the five variables $x, y, z, \lambda_z, \lambda_2$.

$$x^2 - y^2 - z = 0$$

$$\sqrt{x^2 + y^2 + z^2} - 1 = 0$$

$$1 = 2\lambda_1 x + \frac{\lambda_2 x}{\sqrt{x^2 + y^2 + z^2}}$$

$$1 = -2\lambda_1 y + \frac{\lambda_2 y}{\sqrt{x^2+y^2+z^2}}$$
$$1 = -\lambda_1 + \frac{\lambda_2 z}{\sqrt{x^2+y^2+z^2}}$$

This system can be analytically or numerically solved for $x, y, z, \lambda_z, \lambda_2$, but in practice, we are likely only concerned with the values of x, y, z.

4.2 A Bit About Newton's Multivariate Rootfinding Method

Continuously differentiable multivariate functions $f(\vec{x})$ (in figure 5, the red surface) may be linearly approximated for small changes of $\vec{x}$ like so:

$$f(\vec{x}+\Delta\vec{x}) = f(\vec{x}) + \sum_i^n \frac{\partial f}{\partial x_i}\,\Delta x_i + \sum_i^n \sum_j^n \frac{\partial^2 f}{\partial x_i \partial x_j}\,\Delta x_i\,\Delta x_j + \cdots \approx f(\vec{x}) + \sum_n \frac{\partial f}{\partial x_n}\,\Delta x_n$$

If our changes of $\vec{x}$ $(\Delta\vec{X})$ are small, then when we start to multiply Δx_i together as we do in later terms of the exact expansion, we will find these later terms to become vanishingly small. Thus, for small $\Delta\vec{x}$, a linear approximation is reasonable.

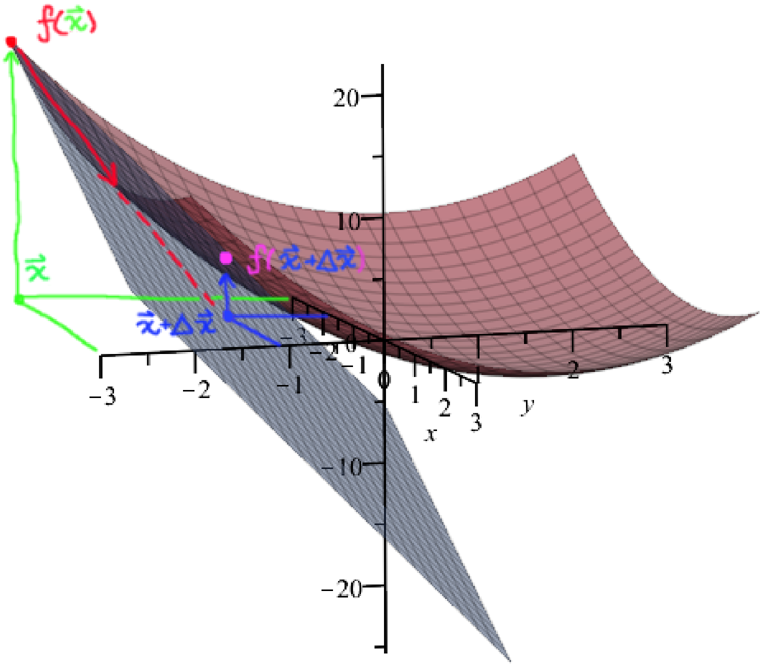

Figure 5: A conceptual figure illustrating how Newton's multivariate method works.

Geometrically, we have obtained a tangent plane (in figure 5, the gray plane) at this point (the green coordinates). If we follow that tangent plane by the line of steepest descent (the red arrow) to where it intersects zero,

$$f(\vec{x}+\Delta\vec{x}) = 0$$

Find the approximate root,

$$\sum_n \frac{\partial f}{\partial x_n}\,\Delta x_n = -f(\vec{x})$$

And then solve for how far we "moved" (these are our Δx_n variables), we can re-approximate the function at this new, closer point, by updating $\vec{x} := \vec{x} + \Delta\vec{x}$ (the blue coordinates) and obtaining a new linear approximation to the function. This iterative method is Newton's rootfinding method, and the Δx_i are known as our iteration variables. When the iteration variables are small enough, we declare the approximation sufficiently accurate and terminate the iteration.

For a multivariate system of functions, we can exploit the linearity of this approximation and solve for roots of the approximation in parallel using matrix representation. We begin with a column vector of variables $\vec{x}$ and a system of functions of $\vec{x}$, f:

$$\vec{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \text{ and } f = \begin{cases} f_1(\vec{x}) \\ \vdots \\ f_m(\vec{x}) \end{cases}$$

And then solve the matrix

$$\begin{bmatrix} \frac{\partial f_1}{\partial x_1}(\vec{x}) & \dots & \frac{\partial f_1}{\partial x_n}(\vec{x}) \\ \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1}(\vec{x}) & \dots & \frac{\partial f_m}{\partial x_n}(\vec{x}) \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \vdots \\ \Delta x_n \end{bmatrix} = \begin{bmatrix} -f_1(\vec{x}) \\ \vdots \\ -f_m(\vec{x}) \end{bmatrix}$$

To obtain Δx-values. The variables $\vec{x}$ are iterated,

$$\vec{x} := \begin{bmatrix} x_1 + \Delta x_1 \\ \vdots \\ x_n + \Delta x_n \end{bmatrix}$$

And the matrix is re-solved; iteration is repeated until $\Delta x \to 0$.

5 Solving the Chemical Problem

With these two tools, we are ready to seek chemical equilibrium through numerical methods. We will begin by approaching the problem directly from first principles in a straightforward approach, and later seek to streamline our solution.

5.1 A Straightforward Approach

5.1.1 A Better Expression for the Gibbs

From our thermodynamic arguments before, we see that reaction equilibrium is equivalent to minimization of the Gibbs energy. Thus, we seek to minimize

$$G = U - TS + PV.$$

First, we will substitute in $H = U + PV$:

$$G = H - TS$$

Next, we will divide each term by mass, making G, H, and S into their specific forms:

$$g = h - Ts$$

Note that, as defined in the Lagrange transformation that beget the Gibbs energy, g, h, s are all functions of temperature (T), pressure (P), and specific composition (the specific mol fractions n_j for all species j).

Enthalphy (H) and entropy (S) are extensive properties. Thus, total specific enthalpy, h, is the sum of species-j-specific enthalpy, h_j, times the mol fraction of species j, n_j. The same argument holds for entropy. Thus, we may rewrite the specific Gibbs function as:

$$g = \sum_j h_j n_j - T \sum_j s_j n_j$$

Where species-specific enthalpy and entropy, h_j and s_j are functions of only pressure and temperature, not composition.

By definition, enthalpy is not a function of pressure (this is why values for enthalpy are more commonly obtained than that for internal energy). That is, for any species j:

$$h_j = h_j^\circ$$

Where the superscripted circle indicates constant pressure at 1 bar (standard pressure).

Entropy is a bit trickier. For condensed species, entropy is not a function of pressure, giving us:

$$s_j = s_j^\circ$$

However, for gasses, pressure is an important factor. For gaseous species j (assumed ideal):

$$s_j = s_j^\circ - R \ln \left(\frac{P_j}{P^\circ} \right)$$

Where P_j is the partial pressure of species j and P° is the standard pressure of 1 bar. If we keep our units consistent, then P° can drop out. Partial pressure for ideal gases is a straightforward affair, where:

$$P_j = P \cdot \frac{n_j}{n}$$

Going forward, we will continue to use the symbol n referring to the sum of mol fractions for all gaseous species:

$$n = \sum_j^{\text{gas.}} n_j$$

By substituting the three above expressions into each other , we obtain the expressions for species-specific entropy:

$$s_j = \begin{cases} s_j^\circ & \text{if } j \text{ is condensed} \\ s_j^\circ - R \ln \left(P \cdot \frac{n_j}{n} \right) & \text{if } j \text{ is gaseous} \end{cases}$$

This pressure dependence in gaseous species is due to the "entropy of mixing" for ideal gases, a concept we will not further discuss.

Finally, we substitute the expressions for h_j and s_j into the Gibbs expression, obtaining the following definition for the Gibbs:

$$g = \sum_j h_j^\circ n_j - T \sum_j^{\text{cond.}} s_j^\circ n_j - T \sum_j^{\text{gas.}} \left(s_j^\circ - R \ln \left(P \cdot \frac{n_j}{n} \right) \right) n_j$$

Here, h_j° and s_j° are functions of temperature only. Values for these thermodynamic functions will be interpolated from data accumulated and distributed by NIST, the National Institute of Standards and Technology (*Linstrom and Mallard*). We can rearrange this expression slightly to produce this final definition for the Gibbs energy function. By now, it is clear that the Gibbs energy is a function of P, T, and $n_{1...j}$.

$$g = \sum_j^{\text{cond.}} \left(h_j^\circ - T s_j^\circ \right) n_j + \sum_j^{\text{gas.}} \left(h_j^\circ - T s_j^\circ + RT \ln \left(P \cdot \frac{n_j}{n} \right) \right) n_j$$

5.1.2 Constraints of the System

Our system is subject to constant temperature, constant pressure, and atomic conservation constraints. We can express this set of constraints as $2 + i$ expressions, where i is the number of elements (types of atom) in the system. These expressions are given by the two isothermal and isobaric constraints

$$T = T_0, \text{ or the function } t = 0, \text{ where } t = T - T_0$$

$$P = P_0, \text{ or the function } p = 0, \text{ where } p = P - P_0$$

And the set of i atomic conservation constraints

$$\sum_j a_{ij} n_j = b_{0,i}, \text{ or the functions } e_i = 0 \text{ where } e_i = \sum_j a_{ij} n_j - b_{0,i}.$$

Here, a_{ij} is the number of atoms of element i in a molecule of species j and $b_{0,j}$ is the initial specific number of atoms j.

5.1.3 The System of Lagrange Equations

We have the constraint functions t, p, and $e_{1\dots i}$, and the function to be minimized, g. This gives us the following system of equations:

$$\begin{cases} t = 0 \\ p = 0 \\ e_i = 0, \ \forall i \\ \nabla g = \sum_i \lambda_i \nabla e_i + \lambda_t \nabla t + \lambda_p \nabla p \end{cases}$$

Since t, p, e_i, g are functions of T, P, and $\{n_j\}$, we can expand the gradient notation to find the following system of $4 + i + j$ equations:

$$\begin{cases} t = 0 \\ p = 0 \\ e_i = 0, \ \forall \text{ elements } i \\ \dfrac{\partial g}{\partial T} = \sum_i \lambda_i \dfrac{\partial e_i}{\partial T} + \lambda_t \dfrac{\partial t}{\partial T} + \lambda_p \dfrac{\partial p}{\partial T} \\ \dfrac{\partial g}{\partial P} = \sum_i \lambda_i \dfrac{\partial e_i}{\partial P} + \lambda_t \dfrac{\partial t}{\partial P} + \lambda_p \dfrac{\partial p}{\partial P} \\ \dfrac{\partial g}{\partial n_j} = \sum_i \lambda_i \dfrac{\partial e_i}{\partial n_j} + \lambda_t \dfrac{\partial t}{\partial n_j} + \lambda_p \dfrac{\partial p}{\partial n_j}, \ \forall \text{ species } j \end{cases}$$

This is now a system of the $4 + i + j$ variables T, P, $\{n_j\}$, $\{\lambda_i\}$, λ_t, and λ_p.

We calculate the appropriate partials and substitute to find the final system:

$$\begin{cases} T - T_0 = 0 \\ P - P_0 = 0 \\ \sum_j a_{ij} n_j - b_{0,i} = 0, \ \forall \text{ elements } i \\ nR - \sum_j s_j n_j = \lambda_t \\ \dfrac{nRT}{P} = \lambda_p \\ h_j^\circ - T s_j^\circ = \sum_i \lambda_i a_{ij}, \ \forall \text{ condensed species } j \\ h_j^\circ - T s_j^\circ + RT \ln\left(P \cdot \dfrac{n_j}{n}\right) = \sum_i \lambda_i a_{ij}, \ \forall \text{ gaseous species } j \end{cases}$$

The chemically inclined reader may note that λ_t is equal to some Gibbs-like unmeasureable value equivalent to the internal energy for equilibrium problems, and λ_p is simply equal to the volume of the system.

5.1.4 Finding a Root of the Lagrange System

We now have the system expressed above. With a bit of rearranging, we can express this as a set of functions of which we seek the root. These functions are named with a * to keep them distinct from previous functions we have defined and named similarly.

$$\begin{cases} T^*(T, P, \{n_j\}, \{\lambda_i\}, \lambda_t, \lambda_p) = T - T_0 \\ P^*(T, P, \{n_j\}, \{\lambda_i\}, \lambda_t, \lambda_p) = P - P_0 \\ E_i^*(T, P, \{n_j\}, \{\lambda_i\}, \lambda_t, \lambda_p) = \sum_j a_{ij} n_j - b_{0,i}, \ \forall \text{ elements } i \\ t^*(T, P, \{n_j\}, \{\lambda_i\}, \lambda_t, \lambda_p) = nR - \sum_j s_j n_j - \lambda_t \\ p^*(T, P, \{n_j\}, \{\lambda_i\}, \lambda_t, \lambda_p) = \dfrac{nRT}{P} - \lambda_p \\ C_j^*(T, P, \{n_j\}, \{\lambda_i\}, \lambda_t, \lambda_p) = h_j^\circ - Ts_j^\circ - \sum_i \lambda_i a_{ij}, \ \forall \text{ condensed species } j \\ G_j^*(T, P, \{n_j\}, \{\lambda_i\}, \lambda_t, \lambda_p) = h_j^\circ - Ts_j^\circ + RT\ln\left(P \cdot \dfrac{n_j}{n}\right) - \sum_i \lambda_i a_{ij}, \ \forall \text{ gaseous species } j \end{cases}$$

We do this by the the multivariate Newton's method. Recall that we ought to find a matrix

$$\begin{bmatrix} \dfrac{\partial f_1}{\partial x_1}(\vec{x}) & \dots & \dfrac{\partial f_1}{\partial x_n}(\vec{x}) \\ \vdots & & \vdots \\ \dfrac{\partial f_m}{\partial x_1}(\vec{x}) & \dots & \dfrac{\partial f_m}{\partial x_n}(\vec{x}) \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \vdots \\ \Delta x_n \end{bmatrix} = \begin{bmatrix} -f_1(\vec{x}) \\ \vdots \\ -f_m(\vec{x}) \end{bmatrix}$$

Where our variable vector is the (re-ordered as shown) column vector

$$\vec{x} = \begin{bmatrix} \lambda_t & \lambda_p & \{n_j \text{ (cond. } j)\} & \{n_j \text{ (gas. } j)\} & \{\lambda_i\} & P & T \end{bmatrix}^{\mathrm{T}}$$

And our fs are the system above, similarly re-ordered as

$$f = \{t^*, p^*, \{E_i^*\}, \{G_j^* \text{ (gas. } j)\}, \{C_j^* \text{ (cond. } j)\}, P^*, T^*\}.$$

The reason for this re-ordering is because it makes our final matrix more closely upper-triangular.

So we ought to find the (possibly very large matrix):

$$A = \begin{array}{ccccccc|l}
 & & \overbrace{}^{j\ \text{(cond.)}} & \overbrace{}^{j\ \text{(gas.)}} & \overbrace{}^{i} & & & \\
\frac{\partial t^*}{\partial \lambda_t} & \frac{\partial t^*}{\partial \lambda_p} & \frac{\partial t^*}{\partial n_j} & \frac{\partial t^*}{\partial n_j} & \frac{\partial t^*}{\partial \lambda_i} & \frac{\partial t^*}{\partial P} & \frac{\partial t^*}{\partial T} & \\
\frac{\partial p^*}{\partial \lambda_t} & \frac{\partial p^*}{\partial \lambda_p} & \frac{\partial p^*}{\partial n_j} & \frac{\partial p^*}{\partial n_j} & \frac{\partial p^*}{\partial \lambda_i} & \frac{\partial p^*}{\partial P} & \frac{\partial p^*}{\partial T} & \\
\frac{\partial E_i^*}{\partial \lambda_t} & \frac{\partial E_i^*}{\partial \lambda_p} & \frac{\partial E_i^*}{\partial n_j} & \frac{\partial E_i^*}{\partial n_j} & \frac{\partial E_i^*}{\partial \lambda_i} & \frac{\partial E_i^*}{\partial P} & \frac{\partial E_i^*}{\partial T} & \big\}\ i \\
\frac{\partial G_j^*}{\partial \lambda_t} & \frac{\partial G_j^*}{\partial \lambda_p} & \frac{\partial G_j^*}{\partial n_j} & \frac{\partial G_j^*}{\partial n_j} & \frac{\partial G_j^*}{\partial \lambda_i} & \frac{\partial G_j^*}{\partial P} & \frac{\partial G_j^*}{\partial T} & \big\}\ j\ \text{(gas.)} \\
\frac{\partial C_j^*}{\partial \lambda_t} & \frac{\partial C_j^*}{\partial \lambda_p} & \frac{\partial C_j^*}{\partial n_j} & \frac{\partial C_j^*}{\partial n_j} & \frac{\partial C_j^*}{\partial \lambda_i} & \frac{\partial C_j^*}{\partial P} & \frac{\partial C_j^*}{\partial T} & \big\}\ j\ \text{(cond.)} \\
\frac{\partial P^*}{\partial \lambda_t} & \frac{\partial P^*}{\partial \lambda_p} & \frac{\partial P^*}{\partial n_j} & \frac{\partial P^*}{\partial n_j} & \frac{\partial P^*}{\partial \lambda_i} & \frac{\partial P^*}{\partial P} & \frac{\partial P^*}{\partial T} & \\
\frac{\partial T^*}{\partial \lambda_t} & \frac{\partial T^*}{\partial \lambda_p} & \frac{\partial T^*}{\partial n_j} & \frac{\partial T^*}{\partial n_j} & \frac{\partial T^*}{\partial \lambda_i} & \frac{\partial T^*}{\partial P} & \frac{\partial T^*}{\partial T} &
\end{array}$$

Calculating all these partials, we find the matrix:

$$\begin{array}{ccccccc|l}
 & & \overbrace{}^{j\ \text{(cond.)}} & \overbrace{}^{j\ \text{(gas.)}} & \overbrace{}^{i} & & & \\
-1 & 0 & -s_j^\circ & R - s_j^\circ + R\ln\left(P \cdot \frac{n_j}{n}\right) & 0 & \frac{nR}{P} & \frac{nR}{T} - \sum_j \frac{C_{p,j}^\circ n_j}{T} & \\
0 & -1 & 0 & \frac{RT}{P} & 0 & -\frac{nRT}{P^2} & \frac{nR}{P} & \\
0 & 0 & a_{ij} & a_{ij} & 0 & 0 & 0 & \big\}\ i \\
0 & 0 & 0 & \begin{cases} \frac{RT}{n_j} - \frac{RT}{n} & \text{on diag.} \\ -\frac{RT}{n} & \text{off-diag.} \end{cases} & -a_{ij} & \frac{RT}{P} & R - s_j^\circ + R\ln\left(P \cdot \frac{n_j}{n}\right) & \Big\}\ j\ \text{(gas.)} \\
0 & 0 & 0 & 0 & -a_{ij} & 0 & -s_j^\circ & \big\}\ j\ \text{(cond.)} \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & \\
0 & 0 & 0 & 0 & 0 & 0 & 1 &
\end{array}$$

$$\cdot \begin{array}{c|l}
\Delta\lambda_t & \\
\Delta\lambda_p & \\
\Delta n_j & \big\}\ j\ \text{(cond.)} \\
\Delta n_j & \big\}\ j\ \text{(gas.)} \\
\Delta\lambda_i & \big\}\ i \\
\Delta P & \\
\Delta T &
\end{array}
= \begin{array}{c|l}
\lambda_p - nR + \sum_j s_j n_j & \\
\lambda_p - \frac{nRT}{P} & \\
b_{0,i} - \sum_j a_{ij} n_j & \big\}\ i \\
\sum_i \lambda_i a_{ij} - h_j^\circ + T s_j^\circ - RT\ln\left(P \cdot \frac{n_j}{n}\right) & \Big\}\ j\ \text{(gas.)} \\
\sum_i \lambda_i a_{ij} - h_j^\circ + T s_j^\circ & \big\}\ j\ \text{(cond.)} \\
P_0 - P & \\
T_0 - T &
\end{array}$$

This straightforward derivation of the iteration system clearly defines the system without any special tricks, but we can improve its efficency and clarity.

5.2 A Smaller Iteration System

The first step towards an improved method is to reduce the size of the matrix which needs to be solved. We can knock a few obvious ones out right away.

5.2.1 Streamlining the Lagrange System

In the Lagrange system we found earlier, there are two meaningless variables introduced.

$$\begin{cases} T - T_0 = 0 \\ P - P_p = 0 \\ \sum_j a_{ij} n_j - b_{0,i} = 0, \ \forall \text{ elements } i \\ nR - \sum_j s_j n_j = \lambda_t \\ \dfrac{nRT}{P} = \lambda_p \\ h_j^\circ - T s_j^\circ = \sum_i \lambda_i a_{ij}, \ \forall \text{ condensed species } j \\ h_j^\circ - T s_j^\circ + RT \ln\left(P \cdot \dfrac{n_j}{n}\right) = \sum_i \lambda_i a_{ij}, \ \forall \text{ gaseous species } j \end{cases}$$

It can be seen that the variables λ_t, λ_p only appear in a single equation each. We do not actually care about these multipliers, and they do not affect the system in any way insofar as it constrains the physical variables T, P, and $\{n_j\}$. We can simply discard these variables, and their associated set of functions which we then include in the system upon which we rootfind. Thus, we end up with the nicer iteration system, two columns and two rows smaller:

$$\begin{array}{c} \begin{array}{ccccc} \overbrace{\qquad}^{j\text{ (cond.)}} & \qquad\qquad \overbrace{\qquad}^{j\text{ (gas.)}} \qquad\qquad & \overbrace{\qquad}^{i} & \qquad & \qquad\qquad\qquad \end{array} \\ \left[\begin{array}{ccccc} a_{ij} & a_{ij} & 0 & 0 & 0 \\ 0 & \begin{cases} \frac{RT}{n_j} - \frac{RT}{n} & \text{on diag.} \\ -\frac{RT}{n} & \text{off-diag.} \end{cases} & -a_{ij} & \dfrac{RT}{P} & R - s_j^\circ + R\ln\left(P \cdot \dfrac{n_j}{n}\right) \\ 0 & 0 & -a_{ij} & 0 & -s_j^\circ \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{array}\right] \begin{array}{l} \}\ i \\ \}\ j\text{ (gas.)} \\ \}\ j\text{ (cond.)} \\ \\ \\ \end{array} \end{array}$$

$$\cdot \begin{bmatrix} \Delta n_j \\ \Delta n_j \\ \Delta \lambda_i \\ \Delta P \\ \Delta T \end{bmatrix} \begin{array}{l} \}\ j\text{ (cond.)} \\ \}\ j\text{ (gas.)} \\ \}\ i \\ \\ \\ \end{array} = \begin{bmatrix} b_{0,i} - \sum_j a_{ij} n_j \\ \sum_i \lambda_i a_{ij} - h_j^\circ + T s_j^\circ - RT\ln\left(P \cdot \dfrac{n_j}{n}\right) \\ \sum_i \lambda_i a_{ij} - h_j^\circ + T s_j^\circ \\ P_0 - P \\ T_0 - T \end{bmatrix} \begin{array}{l} \}\ i \\ \}\ j\text{ (gas.)} \\ \}\ j\text{ (cond.)} \\ \\ \\ \end{array}$$

5.2.2 Neglecting Pressure and Temperature

Next, we can observe that the bottom two rows of the matrix are no more than the independent system

$$\begin{cases} \Delta P = P_0 - P \\ \Delta T = T_0 - T \end{cases} .$$

If we simply take the care to begin the iteration of our P, T variables at P_0, T_0, then we can neglect the iteration of these variables entirely–as we ought to, given that we have a prescribed isobaric and isothermal environment. This eliminates another two rows and columns from our iteration system. We now have the remarkably straightforward $(i+j)\times(i+j)$ system:

$$
\begin{array}{c}
\begin{array}{ccc}
\overbrace{\hphantom{a_{ij}}}^{j\text{ (cond.)}} & \overbrace{\hphantom{a_{ij}}}^{j\text{ (gas.)}} & \overbrace{\hphantom{a_{ij}}}^{i}
\end{array}
\\
\left[\begin{array}{ccc}
a_{ij} & a_{ij} & 0 \\
0 & \begin{cases} \frac{RT}{n_j}-\frac{RT}{n} & \text{on diag.} \\ -\frac{RT}{n} & \text{off-diag.} \end{cases} & -a_{ij} \\
0 & 0 & -a_{ij}
\end{array}\right]
\begin{array}{l}
\}\ i \\
\}\ j\text{ (gas.)} \\
\}\ j\text{ (cond.)}
\end{array}
\end{array}
\cdot
\begin{bmatrix} \Delta n_j \\ \Delta n_j \\ \Delta\lambda_i \end{bmatrix}
\begin{array}{l}
\}\ j\text{ (cond.)} \\
\}\ j\text{ (gas.)} \\
\}\ i
\end{array}
$$

$$
= \begin{bmatrix}
b_{0,i}-\sum_j a_{ij}n_j \\
\sum_i \lambda_i a_{ij} - h_j^\circ + Ts_j^\circ - RT\ln\left(P\cdot\frac{n_j}{n}\right) \\
\sum_i \lambda_i a_{ij} - h_j^\circ + Ts_j^\circ
\end{bmatrix}
\begin{array}{l}
\}\ i \\
\}\ j\text{ (gas.)} \\
\}\ j\text{ (cond.)}
\end{array}
$$

Yet, we have only removed four columns and as many rows. We can drastically reduce the size of this system by taking several clever tricks out of Gordon and McBride's book.

6 Model Evaluation

We have developed our model and solver sufficiently to produce a script and run it. We implement our method in Python 3 (*Van Rossum and Drake*, 2009), according to the pseudocode presented in algorithm 1 (see appendix B for the complete source code). By comparing it to a well-vetted, third-party method, we can ascertain the quality of our solver without having to get our hands dirty with physical verification. We will compare our method to CEA, the method described by *Gordon and McBride* (1994).

Algorithm 1 Pseudocode for our implementation.

```
Pull thermodynamic information from the online NIST database.
Initialize thermodynamic functions of each species.
Define constants, including precision.
Define initial (reactant) composition as ${n_j}$.
${\lambda_i} \leftarrow$ a i-long array of zeros. These are our Lagrange multipliers.
${\mu_j} \leftarrow$ a $j$-long array of ones. This will be used for damping non-physical leaps Newton's method
may attempt.
proceed $\leftarrow$ true
while proceed is true do
    Build the matrix $A$ and column vector $\vec{b}$.
    Using GE-SPP algorithm, solve $A\vec{x} = \vec{b}$ for ${\Delta n_j}$ and ${\Delta\lambda_i}$.
    for index $j$ in species do
        while $n_j + \Delta n_j \cdot \mu_j \leqslant 0$ do This is the case that we overshoot with the Newton's approximation
        and obtain nonphysical negative values that would break the system.
            $\mu_j \leftarrow 0.5\mu_j$ Until we stop getting negative values, we make our damping variable $\mu$ for
            that problematic value twice as damping.
        $n_j \leftarrow n_j + \Delta n_j \cdot \mu_j$ Because of the previous while loop, this assignment only occurs if it is
        sufficient damped to result in a strictly positive value.
        if $\mu_j \neq 1$ then
            $\mu_j \leftarrow 2\mu_j$ We then increase the damping variable again to prevent us from being stuck
            with a very sluggish system longer than we need to.
    for index $i$ in elements do
        $\lambda_i \leftarrow \lambda_i + \Delta\lambda_i$
    Let proceed be false iff all $\Delta n_j \leqslant$ precision.
return ${n_j}$
```

	CEA	**Our Method**
H_2	4.9476000E-01	4.9506127E-01
N_2	2.5040000E-01	2.5040559E-01
CO	2.4750000E-01	2.4767425E-01
H_2O	3.1579000E-03	2.9432857E-03
CH_4	1.7431000E-03	1.6540116E-03
HCN	1.5955000E-03	1.5125876E-03
CO_2	3.4484000E-04	3.2292458E-04
NH_3	2.1157000E-04	2.1523088E-04
H	1.4791000E-04	1.4791430E-04
HNC	8.0486000E-05	
C_2H_2	2.8479000E-05	3.7026196E-05
CH_3	1.0286000E-05	1.2188264E-05
C_2H_4	5.7852000E-06	6.6733152E-06
HCHO	3.0387000E-06	4.8551226E-06
HNCO	2.9206000E-06	1.0843837E-06
CH_3CN	1.7445000E-06	
CH_2CO	1.1544000E-06	
HCO	7.7372000E-07	7.5516032E-07
NH_2	2.2358000E-07	1.8689346E-07
OH	1.0648000E-07	8.9667035E-08
C_2H_6	8.7987000E-08	
C_2N_2	3.3972000E-08	3.4242423E-08
HCOOH	1.8625000E-08	
C_3H_4,propyne	1.5397000E-08	
CH_3OH	1.2044000E-08	
C_2H_3	1.1406000E-08	
CN	1.0988000E-08	1.4565714E-08
C_3H_3,2-propynl	6.6709000E-09	
C_3H_4,allene	6.2396000E-09	
CH_3CHO	4.8110000E-09	
CH_2	4.5860000E-09	5.7875791E-09
C_2H_5	3.8599000E-09	
C_3H_6,propylene	2.6811000E-09	
C_2H_2,vinylidene	2.6270000E-09	
NO	2.2370000E-09	2.2166039E-09
NH	1.8463000E-09	5.7060281E-10
CH_3CO	1.1597000E-09	
COOH	8.3080000E-10	
HCCO	8.2420000E-10	
C_4H_2	6.5430000E-10	
CH_2OH	5.8310000E-10	

Table 1: Hot, high pressure CHON system. 2000 K, 60 bar. 1 mols N_2O; 1 mols CH_4. Results in mol fractions. Results truncated for size.

When offered the same, arbitrarily selected problem, the reaction of a 50-50 molar mixture of methane, CH_4, with nitrous oxide, N_2O, at a pressure of 20 bar and a temperature of 3000 K, our code and CEA agreed to within 4 decimal places (see Table 1).

The majority of variation appears to be from our NIST-sourced data being more incomplete than the database CEA draws from. Several species are present in the NIST data which are not in the CEA data, such as HCNO and most cyclic hydrocarbons, and others have worse data–for example, HCN, hydrogen cyanide, is present in the NIST database, but its tautomer, HCN, hydrogen isocyanide, is not. While the missing species are minor, their absence still slightly affects calculations for other major species.

	CEA	**Our Method**
H_2O	6.157600E-01	6.119556E-01
H_2	3.641800E-01	3.611154E-01
H	1.232100E-02	1.679517E-02
OH	7.495000E-03	9.675438E-03
O	1.426800E-04	2.719612E-04
O_2	9.540200E-05	1.844924E-04
HO_2	9.338400E-07	1.948192E-06
H_2O_2	8.136600E-07	
O_3	9.708000E-12	1.366240E-11

Table 2: Hot hydrolox. 3000 K, 32 bar. 1 mols O; 3.17 mols H2. Results in mol fractions.

	CEA	**Our Method**
H_2	3.7001000E-01	3.6908517E-01
H_2O	2.3145000E-01	6.3091483E-01
$H_2O(L)$	3.9854000E-01	

Table 3: Cold hydrolox. 500 K, 32 bar. 1 mols O; 3.17 mols H2. Results in mol fractions.

Meanwhile, our code and and CEA agreed well but not as closely for a simple, standard, hot (realistic engine) temperature hydrogen-oxygen (hydrolox) reaction (Table 2). However, they differed substantially for a colder reaction, akin to that of an open flame not burning in an engine (Table 3). This error is caused because of our model not accounting for the condensed liquid water species–it assumes all water will fully vaporize. This is an issue with condensed species generally which we were unable to work out.

From first thermodynamic principles (equation of state), we have:

1. Derived an expression for the **Gibbs energy** with respect to the **enthalphy and entropy of each species**.
2. Re-expressed the function as a system of T, P, composition ($\{n_j\}$), and **Lagrange multipliers**, including expressions for **isothermal**, **isobaric** and **elemental constraints**.
3. Produced a matrix expression of the **simultaneous multivariate Newton iteration equation** for all the equations in our system.
4. Obtained thermodynamic data from the **National Institute of Standards and Technology** (NIST) and solved the system in Python..

Our method is in broad agreement with trusted models, and further improvements remain primarily in the intensive (and possibly today unnecessarily) optimization in the spirit of *Gordon and McBride* (1994)

Bibliography

Callen, H. B. (1998), *Thermodynamics and an Introduction to Thermostatistics*, American Association of Physics Teachers.

Cengel, Y. A., and M. A. Boles (2007), *Thermodynamics: An Engineering Approach 6th Editon (SI Units)*, The McGraw-Hill Companies, Inc., New York.

Clark, J. D. (1972), *Ignition!: an informal history of liquid rocket propellants*, Rutgers University Press.

Frautschi, S. (1982), Entropy in an expanding universe, *Science*, *217*(4560), 593–599.

Gordon, S., and B. J. McBride (1994), Computer program for calculation of complex chemical equilibrium compositions and applications. part 1: Analysis.

Heidman, K. (2016), Sanford gordon and bonnie j. mcbride.

Khan, M. F., Z. Quadri, P. Kulkarni, U. Guven, S. Bhat, and K. Sundarraj (2013), Cfd simulation of a liquid rocket propellant (lh2/lox) combustion chamber, *15th Annual CFD Sympos.*

Linstrom, P., and W. Mallard (), *NIST Standard Reference Database Number 69*, National Institute of Standards and Technology, Gaithersburg MD, 20899, https://doi.org/10.18434/T4D303.

Reynolds, W. (1981), *STANJAN: Interactive computer programs for chemical equilibrium analysis.*

Sutton, G. P., and O. Biblarz (2016), *Rocket propulsion elements*, John Wiley & Sons.

Van Rossum, G., and F. L. Drake (2009), *Python 3 Reference Manual*, CreateSpace, Scotts Valley, CA.

Appendices

A Derivation of the *Gordon and McBride* (1994) Equations

The following appendix contains preliminary typeset work pertaining to the derivation of the iteration equations for constant-T, P systems found in *Gordon and McBride* (1994), *Computer Program for Calculation of Complex Chemical Equilibrium Compositions and Applications.*

These equations numerically solve for the flame temperature and composition of any combusting fluids to high precision. The method was originally developed for NASA for propulsion technology modeling. However, in the 1994 publication, the mathematical explanation for the equations presented is terse to the point of absence. In my honors thesis work, I explain how Gordon & McBride derived these equations.

The equations derived in this appendix may be found originally presented in section 2.3 of *Gordon and McBride* (1994).

A.1 The Constrained Minimization Problem

Solving for thermodynamic equilibrium in a Gibbs (constant pressure and temperature) environment is a constrained minimization problem. Our goal is to minimize the Gibbs energy

$$g = \sum_j \mu_j n_j, \text{ where } \mu_j = \left(\frac{\partial g}{\partial n_j}\right)_{T,P,n_{i\neq j}}$$

Of the system, subject to: atomic conservation (the elemental constraint):

$$\sum_j a_{ij} n_j = b_i^\circ, \ \forall \text{ elements } i$$

The molar sum for gases:

$$\sum_j^{\text{gas.}} n_j = n$$

The conservation of enthalphy:

$$h = h_0, \ h = \sum_j H_j^\circ n_j$$

And the isentropic constraint:

$$s = s_0, \ s = \sum_j S_j n_j$$

Let us define a new term, using Lagrange multipliers to implement the elemental constraint:

$$g^* = g + \sum_i \lambda_i \left(\sum_j (a_{ij} n_j) - b_i^\circ\right)$$

Note that g^* is a function of the species concentration variables n_j and the Lagrange multipliers λ_i. Thus, its total differential is generally:

$$\delta g^* = \sum_j \left(\frac{\partial g^*}{\partial n_j}\right)_{n_{i\neq j},\lambda_i} \delta n_j + \sum_i \left(\frac{\partial g^*}{\partial \lambda_i}\right)_{\lambda_{j\neq i},n_j} \delta\lambda_i$$

And specifically:

$$\delta g^* = \sum_j \left(\mu_j + \sum_i \lambda_i a_{ij}\right) \delta n_j + \sum_i \left(\sum_j (a_{ij} n_j) - b_i^\circ\right) \delta\lambda_i$$

Note that at equilibrium, $\delta g^* = 0$. Assuming that the the species concentrations n_j are independent of the Lagrange multipliers λ_i, then at equilibrium we obtain the following important new definition for the chemical potentials μ_j:

$$\mu_j = -\sum_i \lambda_i a_{ij},\ \forall j$$

A.1.1 Another Definition for the Chemical Potential

At constant temperature and pressure, the specific Gibbs equation holds true. Taking its derivative with respect to the species concentrations reforms it as such:

$$\frac{\partial g}{\partial n_j} = \frac{\partial h}{\partial n_j} - T\frac{\partial s}{\partial n_j}$$

If we substitute our previously defined values for the total specific enthalphy, h, and the total specific entropy, s, we obtain:

$$\frac{\partial g}{\partial n_j} = \frac{\partial}{\partial n_j}\left[\sum_j H_j^\circ n_j\right] - T\frac{\partial}{\partial n_j}\left[\sum_j S_j n_j\right]$$

For those j which are gaseous species, we have been given (from where??):

$$S_j = S_j^\circ - R\ln\frac{n_j}{n} - R\ln P,\ \forall \text{ gaseous } j$$

Substituting that in, distributing, and breaking down one of the logarithms gives us:

$$\frac{\partial g}{\partial n_j} = \frac{\partial}{\partial n_j}\left[\sum_j H_j^\circ n_j\right] + \frac{\partial}{\partial n_j}\left[\sum_j -TS_j^\circ n_j + RTn_j\ln n_j - RTn_j\ln n + RTn_j\ln P\right]$$

Which becomes

$$\mu_j = \frac{\partial g}{\partial n_j} = H_j^\circ - TS_j^\circ + RT\ln n_j + RT - RT\ln n + RT\ln P,\ \forall \text{ gaseous } j$$

Note that from an earlier given definition of chemical potential in gaseous species:

$$\mu_j = \mu_j^\circ + RT\ln\frac{n_j}{n} + RT\ln P \implies \mu_j^\circ = H_j^\circ - TS_j^\circ + RT,\ \forall \text{ gaseous } j$$

For condensed species, the process is very similar but a good deal simpler. We begin with the entropy of condensed species:

$$S_j = S_j^\circ,\ \forall \text{ condensed } j$$

And we arrive at

$$\mu_j = H_j^\circ - TS_j^\circ,\ \forall \text{ condensed } j$$

A.2 A Bit About Newton's Multivariate Rootfinding

Recall that we seek to minimize $g = \sum_j \mu_j n_j$, where our chemical potentials μ_j are simply the differentials of g with respect to n_j. Thus, we can seek a root for the system of equations μ_j. Continuously differentiable multivariate functions $f(\vec{x})$ may be linearly approximated for small (but not necessarily infinitesimal) changes of $\vec{x}$ like so:

$$f(\vec{x} + \Delta\vec{x}) \approx f(\vec{x}) + \sum_n \frac{\partial f}{\partial x_n}\,\Delta x_n$$

Because we are seeking a root,

$$f(\vec{x} + \Delta\vec{x}) = 0$$

Implying that

$$\sum_n \frac{\partial f}{\partial x_n}\,\Delta x_n \approx -f(\vec{x})$$

A.3 Our Equations and a Couple Dirty Tricks

Our system of equations to solve are:

$$\sum_j a_{ij} n_j - b_i^\circ = 0,\ \forall i$$

$$\sum_j^{\text{gas.}} n_j - n = 0$$

$$\sum_j H_j^\circ n_j - h_0 = 0$$

$$\sum_j S_j n_j - s_0 = 0$$

And finally, from

$$\mu_j + \sum_i \lambda_i a_{ij} = 0,\ \forall j$$

We obtain the two sets of equations:

$$H_j^\circ - TS_j^\circ + RT \ln n_j + RT - RT \ln n + RT \ln P + \sum_i \lambda_i a_{ij} = 0,\ \forall \text{ gaseous } j$$

$$H_j^\circ - TS_j^\circ + \sum_i \lambda_i a_{ij} = 0,\ \forall \text{ condensed } j$$

All things considered, this is a fairly elegant system. However, in the name of linearization, Gordon & McBride employ a dirty math trick and divide the last four equations by RT (or by just R, in the case of the fourth equation, the isentropic constraint). Imitating their technique provides us with a notably uglier set of equations:

$$\sum_j a_{ij} n_j - b_i^\circ = 0,\ \forall i$$

$$\sum_j^{\text{gas.}} n_j - n = 0$$

$$\sum_j \frac{H_j^\circ n_j}{RT} - \frac{h_0}{RT} = 0$$

$$\sum_j \frac{S_j n_j}{R} - \frac{s_0}{R} = 0$$

$$\frac{H_j^\circ}{RT} - \frac{S_j^\circ}{R} + \ln n_j + 1 - \ln n + \ln P + \sum_i \frac{\lambda_i a_{ij}}{RT} = 0,\ \forall \text{ gaseous } j$$

$$\frac{H_j^\circ}{RT} - \frac{S_j^\circ}{R} + \sum_i \frac{\lambda_i a_{ij}}{RT} = 0,\ \forall \text{ condensed } j$$

Consider the equations to be of the form $f = 0$, where f is a function of some variables. Behold the next dirty math trick: Gordon & McBride consider these f functions of the "variables" $\ln n$, n_j for those j *which are condensed species*, $\ln n_j$ for those j *which are gaseous species*, $\ln T$, and of course the Lagrangian multipliers λ_i.

Recall that the Newton rootfinding approximation is

$$\sum_n \frac{\partial f}{\partial x_n} \Delta x_n \approx -f(\vec{x})$$

To develop our Newton iteration equations, we must find each functions' partials with respect to these variables.

A.3.1 An Aside on Differentiation

Suppose there exists some function $\varphi(x)$ such that

$$\frac{d}{dx} = \varphi(x)\ \frac{d}{d\ln x}$$

Then we can abuse notation to see that:

$$\frac{d\ln x}{dx} = \varphi(x) = \frac{1}{x}$$

$$\text{So } \frac{d}{dx} = \frac{1}{x}\ \frac{d}{d\ln x} \text{ and } x\ \frac{d}{dx} = \frac{d}{d\ln x}.$$

This will be useful in determining partials of logarithmic pseudo-variables.

A.3.2 An Aside on Specific Heat

Specific heat is the amount of heat energy transferred (Δq) per change in system temperature (dT). For experimental reasons, it becomes useful to define two specific heats, one under constant pressure and one under constant volume conditions.

$$C_p = \left(\frac{\Delta q}{dT}\right)_p\ C_v = \left(\frac{\Delta q}{dT}\right)_v$$

Given our constant-T, P conditions, we are only concerned with the former. Now, we note that at constant pressure,

$$C_p\ dT = \Delta q$$

Now, the second law may be expressed as

$$\Delta q = T\ ds$$

Further, the Gibbs relationship, $dg = dh - T\ ds$, at equilibrium ($dg = 0$) becomes

$$0 = dh - T\ ds \implies T\ ds = dh$$

So at equilibrium and constant pressure,

$$C_p\ dT = \Delta q = T\ ds = dh$$

We roll Leibniz over in his grave once more to find

$$C_p = \frac{dh}{dT} \text{ and } C_p = T\frac{ds}{dT} = \frac{ds}{d\ln T}$$

A.4 Derivation of Iteration Equations

A.4.1 Iteration Equations for the Elemental Constraints

From partials of the set of functions:

$$f_i = \sum_j a_{ij} n_j - b_i^\circ,\ \forall i$$

Being:

$$\frac{\partial f_i}{\partial \ln n} = 0,\ \forall i$$

$$\frac{\partial f_i}{\partial n_j} = a_{ij},\ \forall i,\ \text{condensed } j$$

$$\frac{\partial f_i}{\partial \ln n_j} = a_{ij} n_j, \ \forall i, \ \text{gaseous } j$$

$$\frac{\partial f_i}{\partial \ln T} = 0, \ \forall i$$

$$\frac{\partial f_i}{\partial \lambda_i} = 0, \ \forall i \text{ taken independently}$$

We obtain the set of iteration equations:

$$\sum_j^{\text{gas.}} (a_{ij} n_j \ \Delta \ln n_j) + \sum_j^{\text{cond.}} (a_{ij} \ \Delta n_j) = b_i^\circ - \sum_j a_{ij} n_j, \ \forall i$$

A.4.2 Iteration Equation for the Gaseous Molar Sum

From partials of the function:

$$f = \sum_j^{\text{gas.}} n_j - n$$

Being:

$$\frac{\partial f}{\partial \ln n} = -n$$

$$\frac{\partial f}{\partial n_j} = 0, \ \forall \text{ condensed } j$$

$$\frac{\partial f}{\partial \ln n_j} = n_j, \ \forall \text{ gaseous } j$$

$$\frac{\partial f}{\partial \ln T} = 0$$

$$\frac{\partial f}{\partial \lambda_i} = 0$$

We obtain the iteration equation:

$$\sum_j^{\text{gas.}} (n_j \ \Delta \ln n_j) - n \ \Delta \ln n = n - \sum_j^{\text{gas.}} n_j$$

A.4.3 Iteration Equation for Enthalpy Conservation

From partials of the function:

$$f = \sum_j \frac{H_j^\circ n_j}{RT} - \frac{h_0}{RT}$$

Being:

$$\frac{\partial f}{\partial \ln n} = 0$$

$$\frac{\partial f}{\partial n_j} = \frac{H_j}{RT}, \ \forall \text{ condensed } j$$

$$\frac{\partial f}{\partial \ln n_j} = \frac{H_j n_j}{RT}, \ \forall \text{ gaseous } j$$

$$\frac{\partial f}{\partial \ln T} = T \frac{\partial}{\partial T} \left[\sum_j \frac{H_j^\circ n_j}{RT} - \frac{h_0}{RT} \right] = T \left(\sum_j \frac{\partial}{\partial T} \left[\frac{H_j^\circ n_j}{RT} \right] \right) - \frac{h_0}{RT}$$

$$\frac{\partial f}{\partial \lambda_i} = 0$$

Here we must pause and consider the partial with respect to $\ln T$. H_j° is also a function of T. Let us take a closer look at the terms within the summation, considering the chain rule.

$$\frac{\partial}{\partial T}\left[\frac{H_j^\circ n_j}{RT}\right] = \frac{\partial H_j^\circ}{\partial T}\frac{n_j}{RT} + H_j^\circ \frac{\partial}{\partial T}\left[\frac{n_j}{RT}\right]$$

Now, with our discussion on specific heats in mind, observe that

$$\frac{\partial H_j^\circ}{\partial T} = C_{p,j}^\circ$$

Thus,

$$\frac{\partial}{\partial T}\left[\frac{H_j^\circ n_j}{RT}\right] = \frac{C_{p,j}^\circ n_j}{RT} + \frac{H_j^\circ n_j}{RT^2}$$

And

$$\frac{\partial f}{\partial \ln T} = T\sum_j \left(\frac{C_{p,j}^\circ n_j}{RT} + \frac{H_j^\circ n_j}{RT^2}\right) - \frac{h_0}{RT} = \sum_j \frac{C_{p,j}^\circ n_j}{R} + \sum_j \frac{H_j^\circ n_j}{RT} - \frac{h_0}{RT}$$

But note that to satisfy the enthalphy conservation constraint,

$$\sum_j H_j^\circ n_j = h_0 \implies \sum_j \frac{H_j^\circ n_j}{RT} = \frac{h_0}{RT}$$

So those last two terms cancel and our partial with respect to $\ln T$ simplifies to

$$\frac{\partial f}{\partial \ln T} = \sum_j \frac{C_{p,j}^\circ n_j}{R}$$

After all that labor, we obtain the iteration equation:

$$\sum_j^{\text{gas.}} \left(\frac{H_j n_j}{RT}\,\Delta \ln n_j\right) + \sum_j^{\text{cond.}} \left(\frac{H_j}{RT}\,\Delta n_j\right) + \left(\sum_j \frac{C_{p,j}^\circ n_j}{R}\right)\Delta \ln T$$

$$= \frac{h_0}{RT} - \sum_j \frac{H_j^\circ n_j}{RT}$$

A.4.4 Iteration Equation for Isentropic Constraint

Before we can determine the iteration equations from the isentropic constraint function

$$f = \sum_j \frac{S_j n_j}{R} - \frac{s_0}{R}$$

We must rewrite f considering that S_j is a function of n_j, n, T, P for gaseous j and a constant for condensed j. S_j is given by

$$S_j = S_j^\circ - R\ln\frac{n_j}{n} - R\ln P,\ \forall \text{ gaseous } j$$

$$S_j = S_j^\circ,\ \forall \text{ condensed } j$$

To substitute this in, we break the summation of f into two parts, obtaining the more explicit expression

$$f = \sum_j^{\text{gas.}} \frac{n_j}{R}\left(S_j^\circ - R\ln\frac{n_j}{n} - R\ln P\right) + \sum_j^{\text{cond.}} \frac{S_j^\circ n_j}{R} - \frac{s_0}{R}$$

Or (by simplifying the logarithm, distributing, and rearranging terms):

$$f = \sum_j^{\text{gas.}} \frac{S_j^\circ n_j}{R} - \sum_j^{\text{gas.}} n_j \ln n_j + \sum_j^{\text{gas.}} n_j \ln n - \sum_j^{\text{gas.}} n_j \ln P + \sum_j^{\text{cond.}} \frac{S_j^\circ n_j}{R} - \frac{s_0}{R}$$

Now we can obtain partials of the function, being:

$$\frac{\partial f}{\partial \ln n} = \sum_j^{\text{gas.}} n_j = n$$

$$\frac{\partial f}{\partial n_j} = \frac{S_j^\circ}{R},\ \forall \text{ condensed } j$$

$$\frac{\partial f}{\partial \ln n_j} = \frac{S_j^\circ n_j}{R} - (n_j + n_j \ln n_j) + n_j \ln n - n_j \ln P,\ \forall \text{ gaseous } j$$

$$\frac{\partial f}{\partial \ln T} = \frac{\partial}{\partial \ln T}\left[\sum_j^{\text{gas.}} \frac{S_j^\circ n_j}{R} + \sum_j^{\text{cond.}} \frac{S_j^\circ n_j}{R}\right]$$

$$\frac{\partial f}{\partial \lambda_i} = 0$$

Clearly, we need to look at our partials with respect to $\ln n_j$s and $\ln T$ before we continue. The partials with respect to the $\ln n_j$s can be rewritten as

$$\frac{\partial f}{\partial \ln n_j} = \frac{S_j^\circ n_j}{R} - \frac{n_j R \ln n_j}{R} + \frac{n_j R \ln n}{R} - \frac{n_j R \ln P}{R} - n_j,\ \forall \text{ gaseous } j$$

Which becomes

$$\frac{\partial f}{\partial \ln n_j} = \frac{n_j}{R}\left(S_j^\circ - R\ln n_j + R\ln n - R\ln P\right) - n_j = \frac{S_j n_j}{R} - n_j,\ \forall \text{ gaseous } j$$

Now, the partials with respect to $\ln T$ can be collapsed into a single sum. From there, remembering what we know about specific heats, it quickly simplifies down

$$\frac{\partial f}{\partial \ln T} = \frac{\partial}{\partial \ln T}\left[\sum_j \frac{S_j^\circ n_j}{R}\right] = \sum_j \left(\frac{n_j}{R}\frac{\partial}{\partial \ln T}\left[S_j^\circ\right]\right) = \sum_j \frac{n_j}{R} C_{p,j}^\circ$$

So now we have obtained the monstrous iteration equation:

$$n\,\Delta \ln n + \sum_j^{\text{gas.}} \left(\frac{S_j n_j}{R}\,\Delta \ln n_j\right) + \sum_j^{\text{cond.}} \left(\frac{S_j^\circ}{R}\,\Delta n_j\right) - \sum_j^{\text{gas.}} (n_j\,\Delta \ln n_j)$$
$$+ \left(\sum_j \frac{C_{p,j}^\circ n_j}{R}\right)\Delta \ln T = \frac{s_0}{R} - \sum_j \frac{S_j n_j}{R}$$

However, we can make the left-hand side a bit nicer at the expense of the right-hand side by adding to it the iteration equation for the gaseous molar sum.

$$\sum_j^{\text{gas.}} \left(\frac{S_j n_j}{R}\,\Delta \ln n_j\right) + \sum_j^{\text{cond.}} \left(\frac{S_j^\circ}{R}\,\Delta n_j\right) + \left(\sum_j \frac{C_{p,j}^\circ n_j}{R}\right)\Delta \ln T$$
$$= \frac{s_0}{R} - \sum_j \frac{S_j n_j}{R} + n - \sum_j^{\text{gas.}} n_j$$

A.4.5 Iteration Equations for the Gaseous Chemical Potentials

From partials of the set of functions:

$$f_j = \frac{\mu_j}{RT} = \frac{H_j^\circ}{RT} - \frac{S_j^\circ}{R} + \ln n_j + 1 - \ln n + \ln P + \sum_i \frac{\lambda_i a_{ij}}{RT}, \ \forall \text{ gaseous } j$$

Being:

$$\frac{\partial f_j}{\partial \ln n} = -1, \ \forall \text{ gaseous } j$$
$$\frac{\partial f_j}{\partial \ln n_j} = 1, \ \forall \text{ gaseous } j$$
$$\frac{\partial f_j}{\partial \ln T} = -\frac{H_j^\circ}{RT}, \ \forall \text{ gaseous } j$$
$$\frac{\partial f_j}{\partial \lambda_i} = \frac{a_{ij}}{RT}, \ \forall i, \ \text{gaseous } j$$

We obtain the set of iteration equations:

$$-\Delta \ln n + \Delta \ln n_j + \sum_i \left(\frac{a_{ij}}{RT} \, \Delta\lambda_i \right) - \frac{H_j^\circ}{RT} \, \Delta \ln T = -\frac{\mu_j}{RT}, \ \forall \text{ gaseous } j$$

A.4.6 Iteration Equations for the Condensed Chemical Potentials

And finally, from partials of the set of functions:

$$f_j = \frac{\mu_j}{RT} = \frac{H_j^\circ}{RT} - \frac{S_j^\circ}{R} + \sum_i \frac{\lambda_i a_{ij}}{RT}, \ \forall \text{ condensed } j$$

Being:

$$\frac{\partial f_j}{\partial \ln n} = 0, \ \forall \text{ condensed } j$$
$$\frac{\partial f_j}{\partial n_j} = 0, \ \forall \text{ condensed } j$$
$$\frac{\partial f_j}{\partial \ln T} = -\frac{H_j^\circ}{RT}, \ \forall \text{ condensed } j$$
$$\frac{\partial f_j}{\partial \lambda_i} = \frac{a_{ij}}{RT}, \ \forall i, \ \text{condensed } j$$

We obtain the set of iteration equations:

$$\sum_i \left(\frac{a_{ij}}{RT} \, \Delta\lambda_i \right) - \frac{H_j^\circ}{RT} \, \Delta \ln T = -\frac{\mu_j}{RT}, \ \forall \text{ condensed } j$$

A.5 Compiled Iteration Equations

The following equations, functions of $\Delta \ln n$, Δn_j for condensed j, $\Delta \ln n_j$ for gaseous j, $\Delta \ln T$, and $\Delta\lambda_i$ for all i have been obtained. They may be solved analytically; using Gaussian elimination methods on a matrix with respect to a column vector of the variables is most straight-forward.

$$\sum_j^{\text{gas.}} (a_{ij} n_j \, \Delta \ln n_j) + \sum_j^{\text{cond.}} (a_{ij} \, \Delta n_j) = b_i^\circ - \sum_j a_{ij} n_j, \ \forall i$$

$$\sum_j^{\text{gas.}} (n_j \, \Delta \ln n_j) - n \, \Delta \ln n = n - \sum_j^{\text{gas.}} n_j$$

$$\sum_j^{\text{gas.}} \left(\frac{H_j n_j}{RT} \, \Delta \ln n_j \right) + \sum_j^{\text{cond.}} \left(\frac{H_j}{RT} \, \Delta n_j \right) + \left(\sum_j \frac{C^\circ_{p,j} n_j}{R} \right) \Delta \ln T$$

$$= \frac{h_0}{RT} - \sum_j \frac{H^\circ_j n_j}{RT}$$

$$\sum_j^{\text{gas.}} \left(\frac{S_j n_j}{R} \, \Delta \ln n_j \right) + \sum_j^{\text{cond.}} \left(\frac{S^\circ_j}{R} \, \Delta n_j \right) + \left(\sum_j \frac{C^\circ_{p,j} n_j}{R} \right) \Delta \ln T$$

$$= \frac{s_0}{R} - \sum_j \frac{S_j n_j}{R} + n - \sum_j^{\text{gas.}} n_j$$

$$-\Delta \ln n + \Delta \ln n_j + \sum_i \left(\frac{a_{ij}}{RT} \, \Delta \lambda_i \right) - \frac{H^\circ_j}{RT} \, \Delta \ln T = -\frac{\mu_j}{RT}, \ \forall \text{ gaseous } j$$

$$\sum_i \left(\frac{a_{ij}}{RT} \, \Delta \lambda_i \right) - \frac{H^\circ_j}{RT} \, \Delta \ln T = -\frac{\mu_j}{RT}, \ \forall \text{ condensed } j$$

However, such a matrix would be both very large (of dimension $i + j + 3$ by $i + j + 2$) and very sparse. By clever substitution, we can both reduce the size and the sparsity of the matrix to be solved.

B Python Source Code

Scripts are written to be run in Python 3 (*Van Rossum and Drake*, 2009).

B.1 combustion.py

This is the main script.

```
import numpy as Math
import solver
from nist import elementalComposition
from nist import getElementsInSpecies as aij
from nist import getEnthalpyAtT as h
from nist import getEntropyAtT as s

# Solver parameters
precision = 1e-15

# System parameters
R = 8.31446261815324    # J / K / mol
temperature = T0 = 3000.  # K
pressure = P0 = 32.     # bar
composition = {       # in mols
  ’H2’: 3.17,
  ’O2’: 1,
}

# Start tracking all the things we’ve got to track
elements, gaseous, condensed = set(), set(), set()

# Load up the elements automagically
for formula in composition:
  for element in elementalComposition[formula]:
    elements.add(element)
elements = list(elements)
elements.sort()

# Load up the species automagically
for formula in elementalComposition:
  if Math.all([element in elements for element in elementalComposition[formula]]):
    gaseous.add(formula)
gaseous, condensed = list(gaseous), list(condensed)
gaseous.sort()
condensed.sort()

# Get the full list of species
species = gaseous + condensed
I, J = len(elements), len(species)

# Initial element amounts
elementMols = {}
for i in elements:
  elementMols[i] = Math.sum([composition[j] * aij(i, j) for j in composition])

# Start tracking iteration variables
mols, lagrange_is= {}, {}
for j in species:
  mols[j] = 1
for i in elements:
  lagrange_is[i] = 0

# Iterate!
print("Beginning iteration...")
proceed, iteration, damps = True, 0, [1 for i in range(J)]
while proceed:

  # Solve the matrix
  try:
    x = solver.solve(temperature, pressure, elements, gaseous, condensed, mols,
                                            elementMols, lagrange_is)
  except ValueError as e:
    print(e, ’Iteration halted at i =’, iteration)
    break

  # Update variables
  for index in range(J):
    while mols[species[index]] + x[index] * damps[index] <= 0:
      damps[index] /= 2
    mols[species[index]] += x[index] * damps[index]
    if damps[index] != 1:
      damps[index] *= 2
  for index in range(I):
    lagrange_is[elements[index]] += x[index+J]

  # Keep going?
  iteration += 1
  proceed = Math.any([Math.abs(delta) > precision for delta in x[:J]])

  # Print thingy
  if Math.mod(iteration, 1000)==0:
    print("...n="+str(iteration))

# A function (NOTE: returns in kJ NOT J!!!)
def gibbs(comp):
  tot = 0
  n = 0
  for j in comp:
    if j in gaseous:
      n += comp[j]
  for j in comp:
    potiential = h(j, temperature) - temperature * s(j, temperature)
    if j in gaseous:
      potiential += R * temperature * Math.log(pressure * comp[j] / n)
    tot += potiential * comp[j]
  return tot*1e-4

# Returns in J/K
def entropy(comp):
  tot = 0
  n = 0
  for j in comp:
    if j in gaseous:
      n += comp[j]
  for j in comp:
    S = s(j, temperature)
    if j in gaseous:
      S -= R * Math.log(pressure * comp[j] / n)
    tot += S * comp[j]
  return tot

# Clean up & sort
sortedSpecies = [{’name’: j, ’amount’: mols[j]} for j in species]
sortedSpecies.sort(key=lambda e: -e[’amount’])

# Interface
print(’\n- Iteration Complete (n=’+str(iteration)+’) -’)
print(’T:’, temperature, ’K’)
print(’P:’, pressure, ’bar’)

print(’\n- Reactant Composition -’)
for j in composition:
  print(j+’:’, composition[j], ’mol’)

totalMols = Math.sum([j[’amount’] for j in sortedSpecies])
print(’\n- Product Composition -’)
```

```
for val in sortedSpecies:
  if val[’amount’] > precision:
    print((val[’name’]+’:’).ljust(8), val[’amount’]/totalMols, ’mol fraction’)
print(’Total:’, totalMols, ’mol’)

print(’\n- Delta G, Delta S -’)
print(gibbs(mols) - gibbs(composition), ’kJ’)
print(entropy(mols) - entropy(composition), ’J/K’)


print(’\n- Elemental Residuals -’)
for i in elements:
  residual = Math.sum([mols[j] * aij(i, j) for j in species]) - elementMols[i]
  print(i+’:’, residual, ’mol’)
```

B.2 data.py

This script is used for preparing raw NIST data.

```
import urllib.request
import csv, os

print("Obtaining NIST thermodynamic data...")

thermodynamicData = dict()

# URL References
NISTCodes = {

  'H':  'H-001',
  'NH': 'H-026',
  'HNO':  'H-027',
  'HNO2 (Cis)': 'H-028',
  'HNO2 (Trans)': 'H-029',
  'HNO3': 'H-030',
  'OH': 'H-038',
  'HO2':  'H-043',
  'H2': 'H-050',
  'NH2':  'H-060',
  'H2O':  'H-064',
  # 'H2O2': 'H-070',
  'NH3':  'H-083',
  'N2H4': 'H-091',

  'O':  'O-001',
  'O2': 'O-029',
  'O3': 'O-056',

  'CO': 'C-093',
  'CO2':  'C-095',
  'CH': 'C-043',
  'HCN':  'C-052',
  'HNCO': 'C-053',
  'HCO':  'C-054',
  'CH2':  'C-057',
  'H2CO': 'C-061',
  'CH3':  'C-062',
  'CH4':  'C-067',
  'CN': 'C-080',
  'CNO':  'C-087',
  'CN2':  'C-088',
  'C2': 'C-113',
  'C2H':  'C-124',
  'C2H2': 'C-127',
  'C2H4': 'C-128',
  'C2H4O': 'C-129',
  'C2N':  'C-133',
  'C2N2': 'C-134',
  'C3': 'C-138',

  'N':  'N-002',
  'NO': 'N-005',
  'NO2':  'N-007',
  'NO3':  'N-009',
  'N2': 'N-023',
  'N2O':  'N-026',
  'N3': 'N-034',

}

def parseCompositionFromEquation(code):
  composition = {}
  lastchar = ''
  for char in code:
    if char == ' ':
      break
    if char == '-':
      composition['e-'] = 1
      continue
    if char == '+':
      composition['e-'] = -1
      continue
    if char.isalpha():
      composition[char] = 1
      lastchar = char
      continue
    if char.isdigit():
      composition[lastchar] = int(char)
      continue

  return composition

# This will be used in many other scripts
elementalComposition = {code: parseCompositionFromEquation(code) for code in NISTCodes}

# Load enthalpies from the web
for species in NISTCodes:

  code, file = NISTCodes[species], os.getcwd() + '/nist/'+species+'.csv'

  # Data structure for the data
  thermodynamicData[species] = dict()

  # Obtain from remote site
  if not os.path.isfile('filename.txt'):
    url = 'https://janaf.nist.gov/tables/' + code + '.txt'
    urllib.request.urlretrieve(url, file)

  # Open & parse
  with open(file) as csvFile:

    # Open file
    csvReader = csv.reader(csvFile, delimiter='\t')

    # Skip the first two rows
    rowskip = 2

    # Extract enthalphy information about each species
    for row in csvReader:

      # Skip the first two rows
      if rowskip:
        rowskip -= 1
        continue

      # Reparse their infinity name
      if row[7] == 'INFINITE':
        row[7] = 'Infinity'

      # Store values
      try:
        thermodynamicData[species][float(row[0])] = {
          'Cp': float(row[1]), # specific heat at constant pressure
          'S': float(row[2]), # entropy (J/mol/K)
          'H': float(row[4])*1000, # enthalpy (J/mol)
          'Hf': float(row[5])*1000, # enthalpy of formation (J/mol)
        }
      except ValueError:
        if (row[0] == ''):
          continue
        else:
          print(row)
          exit('Critical problem parsing data for species ' + species + '.')


# What this package is all about: a method to retrieve interpolated thermo data
def getThermodynamicData(species, temperature):

  # Load all temperatures
  temperatures = [t for t in thermodynamicData[species]]

  # Get the coldest temperature we have
  index = 0
  for i in range(len(temperatures)):
    if temperature < temperatures[i]:
      break
    index += 1

  # These are the temperatures to interpolate between
  floorTemp = temperatures[index-1]
  ceilTemp = temperatures[index]

  # Interpolation value
  interp = (temperature-floorTemp) / (ceilTemp-floorTemp)

  # Prepare to output
  output = {}

  # Get interpolated values
  for key in thermodynamicData[species][floorTemp]:
    floorVal = thermodynamicData[species][floorTemp][key]
    ceilVal = thermodynamicData[species][ceilTemp][key]
    output[key] = (1-interp) * floorVal + interp * ceilVal

  # Wrap up
  return output

def getElementsInSpecies(element, species):
  try:
    val = elementalComposition[species][element]
  except KeyError:
    val = 0
  return val

def getEntropyAtT(species, temperature):
  try:
    return getThermodynamicData(species, temperature)['S']
  except IndexError:
    exit("Error finding entropy of "+species+" at T="+str(temperature))

def getEnthalpyAtT(species, temperature):
  try:
    return getThermodynamicData(species, 298.15)['Hf'] + getThermodynamicData(species,
                                            temperature)['H']
  except IndexError:
    exit("Error finding enthalpy of "+species+" at T="+str(temperature))

def getSpecificHeatAtT(species, temperature):
  try:
    return getThermodynamicData(species, temperature)['Cp']
  except IndexError:
    exit("Error finding specific heat of "+species+" at T="+str(temperature))
```

B.3 solver.py

This script generates and solve the iteration matrix of the model.

```
import numpy as Math
from numpy import log as ln
from nist import getElementsInSpecies as aij
from nist import getEnthalpyAtT as h
from nist import getEntropyAtT as s

R = 8.31446261815324

# Gaussian scaled partial pivot
def gaussian(A, b):

  n = len(b)

  # Keeps track of row order
  L = [i for i in range(n)]
  # Used for scaling the rows
  S = [0.0 for i in range(n)]
  # Solving for x vect
  x = [0.0 for i in range(n)]

  # Find scaling values
  for i in range(n-1):
    for j in range(n-1):
      S[i] = max(S[i], abs(A[i][j]))

  # Forward Elimination
  for k in range(n-1):
    R = 0.0
    for i in range(k, n-1):
      temp = abs(A[L[i]][k] / S[L[i]])
      if temp > R:
        R = temp
        index = i

    # Do the swap
    temp = L[index]
    L[index] = L[k]
    L[k] = temp

    # Zero out below the diagonal
    for i in range (k+1, n):
      xmult = A[L[i]][k]/A[L[k]][k]
      for j in range(k+1, n):
        A[L[i]][j] = A[L[i]][j] - xmult * A[L[k]][j]
      b[L[i]] = b[L[i]] - xmult * b[L[k]]

  # Back substitution
  x[n-1] = b[L[n-1]] / A[L[n-1]][n-1]

  for k in range(n-1, -1, -1):
    sum = b[L[k]]
    for j in range(k+1, n):
      sum = sum - A[L[k]][j] * x[j]
    x[k] = sum / A[L[k]][k]

  return x

# The iterator function
def solve (T, P, elements, gaseous, condensed, moles, b_0s, lambda_is):

  species = gaseous+condensed

  # Make some lil functions
  l = lambda i: lambda_is[i]
```

```
b = lambda i: b_0s[i]
n = lambda j: moles[j]
N = Math.sum([n(j) for j in gaseous])

if Math.min([n(j) for j in species]) <= 0:
  raise ValueError(’Negative or zero mols of something.’)

# Build the iteration matrix
A = (
  ([
    [aij(i_, j) for j in condensed] +
    [aij(i_, j) for j in gaseous] +
    [0 for i in elements]
    for i_ in elements
  ]) + ([
    [0 for j in condensed] +
    [R*T/n(j) - R*T/N if j_==j else -R*T/N for j in gaseous] +
    [-aij(i, j_) for i in elements]
    for j_ in gaseous
  ]) + ([
    [0 for j in condensed] +
    [0 for j in gaseous] +
    [-aij(i, j_) for i in elements]
    for j_ in condensed
  ])
)

# Set up the right-hand-side
b = (
  [b(i) - Math.sum([aij(i, j)*n(j) for j in species]) for i in elements] +
  [Math.sum([l(i)*aij(i, j) for i in elements]) - h(j, T) + T*s(j, T) - R*T*ln(P*n(j)/
                                          N) for j in gaseous] +
  [Math.sum([l(i)*aij(i, j) for i in elements]) - h(j, T) + T*s(j, T)for j in
                                          condensed]
)

# Solve
return gaussian(A, b)
```