

# Contents

<b>Nomenclature</b>	<b>3</b>
<b>1 Minimization Procedures</b>	<b>4</b>
1.1 Constraints for Both Procedures . . . . .	4
1.2 The Lagrangian . . . . .	4
1.3 Newton-Raphson Iterative Method . . . . .	5
1.4 Gibbs Minimization . . . . .	5
1.4.1 Constraints . . . . .	6
1.4.2 Derivatives . . . . .	7
1.4.3 Derivatives of $f_1$ . . . . .	8
1.4.4 Derivatives of $f_2$ . . . . .	9
1.4.5 Derivatives of $f_3$ . . . . .	9
1.4.6 Derivatives of $f_4$ . . . . .	9
1.4.7 Derivatives of $f_5$ . . . . .	10
1.4.8 Derivatives of $f_6$ . . . . .	10
1.4.9 Reduced Gibbs Equations . . . . .	11
1.5 Helmholtz Minimization . . . . .	13
1.5.1 Constraints . . . . .	13
1.5.2 Derivatives . . . . .	14
1.5.3 Derivatives for $f_1$ . . . . .	15
1.5.4 Derivatives for $f_2$ . . . . .	15
1.5.5 Derivatives for $f_3$ . . . . .	15
1.5.6 Derivatives for $f_4$ . . . . .	16
1.5.7 Derivatives for $f_5$ . . . . .	16
1.5.8 Reduced Helmholtz Equations . . . . .	17
1.6 Useful Relationships . . . . .	19
<b>2 Standard State Chemical Potential Derivatives</b>	<b>19</b>
2.1 Gibbs Derivative . . . . .	19
2.2 Helmholtz Derivative . . . . .	20
<b>3 Quick code overview</b>	<b>21</b>
<b>4 Code specifics</b>	<b>22</b>
4.1 Overview . . . . .	22
4.2 Function details . . . . .	25

4.2.1	Constructor function . . . . .	25
4.2.2	compute_equilibriumXY() functions . . . . .	25
4.2.3	NASA_fits() . . . . .	26
4.2.4	Namespace functions . . . . .	26
4.2.5	compute_mixture_properties() . . . . .	26
4.2.6	compute_derivatives() . . . . .	26
4.2.7	compute_derivativesCFD() . . . . .	26
4.2.8	check_convergence() . . . . .	26
4.2.9	compute_damping() . . . . .	26

## Nomenclature

		$F$	Helmholtz free-energy
$\hat{R}$	Universal gas constant $\approx 8314$ kJ/kg-K	$G$	Gibbs free-energy
$\lambda$	Lagrange multiplier	$h'$	Sum of contributions from each species towards enthalpy constraint, i.g. $\sum_j^{\text{NS}} \mathcal{N}_j H_j^\circ$
$\mathcal{L}$	The Lagrangian	$h'_0$	User-specified enthalpy (J/kg)
$\mathcal{N}$	Total kg-moles per unit mass in the system	$H_j^\circ$	Standard state enthalpy of species $j$ (J/kg)
$\mathcal{N}_j$	Number of kg-moles per unit mass of species $j$ in the system	$MW_j$	Molecular weight of species $j$ in kg / kg-mole
$\mu_j$	Chemical potential of species $j$	$p_{\text{ref}}$	Reference temperature used to compute chemical potential
$\mu_j^\circ$	Standard state chemical potential of species $j$	$q_j$	Charge of species $j$ (0 for neutrals, 1 for ions, -1 for electrons)
$\rho$	Density ( $\text{kg}/\text{m}^3$ )	$T$	Temperature (K)
$a_{ij}$	Stoichiometric coefficients, e.g. number of atoms of element $i$ in species $j$ . Weighted by element $i$ 's moelcular weight	$u'$	Sum of contributions from each species towards internal energy constraint, i.g. $\sum_j^{\text{NS}} \mathcal{N}_j U_j^\circ$
$b_i$	Sum of contributions from each species towards constraint for element $i$ , i.g. $\sum_{j=1}^{\text{NS}} a_{i,j} \mathcal{N}_j$	$u'_0$	User-specified internal energy (J/kg)
$b_i^\circ$	Specified number of moles of element $i$	$U_j^\circ$	Standard state internal energy of species $j$ (J/kg)
$c_{p,j}^\circ$	Standard state specific heat of species $j$	$V$	Specific volume ( $\text{m}^3/\text{kg}$ )
$c_{v,j}^\circ$	Standard state specific heat of species $j$		

# 1 Minimization Procedures

## 1.1 Constraints for Both Procedures

There are two constraints that are used the most in both energy minimization procedures. The first, equation [1.1], is always used, while the charge constraints in equation [1.2] is only used when ions are present:

$$\sum_{j=1}^{\text{NS}} a_{ij} \mathcal{N}_j - b_i^\circ = 0 \quad (1.1)$$

$$\sum_{j=1}^{\text{NS}} q_j \mathcal{N}_j = 0 \quad (1.2)$$

Since these are used in abundance for both procedures, their constraints are placed in the Lagrangian term. Other constraints (such as specified volume, temperature, internal energy, and entropy) are given their own residual equations in the Newton solve without addition into the Lagrangian.

Equation [1.1] says that the number of moles of element  $i$  must remain constant during the minimization process, as atoms can not be created or destroyed without nuclear reactions being involved.  $b_i^\circ$  is the number of moles of element  $i$  before minimizing, and  $a_{ij}$  is the stoichiometric coefficient, i.e. how many atoms of element  $i$  are in species  $j$ . In this code, the stoichiometric coefficient is actually multiplied by the element's molecular weight. The current reason is not known, though it needs to be figured out why this method yields correct results.

Equation [1.2] says that the charge of the system must remain neutral, i.e., if cations are formed, than there must be an equal number of free electrons in the gas to balance the charge.

## 1.2 The Lagrangian

The Lagrangian is defined as;

$$\mathcal{L} = f + \sum_j \lambda_j g_j \quad (1.3)$$

where  $f$  is the function we want to minimize,  $\lambda_j$  are Lagrange multipliers, and

$g_j$  are the constraint functions equal to 0. Instead of minimizing just  $f$ , we aim to minimize the Lagrangian  $\mathcal{L}$ .

### 1.3 Newton-Raphson Iterative Method

A Newton-Raphson iterative procedure is used to minimize the energies. Since this is essentially a root finding system of equations, and we are seeking the solution update, the method looks like the following after using a taylor expansion on the function  $f$ :

$$\sum_{i=1}^N \frac{\partial f}{\partial x_i} \Delta x_i = -f(x) \quad (1.4)$$

Here,  $f(x)$  is the function that is being minimized. In the case of Gibbs and Helmholtz energy minimization,  $f$  takes the form of  $\partial \mathcal{L} / \partial x$ , with  $x$  being non-linear variables used to guide the convergence of the system. The non-linear correction variables are in the form of  $\Delta \ln \mathcal{N}_j$ ,  $\Delta \ln \mathcal{N}$ ,  $\Delta \ln T$ , and  $\pi_{i,q} = \Delta \pi_{i,q}$ . For the last correction variable, it is argued that setting  $\pi$  to 0 at the beginning of each update does not influence the solution for the non-reduced equations, so the  $\Delta$  is dropped for that term. This will influence the form of the update equations, which will be talked about in due time.

### 1.4 Gibbs Minimization

The Gibbs energy is a function of temperature, pressure, and composition of a gas:

$$G = G(T, p, \mathcal{N}_1, \mathcal{N}_2, \dots, \mathcal{N}_{NS}) = \sum_{j=1}^{NS} \mathcal{N}_j \mu_j \quad (1.5)$$

Where NS is the number of gas species,  $\mathcal{N}_j$  is the number of kg-moles of species  $j$  per kilogram (i.e., the mole-mass fraction), and  $\mu_j$  is the chemical potential of species  $j$  defined by:

$$\mu_j = \frac{\partial G}{\partial \mathcal{N}_j} = \mu_j^\circ + \hat{R}T \left[ \ln \left( \frac{p}{p_{ref}} \right) + \ln \left( \frac{\mathcal{N}_j}{\mathcal{N}} \right) \right] \quad (1.6)$$

Here,  $\hat{R}$  is the universal gas constant,  $T$  and  $p$  are the temperature and pressure of the system,  $p_{ref}$  is the reference pressure (usually takes as 101,325 Pa, or 1 bar depending on the literature), and  $\mathcal{N}$  is the number of kg-moles

per unit mass in the system. In order to find chemical equilibrium, the derivative of  $G$  wrt  $\mathcal{N}_j$  is set to 0 for all species. However, a number of constraints need to be added in.

#### 1.4.1 Constraints

Firstly, the elemental constraint from equation [1.1] is added into the Lagrangian for our Gibbs function. If ions are present, the charge constraint [1.2] is also put in. Therefore, our Lagrangian is:

$$\mathcal{L} = \sum_{j=1}^{NS} \mu_j \mathcal{N}_j + \sum_{i=1}^{NE} \lambda_i \left[ \sum_{j=1}^{NS} a_{ij} \mathcal{N}_j - b_i^\circ \right] + \lambda_q \left[ \sum_{j=0}^{NS} q_j \mathcal{N}_j \right] \quad (1.7)$$

The constraints available for the Gibbs energy are:

- Hold temperature (T) and pressure (P) constant.
- Hold enthalpy (H) and pressure (P) constant.
- Hold entropy (S) and pressure (P) constant.

For all Gibbs minimizations, we need the molar constraint:

$$\sum_{j=1}^{NS} \mathcal{N}_j - \mathcal{N} = 0 \quad (1.8)$$

For (TP) minimization, only the elemental and charge (if present) constraints are necessary. For (HP) minimization, the constraint is defined by:

$$h' - h'_0 = 0 \quad (1.9)$$

Where  $h'_0$  is the specified enthalpy in J/kg, and  $h'$  is solved for during the minimization process as:

$$h' = \sum_{j=1}^{NS} \mathcal{N}_j H_j^\circ \quad (1.10)$$

For (SP) minimization, the constraint is defined by:

$$s' - s'_0 = 0 \quad (1.11)$$

Where  $s'_0$  is the specified entropy and  $s'$  is solved for in the minimization process as:

$$s' = \sum_{j=1}^{NS} \mathcal{N}_j S_j \quad (1.12)$$

where:

$$S_j = S_j^\circ - \hat{R} \ln \frac{\mathcal{N}_j}{\mathcal{N}} - \hat{R} \ln p \quad (1.13)$$

#### 1.4.2 Derivatives

We now take the derivatives of  $\mathcal{L}$  wrt  $\mathcal{N}_j$ , and each  $\lambda$  and denote them as  $f$ :

$$f_1 = \frac{\partial \mathcal{L}}{\partial \mathcal{N}_j} = \mu_j + \sum_{i=1}^{NE} \lambda_i a_{ij} + \lambda_q q_j = 0 \quad (1.14)$$

$$f_2 = \frac{\partial \mathcal{L}}{\partial \lambda_i} = \sum_{j=1}^{NS} a_{ij} \mathcal{N}_j - b_i = 0 \quad (1.15)$$

$$f_3 = \frac{\partial \mathcal{L}}{\partial \lambda_q} = \sum_{j=0}^{NS} q_j \mathcal{N}_j = 0 \quad (1.16)$$

These three equations are our minimization functions. Through use of the aforementioned Newton-Raphson method, we can solve these equations. Additional residual equations for total moles of the system, enthalpy, and entropy are:

$$f_4 = \sum_{j=1}^{NS} \mathcal{N}_j - \mathcal{N} = 0 \quad (1.17)$$

$$f_5 = \frac{h' - h'_0}{\hat{R}T} = \frac{\sum_{j=1}^{NS} \mathcal{N}_j H_j^\circ - h'_0}{\hat{R}T} \quad (1.18)$$

$$f_6 = \frac{s' - s'_0}{\hat{R}} = \frac{\sum_{j=1}^{NS} \mathcal{N}_j S_j - s'_0}{\hat{R}} \quad (1.19)$$

Special care needs to be taken in the enthalpy constraint to include the enthalpies of formation and/or reference enthalpies, if applicable.

### 1.4.3 Derivatives of $f_1$

First, we must do some algebraic manipulation. We start by expanding equation [1.14] with equation [1.6]:

$$\frac{\partial \mathcal{L}}{\partial \mathcal{N}_j} = \mu_j^\circ + \hat{R}T \left[ \ln \left( \frac{p}{p_{\text{ref}}} \right) + \ln \left( \frac{\mathcal{N}_j}{\mathcal{N}} \right) \right] + \sum_{i=1}^{\text{NE}} \lambda_i a_{ij} + \lambda_q q_j = 0 \quad (1.20)$$

Dividing through by  $\hat{R}T$  and setting  $\pi = -\lambda/\hat{R}T$  gives:

$$f_1 = \frac{\mu_j^\circ}{\hat{R}T} + \ln \frac{p}{p_{\text{ref}}} + \ln \mathcal{N}_j - \ln \mathcal{N} - \sum_{i=1}^{\text{NE}} \pi_i a_{i,j} - \pi_q q_j = 0 \quad (1.21)$$

We now need to take the partial derivatives of our functions wrt our non-linear variables as well as each  $\pi$ , and then multiple it by the correction variables. You can easily take derivatives wrt  $\ln x$  by converting from  $x$  to  $\exp(\ln x)$ .

$$\frac{\partial f_1}{\partial [\ln \mathcal{N}_j]} \Delta \ln \mathcal{N}_j = \Delta \ln \mathcal{N}_j \quad (1.22)$$

$$\frac{\partial f_1}{\partial [\ln \mathcal{N}]} \Delta \ln \mathcal{N} = -\Delta \ln \mathcal{N} \quad (1.23)$$

$$\frac{\partial f_1}{\partial [\ln T]} \Delta \ln T = -\frac{H_j^\circ}{\hat{R}T} \Delta \ln T \quad (1.24)$$

$$\frac{\partial f_1}{\partial [\pi_i]} \pi_i = -a_{ij} \pi_i \quad (1.25)$$

$$\frac{\partial f_1}{\partial [\pi_q]} \pi_q = -q_j \pi_q \quad (1.26)$$

Combining these into the form of equation [1.4] gives

$$\Delta \ln \mathcal{N}_j - \Delta \ln \mathcal{N} - \sum_{i=1}^{\text{NE}} a_{ij} \pi_i - q_j \pi_q - \frac{H_j^\circ}{\hat{R}T} \Delta \ln T = \mu_j + \sum_{i=1}^{\text{NE}} \pi_i a_{ij} + \pi_q q_j \quad (1.27)$$

From before, the argument is made that  $\pi_i = 0$  at the start of every iteration, so this equation becomes:

$$\boxed{\Delta \ln \mathcal{N}_j - \Delta \ln \mathcal{N} - \sum_{i=1}^{\text{NE}} a_{ij} \pi_i - q_j \pi_q - \frac{H_j^\circ}{\hat{R}T} \Delta \ln T = -\frac{\mu_j}{\hat{R}T}}$$

There will be one of these equation for each species.

#### 1.4.4 Derivatives of $f_2$

$$\frac{\partial f_2}{\partial [\ln \mathcal{N}_j]} \Delta \ln \mathcal{N}_j = a_{ij} \mathcal{N}_j \Delta \ln \mathcal{N}_j \quad (1.28)$$

$$\frac{\partial f_2}{\partial [\ln \mathcal{N}]} \Delta \ln \mathcal{N} = \frac{\partial f_2}{\partial [\pi_i]} \pi_i = \frac{\partial f_2}{\partial [\ln T]} = \frac{\partial f_2}{\partial [\pi_q]} \pi_q = 0 \quad (1.29)$$

Which gives:

$$\boxed{\sum_{j=1}^{\text{NS}} a_{ij} \mathcal{N}_j \Delta \ln \mathcal{N}_j = b_i^\circ - \sum_{j=1}^{\text{NS}} a_{ij} \mathcal{N}_j}$$

#### 1.4.5 Derivatives of $f_3$

$$\frac{\partial f_3}{\partial [\ln \mathcal{N}_j]} \Delta \ln \mathcal{N}_j = q_j \mathcal{N}_j \Delta \ln \mathcal{N}_j \quad (1.30)$$

$$\frac{\partial f_3}{\partial [\ln \mathcal{N}]} \Delta \ln \mathcal{N} = \frac{\partial f_3}{\partial [\ln T]} = \frac{\partial f_3}{\partial [\pi_i]} \pi_i = \frac{\partial f_3}{\partial [\pi_q]} \pi_q = 0 \quad (1.31)$$

Giving:

$$\boxed{\sum_{j=1}^{\text{NS}} q_j \mathcal{N}_j \Delta \ln \mathcal{N}_j = - \sum_{j=1}^{\text{NS}} q_i \mathcal{N}_j}$$

#### 1.4.6 Derivatives of $f_4$

$$\frac{\partial f_4}{\partial [\ln \mathcal{N}_j]} \Delta \ln \mathcal{N}_j = \mathcal{N}_j \Delta \ln \mathcal{N}_j \quad (1.32)$$

$$\frac{\partial f_4}{\partial [\ln \mathcal{N}]} \Delta \ln \mathcal{N} = -\mathcal{N} \Delta \ln \mathcal{N} \quad (1.33)$$

$$\frac{\partial f_4}{\partial [\ln T]} = \frac{\partial f_4}{\partial [\pi_i]} \pi_i = \frac{\partial f_4}{\partial [\pi_q]} \pi_q = 0 \quad (1.34)$$

$$\boxed{\sum_{j=1}^{\text{NS}} \mathcal{N}_j \Delta \ln \mathcal{N}_j - \mathcal{N} \Delta \ln \mathcal{N} = \mathcal{N} - \sum_{j=1}^{\text{N}} \mathcal{N}_j} \quad (1.35)$$

### 1.4.7 Derivatives of $f_5$

$$\frac{\partial f_5}{\partial [\ln \mathcal{N}_j]} \Delta \ln \mathcal{N}_j = \mathcal{N}_j \frac{H_j^\circ}{\hat{R}T} \Delta \ln \mathcal{N}_j \quad (1.36)$$

$$\frac{\partial f_5}{\partial [\ln T]} = \sum_{j=1}^{NS} \mathcal{N}_j \frac{c_{p,j}^\circ}{\hat{R}} \Delta \ln T \quad (1.37)$$

$$\frac{\partial f_5}{\partial [\ln \mathcal{N}]} \Delta \ln \mathcal{N} = \frac{\partial f_5}{\partial [\pi_i]} \pi_i = \frac{\partial f_5}{\partial [\pi_q]} \pi_q = 0 \quad (1.38)$$

Giving:

$$\boxed{\sum_{j=1}^{NS} \mathcal{N}_j \frac{H_j^\circ}{\hat{R}T} \Delta \ln \mathcal{N}_j + \sum_{j=1}^{NS} \mathcal{N}_j \frac{c_{p,j}^\circ}{\hat{R}} \Delta \ln T = \frac{h' - h'_0}{\hat{R}T}}$$

### 1.4.8 Derivatives of $f_6$

$$\frac{\partial f_6}{\partial [\ln \mathcal{N}_j]} \Delta \ln \mathcal{N}_j = \mathcal{N}_j \frac{S_j}{\hat{R}} \Delta \ln \mathcal{N}_j \quad (1.39)$$

$$\frac{\partial f_6}{\partial [\ln T]} = \sum_{j=1}^{NS} \mathcal{N}_j \frac{c_{p,j}^\circ}{\hat{R}} \Delta \ln T \quad (1.40)$$

$$\frac{\partial f_6}{\partial [\ln \mathcal{N}]} \Delta \ln \mathcal{N} = \frac{\partial f_6}{\partial [\pi_i]} \pi_i = \frac{\partial f_6}{\partial [\pi_q]} \pi_q = 0 \quad (1.41)$$

Giving

$$\boxed{\sum_{j=1}^{NS} \mathcal{N}_j \frac{S_j}{\hat{R}} \Delta \ln \mathcal{N}_j + \sum_{j=1}^{NS} \mathcal{N}_j \frac{c_{p,j}^\circ}{\hat{R}} \Delta \ln T = \frac{s'_0 - s'}{\hat{R}} + \mathcal{N} - \sum_{j=1}^{NS} \mathcal{N}_j}$$

All six of the equations are now listed for convenience:

$$\Delta \ln \mathcal{N}_j - \Delta \ln \mathcal{N} - \sum_{i=1}^{NE} a_{ij} \pi_i - q_j \pi_q - \frac{H_j^\circ}{\hat{R}T} \Delta \ln T = -\frac{\mu_j}{\hat{R}T} \quad (1.42)$$

$$\sum_{j=1}^{NS} \mathcal{N}_j \Delta \ln \mathcal{N}_j - \mathcal{N} \Delta \ln \mathcal{N} = \mathcal{N} - \sum_{j=1}^{NS} \mathcal{N}_j \quad (1.43)$$

$$\sum_{j=1}^{NS} a_{ij} \mathcal{N}_j \Delta \ln \mathcal{N}_j = b_i^\circ - \sum_{j=1}^{NS} a_{ij} \mathcal{N}_j \quad (1.44)$$

$$\sum_{j=1}^{NS} q_j \mathcal{N}_j \Delta \ln \mathcal{N}_j = - \sum_{j=1}^{NS} q_j \mathcal{N}_j \quad (1.45)$$

$$\sum_{j=1}^{NS} \mathcal{N}_j \frac{H_j^\circ}{\hat{R}T} \Delta \ln \mathcal{N}_j + \sum_{j=1}^{NS} \mathcal{N}_j \frac{c_{p,j}^\circ}{\hat{R}} \Delta \ln T = \frac{h_0' - h'}{\hat{R}T} \quad (1.46)$$

$$\sum_{j=1}^{NS} \mathcal{N}_j \frac{S_j}{\hat{R}} \Delta \ln \mathcal{N}_j + \sum_{j=1}^{NS} \mathcal{N}_j \frac{c_{p,j}^\circ}{\hat{R}} \Delta \ln T = \frac{s_0' - s'}{\hat{R}} + \mathcal{N} - \sum_{j=1}^{NS} \mathcal{N}_j \quad (1.47)$$

#### 1.4.9 Reduced Gibbs Equations

A shortcut can be made to make this system of equations smaller. This is achieved by solving equation [1.42] for  $\Delta \ln \mathcal{N}_j$  and pluggin it into equations [1.43] - [1.47]

$$\Delta \ln \mathcal{N}_j = \Delta \ln \mathcal{N} + \sum_{i=1}^{NE} a_{ij} \pi_i - \frac{\mu_j}{\hat{R}T} + q_j \pi_q + \frac{H_j^\circ}{\hat{R}T} \Delta \ln T \quad (1.48)$$

Substitution into equation [1.44] - [1.47] yields:

$$\begin{aligned} & \sum_{i=1}^{NE} \left[ \sum_{j=1}^{NS} a_{kj} a_{ij} \mathcal{N}_j \right] \pi_i + \left[ \sum_{j=1}^{NS} a_{kj} \mathcal{N}_j \right] \Delta \ln \mathcal{N} + \left[ \sum_{j=1}^{NS} a_{kj} q_j \mathcal{N}_j \right] \pi_q + \left[ \sum_{j=1}^{NS} a_{kj} \mathcal{N}_j \frac{H_j^\circ}{\hat{R}T} \right] \Delta \ln T \\ &= b_k^\circ - \sum_{j=1}^{NS} a_{kj} \mathcal{N}_j + \sum_{j=1}^{NS} a_{kj} \mathcal{N}_j \frac{\mu_j}{\hat{R}T} \quad (1.49) \end{aligned}$$

$$\begin{aligned}
& \sum_{i=1}^{\text{NE}} \left[ \sum_{j=1}^{\text{NS}} a_{ij} \mathcal{N}_j \right] \pi_i + \left[ \sum_{j=1}^{\text{NS}} \mathcal{N}_j - \mathcal{N} \right] \Delta \ln \mathcal{N} + \left[ \sum_{j=1}^{\text{NS}} q_j \mathcal{N}_j \right] \pi_q + \left[ \sum_{j=1}^{\text{NS}} \mathcal{N}_j \frac{H_j^\circ}{\hat{R}T} \right] \Delta \ln T \\
& = \mathcal{N} - \sum_{j=1}^{\text{NS}} \mathcal{N}_j + \sum_{j=1}^{\text{NS}} \mathcal{N}_j \frac{\mu_j}{\hat{R}T} \quad (1.50)
\end{aligned}$$

$$\begin{aligned}
& \sum_{i=1}^{\text{NE}} \left[ \sum_{j=1}^{\text{NS}} a_{ij} q_j \mathcal{N}_j \right] \pi_i + \left[ \sum_{j=1}^{\text{NS}} q_j \mathcal{N}_j \right] \Delta \ln \mathcal{N} + \left[ \sum_{j=1}^{\text{NS}} q_j^2 \mathcal{N}_j \right] \pi_q + \left[ \sum_{j=1}^{\text{NS}} q_j \mathcal{N}_j \frac{H_j^\circ}{\hat{R}T} \right] \Delta \ln T \\
& = \sum_{j=1}^{\text{NS}} q_j \mathcal{N}_j \frac{\mu_j}{\hat{R}T} - \sum_{j=1}^{\text{NS}} q_j \mathcal{N}_j \quad (1.51)
\end{aligned}$$

$$\begin{aligned}
& \sum_{i=1}^{\text{NE}} \left[ \sum_{j=1}^{\text{NS}} a_{ij} \mathcal{N}_j \frac{H_j^\circ}{\hat{R}T} \right] \pi_i + \left[ \sum_{j=1}^{\text{NS}} \mathcal{N}_j \frac{H_j^\circ}{\hat{R}T} \right] \Delta \ln \mathcal{N} + \left[ \sum_{j=1}^{\text{NS}} q_j \mathcal{N}_j \frac{H_j^\circ}{\hat{R}T} \right] \pi_q \\
& + \left[ \sum_{j=1}^{\text{NS}} \mathcal{N}_j \left( \frac{H_j^\circ}{\hat{R}T} \right)^2 + \sum_{j=1}^{\text{NS}} \mathcal{N}_j \frac{c_{p,j}^\circ}{\hat{R}} \right] \Delta \ln T = \frac{h'_0 - h'}{\hat{R}T} + \sum_{j=1}^{\text{NS}} \mathcal{N}_j \frac{H_j^\circ}{\hat{R}T} \frac{\mu_j}{\hat{R}T} \quad (1.52)
\end{aligned}$$

$$\begin{aligned}
& \sum_{i=1}^{\text{NE}} \left[ \sum_{j=1}^{\text{NS}} a_{ij} \mathcal{N}_j \frac{S_j}{\hat{R}} \right] \pi_i + \left[ \sum_{j=1}^{\text{NS}} \mathcal{N}_j \frac{S_j}{\hat{R}} \right] \Delta \ln \mathcal{N} + \left[ \sum_{j=1}^{\text{NS}} q_j \mathcal{N}_j \frac{S_j}{\hat{R}} \right] \pi_q \\
& + \left[ \sum_{j=1}^{\text{NS}} \mathcal{N}_j \frac{S_j}{\hat{R}} \frac{H_j^\circ}{\hat{R}T} + \sum_{j=1}^{\text{NS}} \mathcal{N}_j \frac{c_{p,j}^\circ}{\hat{R}} \right] \Delta \ln T = \frac{s'_0 - s'}{\hat{R}} + \mathcal{N} - \sum_{j=1}^{\text{NS}} \mathcal{N}_j + \sum_{j=1}^{\text{NS}} \mathcal{N}_j \frac{S_j}{\hat{R}} \frac{\mu_j}{\hat{R}T} \quad (1.53)
\end{aligned}$$

There are NE number of equation [1.50], and only one of all others. Once this system has been solved, you recover  $\mathcal{N}_j$  with equation [1.48].

The equations are colored according to when they are needed. **Red coloring** denotes terms that arise from the elemental constraint condition. These are included no matter what. **Blue** denotes terms that are added when you include the charge constraint. **Violet** terms are added for both HP and SP constraints.

**orange** and **teal** are the rows added for enthalpy or entropy constraints, respectively.

## 1.5 Helmholtz Minimization

The Helmholtz energy is defined as:

$$F = G - pV \quad (1.54)$$

Where  $G$  is the Gibbs free energy,  $p$  is the pressure, and  $V$  is the volume.

Substitution of  $G$  yields:

$$F = \sum_{j=1}^{NS} \mu_j \mathcal{N}_j - pV \quad (1.55)$$

Where the chemical potential is redefined as:

$$\mu_j = \mu_j^\circ + \hat{R}T \ln \left( \frac{\mathcal{N}_j R' T}{V} \right) \quad (1.56)$$

and  $R' = \hat{R} \cdot 10^{-5}$  is the same as dividing  $\hat{R}$  by the reference pressure of 1 bar.

### 1.5.1 Constraints

The constraints available for the Helmholtz energy are:

- Hold temperature (T) and volume (V) constant.
- Hold internal energy (U) and volume (V) constant.
- Hold entropy (S) and volume (V) constant.

For (TV) minimization, only the elemental and charge (if present) constraints are necessary. For (UV) minimization, the constraint is defined by:

$$u' - u'_0 = 0 \quad (1.57)$$

Where  $u'_0$  is the specified internal energy, and  $u'$  is solved for during the minimization process as:

$$u' = \sum_{j=1}^{NS} \mathcal{N}_j U_j^\circ = u'_0 \quad (1.58)$$

As stated before, special care needs to be taken to include the enthalpies of formation and reference enthalpies if applicable. For (SV) minimization, the constraint is defined by:

$$s' - s'_0 = 0 \quad (1.59)$$

Where  $s'_0$  is the specified entropy and  $s'$  is solved for during the minimization process as:

$$s' = \sum_{j=1}^{NS} \mathcal{N}_j S_j \quad (1.60)$$

where:

$$S_j = S_j^\circ - \hat{R} \ln \mathcal{N}_j - \hat{R} \ln \left( \frac{R'T}{V} \right) \quad (1.61)$$

The Lagrangian for the Helmholtz energy minimization process is then defined as:

$$\mathcal{L} = \sum_{j=1}^{NS} \mu_j \mathcal{N}_j - pV + \sum_{i=1}^{NE} \lambda_i \left[ \sum_{j=1}^{NS} a_{ij} \mathcal{N}_j - b_i^\circ \right] + \lambda_q \left[ \sum_{j=1}^{NS} q_j \mathcal{N}_j \right] \quad (1.62)$$

### 1.5.2 Derivatives

We first find our functions  $f$  that are equal to 0 by taking the derivatives wrt  $\mathcal{N}_j$ ,  $\lambda_i$ , and  $\lambda_q$ :

$$f_1 = \frac{\partial \mathcal{L}}{\partial \mathcal{N}_j} = \mu_j + \sum_{i=1}^{NE} \lambda_i a_{ij} + \lambda_q q_j = 0 \quad (1.63)$$

$$f_2 = \frac{\partial \mathcal{L}}{\partial \lambda_i} = \sum_{j=1}^{NS} a_{ij} \mathcal{N}_j - b_i^\circ = 0 \quad (1.64)$$

$$f_3 = \frac{\partial \mathcal{L}}{\partial \lambda_q} = \sum_{j=1}^{NS} q_j \mathcal{N}_j - 0 = 0 \quad (1.65)$$

We now form the second derivatives that define our Jacobian in the Newton step. We expand the chemical potential term using equation 1.56 and non-dimensionalize the equation by  $\hat{R}T$ . We also set  $\pi = -\lambda/\hat{R}T$ .

### 1.5.3 Derivatives for $f_1$

The derivatives for  $f_1$  are then:

$$\frac{\partial f_1}{\partial [\ln \mathcal{N}_j]} \Delta \ln \mathcal{N}_j = \Delta \ln \mathcal{N}_j \quad (1.66)$$

$$\frac{\partial f_1}{\partial [\ln T]} \Delta \ln T = -\frac{U_j^\circ}{\hat{R}T} \ln T \quad (1.67)$$

$$\frac{\partial f_1}{\partial \pi_i} \pi_i = -a_{ij} \pi_i \quad (1.68)$$

$$\frac{\partial f_1}{\partial \pi_q} \pi_q = -q_j \pi_q \quad (1.69)$$

The derivation for equation [1.67] is carried out in more depth in section [2.2]. Therefore, our final result is:

$$\Delta \ln \mathcal{N}_j - \sum_{i=1}^{\text{NE}} a_{ij} \pi_i - q_j \pi_q - \frac{U_j^\circ}{\hat{R}T} \Delta \ln T = -\frac{\mu_j}{\hat{R}T}$$

There will be NS number of these equations in our system.

### 1.5.4 Derivatives for $f_2$

Next, we take the derivatives of  $f_2$ :

$$\frac{\partial f_2}{\partial [\ln \mathcal{N}_j]} \Delta \ln \mathcal{N}_j = \mathcal{N}_j a_{i,j} \Delta \ln \mathcal{N}_j \quad (1.70)$$

$$\frac{\partial f_2}{\partial [\ln T]} \Delta \ln T = \frac{\partial f_2}{\partial \pi_i} \pi_i = \frac{\partial f_2}{\partial \pi_q} \pi_q = 0 \quad (1.71)$$

Which gives:

$$\sum_{j=1}^{\text{NS}} \mathcal{N}_j a_{ij} \Delta \ln \mathcal{N}_j = b_i^\circ - \sum_{j=1}^{\text{NS}} a_{ij} \mathcal{N}_j$$

There will be NE of these equations.

### 1.5.5 Derivatives for $f_3$

For  $f_3$  we get:

$$\frac{\partial f_3}{\partial [\ln \mathcal{N}_j]} \Delta \ln \mathcal{N}_j = \mathcal{N}_j q_j \Delta \ln \mathcal{N}_j \quad (1.72)$$

$$\frac{\partial f_3}{\partial [\ln T]} \Delta \ln T = \frac{\partial f_3}{\partial \pi_i} \pi_i = \frac{\partial f_3}{\partial \pi_q} \pi_q = 0 \quad (1.73)$$

Which gives only one equation:

$$\boxed{\sum_{j=1}^{\text{NS}} \mathcal{N}_j q_j \Delta \ln \mathcal{N}_j = - \sum_{j=1}^{\text{NS}} q_j \mathcal{N}_j}$$

### 1.5.6 Derivatives for $f_4$

For internal energy, we have:

$$f_4(\ln \mathcal{N}_j, \ln T) = \sum_{j=1}^{\text{NS}} \mathcal{N}_j U_j^\circ - u'_0 = u' - u'_0 \quad (1.74)$$

Using identity [2.1] helps with the temperature derivative. The derivatives are then:

$$\frac{\partial f_4}{\partial [\ln \mathcal{N}_j]} \Delta \ln \mathcal{N}_j = \frac{\mathcal{N}_j U_j^\circ}{\hat{R}T} \Delta \ln \mathcal{N}_j \quad (1.75)$$

$$\frac{\partial f_4}{\partial [\ln T]} \Delta \ln T = \sum_{j=1}^{\text{NS}} [T \mathcal{N}_j c_{v,j}^\circ] \Delta \ln T \quad (1.76)$$

$$\frac{\partial f_4}{\partial \pi_i} \pi_i = \frac{\partial f_4}{\partial \pi_q} \pi_q = 0 \quad (1.77)$$

After non-dimensionalizing, we are left with:

$$\boxed{\sum_{j=1}^{\text{NS}} \frac{\mathcal{N}_j U_j^\circ}{\hat{R}T} \Delta \ln \mathcal{N}_j + \sum_{j=1}^{\text{NS}} \frac{\mathcal{N}_j c_{v,j}^\circ}{R} \Delta \ln T = \frac{u'_0 - u'}{\hat{R}T}}$$

### 1.5.7 Derivatives for $f_5$

For the entropy constraint we have:

$$f_5(\ln \mathcal{N}_j, \ln T) = \sum_{j=1}^{\text{NS}} \mathcal{N}_j S_j - s'_0 = s' - s'_0 \quad (1.78)$$

Again, identity [2.1] assists us. The derivates are:

$$\frac{\partial f_5}{\partial [\ln \mathcal{N}_j]} \Delta \ln \mathcal{N}_j = \mathcal{N}_j [S_j - \hat{R}] \Delta \ln \mathcal{N}_j \quad (1.79)$$

$$\frac{\partial f_5}{\partial [\ln T]} \Delta \ln T = \mathcal{N}_j c_{v,j}^\circ \Delta \ln T \quad (1.80)$$

$$\frac{\partial f_5}{\partial \pi_i} \pi_i = \frac{\partial f_5}{\partial \pi_q} \pi_q = 0 \quad (1.81)$$

Which, after non-dimensionalizing gives the singular equation:

$$\boxed{\sum_{j=1}^{\text{NS}} \mathcal{N}_j \left[ \frac{S_j}{\hat{R}} - 1 \right] \Delta \ln \mathcal{N}_j + \mathcal{N}_j \frac{c_{v,j}^\circ}{\hat{R}} \Delta \ln T = \frac{s'_0 - s'}{\hat{R}}} \quad (1.82)$$

For convenience, they are all listed here together:

$$\Delta \ln \mathcal{N}_j - \sum_{i=1}^{\text{NE}} a_{ij} \pi_i - q_j \pi_q - \frac{U_j^\circ}{\hat{R}T} \Delta \ln T = -\frac{\mu_j}{\hat{R}T} \quad (1.83)$$

$$\sum_{j=1}^{\text{NS}} \mathcal{N}_j a_{ij} \Delta \ln \mathcal{N}_j = b_i^\circ - \sum_{j=1}^{\text{NS}} a_{ij} \mathcal{N}_j \quad (1.84)$$

$$\sum_{j=1}^{\text{NS}} \mathcal{N}_j q_j \Delta \ln \mathcal{N}_j = - \sum_{j=1}^{\text{NS}} q_j \mathcal{N}_j \quad (1.85)$$

$$\sum_{j=1}^{\text{NS}} \frac{\mathcal{N}_j U_j^\circ}{\hat{R}T} \Delta \ln \mathcal{N}_j + \sum_{j=1}^{\text{NS}} \frac{\mathcal{N}_j c_{v,j}^\circ}{R} \Delta \ln T = \frac{u'_0 - u'}{\hat{R}T} \quad (1.86)$$

$$\sum_{j=1}^{\text{NS}} \mathcal{N}_j \left[ \frac{S_j}{\hat{R}} - 1 \right] \Delta \ln \mathcal{N}_j + \mathcal{N}_j \frac{c_{v,j}^\circ}{\hat{R}} \Delta \ln T = \frac{s'_0 - s'}{\hat{R}} \quad (1.87)$$

### 1.5.8 Reduced Helmholtz Equations

A "simplification" can be made to the above system of equations by solving equations [1.83] for  $\Delta \ln \mathcal{N}_j$ :

$$\Delta \ln \mathcal{N}_j = \sum_{i=1}^{NE} a_{ij} \pi_i - \frac{\mu_j}{\hat{R}T} + q_j \pi_q + \frac{U_j^\circ}{\hat{R}T} \Delta \ln T \quad (1.88)$$

By substituting this into equations [1.84] - [1.87], we can shrink the system of equations dramatically. The resulting set of equations is:

$$\sum_{i=1}^{NE} \left[ \sum_{j=1}^{NS} a_{kj} a_{ij} \mathcal{N}_j \right] \pi_i + \left[ \sum_{j=1}^{NS} a_{kj} q_j \mathcal{N}_j \right] \pi_q + \left[ \sum_{j=1}^{NS} a_{kj} \mathcal{N}_j \frac{U_j^\circ}{\hat{R}T} \right] \Delta \ln T = b_k^\circ - \sum_{j=1}^{NS} a_{kj} \mathcal{N}_j + \sum_{j=1}^{NS} a_{kj} \mathcal{N}_j \frac{\mu_j}{\hat{R}T} \quad (1.89)$$

$$\sum_{i=1}^{NE} \left[ \sum_{j=1}^{NS} a_{ij} q_j \mathcal{N}_j \right] \pi_i + \left[ \sum_{j=1}^{NS} q_j^2 \mathcal{N}_j \right] \pi_q + \left[ \sum_{j=1}^{NS} q_j \mathcal{N}_j \frac{U_j^\circ}{\hat{R}T} \right] \Delta \ln T = \sum_{j=1}^{NS} q_j \mathcal{N}_j \frac{\mu_j}{\hat{R}T} - \sum_{j=1}^{NS} q_j \mathcal{N}_j \quad (1.90)$$

$$\begin{aligned} & \sum_{i=1}^{NE} \left[ \sum_{j=1}^{NS} a_{ij} \mathcal{N}_j \frac{U_j^\circ}{\hat{R}T} \right] \pi_i + \left[ \sum_{j=1}^{NS} q_j \mathcal{N}_j \frac{U_j^\circ}{\hat{R}T} \right] \pi_q + \left[ \sum_{j=1}^{NS} \mathcal{N}_j \left( \frac{U_j^\circ}{\hat{R}T} \right)^2 + \sum_{j=1}^{NS} \mathcal{N}_j \frac{c_{v,j}^\circ}{\hat{R}} \right] \Delta \ln T \\ &= \frac{u'_0 - u'}{\hat{R}T} + \sum_{j=1}^{NS} \mathcal{N}_j \frac{U_j^\circ}{\hat{R}T} \frac{\mu_j}{\hat{R}T} \end{aligned} \quad (1.91)$$

$$\begin{aligned} & \sum_{i=1}^{NE} \left[ \sum_{j=1}^{NS} a_{ij} \mathcal{N}_j \left( \frac{S_j}{\hat{R}} - 1 \right) \right] \pi_i + \left[ \sum_{j=1}^{NS} q_j \mathcal{N}_j \left( \frac{S_j}{\hat{R}} - 1 \right) \right] \pi_q \\ &+ \left[ \sum_{j=1}^{NS} \mathcal{N}_j \left( \frac{S_j}{\hat{R}} - 1 \right) \frac{U_j^\circ}{\hat{R}T} + \sum_{j=1}^{NS} \mathcal{N}_j \frac{c_{v,j}^\circ}{\hat{R}} \right] \Delta \ln T = \frac{s'_0 - s'}{\hat{R}} + \sum_{j=1}^{NS} \mathcal{N}_j \frac{\mu_j}{\hat{R}T} \left( \frac{S_j}{\hat{R}} - 1 \right) \end{aligned} \quad (1.92)$$

There will be NE number of equations 1.89, and one of 1.90 if charge is added, and one of 1.91/1.92 depending on which one you made need.

The equations are colored according to when they are needed. **Red coloring** denotes terms that arise from the elemental constraint condition. These are included no matter what. **Blue** denotes terms that are added when you include

the charge constraint. **Violet** terms are added for both UV and SV constraints. **orange** and **teal** are the rows added for either internal energy or entropy constraints, respectively.

## 1.6 Useful Relationships

$$\frac{c_v^\circ}{\hat{R}} = \frac{c_{p,j}^\circ}{\hat{R}} - 1 \quad (1.93)$$

$$\frac{\mu_j^\circ}{\hat{R}T} = \frac{H_j^\circ}{\hat{R}T} - \frac{S_j^\circ}{\hat{R}} \quad (1.94)$$

$$\frac{U_j^\circ}{\hat{R}T} = \frac{H_j^\circ}{\hat{R}T} - 1 \quad (1.95)$$

## 2 Standard State Chemical Potential Derivatives

Some important derivatives are derived here. First, we state a chain rule identity that is very useful:

$$\frac{\partial}{\partial T} = \frac{\partial[\ln(T)]}{\partial T} \frac{\partial}{\partial[\ln(T)]} \rightarrow \frac{\partial}{\partial[\ln(T)]} = T \frac{\partial}{\partial T} \quad (2.1)$$

Then, taking the derivative of  $\mu_j^0$  gives:

$$\frac{\partial}{\partial[\ln(T)]} \left( \frac{\mu_j^0}{RT} \right) = T \frac{\partial}{\partial T} \left( \frac{\mu_j^0}{RT} \right) \quad (2.2)$$

We expand  $\mu_j^0/RT$  as one of the following two equations depending on if we are using Gibbs or Helmholtz minimization:

$$\frac{\mu_j^0}{RT} = \frac{H_j^0}{RT} - \frac{S_j^0}{R} \quad (2.3)$$

$$\frac{\mu_j^0}{RT} = \frac{U_j^0}{RT} - \frac{S_j^0}{R} + 1 \quad (2.4)$$

### 2.1 Gibbs Derivative

First, we take the derivative of [2.3] for Gibbs minimization.

$$T \frac{\partial}{\partial T} \left( \frac{H_j^0}{RT} - \frac{S_j^0}{R} \right) = T \left( \frac{\partial_T(H_j^0) \cdot RT - H_j^0 \cdot R}{(RT)^2} - \frac{\partial_T(S_j^0)}{R} \right) \quad (2.5)$$

Here,  $\partial_T$  denotes  $(\partial/\partial T)$ . If:

$$c_{p,j}^0 = \frac{\partial H_j^0}{\partial T}, \text{ and } \frac{c_{p,j}^0}{T} = \frac{\partial S_j^0}{\partial T} \quad (2.6)$$

This reduces to:

$$T \left[ \frac{c_{p,j}^0}{RT} - \frac{H_j^0}{RT^2} - \frac{c_{p,j}^0}{RT} \right] = -\frac{H_j^0}{RT} \quad (2.7)$$

Therefore:

$$\boxed{\frac{\partial}{\partial[\ln(T)]} \left( \frac{\mu_j^0}{RT} \right) = -\frac{H_j^0}{RT}} \quad (2.8)$$

## 2.2 Helmholtz Derivative

Using equation [2.4] gives us:

$$T \frac{\partial}{\partial T} \left( \frac{U_j^0}{RT} + 1 - \frac{S_j^0}{R} \right) = T \left( \frac{\partial_T(U_j^0) \cdot RT - U_j^0 \cdot R}{(RT)^2} - \frac{\partial_T(S_j^0)}{R} \right) \quad (2.9)$$

Recalling  $\partial_T(S_j^0)$  from equation [2.6] and using:

$$c_{v,j}^0 = \frac{\partial U_j^0}{\partial T} \quad (2.10)$$

Equation [2.9] reduced to:

$$T \left[ \frac{c_{v,j}^0}{RT} - \frac{U_j^0}{RT^2} - \frac{c_{p,j}^0}{RT} \right] \quad (2.11)$$

Since:

$$\frac{c_{v,j}^0}{R} - \frac{c_{p,j}^0}{R} = -1 \quad (2.12)$$

The final derivative is:

$$\boxed{\frac{\partial}{\partial[\ln(T)]} \left( \frac{\mu_j^0}{RT} \right) = -\frac{U_j^0}{RT} - 1} \quad (2.13)$$

### 3 Quick code overview

Currently, this code is focused on Gibbs minimization holding TP constant, and Helmholtz minimization holding TV, and UV constant. Further development will include the other three auxiliary constraints. It also includes the charge constraint if ions are present in the mixture. The code comes with easily accessible functions for common air mixtures such as 5, 7, and 11 species air for applications in the aerospace industry. The ability to create mixtures from any user-specified mixture is available if necessary.

Access to all thermodynamic data is readily available through use of the object `mix`. By defining an object named 'gas' of type `mix`, the user can find all thermodynamic variables necessary after minimization by appending the correct suffix to 'gas', e.g., `gas.rho` contains mixture density, `gas.R` contains mixture gas constant, `gas.Y[j]` shows mass fraction of species  $j$ , etc. A function named `print_properties(gas)` will output all thermodynamic variables of the gas mixture to the terminal.

In the bin directory of this code, there are three programs ready to be run and/or edited after you build. The first is an example program (fittingly named `example.cpp`) that has everything defined for you to run it and see the minimization results. You can toy with options to understand how to use correct 'enum class' such as `ConstraintType::` (both defined below) and access results.

The second program is the command-line driven code that lets you choose options and set mixtures for the chemical equilibrium process. The program has a "help" command that will tell you what you need to run. An overview is given here as well. When you start the program, it will give you some options right off the bat. In order to run minimization, all fields need to be filled in that appear when you type "show".

The '-mode' option tells the program which mode it is running in. By

inputting ‘–mode= sweep’ you can change the program from a standalone minimization given a single value for minimization parameters, to a version that sweeps through an user-input min and max energy value and then plots the results to a user-named file. It is initialized by default to used single values and not sweep.

The ‘–constraint’ option tells the programs which minimization procedure to use. TP, TV, UV, CFD are the current ones available. It needs to be in caps when input as well. For example ‘–constraint=TP’.

The ‘–species’ option tells the program which species you want to include in the minimization process. You can list your own with comma separated variables as such: ‘–species= N2, O2, NO, N, O’. If you choose to build a species list this way, you will need to specify a couple more things like ‘–elements’ and ‘–Y’. If instead you use an already created mixture (available ones are air5, air7, air11, air13, mars8), you can do so with ‘–species= air11’. This will fill in the ‘–elements’ and ‘–Y’ flags for you as well. You can always double check what needs to be filled in with the ‘show’ command.

If you do need to specify elements and mass fractions, you can do so with ‘–elements= N, O’ and also ‘–Y= 0.7572, 0.2428’. Just make sure that the order that you input the mass fractions lines up with the elemental ordering.

When the configuration input has been filled in, typing ”run” in the command line will run the program for you. You will still need to follow command line prompts for specifying filenames, sweeping values, etc.

Finally, the third program is created for timing the speed of the minimization processes and is really more for developing than for getting values.

## 4 Code specifics

### 4.1 Overview

The code uses a class named `CESolver` to run the minimization process. Most of the functionality uses private-member functions which make use of class member variables so that no inputs to functions are required. Of course, any

function that needs to be accessed outside of the library is a public-member function. A separate file names `functions.h` contains two namespaces for Gibbs and Helmholtz energy minimization procedures, namely for computing the chemical potential and filling the Jacobian and RHS for the Newton solver, since they are done differently in each case.

In order to run the minimization, you need to first create a few objects. First, you need to create an instance of the `mix` type. This contains vectors of size NS of a type called `SpeciesInfo`. `SpeciesInfo` contains all the data that is read in from the NASA thermodynamic table for a single species. Therefore, `mix` contains a vector of this type to easily access. `mix` contains all mixture properties, and also the current solution for  $\mathcal{N}_j$ , and other vectors such as the stoichiometric coefficients. Everything that `SpeciesInfo` and `mix` contains can be found in the header file `thermoObjs.h`. There are only a couple of other `CESolver` member variables besides the main `mix` type, but they are mostly just sizes of the Jacobian and RHS vectors, sizes of species list, element list, etc.

You can instantiate a gas mixture using one of two functions.

`mix gas = common_mixture(GasType::air11)` is an example of the function call to create a `mix` using the enum class `GasType::`, which holds commonly used gas mixtures. Current common mixtures are:

- `GasType::AIR5` - contains  $N_2$ ,  $O_2$ ,  $NO$ ,  $N$ ,  $O$ .
- `GasType::AIR7` - contains  $N_2$ ,  $O_2$ ,  $NO$ ,  $N$ ,  $O$ ,  $NO^+$ ,  $e^-$ .
- `GasType::AIR11` - contains  $N_2$ ,  $O_2$ ,  $NO$ ,  $N$ ,  $O$ ,  $N_2^+$ ,  $O_2^+$ ,  $NO^+$ ,  $N^+$ ,  $O^+$ ,  $e^-$ .
- `GasType::AIR11_AR` - contains  $N_2$ ,  $O_2$ ,  $NO$ ,  $N$ ,  $O$ ,  $Ar$ ,  $Ar^+$ ,  $NO^+$ ,  $N^+$ ,  $O^+$ ,  $e^-$ .
- `GasType::AIR13` - contains  $N_2$ ,  $O_2$ ,  $NO$ ,  $N$ ,  $O$ ,  $Ar$ ,  $Ar^+$ ,  $N_2^+$ ,  $O_2^+$ ,  $NO^+$ ,  $N^+$ ,  $O^+$ ,  $e^-$ .

For creation of your own mixture, you must define three vectors. The first is a vector of strings that contains the names of the species in your mixture. The second, a vector of strings that contains the names of the elements in your mixture. The third is a vector of doubles that hold the mass fractions of your

elements (in the same ordering as the list of your element names). For example:

```
vector<string> species = {"N2", "O2", "NO", "N", "O"};
vector<string> elements = {"N", "O"};
vector<double> Y_initial = {0.7572, 0.2428}
```

You can then create your mix via:

```
mix gas = create_mixture(species, elements, Y_initial);
```

The next step is to use the enum class `ConstraintType`:: to define the constraint you want to use in the minimization process. Currently provided constraints are:

- `ConstraintType::TP` - Minimize Gibbs free-energy holding temperature and pressure constant.
- `ConstraintType::TV` - Minimize Helmholtz free-energy holding temperature and specific volume constant.
- `ConstraintType::UV` - Minimize Helmholtz free-energy holding internal energy and specific volume constant.
- `ConstraintType::CFD` - Minimize Helmholtz free-energy holding internal energy and density constant.

The constructor function automatically checks to see which constraint type pertains to which energy minimization method, so you do not have to specify if you are minimizing with Gibbs or Helmholtz energies. This can be done inside the `CESolver` class-constructor function call:

```
CESolver CE(gas, ConstraintType::TP)
```

Where `gas` is our previously created `mix`, and `ConstraintType::TP` means we want to minimize Gibbs free-free energy holding temperature and pressure constant. This command sets up our solver by letting the solver know which equations need to be used (are ions present, do we need to solve for temperature, which temperature minimization equation are we using). Once

the above is known, it creates a pointer to a function that uses the correct minimization function. Because of the pointer, you can just call

```
compute_equilibrium(1000.0, 101325.0)
```

The inputs are arranged in the order of the `ConstraintType::` that you put into the `CESolver` constructor. Since we used `TP`, the above will set  $T = 1000.0$  K, and  $P = 1013125.0$  Pa. This function is called for every constraint type you use except for `ConstraintType::CFD`, which will force you to call `CFD_equilibrium(e, rho)` instead of the universal `compute_equilibrium()` function.

## 4.2 Function details

### 4.2.1 Constructor function

The constructor function, `CESolver(mix, ConstraintType::)`, checks the `mix` object and constraint type to create the function pointer to the correct minimization function as well as create the sizes of the Newton-Raphson method solver. It does so through checking and changing certain booleans. For example, if the mixture has ionic species present, the `mix` boolean `HAS_IONS` is flagged as true. The `ConstraintType::` will set the `mix` boolean `NEEDS_T` to true or false depending on the constraint. Then, the number of equation that need to be solver for is `J_SIZE = NS + gas.HAS_IONS + gas.NEEDS_T`.

### 4.2.2 `compute_equilibriumXY()` functions

There are a number of these functions at the moment for each separate constraint. Unfortunately, each minimization function is a little different, so thought will need to be put into a method to combine everything into one function to reduce the amount of code duplication. This function takes

- 4.2.3   NASA\_fits()**
- 4.2.4   Namespace functions**
- 4.2.5   compute\_mixture\_properties()**
- 4.2.6   compute\_derivatives()**
- 4.2.7   compute\_derivativesCFD()**
- 4.2.8   check\_convergence()**
- 4.2.9   compute\_damping()**