author: Ocean Wong
(Hoi Yeung Wong)
supervisor: Chantal Nobs, Robin Smith
organization: Culham Centre for Fusion Energy
date: October 2019

Abstract

In this document, the method of particle spectrum unfolding using Tikhonov Regularization is presented. Kullback-Leibler divergence was chosen as the regularizing function. The results has been implemented as a class within the neutron spectrum unfolding suite.

The same framework of minimizing the loss function, linearization of loss function gradient landscape, and proving the uniqueness (or non-uniqueness) of the global minimum, can be re-used by other researches who wishes to create a different regularization program based on another (or multiple others) metric(s).

A general neutron spectrum unfolding algorithm by regularization using relative-entropy

1 Background

regularization algorithm uses a metric, which is a linear mixture of $(\chi^2)_{reactionrates}$ and the deviation of the solution spectrum from the *a priori* spectrum. The mixing ratio between these two quantities is controlled by the Tikhonov parameter τ, which more or less weight on the relative-entropy.

This method of particle spectra unfolding is applicable to any general unfolding problem, as nothing except the reaction rates, response matrix, and *a priori* is known. This is particularly useful for fusion neutron spectrum unfolding, where there are not many experimental data available, thus no reasonable parametrisation or other sophisticated method of simplifying the neutron spectrum can be used.

$$loss(\phi_{sol}) = (\chi^2)_{\text{reaction rates}}(\phi_{sol}) + \tau \cdot RelativeEntropy_{\phi_0}(\phi_{sol}) \tag{1}$$

where
ϕ_0 is the *a priori* spectrum,
ϕ_{sol} is the algorithm's output spectrum,
$Loss(\phi_{sol})$ is the value outputted by the loss function, sometimes called the loss function

Due to the underdetermination condition of the unfolding problem, there exist multiple (i.e. infinitely many) solutions which will minimize χ^2 (potentially down to zero). By also requiring it to deviate minimally from the *a priori* spectrum, a unique solution to the problem can be found instead.

1.1 Motivation

While some programs of particle spectra unfolding via regularization exist, it is not rigorously derived and do not allow the user to specify a regularization constant. MAXED in particular, forces (χ^2) = number of degree of freedoms [7]. This form of "regularization" leads to no insight to the user of how far off is the *a priori* spectra, and therefore if the spectrum is accurate or not. It also does not guarantee that

the solution is unique, since multiple (infinitely many) solutions with $\chi^2 = 1$ can be found[1]; while for this program, only finitely many (only one) of such minima will be found. (See Appendix B) By carrying out the following derivation, which does not rely on any pre-existing numerical solvers to obtain a solution (i.e. no bin-hopping or simulated annealing), a minimum loss value in equation 1 is guaranteed, without the chance of being trapped at sub-optimal stationary points. A program is created using the result of this derivation.

2 Justification of loss function (relative-entropy + (χ^2)) chosen

χ^2 was used to quantify the deviation of the unfolded solution neutron spectrum's would-have-been reaction rate from the measured reaction rates. This is a widely used metric for measuring the deviation between pairs of numbers that are otherwise uncorrelated, and is generally accepted as the standard metric used to measure the deviation between reaction rates [4][8].

Perhaps a more interesting question is why should we use relative-entropy as the regularizing function in the loss function.

The regularizing function, in the context of neutron spectrum unfolding, refers to the function used to compute the "deviation" of one spectrum from another (from the solution spectrum to the *a priori* spectrum).

When choosing a regularizing function, it is desirable to have the following features:

1. Can naturally obey the physical requirement of having no negative flux without applying additional external constraint onto the set of equations;
2. For the same amount of absolute deviation $\Delta\phi$, harsher penalty should be given to bin with lower *a priori* flux. Penalty contributed by the i-th bin¿ penalty contributed by the j^{th} bin if $|\phi_{sol,i} - \phi_{0,i}| = \Delta\phi = |\phi_{sol,j} - \phi_{0,j}|$

To satisfy both of these requirements,

1. The gradient of the loss function must approach $-\infty$ as the flux approaches 0,
2. The magnitude of the 2^{nd} order derivative of the loss function $\frac{\partial^2(\text{loss})}{\partial\phi^2_{sol,i}}$ must increase with decreasing *a priori* value $\phi_{0,i}$ [2]

Other metrics considered before writing this document includes:

- Fractional deviation (obeys requirement 2 but not 1)
- Mean squared deviation in log-space
- Absolute deviation in log-space

[1]MAXED deals with this problem of infintely-many-possible-solutions by opening up an extra degree of freedom, removing the $\sum\phi_i = 1$ constraint, seeking to minimize the deviation of this sum $\sum\phi_i$from 1 instead. But, as the author of this paper, I disagree with that approach, as it invalidates the premise of having f_i as a "normalized probability distribution".

[2]If the reader disagree with requirement 2 but still would like a method that fulfill requirement 1, then s/he can use a regularizing function with two Tikhonov parameters: Regularizing function = $\tau_1(\chi^2)_{\phi,\phi_0} + \tau_2 S(\phi)$; where $S(\phi)$ is the self-entropy of the probability distribution of the solution spectrum ϕ, and $(\chi^2)_{\phi,\phi_0}$ is the usual definition of χ^2 (sum of square of weighted deviations of flux in each bin, between the solution spectrum and the *a priori* spectrum.)

- Determinant of the Fisher Information Matrix [3]

Relative entropy satisifes both of the requirements above, with the additional benefit of being the only unique and non-self-contradictory metric that can be used to measure deviation between probability distributions (See Appendix A in [6]). Therefore it is chosen as the metric to be used in the regularizing function in this paper.

Serendipiteously, the loss function, when composed in this manner, has no local trapping suboptimal points (See appendix B), allowing to use of Lagrangian multipliers to find the point of minimum minimization, which is detailed in section 4.

There are other implementations of the regularization method in ways that does not require an *a priori*. Take minimum (self-)entropy for example, which is the algorithm used by [1] and [3]. This is only a special implementation of the principle of MaxEnt (Maximum (relative-)Entropy), where a naïve *a priori* is assumed, i.e. it is equivalent to using a flat *a priori* in MAXED.

It is also worth noting that the commonly used neutron spectrum unfolding code GRAVEL obeys both requirements, even though it is not program that uses a regularization code. It takes steps in the log-flux space (thus fulfilling requirement **??**)in order to minimize the χ^2 of the reaction rates; and the starting point in the log-flux space is the *a priori* (thus fulfilling requirement 2).

2.1 Definition of relative-entropy

Before proceeding any further, it is useful to know what relative-entropy is. It is also known as Kullback-Leibler divergence (D_{KL}); and has been misnamed as cross-entropy(H) in some papers[6] (See equation 5 for details). Both measures the deviation of one normalized probability distribution from another. These probability distribution can be continuous or discrete.

For example, let there be two discrete probability distributions Q and P , over x:

$$H(P,Q) = -\sum_x P(x)log(Q(x)) \tag{2}$$

$$S(P) = -\sum_x P(x)log(P(x)) \tag{3}$$

$$D_{KL}(P||Q) = \sum_x P(x)log(\frac{P(x)}{Q(x)}) \tag{4}$$

$$D_{KL}(P||Q) = H(P,Q) - S(P) \tag{5}$$

where $S(P)$ is the (self-)entropy of the distribution. (Intuitively, when analogized to drawing balls from a bag, $S(P)$ is how "surprising" the resulting ball's colour is on average.) When P is fixed (as is the *a priori*), then $S(P)$ is a constant, so $D_{KL}(P||Q)$ and $H(P,Q)$ only differ by a fixed, constant value for any P.

Note the non-commutativity of $D_{KL}(P||Q)$: if the role of P and Q in equation 4 were reversed, a different answer will be obtained; but it will still give a loss function landscape that only has one global minimum.

The choice of the base of the *log* in the equations above is arbitrary. (In the ball-drawing analogy this number is equal to the number of colours available.) In

[3]Fisher information of a variable vector (which the neutron spectrum is) should form a MATRIX, not a scalar, unlike the papers [2][5] mentioned.

the context of information theory and computer science this is usually chosen as 2, and given the unit "shannons"; but for physics application, and for the rest of this text, it will be chosen as base e, and given the unit "nats", where 1 shannon = 0.693 nats.

3 Assumption

Let's assume that we have a response matrix $\underline{\underline{\mathbf{R}}}$ where each element is known to sufficiently high precision. R_{ki} refers to the probability of a nuclide of type k being created upon the incidence of a neutron with energy within energy bin i.

3.1 Notations

In previous texts, the neutron flux is usually represented with $\boldsymbol{\phi}$, and the reaction rates with Z. In this text, however, the neutron flux needs to be normalized to obtain a probability distribution instead. The probability distribution of neutron will be indicated with $\boldsymbol{f}$,

$$\sum_i f_i = 1 \tag{6}$$

in line with the notations used in [6] derivation. Additionally, the reaction rate will not be use; instead the following quantity $\boldsymbol{N}$

$$N_k = \#\text{ of nuclide } k \text{ created per incident neutron} \tag{7}$$

$$= \frac{\text{total \# of nuclide } k \text{ created}}{\text{total n. yield}} = \frac{\text{reaction rate } k}{\text{n. per second}} = \frac{Z_k}{\text{n. per second}} \tag{8}$$

will be use (contrary to previous texts which may use N as the symbol or no. of nuclides in total).

To make the following derivation easier, we assume that the measurements of "# of nuclide k created per incident neutron" N_k are mutually independent, forming a vector $\boldsymbol{N}$ of with a covariance matrix which is purely diagonal,

$$\underline{\underline{\mathbf{S_N}}} = \begin{pmatrix} \sigma_1 & & & & \\ & \ddots & & & \\ & & \sigma_k & & \\ & & & \ddots & \\ & & & & \sigma_m \end{pmatrix} \tag{9}$$

However, if one wishes to account for the covariances between different measured N_k, they can repeat the following derivation with $(\chi^2) = (\boldsymbol{N'} - \boldsymbol{N}) \cdot \underline{\underline{\mathbf{S_N}}} (\boldsymbol{N'} - \boldsymbol{N})$ instead of $(\chi^2) = \sum_k (\frac{N'_k - N_k}{\sigma_k})^2$. One of the possible reasons to do so is to account for the covariance in uncertainty between N_k due to uncertainty of the total n. yield, which is the common denominator to all of these reaction rates. Even if Z_k $\{\forall\ 1 \leq k \leq m\}$ are all obtained via independent measurements, they need to be divided by the same neutron yield as shown in equation 8, introducing a large component of positive covariance between them.

Below are some of the notations which will be used:

number of reactions = m
so that daughter nuclide k has a range $1 \leq k \leq m$

number of energy groups = i
so that the i^{th} or j^{th} energy group $1 \leq i, j \leq n$

A priori probability distribution of neutron spectrum = $\boldsymbol{f^{DEF}}$

probability distribution of neutron spectrum = $\boldsymbol{f}$ (vectors are written in bold italics)

Measured # of nuclide created per incident neutron = $\boldsymbol{N}$ (without prime ($'$))

Response matrix = $\underline{\underline{\mathbf{R}}}$

element of a matrix is indicated with a pair of subscript indices, e.g. = R_{ij}

element of a vector is indicated with a subscript index, e.g. = f_i

Iteration of a vector/matrix is indicated with superscript, e.g. = $\boldsymbol{v}^g$

The definitions of some more specific terms will be made clear further down the derivation. See equation 33, 34, and 66 for the definition of $\boldsymbol{c}, \underline{\underline{\mathbf{M}}}$, and $\underline{\mathbf{J}}$.

4 Derivation

4.1 Formulation (Lagrangian Multiplier)

For a fixed value of τ the following m+1 constraints has to be followed:

$$N'_k = \sum_i^n R_{ki} f_i \qquad \text{for } 1 \leq k \leq m \tag{10}$$

$$\sum_i f_i = 1 \tag{11}$$

where N'_k is the hypothetical number of nuclide k generated per incident neutron given the hypothetical neutron spectrum spectrum $\boldsymbol{f}$.

And the following quantities has to be minimized:

$$\tau(D_{KL}(\boldsymbol{f}, \boldsymbol{f^{DEF}})) = \tau \sum_i f_i ln(\frac{f_i}{f_i^{DEF}}) \tag{12}$$

$$\chi^2(\boldsymbol{N'}, \boldsymbol{N}) = \sum_k (\frac{N'_k - N_k}{\sigma_k})^2 \tag{13}$$

Note that D_{KL} is always positive when constraint 11 is obeyed; and reaches a minimum when $\boldsymbol{f} = \boldsymbol{f^{DEF}}$.

Thus the Lagrangian can be constructed as

$$\mathcal{L}(\boldsymbol{f}, \boldsymbol{\lambda}, \boldsymbol{N'}, \mu) =$$
$$\tau \sum_i f_i ln(\frac{f_i}{f_i^{DEF}}) + \sum_k (\frac{N'_k - N_k}{\sigma_k})^2 + \sum_k \lambda_k [N'_k - \sum_i^n R_{ki} f_i] + \mu [\sum_i f_i - 1] \quad (14)$$

where $\boldsymbol{\lambda}$ is a list of Lagrangian multipliers λ_k, $1 \leq k \leq m$; and μ is a Lagrangian multiplier.

To obtain the minimum loss value where $loss = \chi^2 + \tau(D_{KL})$,

$$\nabla \mathcal{L} = 0 \quad (15)$$

i.e.

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial f_i} = 0 & \text{for } 1 \leq i \leq n \quad (16) \\ \frac{\partial \mathcal{L}}{\partial N'_k} = 0 & \text{for } 1 \leq k \leq m \quad (17) \\ \frac{\partial \mathcal{L}}{\partial \lambda_k} = 0 & \text{for } 1 \leq k \leq m \quad (18) \\ \frac{\partial \mathcal{L}}{\partial \mu} = 0 & (19) \end{cases}$$

This leads to the following n+2m+1 simultaneous equations:

$$\begin{cases} \tau[ln(\frac{f_i}{f_i^{DEF}}) + 1] + \mu + \sum_k^m \lambda_k R_{ki} = 0 & \text{for } 1 \leq i \leq n \quad (20) \\ 2\frac{N'_k - N_k}{\sigma_k^2} - \lambda_k = 0 & \text{for } 1 \leq k \leq m \quad (21) \\ \sum_j^n R_{kj} f_j = N'_k & \text{for } 1 \leq k \leq m \quad (22) \\ \sum_i f_i = 1 & (23) \end{cases}$$

Substituting the 3^{rd} line into the 2^{nd} line removes N'_k from this set of equations entirely; then substituting the 2^{nd} line into the 1^{st} removes the n Lagrangian multipliers λ_k from this set of equations entirely, leaving only a set of equations dependent on the variables f_i $(1 \leq i \leq n)$ and μ, as well as n+1 linearly-independent equations.

$$\begin{cases} \tau[ln(\frac{f_i}{f_i^{DEF}} + 1)] + \mu + \sum_k^m \left[2\frac{R_{ki}}{\sigma_k^2} \left((\sum_j^n R_{kj} f_j) - N_k \right) \right] = 0 & (24) \\ \sum_j^n f_j = 1 & (25) \end{cases}$$

Rearrange the top line (note: each i represent 1 equation, $1 \leq i \leq n$, so the top line entails n equations), such that all linear terms of f_i and μ stays on the LHS, while the constants and non-linear terms in f_i moves to the RHS.

$$\sum_k^m \left[2\frac{R_{ki}}{\sigma_k^2}(\sum_j^n R_{kj} f_j) \right] - \sum_k^m (2\frac{N_k R_{ki}}{\sigma_k^2}) \quad +\mu = -\tau[ln(\frac{f_i}{f_i^{DEF}}) + 1] \tag{26}$$

$$\sum_k^m \left[2\frac{R_{ki}}{\sigma_k^2}(\sum_j^n R_{kj} f_j) \right] \quad +\mu = -\tau[ln(\frac{f_i}{f_i^{DEF}}) + 1] \quad + \sum_k^m (2\frac{N_k R_{ki}}{\sigma_k^2}) \tag{27}$$

Transposing the order of summation in the first term yields

$$\sum_k^m 2\frac{R_{ki}}{\sigma_k^2} \sum_j^n (R_{kj} f_j) = \sum_k^m \sum_j^n [(2\frac{R_{ki}R_{kj}}{\sigma_k^2}) f_j] = \sum_j^n [(2\sum_k^m (\frac{R_{ki}R_{kj}}{\sigma_k^2})) f_j] \tag{28}$$

Now if we construct matrix $\underline{\underline{\mathbf{A}}}$ such that for $1 \leq i, j \leq n$

$$A_{ij} = 2\sum_k^m (\frac{R_{ki}R_{kj}}{\sigma_k^2}) \tag{29}$$

And further rearranging so only the constants are on the RHS, then the set of simultaneous equations can be expressed as:

$$\begin{cases} \sum_j^n A_{ij} f_j + \tau ln(f_i) + \mu = \tau[ln(f_i^{DEF}) - 1] + 2\sum_k^m (\frac{N_k R_{ki}}{\sigma_k^2}) & (30) \\ \sum_j^n f_j = 1 & (31) \end{cases}$$

4.2 How not to solve it

One may attempt to decouple the dependence of the i^{th} term from itself in equations 30, perhaps by moving all terms dependent on f_i in the i^{th} equation to the LHS, and all terms dependent on $f_{j \neq i}$ to the RHS, so that each line can be expressed as $F_{i/LHS}(f_i) = F_{i/RHS}(\boldsymbol{f_{j \neq i}}, constants)$, i.e. equating the result of non-linear function $F_{i/LHS}(f_i)$ to a the result of a (linear) function dependent on f_j.

This seems promising at first, until one tries to solve for f_1, and realising that s/he still knows nothing about the values on the RHS as f_j (for j=2,...,n) are unknown. f_2 is a function of f'_j where $j' = 1, 3, 4, ..., n$, etc. To evaluate f_2 will lead to circular reference towards f_1, leading to infinite recursion.

It is also tempting to give up at this stage and allow a generic minimization algorithm (e.g. L-BFGS) to find the solution of these equations. However, due to the logarithmic term present, most algorithm will inevitably overshoot into the $f_i < 0$ region when exploring the solution space, leading to failure.

4.3 Linearization

Instead, a much more elegant technique is stated as follows.

In general, in a set of fully-determined simultaneous equations, the variables required can be stacked into a column vector $\boldsymbol{v}$, and the constant terms can be moved to the RHS to be stacked into a column vector $\boldsymbol{c}$. Assuming that only linear operations on $\boldsymbol{v}$ is present in the system, then these simultaneous equations can be

expressed in linear algebra notations as $\underline{\underline{\mathbf{M}}}\boldsymbol{v} = \boldsymbol{c}$, where $\underline{\underline{\mathbf{M}}}$ is a non-singular square matrix, so $\boldsymbol{v} = \underline{\underline{\mathbf{M}}}^{-1}\boldsymbol{c}$.

For the regularization problem described by equation 30 and 31, the set of simultaneous equations is fully determined, but not expressible as a linear operator $\underline{\underline{\mathbf{M}}}$ operating on a vector, as it involves the non-linear operator ln.

One way of getting around this is via Taylor expansion. By incorperating the first term of the Taylor expansion into the LHS of equation 30 and ignoring all higher order terms, an approximate solution can be obtained.

This process can be repeatedly applied onto the result obtained to iteratively improve the accuracy of this approximation until it is much higher than the precision of the floating point.

To more succinctly express the set of equations above (30 and 31), we will relabel them into a matrix multiplication equation, $\underline{\underline{\mathbf{M}}}\boldsymbol{v} = \boldsymbol{c}$.

The most obvious next step is to turn the LHS of 30 and 31 $\underline{\underline{\mathbf{M}}}\boldsymbol{v}$ where $\underline{\underline{\mathbf{M}}}$ has size (n+1)×(n+1), variable vector $\boldsymbol{v}$ has length (n+1), and the RHS into a constant vector $\boldsymbol{c}$ has length (n+1). However, we an alternative, which is to eliminates the variable μ from the set of equation by subtracting the $(i+1)^{th}$ line from the i^{th} line. Note that μ has no immediately obvious physical meaning (it represents the steepness of the function in the direction normal to the constraint hyperplane), so it is not of interest to us. Then the set of n simultaneous equations in (30) can be reduced to the following $n-1$ equations, where $1 \leq i \leq n-1$:

$$\sum_{j}^{n}[A_{ij} - A_{(i+1)j}]f_j + \tau[ln(f_i) - ln(f_{i+1})] \quad = \tau[ln(f_i^{DEF}) - ln(f_{i+1}^{DEF})] + 2\sum_{k}^{m}[\frac{N_k(R_{ki} - R_{k(i+1)})}{\sigma_k^2}] \tag{32}$$

Such that equation 31 and 32 together forms n simultaneous equations in n variables, i.e. a fully determined system.

In attempt to condense them further into vector and matrix representations, we can use the following notations:

$$M_{ij} = \begin{cases} A_{ij} - A_{(i+1)j} & \text{for } 1 \leq i \leq n-1, 1 \leq j \leq n \\ 1 & \text{for } i = n, 1 \leq j \leq n \end{cases} \tag{33}$$

$$c_i = \begin{cases} \tau[ln(f_i^{DEF}) - ln(f_{i+1}^{DEF})] + 2\sum\limits_{k}^{m}(\frac{N_k(R_{ki} - R_{k(i+1)})}{\sigma_k^2}) & \text{for } 1 \leq i \leq n \\ 1 & \text{for } i = n \end{cases} \tag{34}$$

such that

$$\begin{cases} \sum\limits_{j}^{n} M_{ij}f_j + \tau[ln(f_i) - ln(f_{i+1})] & -c_i = 0 \quad \text{for} \quad 1 \leq i \leq n-1 \quad (35) \\ \sum\limits_{j}^{n} M_{ij}f_j & -c_i = 0 \quad \text{for} \quad i = n \quad (36) \end{cases}$$

Express the i^{th} line of LHS of equations 35 and 36 as a function F_i. Define $\boldsymbol{u}$ to be the test solution vector, while the $\boldsymbol{v}$ is the correct solution vector, such that

$$F_i(\boldsymbol{u}) = 0 \tag{37}$$

which can be written in vector form

$$\boldsymbol{F}(\boldsymbol{u}) = \mathbf{0} \tag{38}$$

if $\boldsymbol{u}$=solution$\equiv \boldsymbol{v}$.

For convenience the vector $\boldsymbol{\Delta}$ is defined:

$$\boldsymbol{\Delta} \equiv (\boldsymbol{v} - \boldsymbol{u}) \tag{39}$$

When searching for the solution, the true value of $\boldsymbol{v}$ is not known; but guess values of $\boldsymbol{u}$ can be used.

Consider the taylor expansion of $ln(v_i)$ at $ln(u_i)$,

$$ln(v_i) = ln(u_i) + \frac{v_i - u_i}{u_i} - O(u_i, \Delta_i) \tag{40}$$

$$\tau ln(v_i) = \frac{\tau}{u_i} v_i - \tau(-ln(u_i) + 1 + O(u_i, \Delta_i)) \tag{41}$$

where the higher order terms $O(u_i, \Delta_i)$

$$-O(u_i, \Delta_i) = -\frac{1}{2}\left(\frac{\Delta_i}{u_i}\right)^2 + \frac{1}{3}\left(\frac{\Delta_i}{u_i}\right)^3 - \frac{1}{4}\left(\frac{\Delta_i}{u_i}\right)^4 + \frac{1}{5}\left(\frac{\Delta_i}{u_i}\right)^5 \tag{42}$$

$$= -\sum_{l=2}^{\infty} \frac{1}{l}\left(\frac{\Delta_i}{u_i}\right)^l \tag{43}$$

Therefore Taylor expanding the logarithms in 35 gives:

$$\sum_j^n M_{ij} v_j + \frac{\tau}{u_i} v_i - \frac{\tau}{u_{i+1}} v_{i+1}$$
$$= c_i - \tau[ln(u_i) - ln(u_{i+1})] + \tau[O(u_i, \Delta_i) - O(u_{i+1}, \Delta_{i+1})] \tag{44}$$

expressing the equation 44 in terms of $\boldsymbol{\Delta}$ and $\boldsymbol{u}$ only,

$$\sum_j^n M_{ij} \Delta_j + \frac{\tau}{u_i} \Delta_i - \frac{\tau}{u_{i+1}} \Delta_{i+1} + \sum_j^n M_{ij} u_j + \tau[\frac{u_i}{u_i} - \frac{u_{i+1}}{u_{i+1}}]$$
$$= c_i - \tau[ln(u_i) - ln(u_{i+1})] + \tau[O(u_i, \Delta_i) - O(u_{i+1}, \Delta_{i+1})] \tag{45}$$

Manipulate equation 45, leaving only terms linearly dependent on $\boldsymbol{\Delta}$ on the left,

$$\sum_j^n M_{ij} \Delta_j + \frac{\tau}{u_i} \Delta_i - \frac{\tau}{u_{i+1}} \Delta_{i+1}$$
$$= c_i - \sum_j^n M_{ij} u_j - \tau[ln(u_i) - ln(u_{i+1})] + \tau[O(u_i, \Delta_i) - O(u_{i+1}, \Delta_{i+1})] \tag{46}$$

Meanwhile, the nth line of the system of simultaneous equations (equation 36) gives

$$\sum_j^n M_{ij} u_j - c_i = 0 \tag{47}$$

$$\sum_j^n M_{ij} v_j - c_i = 0 \tag{48}$$

$$\sum_j^n M_{ij} \delta_j = 0 \tag{49}$$

where $i = n$. Note $M_{nj} = 1 \; \forall 1 \leq j \leq n$ and $c_n = 1$ as defined in 33 and 34; so all of the M_{ij} and c_i in the three equations above are unity.

4.4 Convergence condition and underrelaxation

Rewrite equation 39 as

$$\boldsymbol{v} = \boldsymbol{u} + \boldsymbol{\Delta} \tag{50}$$

Where $\boldsymbol{\Delta}$ is the deviation of $\boldsymbol{u}$ from the true root $\boldsymbol{v}$ in the domain space. It is difficult to accurately calculate $\boldsymbol{\Delta}$. However, if we take the approximation of it up to the first order as shown in equation 44 to 46, and call it $\boldsymbol{\delta}$, we can rewrite equation 50 as

$$\boldsymbol{f}^g = \boldsymbol{f}^{g-1} + \alpha\boldsymbol{\delta}^{g-1} \tag{51}$$

for the gth iteration, where $\boldsymbol{\delta}^{g-1}$ is the vector proportional to the difference between the solution vector (i.e. solution flux) at the (g-1)-th and g-th iterations.

Note that we have introduced an underrelaxation factor $0 < \alpha \leq 1$ to slow down the convergence as that is necessary to ensure numerical stability. This is explained in appendix A.

Ignoring the higher order terms, equation 46 becomes:

$$\sum_j^n M_{ij}\delta_j^{g-1} + \frac{\tau}{u_i}\delta_i^{g-1} - \frac{\tau}{u_{i+1}}\delta_{i+1}^{g-1} = c_i - \sum_j^n M_{ij}f_j^{g-1} - \tau[ln(f_i^{g-1}) - ln(f_{i+1}^{g-1})] \tag{52}$$

for $1 \leq i \leq n-1$, and equation 36 becomes

$$\sum_j^n f_i^g = 1 \tag{53}$$

$$\sum_j^n \delta_i^{g-1} = 1 - \sum_i^n f_i^{g-1} \tag{54}$$

$$= 0 \tag{55}$$

for i = n, assuming $\boldsymbol{f}^{g-1}$ is normalized.

A collorary of the definition 51 above is that at each iteration g, $\boldsymbol{f}^g$ will always deviation from the correct solution $\boldsymbol{v}$ as follows:

Recalling that the LHS of equations 35 and 36 can be written as function F_i, and then vectorized into $\boldsymbol{F}(\boldsymbol{v}) = \boldsymbol{0}$ as seen in equation 38,

$$\boldsymbol{F}(\boldsymbol{f}^g) = \boldsymbol{F}(\boldsymbol{f}^{g-1} + \alpha\boldsymbol{\delta}^{g-1}) = \boldsymbol{O}^{g-1} \tag{56}$$

where the deviation from the origin in codomain space $\boldsymbol{O}^{g-1}$ is

$$(\boldsymbol{O}^{g-1})_i = \begin{cases} \sum_i^n M_{ij}(f_j^{g-1} + \alpha\delta_j^{g-1}) + \tau[ln(f_i^{g-1} + \alpha\delta_i^{g-1}) - ln(f_{i+1}^{g-1} + \alpha\delta_{i+1}^{g-1})] - c_i & \text{for } 1 \leq i \leq n-1 \\ \sum_i^n M_{ij}(f_j^{g-1} + \alpha\delta_j^{g-1}) - c_i & \text{for } i = n \end{cases} \tag{57}$$

Thus, for the answer to converge, we will require:

1. f_i^g to stay in the physically allowed bounds at the end, i.e. $0 \leq f_i^g \leq 1$ as $g \to \infty$
2. $\boldsymbol{O}^g$ to be bounded,
3. $\boldsymbol{O}^g$ to decrease with every iteration, converging to zero,

Condition 3 is more difficult to define, since there are n different dimensions to consider, each dimension can converge towards zero at different rates and manner (oscillatorily or monotonically); and if we use its p-norm, we can choose

$\lim_{g\to\infty} ||\boldsymbol{O}||_1 = \sum_i (f_i) \to 0$, $\lim_{g\to\infty} ||\boldsymbol{O}||_2 = \sqrt{\sum_i (f_i)^2} \to 0$,

or any other positive values of p $\lim_{g\to\infty} ||\boldsymbol{O}||_p \to 0$, as the measurement of "convergence".

In practice condition 3 will not be implemented as $\boldsymbol{O}^g < \boldsymbol{O}^{g-1}$ is not guaranteed when $\boldsymbol{f}$ approaches the solution and becomes

Condition 2 is easier to impose: as long as $f_i^g > 0$, $ln(f_i^g)$, and therefore, $\sum_{l=2}^{\infty} \frac{1}{l}(\frac{\delta_i^{g-1}}{f_i^{g-1}})^l$, will not diverge to infinity, staying bounded. This can be achieved by ensuring that

$$f_i^g = f_i^{g-1} + \alpha \delta_i^{g-1} > 0 \qquad \forall 1 \leq i \leq n \tag{58}$$

Given that $\boldsymbol{f}_i^{DEF}$ and $\boldsymbol{f}_i^g$ are both positive, condition 1 is obeyed naturally when the inequality 58 is obeyed, due to the normalization constraint imposed by equation 31.

The details of how to impose inequality 58 will be stated in appendix A.

When an appropriate choice of α has been made, the procedure of finding δ can be stated as follows: Starting from equation 52 and 54,

$$\sum_j^n M_{ij}\delta_j^{g-1} + \frac{\tau}{u_i}\delta_i^{g-1} - \frac{\tau}{u_{i+1}}\delta_{i+1}^{g-1} = c_i - \sum_j^n M_{ij} f_j^{g-1} - \tau[ln(f_i^{g-1}) - ln(f_{i+1}^{g-1})] \tag{59}$$

$$\text{for } 1 \leq i \leq n-1$$

$$\sum_j^n \delta_j^{g-1} = c_i - \sum_j^n M_{ij} f_j^{g-1} \tag{60}$$

$$\text{for } i = n \ .$$

To express equations 59 and 60 in vector form,

$$[\underline{\underline{\mathbf{M}}} + \tau \underline{\underline{\boldsymbol{\Lambda}}}^{g-1}]\boldsymbol{\delta}^{g-1} = \boldsymbol{c} - \underline{\underline{\mathbf{M}}}\boldsymbol{f}^{g-1} - \tau \underline{\underline{\mathbf{L}}}(\boldsymbol{f}^{g-1}) \tag{61}$$

where each of the symbol is defined as follows:

$\underline{\underline{\mathbf{M}}}$ and $\boldsymbol{c}$ are as defined in equation 33 and 34, τ (the Tikhonov parameter) is a scalar,

$$(\underline{\underline{\mathbf{L}}}(\boldsymbol{f}^{g-1}))_i = \begin{cases} ln(f_i^{g-1}) - ln(f_{i+1}^{g-1}) & \text{for } 1 \leq i \leq n-1 \\ 0 & \text{for } i = n \end{cases} \tag{62}$$

$$(\underline{\underline{\boldsymbol{\Lambda}}}^{g-1})_{ij} = \begin{cases} \frac{\delta_{ij}^*}{f_j^{g-1}} - \frac{\delta_{(i+1)j}^*}{f_j^{g-1}} & \text{for } 1 \leq i \leq n-1 \\ 0 & \text{for } i = n \end{cases} \tag{63}$$

where δ^* is the Kronecka delta; the * is used to prevent clashing symbol with the vector $\boldsymbol{\delta}^{g-1}$ previously used.

The first $n-1$ rows of $\underline{\underline{\mathbf{\Lambda}}}^{g-1}$ forms a double diagonal matrix (populated only at the the main diagonal and the diagonal line immediately above it).

$$\tau\underline{\underline{\mathbf{\Lambda}}}^{g-1} = \begin{pmatrix} \frac{\tau}{f_1^{g-1}} & \frac{-\tau}{f_2^{g-1}} & & & & \\ & \ddots & \ddots & & & \\ & & \frac{\tau}{f_i^{g-1}} & \frac{-\tau}{f_{i+1}^{g-1}} & & \\ & & & \ddots & \ddots & \\ & & & & \frac{\tau}{f_{n-1}^{g-1}} & \frac{-\tau}{f_n^{g-1}} \\ 0 & \cdots & 0 & \cdots & 0 & 0 \end{pmatrix} \tag{64}$$

Note that $\underline{\underline{\mathbf{L}}}(\boldsymbol{f})$ is actually a non-linear operation on $\boldsymbol{f}$, but still outputs a vector. It can be similarly expressed as

$$\tau\underline{\underline{\mathbf{L}}} = \begin{pmatrix} \tau ln & -\tau ln & & & & \\ & \ddots & \ddots & & & \\ & & \tau ln & -\tau ln & & \\ & & & \ddots & \ddots & \\ & & & & \tau ln & -\tau ln \\ 0 & \cdots & 0 & \cdots & 0 & 0 \end{pmatrix} \tag{65}$$

The operator on the LHS of equation 61 is the Jacobian matrix,

$$\underline{\underline{\mathbf{J}}}^{g-1} = \underline{\underline{\mathbf{M}}} + \tau\underline{\underline{\mathbf{\Lambda}}}^{g-1} \tag{66}$$

4.4.1 Summary

From 61,

$$\underline{\underline{\mathbf{J}}}^{g-1}\boldsymbol{\delta}^{g-1} = \boldsymbol{c} - \underline{\underline{\mathbf{M}}}\boldsymbol{f}^{g-1} - \tau\underline{\underline{\mathbf{L}}}(\boldsymbol{f}^{g-1}) \tag{67}$$

At each step, when f^{g-1} is known, the RHS is evaluated, and then the inverse of the Jacobian is left-multiplied onto both sides to obtain $\boldsymbol{\delta}^{g-1}$

$$\boldsymbol{\delta}^{g-1} = \left(\underline{\underline{\mathbf{J}}}^{g-1}\right)^{-1}\left[\boldsymbol{c} - \underline{\underline{\mathbf{M}}}\boldsymbol{f}^{g-1} - \tau\underline{\underline{\mathbf{L}}}(\boldsymbol{f}^{g-1})\right] \tag{68}$$

And then, given the underrelaxation factor α, the next iteration's $\boldsymbol{f}$ can be evaluated using equation 51

$$\boldsymbol{f}^g = \boldsymbol{f}^{g-1} + \alpha\boldsymbol{\delta}^{g-1} \tag{69}$$

For the first iteration $g = 1$, a very obvious choice of $\boldsymbol{f}^{g-1} = \boldsymbol{f}^0$ is the *a priori* , i.e. $\boldsymbol{f}^0 = \boldsymbol{f}^{DEF}$. When $\boldsymbol{f}^g$ settles onto an equilibrium, a check $\boldsymbol{F}(\boldsymbol{f}^g) = \mathbf{0} \pm$ tolerance is performed. If it passes this check, then the program can terminate, returning this value as the output.

5 Benchmarking/Verification

6 Renormalization

A normalization factor Y can be added into the derivation, scaling the list of calculated reaction rates up/down, minimizing the χ^2 at the same time:

$$Y = \text{n. yield} \tag{70}$$

$$YN_k = a_k \tag{71}$$

where $\boldsymbol{a}$ is the list of total reaction rates measurements at t=0, so that now χ^2 is defined as

$$\chi^2 = \sum_k^m \frac{(YN'_k - a_k)^2}{\sigma_k^2} \tag{72}$$

$$Y = \arg\min_Y(\chi^2) \tag{73}$$

This means this regularization neutron spectrum unfolding method can simultaneously output an estimate of the n. yield.

A similar procedure is carried out by MAXED at the start of each run; but this n. yield estimate is based on its comparison with the *a priori* 's reaction rates, no the final reaction rates. By imposing the constraint stated in equation 73 at each iteration, a more accurate n. yield can be obtained.

7 Using multiple a priori

8 Future direction for development

As briefly mentioned in section 3.1, the covariances between daughter nuclides resulting from the irradiation of the same foil, or from division by the n. yield diagnostics, can be accounted for by using a more sophisticated model.

Additionally, the error from the reaction rates measurements (and potentially from the cross-section values as well) can be propagated, analytically (instead of using Monte Carlo methods), through the unfolding process to obtain a covariance matrix of the flux values.

References

[1] J Cvachovec and F Cvachovec. Maximum likelihood estimation of a neutron spectrum and associated uncertainties. Advances in Military Technology, 1(2):67–79, 2008.

[2] Fatma Zohra Dehimi, Abdeslam Seghour, and Saddik El Hak Abaidia. Unfolding of neutron energy spectra with fisher regularisation. IEEE Transactions on Nuclear Science, 57(2):768–774, 2010.

[3] Shigetaka Maeda and Tetsuo Iguchi. A new unfolding code combining maximum entropy and maximum likelihood for neutron spectrum measurement. Journal of Nuclear Science and Technology, 50(4):381–386, 2013.

[4] M. Matzke. Unfolding procedures. Radiation Protection Dosimetry, 107(1-3):155–174, 2003.

[5] Jan Mlynar, John M. Adams, Luciano Bertalot, and Sean Conroy. First results of minimum fisher regularisation as unfolding method for jet ne213 liquid scintillator neutron spectrometry. Fusion Engineering and Design, 74(1):781 – 786, 2005. Proceedings of the 23rd Symposium of Fusion Technology.

[6] M. Reginatto and P. Goldhagen. Maxed, a computer code for maximum entropy deconvolution of multisphere neutron spectrometer data. Health Phys., 77(5):579–83, 1999.

[7] Marcel Reginatto, Paul Goldhagen, and Sonja Neumann. Spectrum unfolding, sensitivity analysis and propagation of uncertainties with the maximum entropy deconvolution code maxed. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 476(1-2):242–246, 2002.

[8] Wolfgang von der Linden. Maximum-entropy data analysis. Applied Physics A, 60(2):155–165, 1995.

Appendices

A Choosing an underrelaxation factor

Consider equation 46, after the linearization approximation and removing O (higher order terms), the algorithm assumes that this is a simple linear algebra problem, i.e. the gradients in the landscapes stays constant everywhere. Therefore it takes a step as big as it think it needs to according to the gradient evaluated at $\boldsymbol{u}$ when there is no underrelaxation, i.e. $\alpha = 1$.

To fulfill Inequality 58, i.e. no flux values overshoots into the negative flux region, we can first define the vector intended fractional increase as $\boldsymbol{w}$

$$w_i = \frac{\delta_i^{g-1}}{f_i^{g-1}} \tag{74}$$

Obviously, if the intended fractional increase for the i^{th} element is less than -100% or worse, then the resulting f_i^g will definitely overshoot into the negative flux region if α is kept at 1.

To moderate this, we can choose $0 < \alpha < 1$ such that the worst fractional decrease is still less than -1, so $O(u_i, \Delta_i)$ doesn't diverge too much.

$$x = |min(\boldsymbol{w})| \tag{75}$$

$$\alpha = min(\frac{1}{2}\frac{1}{x}, 1) \tag{76}$$

The factor of 2 in the denominator of equation 76 is chosen to minimize the amount of deviation due to the term $O(u_i, \Delta_i)$. A conservative algorithm will use a larger denominator (e.g. 100), sliding down the gradient smoothly and slowly, evaluating the gradient at every point along the way; while a greedy algorithm will use a smaller denominator (e.g. 1.255 [4]) and therefore larger steps, occasionally overshoot due to underestimation of the steepness of the loss-value landscape (due to the linearization assumption), taking an oscillatory path to the correct answer.

[4] To ensure that the higher order terms in the Taylor expansion of ln() is still smaller than the first order term, $O(\frac{\delta}{x}) = |-ln(\frac{\delta}{x}+1)+1| < |\frac{\delta}{x}|$. Solving this inequality graphically gives $\frac{1}{|\frac{\delta}{x}|} > \frac{1}{0.7968} \approx 1.255$.

Note that since $\boldsymbol{w}$, which should be written as $\boldsymbol{w}^{g-1}$ if one were to be scrupulous, is different across different iterations. So the maximum value that α (again, should be written as α^{g-1} if one were to be scrupulous) can take is also dependent on the iteration.

B Uniqueness of global minimum

If one chooses the wrong loss function, it may lead to the existance of multiple local minima. Therefore it is important to show, under what condition will the loss function landscape have multiple local minima.

The following procedure can be applied onto any loss function with a loss function gradient composed of one linear part and one non-linear part. For our loss function $Loss(\phi_{sol}) = (\chi^2)_{\text{reaction rates}}(\phi_{sol}) + \tau \cdot RelativeEntropy_{\phi_0}(\phi_{sol})$, χ^2 is the linea part, relative-entropy is the non-linear part.

The approach used in the following derivation is to think of the solution space $\boldsymbol{f}$ as the domain, the gradient on the loss function $\boldsymbol{G} = \nabla(loss) + const.$ as the codomain. Note that $\nabla loss$ is a function of $\boldsymbol{f}$

First, it will be beneficial to review some of the basic terminology used. Additionally, for convenient communication, I will create a term called "hyperedge".

word	meaning	e.g. in 1D	3D	in the current context
domain	input space	x	$\boldsymbol{u} = (u_1, u_2, u_3)^T$	$\boldsymbol{f} = (f_1, ...f_n)^T$
codomain	output space	y=f(x)	$\boldsymbol{v} = \boldsymbol{F}(\boldsymbol{u})$	$\boldsymbol{G} = \nabla(loss) + const.$
Jacobian $\underline{\underline{\mathbf{J}}}$	how much is the domain "stretched" to form the codomain	$\frac{dy}{dx}$	$\underline{\underline{\mathbf{J}}} = \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{u}}; J_{ij} = \frac{\partial v_i}{\partial u_j}$	$J_{ij} = \frac{\partial G_i}{\partial f_j}$
hyperplane	an n-1 dimensional object	a point	a plane	
hyperedge	an n-2 dimensional object	N/A	a line	
simplex	object formed by n+1 vertices	a line	a tetrahedron	

Additionally, the definition of some less-commonly seen matrix decomposition methods is provided below:

SVD : singular-value-decomposition, equivalent of eigenvalue decomposition for a singular matrix.

Left-singular-vector: equivalent of a eigenvector for a singular matrix.

In the following derivation, it is helpful to think of the domain and codomain as separate vector spaces. Trying to plot f_i against G_i will only confuse the reader. Instead, we should think about the transformation from $\boldsymbol{f}$ into $\boldsymbol{G}$, where $\boldsymbol{f}$ and $\boldsymbol{G}$ are n-dimensional vector spaces. For n=2 & 3, the transformation is trivial when it's still linear; and slightly more difficult to imagine when it's non-linear. If you are adventurous, you can try to imagine n=4.

In section 4.1, the procedure of using Lagrangian multipliers led to equation 30 and 31. It is not difficult to see that equation 31 is a constraint that forces the solution to exist within a hyperplane in the domain space. However, perhaps it is more difficult to see that 30 is an equation that requires our solution to exist on a line $\boldsymbol{c} - \mu\hat{\boldsymbol{n}}$ in the codomain space. This is because, if we express the LHS of 30 as $\boldsymbol{G}$ (which is a function of $\boldsymbol{f}$), and condense the RHS into a constant vector $\boldsymbol{c'}$, then we have

$$G_i(\boldsymbol{f}) + \mu = c_i' \tag{77}$$

So that in codomain space, we have

$$\boldsymbol{G} - \boldsymbol{c'} = -\mu\hat{\boldsymbol{n}}, \tag{78}$$

where μ is an arbitrary real number, and

$$\hat{\boldsymbol{n}} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \tag{79}$$

The hyperplane in the domain space defined by equation 31 is mapped (in a non-linear manner, as dictated by the loss function's gradient) onto the codomain space, $\boldsymbol{f} \to \boldsymbol{G}$. The intersection points of it with the line $\boldsymbol{c'} - \mu\hat{\boldsymbol{n}}$ are the solutions.

B.1 Linear transformation

Now, if the mapping from $\boldsymbol{f} \to \boldsymbol{G}$ is (almost) linear (i.e. the terms in the loss function with higher than 2^{nd} order dependence on $\boldsymbol{f}$ on are vanishingly small), then we can be sure that there will be:

1. no solution if line $(\boldsymbol{c'} - \mu\hat{\boldsymbol{n}})$ does not intersect the hyperplane in the codomain space
2. 1 unique solution if the line penetrates the hyperplane at 1 point.
3. multiple (infinitely many) solutions if the line lies inside the hyperplane.

If the the hyperplane in the codomain space is parallel to the line $(\boldsymbol{c'} - \mu\hat{\boldsymbol{n}})$, then either situation 3 or situation 1 may occur. But the occurance of situation 1 does not guarantee that the hyperplane is parallel to the line $(\boldsymbol{c'} - \mu\hat{\boldsymbol{n}})$. This is because the allowed range of solution is bounded in the domain space by $f_i > 0$. (In 3D, this hyperplane can be visualized as the trianlge "covering" the origin, with the x-,y-, and z-intercept at 1,1, and 1.) Therefore, when a linear mapping is applied to transform this hyperplane from the domain space into the codomain space, an in appropriate choice of $\boldsymbol{c}$ will lead to no solution. This example (where there is no solution even when the line $(\boldsymbol{c'} - \mu\hat{\boldsymbol{n}})$ is not parallel to the hyperplane) corresponds to the cases where the fluxes in some bins must encroach on the negative fluxes regions in order to achieve optimal loss functions.

B.2 when the linear transformation is singular

Another problem is that $\underline{\underline{\mathbf{A}}}$ will be a singular matrix. Since the matrix $\underline{\underline{\mathbf{A}}}$ is constructed from the sum of m outer products of length n vectors, the number of non-zero eigenvalues of the matrix A is $\leq$ m. Therefore there are at least (n-m) eigenvalues which equal to zero[5].Meaning the linear transformation will map the $\sum f_i = 1$ hyperplane in the domain into a m-dimensional manifold in the codomain. In the rare case that the $\hat{\boldsymbol{n}}$ vector in the codomain is contained by this m- (or fewer-)

[5]Be careful when dealing with singular matrices such as $\underline{\underline{\mathbf{A}}}$, as computer programs in general will compute (incorrectly) to give non-zero eigenvalues, and successfully producing an "inverse" matrix without raising a warning.

dimensional manifold, it is possible to have infinitely many solutions, with at least (n-m) degrees of freedom in the solution space.

(Notice that the measured reaction rates only control $\boldsymbol{c}$. Therefore if the m-manifold is parallel to the $\hat{\boldsymbol{n}}$ vector (i.e. has bases vectors that can span $\hat{\boldsymbol{n}}$) in the codomain (i.e. condition 3 is met), then that means we have a response matrix where only a very specific ratio of measured reaction rates will give a solution at all. Outside of this range of physically possible measured reaction rates ratios, we will reqiure $\sum f_i \neq 1$ in order to fit to it optimally. The fact that $\hat{\boldsymbol{n}}$ is parallel to the manifold means we have a response matrix that causes no change at all in total χ^2 even after changing the fluxes in all manner imaginable. I.e. we must have a flat cross-section in that case.)

A sensible choice of the response matrix $\underline{\underline{\mathbf{R}}}$ (i.e. one without constant cross sections) will lead to a m-manifold that does not span the vector $\hat{\boldsymbol{n}}$. However, if one is feeling paranoid, they can check that their m-manifold object in the codomain space does not span the $\hat{\boldsymbol{n}}$ vector by $\underline{\underline{\mathbf{A}}}$ matrix does not by checking that all m right-singular-vectors $\boldsymbol{\theta}_k$

$$\hat{\boldsymbol{n}} - \sum_{k}^{m} (\hat{\boldsymbol{n}} \cdot \boldsymbol{\theta}_k)\boldsymbol{\theta}_k \neq \mathbf{0} \tag{80}$$

B.3 Non-linear transformation

In this section we will look more closely at loss functions which have a linear part and a non-linear part.

We have designed our loss function to increase logarithmically, so harsher penalty will be given by the loss function as $f_i \to 0^+$, as explained in section 2. Thus, the problems discussed in section B.1 and B.2 are usually avoided when using this loss funciton,

$$LossFunction(\boldsymbol{f}, \boldsymbol{f^{DEF}}) = \underline{\underline{\mathbf{A}}}\boldsymbol{f} + \tau \sum_{i}^{n} (f_i ln(\frac{f_i}{f_i^{DEF}})) \tag{81}$$

The first term in the loss function is linear, while the second term is non-linear.

For the moment, let's assume that $\underline{\underline{\mathbf{A}}}$ is non-singular, and τ is very small, i.e. for values of f_i far away from zero, the second term is effectively zero.

Therefore, the majority of the hyperplane in codomain space will look like $\underline{\underline{\mathbf{A}}}\boldsymbol{f}\{\forall \sum_i f_i = 1\}$. In 3D, this is the same as the triangle formed by $\boldsymbol{A}_1$, $\boldsymbol{A}_2$, $\boldsymbol{A}_3$, which are the three column vectors that makes up $\underline{\underline{\mathbf{A}}}$. Generalized to higher dimension, it is the (n-1) simplex formed by the n column vectors of $\underline{\underline{\mathbf{A}}}$.

At this point, it is convenient to define the following vector $\boldsymbol{H}$, such that

$$\boldsymbol{H} = \underline{\underline{\mathbf{A}}}\boldsymbol{f} \tag{82}$$

is the partial transformation of $\boldsymbol{f}$ into codomain $\boldsymbol{G}$.

$$G_i = H_i + \tau ln(f_i) \tag{83}$$

Note that the Jacobian of these transformations are as follows:

$$(\underline{\underline{\mathbf{J}}}_{\boldsymbol{f}\to\boldsymbol{H}})_{ij} = A_{ij} \tag{84}$$

$$(\underline{\underline{\mathbf{J}}}_{\boldsymbol{f}\to\boldsymbol{G}})_{ij} = A_{ij} + \tau \frac{\delta^*_{ij}}{f_i} \tag{85}$$

where δ^* is, again, the Kronecka delta.

After each of the above transformation, the direction of the hyperplane's normal vector is given as (without proof) $\hat{\boldsymbol{n}} \cdot \underline{\underline{\mathbf{J}}}^{-1}$, where $\hat{\boldsymbol{n}}$ is the unnormalized vector defined in 79. But note that this equation is only meaningful if A is non-singular, which is not true for the case when the is underdetermined.

Instead all we can do is check that the

B.4 Other failed approaches

In science it is equally important to record what failed and why. Here are the list of other starting points that led to nowhere.

- 'Using Stokes theorem as the starting point: leads to no useful results/intuition
- 'Using Divergence theorem as the starting point: only gives information about the curvature of an already known point, not useful for finding points where the gradient = 0
- 'Draw a line $(\boldsymbol{u} + \lambda \boldsymbol{v})$, anchored at a known global minimum $\boldsymbol{u}$, pointed at an arbitrary direction $\boldsymbol{v}$. If the loss function has a point $\frac{\partial(loss)}{\partial \lambda} = 0$ where $\lambda \neq 0$, then we know that a local minimum is present': Not feasible without knowning which directions $\boldsymbol{v}$ to draw the line in in the first place.