# The VALOR Analysis Software Development Kit

Costas Andreopoulos[e,h], Maria Antonova[c], Dom Barker[i], Chistopher Barry[e], Francis Bench[e], Andrew Chappell[j], Thomas Dealtry[d], Steve Dennis[e], Lorena Escudero[a], Nick Grant[j], Tereza Kroupova[g], Thomas Ham[e], Rhiannon Jones[e], Jaiden Parlone[e], Davide Sgalaberna[b], Bethany Slater[e], Raj Shah[f]

[a] *University of Cambridge, Cavendish Laboratory, Cambridge CB3 0HE, UK*
[b] *ETH Zurich, Institute for Particle Physics and Astrophysics, Zurich, Switzerland*
[c] *Instituto de Física Corpuscular (IFIC), Valencia, E-46980 Paterna, Spain*
[d] *Lancaster University, Physics Department, Lancaster, LA1 4YB, UK*
[e] *University of Liverpool, Department of Physics, Liverpool L69 7ZE, UK*
[f] *University of Oxford, Subdepartment of Particle Physics, Oxford, OX1 3RH, UK*
[g] *University of Pennsylvania, Department of Physics and Astronomy, Philadelphia, PA 19104-6396, USA*
[h] *Science and Technology Facilities Council, Rutherford Appleton Laboratory, Particle Physics Department, Oxfordshire OX11 0QX, UK*
[i] *University of Sheffield, Physics Department*
[j] *University of Warwick, Physics Department, Coventry CV4 7AL, UK*

## Abstract

VALOR[1] is a neutrino oscillation fitting group that was first established within the T2K experiment in 2010. The VALOR group has performed numerous iterations of the T2K electron-(anti)neutrino appearance, muon-(anti)neutrino disappearance, and joint 3-flavour analyses and it contributed to nearly all published T2K oscillation results. Through a large-scale code refactoring of the VALOR code, a generic, CPU-efficient and highly flexible oscillation analysis framework, the VALOR Software Development Kit (SDK), was carved out of the leading VALOR T2K analysis. The SDK has streamlined the adoption of the VALOR analysis for the physics exploitation and oscillation sensitivity simulation of several current and future experiments: Currently, VALOR oscillation analyses of T2K beam data, joint analyses of T2K beam data with SuperK atmospheric and NOvA beam data, as well as advanced oscillation sensitivity simulations for SBN, DUNE and

[1]https://valor.pp.rl.ac.uk

HyperK are (or are being) built as a thin layer of experiment-specific definitions and data inputs on top of a common analysis SDK. We present a general description of the VALOR SDK design and toolkit, and describe how the SDK facilitates the process of building, validating and running the VALOR oscillation analysis.

*Keywords:* VALOR, SDK, Neutrino, Oscillations, Data Analysis

## Contents

## Key

- VALOR namespaces: valor::utils
- External packages: e.g. ROOT, GSL
- Object names and function arguments: e.g. covariance_matrix
- External class names: e.g. TLorentzVector
- VALOR class names: e.g. **FitModelABC**
- VALOR application names: e.g. *valor-adaptive_mcmc*
- VALOR library names: e.g. *libValorOsc*
- Filenames and paths (in single quotes): e.g. *'/data/flux/bnb.root'*
- URLs: e.g. http://valor.pp.rl.ac.uk
- String literals (in double quotes): e.g. *"G16_02a_01_000"*
- Function signatures: e.g. *const Registry & Metadata(void) const*
- Typed-in commands: e.g. `% gevdump -f /data/file.ghep.root`
- Fragments of above (in single quotes): eg. `'-n 100'`
- Environmental variables: e.g. `$VALOR_SDK`

Note: A leading % in typed-in commands represents a shell prompt.

## 1. Introduction

VALOR is a modern oscillation analysis framework, that was first established within the T2K experiment in 2010 and named after its initial participants (VALencia-Oxford-Rutherford). That original name was maintained although the group has since expanded substantially. Since 2010, the VALOR group performed around 20 iterations of T2K electron-(anti)neutrino appearance, muon-(anti)neutrino disappearance, and joint 3-flavour analyses and it contributed to nearly all published T2K oscillation results. 2019 saw the completion of a large-scale code refactoring, painstakingly carving a generic, CPU efficient and highly flexible oscillation analysis *framework* out of a leading 3-flavour oscillation analysis. This framework is the VALOR Software Development Kit (SDK), that has streamlined use of the VALOR analysis in several experimental setups without the need to maintain multiple separate forks: Currently several distinct oscillation analyses (T2K, SuperK atmospherics, joint T2K+SuperK, joint T2K+NOvA, DUNE, SBN) are (or are being) built as a thin layer of experiment-specific definitions and data inputs on top of that single VALOR SDK. This document is the official VALOR SDK manual.

## 2. The VALOR SDK

The development of a *physics parameterization* is required for the analysis and interpretation of any physics dataset. Typically, this parameterization takes the form of a likelihood function, $\lambda$, which depends on some aspects of the recorded data and, in general, is a function of M physics parameters $\vec{\theta}$ (parameters of the physics hypothesis used for interpreting the data) and N nuisance (systematic) parameters $\vec{f}$. A successful generic analysis framework should facilitate the implementation of the analysis-specific choices on which aspects of a recorded dataset are to be included in the analysis (such as, for example, which kinematic distributions from which topological selections applied on which detector and/or running conditions), and it should allow the construction of corresponding predictions and the efficient parameterization of the effects of different physics hypotheses as of systematic effects on these predictions. In general, these analysis-specific choices include:

- The definition of topological event samples s. In principle, different topological samples can be used for different detectors included in the analysis, allowing, for example, the higher statistics near detector samples to be further subdivided into several exclusive channels and utilise correlations in order to disentangle systematic effects.

- The definition of reconstructed and true kinematic spaces. This includes the choice of kinematic space dimensionality, the choice of kinematic quantities, and the choice of binning schemes. In principle, the choice of a reconstructed kinematic space can be unique to each topological sample included in the fit.

- The choice of generator-level labels (modes m), whose contributions to the predicted event rate for a given topological sample are individually tracked. This choice can be unique to each topological sample included in the fit and it is crucial (along with the choice of kinematic quantities) so that the application of systematic and physics effects can be finally targeted and expressed with the most natural degrees of freedom.

- The choice of systematic parameters, the parameterization of the effect of systematic parameters on the computed event rates, and the definition of prior systematic parameter constraints.

The general framework used in the VALOR SDK for constructing that physics parameterization, is outlined in Sec. 2.1.

## *2.1. Framework for construction of the likelihood*

In the VALOR SDK an arbitrary number of samples, each corresponding to a given i) detector (or sub-detector region), ii) beam configuration, iii) observed final state, and iv) kinematic phase space, can be fit jointly to determine the parameters of a physics hypothesis in the presence of uncertainties.

A crucial element of parameter and error estimation in the context of an oscillation analysis is the efficient calculation of event rate predictions, both nominal and varied ones corresponding to specific combination of systematic effects and oscillation (or other new physics) hypotheses. Event rate predictions needs to be calculated for each topological event sample included in the fit, as a function of the observed kinematic quantities chosen for each sample. Predictions are constructed from Monte Carlo (MC) templates T, which represent the number of MC events (after event selection cuts were applied on the output of the full event simulation and reconstruction chain) as function of several reconstructed and true quantities, as needed in order to apply physics and systematic effects as function of the most appropriate variables, and in order to enable comparisons with the chosen observed kinematic distributions. Separate MC templates are constructed for each detector d, beam configuration b, and sample s. Each MC template contains information on how the number of events is distributed in the same $N_r$-dimensional space $K_r$ of reconstructed kinematic variables chosen for the fit samples. The same reconstructed kinematic variable binning scheme is used both for the MC templates and the fit distributions. The MC templates provide a mapping between reconstructed and true information. Separate templates are constructed for different true reaction modes m, and each template contains information on how the number of events, for each individual reconstructed bin, is distributed in a chosen $N_t$-dimensional space $K_t$ of true kinematic variables. The true reaction modes, the true kinematic variables, and the kinematic variable binning schemes are defined so that the intended flavour dependencies, reaction mode dependencies and kinematic dependencies of systematics and/or of considered physics hypotheses can be taken into account. Therefore, summarizing all MC template dependencies, we can write

$$T = T_{d;b;s;m}(r,t) \tag{1}$$

where r is a bin in the $K_r$ space of reconstructed kinematic variables, and t is a bin in the $K_t$ space of true kinematic variables. A MC template $T_{d;b;s;m}$ is constructed from a MC sample corresponding to an exposure of $e^{MC}_{d;b}$ and then used to predict observations, or fit data, corresponding to an exposure of $e^{data}_{d;b}$. Here, we will assume that a scaling factor $e^{data}_{d;b}$ / $e^{MC}_{d;b}$ is absorbed in the definition of $T_{d;b;s;m}$.

Using MC templates, the predicted number of events $n^{pred}_{d;b;s}(r;\vec{\theta};\vec{f})$ in a $N_r$-dimensional reconstructed kinematic bin r, for a specific sample s, seen in a detector d exposed in a beam configuration b, and for a particular set of M physics parameters $\vec{\theta} = (\theta_0, \theta_1, ..., \theta_{M-1})$ and N nuisance (systematic) parameters $\vec{f} = (f_0, f_1, ..., f_{N-1})$, is computed as

$$n^{pred}_{d;b;s}(r;\vec{\theta};\vec{f}) = \sum_m \sum_t P_{d;b;m}(t;\vec{\theta}) \cdot R_{d;b;s;m}(r,t;\vec{f}) \cdot T_{d;b;s;m}(r,t) \tag{2}$$

where $P_{d;b;m}(t;\vec{\theta})$ encapsulates the effect of a physics hypothesis (e.g. neutrino oscillations in a 3-flavour framework), and $R_{d;b;s;m}(r,t;\vec{f})$ parameterizes the response of a template bin to systematic variations. In principle, the range of m values in the above sum depends on the sample s. In addition, the number, type and dimensionality of true bins t is a function of both s and m. The above will be assumed implicitly and not written explicitly.

Once the construction of VALOR event rate predictions $n^{pred}_{d;b;s}(r;\vec{\theta};\vec{f})$ is implemented, through the factorization described in Eq. 2, physics is extracted through a comparison with observed (or simulated fake) data, which are denoted as $n^{obs}_{d;b;s}(r)$ and represent the observed event rate for each detector d, beam configuration b, topological sample s and (multi-dimensional) reconstructed kinematic bin r. VALOR, typically, uses a binned likelihood ratio method and, therefore, attempts to minimize the quantity:

$$-2ln\lambda(\vec{\theta};\vec{f}) = -2ln\lambda_0(\vec{\theta};\vec{f}) - 2ln\lambda_p(\vec{\theta};\vec{f}) \tag{3}$$

where

$$-2ln\lambda_o(\vec{\theta};\vec{f}) = 2\sum_{d,b,s,r}\Big(n^{obs}_{d;b;s}(r)\cdot ln\frac{n^{obs}_{d;b;s}(r)}{n^{pred}_{d;b;s}(r;\vec{\theta};\vec{f})} + (n^{pred}_{d;b;s}(r;\vec{\theta};\vec{f}) - n^{obs}_{d;b;s}(r))\Big) \tag{4}$$

and $-2ln\lambda_p$ is a penalty term encapsulating prior constraints from external data and theory. The advantage of the likelihood ratio method, compared with the extended maximum-likelihood method, is that in the large-sample limit, the quantity $-2ln\lambda(\vec{\theta};\vec{f})$ has a $\chi^2$ distribution and it can therefore be used as a goodness-of-fit test. Various strategies are employed for the minimisation of $-2ln\lambda(\vec{\theta};\vec{f})$, the elimination of nuisance and/or interesting physics parameters, and the construction of confidence intervals. These statistical procedures implemented within the VALOR SDK are summarised in Sec. 2.4.

## *2.2. Computing effects of physics hypotheses*

The effect of a physics hypothesis, such as, for example, neutrino oscillations, enters via the term $P_{d;b;m}(t;\vec{\theta})$ in Eq. 2. This term is naturally only a function of true kinematic variables and of neutrino flavour and/or true reaction mode (both of which are encapsulated in m). For multi-detector fits, the dependence on d and b reflects the dependence of $P$ on the appropriate baseline (or on the distribution of baselines, where that is appropriate).

The oscillation probability $P_{d;b;m}(t;\vec{\theta})$ is evaluated in one of a number of different physics frameworks, each with its own set of parameters $\vec{\theta}$. Currently, the VALOR SDK supports oscillation probability calculations in a

3-flavour framework, as well as 3+1 and 3+2 frameworks, considering matter-effects in constant density matter and, optionally, non-standard interactions. In addition, simpler 2-flavour oscillation probability calculations are also supported. The application of the term $P_{d;b;m}(t;\vec{\theta})$ requires some extra elaboration

### *2.2.1. Standard 3-flavour oscillations*

*3-flavour neutrino oscillation probabilities in vacuum*

The 3-flavour neutrino oscillation probabilities in vacuum are calculated from equation 11 in [1]:

$$P(\nu_\alpha \to \nu_\beta) = \delta_{\alpha\beta} - 4\sum_{i>j} Re(U^*_{\alpha i}U_{\beta i}U_{\alpha j}U^*_{\beta j}) \sin^2\left(\frac{\Delta m^2_{ij}L}{4E}\right) + 2\sum_{i>j} Im(U^*_{\alpha i}U_{\beta i}U_{\alpha j}U^*_{\beta j}) \sin\left(\frac{\Delta m^2_{ij}L}{2E}\right) \tag{5}$$

In equation 5, $\Delta m^2_{ij} = m^2_j$ - $m^2_i$, L is the neutrino baseline, E is the neutrino energy, and U is the PMNS matrix in vacuum:

$$U = \begin{pmatrix} c_{12}c_{13} & c_{13}s_{12} & s_{13}e^{-i\delta} \\ -c_{23}s_{12} - s_{13}s_{23}c_{12}e^{i\delta} & c_{12}c_{23} - s_{12}s_{13}s_{23}e^{i\delta} & c_{13}s_{23} \\ s_{12}s_{23} - s_{13}c_{12}c_{23}e^{i\delta} & -s_{23}c_{12} - s_{12}c_{23}s_{13}e^{i\delta} & c_{13}c_{23} \end{pmatrix} \tag{6}$$

with $s_{12} = \sin(\theta_{12})$, $c_{12} = \cos(\theta_{12})$, etc. $P(\bar{\nu}_\alpha \to \bar{\nu}_\beta)$ is calculated by changing the sign of the third term in equation 5 (this assumes that CPT is conserved).

*3-flavour neutrino oscillation probabilities in constant density matter*

All three neutrino flavours undergo neutral-current interactions with protons, neutrons and electrons in matter. These neutral-current interactions have identical amplitudes for all three flavours, and they produce no observable effects on the probabilities of oscillation between one flavour and another. However the oscillation probabilities are changed by coherent forward scattering of electron neutrinos in charged-current interactions with electrons in matter [1].

Using natural units with $\hbar$ = c = 1, the time evolution of neutrino flavour states in vacuum is described by the Schrödinger equation:

$$i\frac{\partial\psi}{\partial t} = H_F\psi \tag{7}$$

where

$$\psi = \begin{pmatrix} \psi_e \\ \psi_\mu \\ \psi_\tau \end{pmatrix} \tag{8}$$

and $H_F$ is the effective Hamiltonian in the flavour-state basis

$$H_F = \frac{1}{2E} UMU^\dagger \tag{9}$$

with M defined by

$$M = \begin{pmatrix} 0 & 0 & 0 \\ 0 & \Delta m^2_{21} & 0 \\ 0 & 0 & \Delta m^2_{31} \end{pmatrix} \tag{10}$$

The calculation of oscillation probabilities requires neutrino flight time to be converted to distance travelled, and this conversion assumes that neutrinos are highly relativistic. In this calculation, $UMU^\dagger$ is used for neutrinos, but its complex conjugate is used for antineutrinos. For neutrinos, the effects of the charged-current interactions between electron neutrinos and electrons are taken into account by adding the potential $2E\sqrt{2}G_F N_e$ to the real part of the first diagonal element of $UMU^\dagger$, where E is the neutrino energy, $G_F$ is the Fermi coupling constant, and $N_e$ is the number density of electrons in matter. For antineutrinos, this potential is subtracted from the real part of the first diagonal element of the complex conjugate of $UMU^\dagger$ (for further details see section 9.2 in [2]).

After the addition or subtraction of this potential, $UMU^\dagger$ (or its complex conjugate) is diagonalised. The eigenvalues of a Hermitian matrix are always real; they are calculated by solving the (cubic) characteristic equation using the method of del Ferro, Tartaglia and Cardano as described in [4]. The differences between the eigenvalues are the effective mass-squared differences in matter. The eigenvectors are calculated using an algebraic method: one of the components is set to be 1.0 (real), and the other two components are calculated using two of the three simultaneous equations in $UMU^\dagger - \lambda I = 0$, where the $\lambda$ are the eigenvalues. After being normalised, the three eigenvectors become the columns of the effective mixing matrix in matter.

The initial flavour state of the neutrino is represented by a 1×3 column vector whose entries are complex; if, for example, the initial flavour state is a muon neutrino, this column vector is

$$\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

The vector representing the initial flavour state is then multiplied by the Hermitian conjugate of the matter mixing matrix to convert it to another 1×3 complex column vector representing the initial mass states. These mass states are propagated to Super-K by multiplying the jth initial mass state by $\exp\left(\frac{-i\Delta m^2_{j1}L}{2E}\right)$, where the $\Delta m^2_{j1}$ are the mass-squared differences in matter, i.e. the differences between the eigenvalues of $UMU^\dagger$. This gives a new 1×3 complex column vector representing the final mass states, and these are converted to the final flavour states by multiplying by the matter mixing matrix. The entries of the resulting 1×3 complex column vector represent the amplitudes of each flavour at Super-K, and the corresponding oscillation probabilities are calculated as the moduli squared of these amplitudes.

*Inputs to the 3-flavour neutrino oscillation probability calculations*

When calculating the 3-flavour oscillation probabilities, the mixing angles can be input either in the form $\sin^2(2\theta_{ij})$ or as $\sin^2(\theta_{ij})$. $\sin(\theta_{ij})$ and $\cos(\theta_{ij})$ are calculated from the input values, and these are used in turn to calculate the elements of $U$. The "solar" mass-squared difference is input as $\Delta m^2_{21}$, and this is $> 0$ for both mass hierarchies. The "atmospheric" mass-squared difference is input either as $|\Delta m^2_{32}|$ or using a definition introduced by G. Fogli, E. Lisi et al. in [**?** ]:

$$\Delta m^2_{FL} = m^2_3 - \frac{m^2_2 + m^2_1}{2} \tag{11}$$

The quantity denoted $\Delta m^2_{FL}$ is useful since its absolute value is the same for both mass hierarchies, and only its sign changes. If the atmospheric mass-squared difference is input as $\Delta m^2_{32}$, the mass hierarchy is set using the sign of $\Delta m^2_{32}$; this is positive for the normal mass hierarchy (NH) and negative for the inverted mass hierarchy (IH). In this case, $\Delta m^2_{31}$ is not input but is calculated as $\Delta m^2_{31} = \Delta m^2_{21} + \Delta m^2_{32}$ for both hierarchies. If the atmospheric mass-squared difference is input as $\Delta m^2_{FL}$, the mass hierarchy is set using the sign of $\Delta m^2_{FL}$. Then $\Delta m^2_{32}$ and $\Delta m^2_{31}$ are calculated from the input values of $\Delta m^2_{FL}$ and $\Delta m^2_{21}$.

*Validation of the 3-flavour oscillation probabilities*

The 3-flavour oscillation probabilities in vacuum were checked using an alternative formulation in equations 13.13 and 13.14 in the PDG review [3]:

$$P(\nu_\alpha \to \nu_\beta) = \sum_j |U_{\beta j}|^2 |U_{\alpha j}|^2 + 2\sum_{j>k} |U_{\beta j} U^*_{\alpha j} U_{\alpha k} U^*_{\beta k}| \cos\left(\frac{\Delta m^2_{jk} L}{2E} - \phi_{\beta\alpha;jk}\right) \tag{12}$$

where $\phi_{\beta\alpha;jk} = \arg(U_{\beta j} U^*_{\alpha j} U_{\alpha k} U^*_{\beta k})$. $P(\bar{\nu}_\alpha \to \bar{\nu}_\beta)$ is calculated by changing the sign of $\phi_{\beta\alpha;jk}$ in the argument of the cosine. The probabilities calculated from equation 12 agreed with those calculated from equation 5 to 14 significant figures.

Several checks of the oscillation probabilities in constant density matter are written into the code that calculates them:

1. The eigenvalues of $UMU^\dagger$ are checked by comparing their sum with the trace of the matrix.
2. Each normalised eigenvector is multiplied in turn by each of the 3 matrices formed by subtracting the eigenvalues from the real parts of the diagonal elements of $UMU^\dagger$, and a check is made of the differences between these products and a zero vector.
3. A check is made that the normalised eigenvectors of $UMU^\dagger$ are orthogonal by calculating the scalar product between them (the scalar product of two complex vectors is the product of the first vector with the complex conjugate of the second vector).
4. A check is also made that the Hermitian conjugate of the mixing matrix in matter is equal to the inverse of the matrix. This is done by multiplying the mixing matrix in matter by its Hermitian conjugate, and checking the differences between the product and an identity matrix.

The values of the oscillation probabilities in constant density matter were checked by comparing them with equivalent probabilities calculated by an independently-written Fortran program [5]. This Fortran program uses different algorithms, and calculates numerically the eigenvalues and eigenvectors of $UMU^\dagger$. Nevertheless there was very good agreement between the two calculations of the probabilities, with the fractional differences being $\approx 2 \times 10^{-6}$.

The values of the matter oscillation probabilities were also compared with those given by Prob3++ [? ], and the fractional differences were $\approx$ 1-5 $\times$ $10^{-4}$. The comparison plots can be seen in [? ].

*Accuracy of the 3-flavour oscillation probabilities*

As stated in section **??**, the 3-flavour oscillation probabilities in vacuum agreed to 14 significant figures between the two formulations.

An estimate of the accuracy of the 3-flavour oscillation probabilities in constant density matter was made in three separate ways:

1. The matter probabilities for zero density were compared with the vacuum probabilities; there was good agreement, with the fractional differences being $\approx$ 2-5 $\times$ $10^{-6}$.
2. The probabilites given when calculating the eigenvectors with simultaneous equations 1 and 2 in $UMU^{\dagger} - \lambda I = 0$ were compared with the probabilities given when calculating the eigenvectors with simultaneous equations 2 and 3. The fractional differences between these two calculations were again $\approx$ 2-5 $\times$ $10^{-6}$.
3. The sum of three matter probabilities, for example P($\nu_\mu \rightarrow \nu_e$) + P($\nu_\mu \rightarrow \nu_\mu$) + P($\nu_\mu \rightarrow \nu_\tau$), was compared with 1.0; again there was good agreement, and the differences were $\approx 2 \times 10^{-6}$.

This means that the matter oscillation probabilities should be considered to be accurate to 5 decimal places. The accuracy of the matter probabilities is less than that of the vacuum probabilities due to the method of calculation of the eigenvectors. This involves calculations of the form

$$\frac{(ab - cd)}{(ef - gh)}$$

where a, b, c, etc. are elements of $UMU^{\dagger} - \lambda I$. Frequently ab is close in value to cd, which means that (ab - cd) is much smaller than ab or cd, and the same applies to (ef - gh). However the advantage of the method of calculation of the eigenvectors described in section **??** is that it allows the worst of these cancellation errors to be avoided; this can be done by calculating the eigenvectors using different pairs of simultaneous equations in $UMU^{\dagger} - \lambda I = 0$ for different combinations of the oscillation parameters. In practice, the eigenvectors are nearly always calculated using equations 1 and 2 in $UMU^{\dagger} - \lambda I = 0$, but equations 2 and 3 are used when 3.10 < $\delta_{CP}$ < 3.18 and $\sin^2(2\theta_{13}) > 10^{-8}$ in order to avoid such cancellation errors.

*Application of oscillation probabilities to MC templates*

In general, in a 3-flavour analysis, the CC MC templates constructed from the unoscillated MC samples are weighted with the corresponding survival probability: The $\nu_\mu$ CC templates are weighted with $P(\nu_\mu \to \nu_\mu)$, the $\bar{\nu}_\mu$ CC templates are weighted with $P(\bar{\nu}_\mu \to \bar{\nu}_\mu)$, the $\nu_e$ CC templates are weighted with $P(\nu_e \to \nu_e)$, and the $\bar{\nu}_e$ CC templates are weighted with $P(\bar{\nu}_e \to \bar{\nu}_e)$, The CC MC templates made from the *oscillated (or swapped)* $\nu_e$ event sample (a $\nu_e$ event sample that was generated with the $\nu_\mu$ flux, as if all $\nu_\mu$'s converted to $\nu_e$) are weighted with $P(\nu_\mu \to \nu_e)$. Similarly, if there was any, CC MC templates made from an *oscillated* $\bar{\nu}_e$ event sample would be weighted with $P(\bar{\nu}_\mu \to \bar{\nu}_e)$, CC MC templates made from an *oscillated* $\nu_\mu$ event sample would be weighted with $P(\nu_e \to \nu_\mu)$, and CC MC templates made from an *oscillated* $\bar{\nu}_\mu$ event sample would be weighted with $P(\bar{\nu}_e \to \bar{\nu}_\mu)$. Contributions from $\nu_\tau$ CC and $\bar{nu}_\tau$ CC are neglected as, even when they are produced via oscillations, have a negligible effect on observed distributions due to the high energy threshold for $\tau$ production. The same oscillation parameters are used for both neutrino and anti-neutrino oscillations. Oscillations are not applied to the NC MC templates since, effectively, they serve as proxies for the mixtures of $\nu_e + \nu_\mu + \nu_\tau$ NC and $\bar{\nu}_e + \bar{\nu}_\mu + \bar{\nu}_\tau$ MC templates which are unchanged under 3-flavour oscillations.

#### *2.2.2. Oscillations in a framework with N sterile neutrinos*

#### *2.2.3. Non-standard interactions*

As described in 2.2.1, neutrino oscillation probabilities are modified by forward coherent scattering of neutrinos off electrons in the matter they propagate through. Additional, flavour-dependent interactions could exist, leading to further modification of oscillation probabilities. These hypothetical interactions, which are usually referred to as Non-Standard Interactions (NSI), could modify the L/E dependence, or couple charged leptons and neutrinos of different flavours.

NSI are described by the following addition to the SM Lagrangian density

$$\mathcal{L}_{NSI} = -2\sqrt{2}G_F \epsilon^{fc}_{\alpha\beta} \left(\bar{f}\gamma^\mu P_c f\right)\left(\bar{\nu}_\alpha \gamma_\mu P_L \nu_\beta\right) \tag{13}$$

where $\epsilon^{fc}_{\alpha\beta}$ are parameters expressing the size of deviation from SM interactions for neutrinos of flavours $\alpha$,$\beta$, and chirality $c$. Typically, in experimental NSI searches, the large parameter space is reduced to a smaller set of effective parameters ($\epsilon_{\alpha\beta}$) by summing $\epsilon^{fc}_{\alpha\beta}$ over chirality and fermion flavour

$$\epsilon_{\alpha\beta} = \sum_{f,c} \epsilon^{fc}_{\alpha\beta} \frac{N_f}{N_e} \tag{14}$$

where $N_f$ is the number density of fermion $f$, and $N_e$ is the electron number density.

In the 3-flavour neutrino oscillation calculation, NSI effects enter as an additional contribution to the standard Hamiltonian shown in Eq.**??**.

$$A_{matter} = 2EV \begin{pmatrix} 1+\epsilon_{ee} & \epsilon^{\star}_{e\mu} & \epsilon^{\star}_{e\tau} \\ \epsilon_{e\mu} & \epsilon_{\mu\mu} & \epsilon^{\star}_{\mu\tau} \\ \epsilon_{e\tau} & \epsilon_{\mu\tau} & \epsilon_{\tau\tau} \end{pmatrix} \tag{15}$$

### *2.3. Framework for parameterization of systematic effects*

Systematic effects enter via the term $R_{d;b;s;m}(r,t;\vec{f})$ in Eq. 2. Typically, but not always, the response $R_{d;b;s;m}(r,t;\vec{f})$ factorises and it can be written as

$$R_{d;b;s;m}(r,t;\vec{f}) = \prod_{i=0}^{N-1} R^{i}_{d;b;s;m}(r,t;f_i) \tag{16}$$

For several systematics the response is linear and, therefore,

$$R^{i}_{d;b;s;m}(r,t;f_i) \propto f_i \tag{17}$$

For non-linear systematics, the response function $R^{i}_{d;b;s;m}(r,t;f_i)$ is usually pre-computed in the [-5$\sigma$, +5$\sigma$] range of the parameter $f_i$ and it is represented internally using Akima or cubic splines. Values of systematic parameters that give a negative predicted number of events in any reconstructed bin in any interaction mode are not allowed.

### *2.4. Statistical tools*

#### *2.4.1. Parameter estimation*

#### *2.4.2. Elimination of nuisance parameters*

#### *2.4.3. Construction of confidence intervals*

## 3. History of the code and releases

## 4. Installation

This describes how it should be, once SDK and DUNE-LBL are consolidated into the SDK

### *4.1. SDK versioning scheme*

### *4.2. Obtaining the source code*

### *4.3. External dependencies*

### *4.4. Preparation of the build environment*

Set the $VALOR_SDK environmental variable to point to the top directory of the SDK installation.

### *4.5. SDK configuration*

```
% cd $VALOR_SDK
% ./configure --enable-fault-on-die --enable-debug
```

Important: Where the last flag, –enable-negative-norm, should only ever be used during the systematic validation (when finished with this, the configure command must the re-run without these flags). This flags allows the parameters to take on either unphysical vales or go into regions where their behaviour may be invalid. We only use these options to test that the systematics are varying in an appropriate way.

### *4.6. Building the SDK*

```
% cd $VALOR_SDK
% make -j
```

### *4.7. Post-installation tests*

## 5. Code structure and analysis workflow in a nutshell

## 6. SDK packages and description of main classes

### *6.1. Conventions*

### *6.2. FitCore*

### *6.3. FitElements*

### *6.4. Fitters*

### *6.5. GENIEUtilities*

### *6.6. Ntuple*

### *6.7. Oscillation*

### *6.8. KineBinning*

### *6.9. CmdLine*

### *6.10. Messenger*

### *6.11. Registry*

### *6.12. Utilities*

### *6.13. VArray*

## 7. Description of configuration files

### *7.1. Main analysis configuration*

#### *7.1.1. Overview*

```

<?xml version="1.0"?>

<valor>

  <element_list name="list_name">
      <element name="element_name">
        ...
      </element>
      ...
  </element_list>

  <parameter_definitions>
     ...
  </parameter_definitions>
  <systematics name="\textcolor{red}{systematics_list_name}">
     ...
  </systematics>

</valor>
```

### *7.1.2. Element list block*

```
  <element_list name="dummy_water_cherenkov_experiment_1_sample">

    <element name="fhc_1rmu">
      <detector>     superk                             </detector>
      <beam>         JPARC_FHC                          </beam>
      <sample>       1rmu                               </sample>
      <observable>   Ereco                              </observable>
      <data>                                            </data>
      <mc>           mc/dummy_mulike.xml                </mc>
      <data_class>   valor::GenericSingleSampleFitData  </data_class>
      <model_class>  valor::GenericSingleSampleFitModel </model_class>
      <binning>      binning/dummy_mulike_bins.xml      </binning>
    </element>

  </element_list>
```

### *7.1.3. Parameter definitions block*

```
  <parameter_definitions>
    <!-- Make sure that any config file containing default values is read first! -
    <filename desc="default_physics_params"> params/default_physics_parameter_valu
    <filename desc="systematic_params">      params/dummy_parameter_specification.
    <filename desc="physics_params">         params/dummy_physics_parameter_specif
  </parameter_definitions>
```

### 7.1.4. Systematics error constraints and tuned values block

```
<systematics name="dummy">

  <group name="nd">
    <!-- This is the list of param names (to be read by AsStringSimple or AsStri
         The order of params in this list must be the same as the order of param
    <format> dummy_nd_constrained_params.xml </format>

    <covariance_matrix>
      <filename> dummy_data/cov/dummy_covariance_matrix.root </filename>
      <objname>  cov                                         </objname>
      <objtype>  TMatrixD                                    </objtype>
    </covariance_matrix>

    <untuned_values>
      <filename> dummy_data/cov/dummy_covariance_matrix.root </filename>
      <objname>  untuned_central_values                      </objname>
    </untuned_values>

    <initial_values>
      <filename> dummy_data/cov/dummy_covariance_matrix.root </filename>
      <objname>  params                                      </objname>
    </initial_values>
  </group>

</systematics>
```

## 7.2. Parameter definitions

## 7.3. MC inputs

### 7.3.1. Event samples

See mc/dummy_mulike.xml

### 7.3.2. Normalisation

### 7.3.3. Response functions

Example:

```
<splines>
  <folder>         dummy_data/xsec/valor_format                  </folder>
  <pattern>        splines_spline2018v1_[BEAM]_[SAMPLE].root     </pattern>
  <stored_as>      TObjArray                                     </stored_as>
  <array_pattern>  spl_[MODE]_[PARAM]                            </array_pattern>
  <method>         curr_tweak                                    </method>
  <!-- This uses weight = spline->Eval(current - untuned) -->
  <type>           TSpline3                                      </type>
  <!-- Can use either ROOT's TSpline3, or various GSL splines. See Interpolator.h
  <sort_knot_order> false                                        </sort_knot_orde
  <!-- Sometimes we get splines with out-of-order knots. These knots are ignored b
  <within_limits>  true                                          </within_limits>
  <!-- Setting this to false allows extrapolation beyond the spline limits. WARNIN
  <allow_negative>  true                                         </allow_negative
  <!-- Splines that go negative are no longer replaced with flat splines, but the
  <set_neg_wght_to> 2e-10                                        </set_neg_wght_t
  <!-- When we would get a negative weight from a spline, force it to this insteac
</splines>
```

### *7.4. Specification of reaction modes*

### *7.5. Specification of binning schemes*

Types of binning: Optimized and simple - explain. Edge-to-edge and Mondrian - explain.

Reco -¿ Optimized and simple True -¿ Simple only

### *7.6. Verbosity level - Message stream thresholds*

## 8. Running and validating a neutrino data analysis

### *8.1. Writing out tables of nominal event rates*

Executable: make_validation_spectrum_tables.py

Code location: $VALOR_EXPT/src/macros/spectra/single_sample/

Expand the following example to a proper command synopsis, listing all the arguments and expected/allowed settings

Synopsis:

```
% make_validation_spectrum_tables.py
  --mixing12 0.304
  --mixing13 0.0212
  --mixing23 0.528
  --cpv -1.601
  --dm12sq 7.53E-5
  --dm23sq 2.509E-3
  --file rates_dummy_AsimovA.tex
  --config analysis/dummy_wc_1sample.xml
  -d 5
```

### *8.2. Write out MC templates and nominal event rate distributions*

Executable: valor$_{d}une - write_{p}df$

Code location:

Expand the following example to a proper command synopsis, listing all the arguments and expected/allowed settings

```
Argument              | Type   |Opt| Default
-------------------------------------------------------------------------
physics-hypothesis    | str    | Y | 3f_osc
fit-param-opt         | str    | Y | NONE
config                | str    | N | -
write-opt             | str    | Y | pdft_vf+pdft_rf+pdfcs_rf+pdfcs_vf+SB
output-name           | str    | Y | valor_pdf
mesg-thresholds       | str    | Y | verbosity.xml
pot-override          | dbl    | Y | -999
(force,no-force)-dst  | {bool} | Y | no-force-dst
```

Example:

```
% valor_dune-write_pdf
  --physics-hypothesis 3f_osc
  --input-mixing12 0.304
  --fix-mixing12
  --input-mixing13 0.0212
  --fix-mixing13
  --input-mixing23 0.528
  --fix-mixing23
  --input-cpv -1.601
  --fix-cpv
  --input-dm12sq 7.53E-5
  --fix-dm12sq
  --input-dm23sq 2.509E-3
  --fix-dm23sq
  --write-opt all+SB
  --config analysis/dummy_wc_1sample.xml
  --output-name write_dummy_all_sb
```

### *8.3. Plot nominal event rate distributions*

Use the *VALOR_EXPT/src/macros/spectra/single_sample/plot_pdf.Cmacro, withtheoutput* *write_pdf*.

Synopsis:

Expand the following example to a proper command synopsis, listing all the arguments and expected/allowed settings

```
% valor -b -q
  $VALOR_EXPT/src/macros/spectra/single_sample/plot_pdf.C'(
     "analysis/dummy_wc_1sample.xml",
     "write_dummy_all_sb.root",
     "valor",
     "Ereco",
     -999,
     "",
     "",
     "SB",
     "dummy_all_sb",
     "pdf",
     1000,
     750,
     3,
     1.25,
     180.
     0,1)'
```

## *8.4. Generate and event rate distributions with systematic variations*

```
valor_dune-tweak_pdf_sigma_var
  --output sigma_var_mode_[PARAMS]
  --config analysis/dummy_wc_1sample.xml
  --physics-hypothesis 3f_osc
  --input-mixing12 0.304
  --input-mixing13 0.0212
  --input-mixing23 0.528
  --input-deltaCP -1.601
  --input-dm12sq 7.53E-5
  --input-dm23sq 2.509E-3
  --normal
  --kinevar Ereco
  --params allsyst
```

## *8.5. Plot event rate distributions with systematic variations*

## *8.6. Plotting uncertainty bands on event rate distributions*

## *8.7. Testing the action of systematic parameters (the tick table)*

## *8.8. Generating fake data*

## *8.9. Generating and fitting toy-MC experiments*

Executable: valor_dune-run_fit

Code location:

Expand the following example to a proper command synopsis, listing all the arguments and expected/allowed settings

Description of command-line arguments:

```
Argument                      | Type   |Opt| Default
-------------------------------------------------------------------------
tag                           | str    | Y | toymc
run                           | int    | Y |   0
ntoymc                        | int    | Y |   1
physics-hypothesis            | str    | Y |   3f_osc
fit-param-opt                 | str    | Y |   NONE
param-values-option           | str    | Y |   randomize-all;prefit-errors
fitter                        | str    | Y |   minuit
config                        | str    | N |   -
mesg-thresholds               | str    | Y |   verbosity.xml
nthreads                      | int    | Y |   -1
(with,without)-stat-fluct     | {bool} | Y |   with-stat-fluct
(store,discard)-spectra       | {bool} | Y |   discard-spectra
start-at-(true,central)       | {bool} | Y |   start-at-true
(rethrow,accept)-unphysical   | {bool} | Y |   rethrow-unphysical
true-from-filename            | str    | Y |   NONE
true-from-objname             | str    | Y |   central_values
pot-override                  | dbl    | Y |   -999
load-prior-from-file          | str    | Y |   none
prior-fit-number              | int    | Y |   0
prior-file                    | str    | N |   -
tweak-parameter-set           | str    | Y |   none
force-tweak-size              | dbl    | Y |   0
flip-true-octant              | bool   | Y |   0
(read,generate)-data          | {bool} | Y |   generate-data
```

Example:

```
% valor_dune-run_fit
  --tag fit_minuit_dummy_1sample
  --second-octant
  --run 1
  --ntoymc 1
  --physics-hypothesis 3f_osc
  --param-values-option prefit-errors
  --fitter minuit
  --config analysis/dummy_wc_1sample.xml
```

```
--mesg-thresholds verbosity.xml
--without-stat-fluct
--normal
--load-prior-from-file
-
--pot-override -999
--store-spectra
--generate-data
--seed-toys-stat 1000
--seed-toys-syst 2000
--seed-toys-osc 3000
--seed-fitter 4000
--seed-other 5000
--prior-file priors/dummy_priors.xml
```

### *8.10. Fitting an input (fake or real) dataset*

### *8.11. Reading the standard output VALOR fit file*

### *8.12. Producing oscillation contours*

## 9. Summary

## References

[1] B. Kayser (2005), Neutrino physics, hep-ph 0506165

[2] C Giunti and C.W. Kim (2007) Fundamentals of neutrino physics and astrophysics, Oxford Uninversity Press

[3] K. Nakamura and S.T. Petcov (2010) Neutrino mass, mixing and oscillations, http://pdg.lbl.gov/2010/reviews/rpp2010-rev-neutrino-mixing.pdf

[4] J. Kopp (2006), Efficient numerical diagonalisation of Hermitian 3x3 matrices, arXiv:physics/0610206

[5] T. Sloan, private communication (June-July 2011)

[6] G. Cowan (1998) Statistical data analysis, Oxford University Press, ISBN 0-19-850155-2.

[7] Numerical Recipes: The art of scientific computing, Cambridge University Press, 3rd Edition, ISBN 978-0-521-88068-0, pp. 100-102.

[8] G.J. Feldman and R.D. Cousins, A Unified approach to the classical statistical analysis of small signals, Phys.Rev.D57:3873-3889,1998; physics/9711021