SLAC-PUB-17563

Inference of neutrino flavor evolution through data assimilation and neural differential equations

Ermal Rrapaj,[1,2,*] Amol V. Patwardhan,[1,3,4,†] Eve Armstrong,[5,6,‡] and George M. Fuller[7,§]

[1] *Department of Physics, University of California, Berkeley, CA 94720, USA*
[2] *School of Physics and Astronomy, University of Minnesota, Minneapolis, MN 55455, USA*
[3] *SLAC National Accelerator Laboratory, 2575 Sand Hill Road, Menlo Park, CA, 94025*
[4] *Institute for Nuclear Theory, University of Washington, Seattle, WA 98115, USA*
[5] *Department of Physics, New York Institute of Technology, New York, NY 10023, USA*
[6] *Department of Astrophysics, American Museum of Natural History, New York, NY 10024, USA*
[7] *Department of Physics, University of California, San Diego, La Jolla, CA 92093-0319, USA*

(Dated: October 5, 2020)

The evolution of neutrino flavor in dense environments such as core-collapse supernovae and binary compact object mergers constitutes an important and unsolved problem. Its solution has potential implications for the dynamics and heavy-element nucleosynthesis in these environments. In this paper, we build upon recent work to explore inference-based techniques for estimation of model parameters and neutrino flavor evolution histories. We combine data assimilation, ordinary differential equation solvers, and neural networks to craft an inference approach tailored for non-linear dynamical systems. Using this architecture, and a simple two-neutrino, two-flavor model, we test various optimization algorithms with the help of four experimental setups. We find that employing this new architecture, together with evolutionary optimization algorithms, accurately captures flavor histories in the four experiments. This work provides more options for extending inference techniques to large numbers of neutrinos.

I. INTRODUCTION

Core-collapse supernovae and binary compact object mergers are extreme physical environments with the potential to serve as valuable laboratories at the intersection of particle theory, dense matter physics, and high-energy astrophysics. Many of the important physical phenomena in these environments, such as shock propagation, bulk matter outflows, and the synthesis of heavy-elements are driven in part by interactions between nuclear matter and the accompanying prodigious flux of emitted neutrinos [1–4].

In these situations, the flavor evolution of the neutrinos is a complicated, nonlinear problem, wherein the flavor histories of neutrinos with different energies and trajectories are coupled to one another. This has been shown to lead to various collective flavor oscillation phenomena [5–15]. In particular, in the last few years, it has been demonstrated that relaxing certain assumptions regarding spatial and temporal symmetries in the neutrino flavor field can lead to flavor instabilities not previously identified, and which have not been well studied [16–40].

Since the flavor evolution of neutrinos is so inextricably linked to the transport of energy and lepton number in these environments, it is important to identify the initial conditions and physical regimes under which these noted flavor-field instabilities and collective phenomena can manifest themselves. Meanwhile, the next generation of terrestrial detectors such as DUNE [41] and Hyper-Kamiokande [42] could potentially provide a detection of a large number ($\sim \mathcal{O}(10^3$–$10^4)$) of neutrinos from Galactic core-collapse supernovae events. Thus it is pertinent to ask what such a detection could reveal about neutrino properties, as well as the physics of the supernova environment itself.

The last decades have seen the rapid development of machine learning (ML) algorithms, many of which utilize "big data" to solve difficult problems such as image recognition [43–45] and natural language processing [46, 47]. The training of most ML algorithms requires large amounts of data, in part because initial conditions are typically not assumed to be well known. Not surprisingly, scientific fields that typically produce large data sets have leveraged these technological advances [48, 49]. Other domains of science and engineering, however, are characterized by sparse data - sparsity that precludes the application of such learning algorithms to problems in these fields. Instead, these fields have long established research traditions which have led to the development of predictive models.

At first glance, then, there appears to be a dichotomy in approach: data-driven machine learning on one hand, and theoretical models with few parameters on the other. If no model knowledge is available but large amounts of data are, the first approach seems very reasonable. When dealing with sparse data, however, ignoring prior knowledge of the system is counterproductive, especially as the sparsity of the available data might preclude the training of the algorithm if it must learn "from scratch". In the physical sciences and engineering fields characterized by highly developed theoretical frameworks, the few unknowns are usually modeled by known functions with few

* ermalrrapaj@gmail.com
† apatward@slac.stanford.edu
‡ evearmstrong.physics@gmail.com
§ gfuller@ucsd.edu

free parameters to be fitted from data. In such cases, the function chosen to represent the unknown part of the model is typically dictated by the researchers' experience and intuition. But, this educated guesswork might not be general enough to account for all possible functional forms that are both useful for the model and appropriately represent the data.

Recently, in calculations of neutrino flavor transformation, there have been attempts [50, 51] to combine the best from both approaches noted above, by utilizing an inference procedure known as statistical data assimilation (SDA). In an SDA procedure, real or simulated data from measurements are considered together with assumed theoretical constraints, to complete a model with one or more unknown parameters. This work continues to explore SDA for constraining neutrino flavor evolution histories inside core-collapse supernovae, with two key extensions of previous work. Firstly, we incorporate into the procedure recent developments in ML, to ascertain whether a "SDA-ML hybrid" ameliorates problems associated with grid discretization. Specifically, we employ a deep neural network to replace any guesswork of the unknown parameters and include a differential equation solver in the neural architecture [52]. Secondly, we perform the calculations using various optimization algorithms, for a comparative analysis of performance.

In section II, we describe the algorithmic setup and specifics of the problem. We perform two sets of inference experiments, for a two-neutrino system. In the first set (sections III and IV), we focus on the estimation of unknown parameters governing the matter potential. The second set (sections V and VI) assumes that the matter potential is known, and instead focuses on the estimation of initial conditions regarding flavor. Finally, we present our conclusions in section VII.

II. INVERSE PROBLEMS AND NEURAL NETWORK OPTIMIZATION

A. General Framework

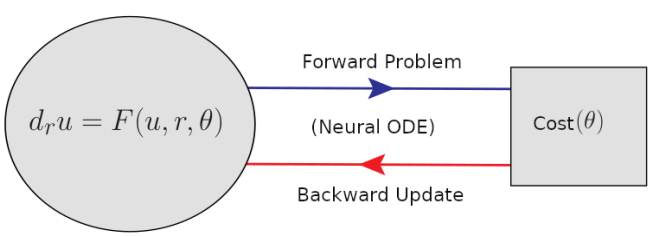

FIG. 1. Data assimilation as neural network architecture. The unknown parameters of the forward problem are denoted by $\boldsymbol{\theta}$. Cost$(\boldsymbol{\theta})$ is the objective function whose minimum denotes the optimal solution of the inverse problem. Blue arrows denote the forward process of prediction, and red arrows denote the backward pass for error correction.

Data assimilation is an inverse problem formulation [53]: a procedure whereby information in measurements is used to complete a model of the system from which the measurements were obtained. For our purposes, the model $\boldsymbol{F}$ is written as a set of ordinary differential equations that evolve in affine parameter r as:

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}r} = \boldsymbol{F}(\boldsymbol{u}, r, \boldsymbol{\theta}) \tag{1}$$

where the vector $\boldsymbol{u}$ is the observable being modeled, with initial value $\boldsymbol{u}_0$. The affine parametrization r may be, for example, time or distance. Any unknown parameter that influences the forward problem (differential equation) is denoted by $\boldsymbol{\theta}$. An observation $\boldsymbol{u}_D$ is made at a detector location R and one seeks to estimate $\boldsymbol{\theta}$ that best fits this observation. This is achieved through minimization of the cost function:

$$\mathrm{Cost}(\boldsymbol{\theta}) = \left[\boldsymbol{u}_D - \boldsymbol{u}_\theta(R)\right]^T \boldsymbol{W} \left[\boldsymbol{u}_D - \boldsymbol{u}_\theta(R)\right] \tag{2}$$

where $\boldsymbol{u}_\theta(R)$ is the prediction from $\boldsymbol{F}$. In practice, we may only observe a subset of the components of the vector $\boldsymbol{u}_D$, while through $\boldsymbol{F}$ all components are evolved in order to predict the final values at the detector. The sparse matrix $\boldsymbol{W}$ is introduced to select these components when computing the cost function.

In [50, 51], the cost function was comprised of three parts: model error, measurement error, and physical constraint terms. The model error, in addition to permitting uncertainty in the model parameter estimates, included terms related to the uniform discretization of the domain and finite difference approximation of the derivative. In addition, as the optimization algorithm took the grid points to be independent, we had imposed co-variation of the model coordinates into the cost function as an equality constraint. Here, we explicitly include only the measurement term in the cost function, as the neural architecture ensures that the other terms are automatically satisfied; this will be explained in detail below. In [51], we also considered $\boldsymbol{u}_0$ to be an input in the form of a measurement. Here, in the first two experiments we follow the same assumption, but without explicitly including $\boldsymbol{u}_0$ as part of the cost function (the cost function implicitly depends on $\boldsymbol{u}_0$, via the dynamical equations). In the last two experiments, we assume all model parameters are known, and we instead optimize the cost function by varying the initial conditions $\boldsymbol{u}_0$.

In recent decades, machine learning has been used to provide solutions to ordinary differential equations [54–58]. As the focus of this work is the inverse problem, the system of differential equations is a building block of our setup. As such, our goal is not to approximate the ODE solution through a neural architecture or discretized grid, but rather to understand which parameters in the ODE definition lead to a solution that best matches observations. Hence, we use the existing vast and established literature on solving ODEs through forward integration in the vein of incorporating model knowledge with machine

learning. This is the motivation for using the recently developed neural ODE [52] network for data assimilation. This network will automatically incorporate our model knowledge of the dynamical system, through Eq. (1).

We include an adaptive step solver method in the neural architecture, removing the need for domain discretization and errors induced by such discretization. Specifically, we employ the Radau method [59]. The forward-problem arrow in Fig. 1 refers to this part of the architecture. The solution $\boldsymbol{u}_\theta(r)$ then satisfies all the physical constraints associated with Eq. (1). Consequently, no model and no constraint terms are needed in the cost function. As such, the cost function in Eq. (2) contains only the measurement term. We have verified that the errors associated with model and constraint terms are within numerical precision ($\lesssim 10^{-16}$). An additional benefit of this setup is the reduction of the number of unknown parameters that require optimization. Obtaining adequate resolution in the previous setup required a rather large number of grid points. That burden is reduced in this new architecture. The points in the domain are automatically chosen by the adaptive Radau method to solve the forward problem to machine precision.

Before moving to the backward update arrow, we comment on the decision to incorporate an ODE solver into a neural architecture. At first glance, this choice may seem puzzling. The most general definition of a neural layer is a differentiable function with tensor input and output. This is the basis of a general differentiable programming architecture. Traditionally, neural layers are superpositions of simple primitive functions, but they need not be. Differential equation solvers naturally fit this framework, as an ODE solver has an input vector $\boldsymbol{u}(r_n)$ that outputs a new vector $\boldsymbol{u}(r_{n+1})$, where the points (r_n and r_{n+1}) and the separation between those points are determined adaptively. In order for the solver to be a neural layer, the output of the ODE solver must be differentiable with respect to the unknown parameters. This is achieved through automatic differentiation [60, 61] and adjoint sensitivity analysis [62, 63]. Automatic differentiation in computer science encompasses a set of techniques for converting a program into a sequence of primitive operations that have specified routines for computing derivatives. It is efficient, in that there exists a linear time cost in computing values, and it is numerically stable. Adjoint sensitivity analysis allows for the automatic differentiation of ordinary differential equations. The code for this work was written in the Julia programming language [64], and we used the DiffEqFlux package [65–67]. While the interested reader may delve deeper into these interesting topics, for the purposes of this work, it suffices to state that the current architecture allows us to compute the Jacobian and Hessian matrices of the dynamical model ($\bigtriangledown_{\boldsymbol{\theta}}\mathrm{Cost}(\boldsymbol{\theta})$ and $\partial_{\theta^i}\partial_{\theta^j}\mathrm{Cost}(\boldsymbol{\theta})$).

Once the forward problem is implemented, one performs the minimization of the cost function. In Fig. 1 this step is represented by the backward update arrow. Typically, in deep learning, the minimization is performed through stochastic gradient descent (SGD) [68] (which requires the computation of the gradients mentioned). In practice, the step size of the parameter update is a hyper parameter of the training procedure, which must be tuned to achieve convergence. Many improvements on SGD have been developed through decades. For instance, AdaGrad [69] and Adam [70] are algorithms widely used in deep learning. LBFGS [71], which also approximates the Hessian of the cost function, is a commonly used alternative to SGD.

The focus of this work is global optimization (finding the optimal value in the entire region of interest), and the algorithms mentioned above work locally, informed by gradients of the cost function. Hence, we also employed a number of global algorithms in our analysis. A list of algorithms used in this work, along with their classification, is given below:

- Monte Carlo based methods (gradient free): "Simulated Annealing" (SAMIN) [72, 73] — based on Metropolis-Hastings algorithm to generate samples from a thermodynamic system,
- Evolutionary algorithms (gradient free):
  1. "Improved Stochastic Ranking Evolution Strategy" (ISRES) [74, 75] — based on a combination of a mutation rule (with a log-normal step-size update and exponential smoothing) and differential variation (update rule similar to Nelder–Mead [76]) ,
  2. "Adaptive Particle Swarm Algorithm" (APS) [77] — improve global coverage and convergence by switching between four evolutionary states: exploration, exploitation, convergence, and jumping out.
- Jacobian and Hessian based methods:
  1. "Interior Point Optimizer" (IPOPT) [78] — a primal-dual interior point method which uses line searches based on filter methods. IPOPT is designed to exploit 1st and 2nd derivative information if provided. If no Hessians are available, IPOPT will approximate them using a quasi-Newton methods, specifically a BFGS update.
  2. "Newton method with Trusted Region Hessian" (NTR) [79] — quadratic approximation of the objective function by means of the hessian with steps restricted to be within a 'trusted' region where the approximation is believed to be valid.
- Combination of global and local optimization:
  1. "Stochastic Global Optimization" (STOGO) [80] — systematically divide the search space (which must be bound-constrained) into smaller hyper-rectangles via

a branch-and-bound technique, and searching them by a gradient-based local-search algorithm.

2. 'Multi-Level Single-Linkage" (MLSL) [81, 82] — global optimization by a sequence of local optimizations from random starting points, in conjunction with local optimizations algorithms
   - BOBYQA [83] — (gradient free) bound-constrained optimization using an iteratively constructed quadratic approximation for the objective function,
   - "Method of Moving Asymptotes" (MMA) [84] — local, convex and separable approximation of the objective function from the gradient,
   - LBFGS [85, 86] — quasi-Newton method that approximates the Broyden–Fletcher–Goldfarb–Shanno algorithm (BFGS) [87] using a limited amount of computer memory.

These algorithms cover a wide range of methodologies. For instance, multi-level algorithms (MLSL) have been used for over three decades in optimization, and nowadays are part of many statistical programming languages. Simulated annealing is another widely used algorithm, with over four decades of applications. In addition to the familiar Newton's method and IPOPT, we have also included evolutionary algorithms like ISRES and APS which are inspired by biological evolution [88]. Clearly, optimization is a rather fascinating and varied field. The numerical implementation can be found in NLopt [89] and Optim [90] packages. In all experiments covered in this work, we set the maximal number of iterations for the optimization procedure to 1000.

B. Specifics of the problem

The neutrino flavor evolution problem has been explained in detail in [51]. Here we summarize the system of differential equations,

$$\boldsymbol{F}_i = \frac{\mathrm{d}\boldsymbol{P}_i}{\mathrm{d}r} = \left(\Delta_i \boldsymbol{B} + V(r)\hat{z} + \mu(r) \sum_{j \neq i} \boldsymbol{P}_j \right) \times \boldsymbol{P}_i \quad (3)$$

Here, $\Delta_i = \delta m^2/(2E_i)$ are the vacuum oscillation frequencies of neutrinos with energies E_i. The mass-squared differences in vacuum are δm^2. The unit vector representing neutrino flavor mixing in vacuum is $\boldsymbol{B} = \sin(2\alpha)\hat{x} - \cos(2\alpha)\hat{z}$, where α is the mixing angle between the flavor and mass eigenstates. The functions $V(r)$ and $\mu(r)$ are the potentials arising from neutrino-matter and neutrino-neutrino interactions, respectively. The "polarization vectors" $\boldsymbol{P}_i$, which contain information about the flavor composition of the neutrinos, play the role of the state variable $\boldsymbol{u}$ from Eq. (1), and the only components that are measured at the detector are the P_z of each neutrino.

In our model, we take the neutrino-neutrino potential to be,

$$\mu(r) = \frac{\mu_0}{(r+\delta_0)^4}. \quad (4)$$

This choice is consistent with how the coupling strength varies in the neutrino bulb model calculations employing the single-angle approximation. In our SDA experiments, μ_0 is taken to be a constant with a known value and $\delta_0 = 10^{-3}$ is added to avoid any numerical singularities at $r_0 = 0$. The matter potential $V(r)$ is chosen to be

$$V(r) = \frac{V_0}{(r+\delta_0)^3}. \quad (5)$$

In the first two experiments, V_0 is treated as an unknown parameter that we optimize. To generate the simulated "detector data" ($\boldsymbol{u}_D$ in Eq. (2)), we use the value $V_0 = \tilde{V}_0$ of the matter potential coefficient, given in table I. As a thought experiment and proof of concept, we study a system of two neutrinos, with all the parameters used for simulated data generation displayed in table I. In future work we intend to study much larger systems.

Parameter	*Value*	*Initial polarization*	*Value*
Δ_1	30	$P_{1,z}(r_0)$	-1.0
Δ_2	55	$P_{2,z}(r_0)$	1.0
μ_0	10.0	*Final polarization*	*Value*
$\tilde{V}_0$	50.0	$P_{1,z}(R)$	0.20575
α	0.15	$P_{2,z}(R)$	-0.96750
r_0	0		
R	5		

TABLE I. **Model parameters used for generating the simulated 'detector' data.** Δ_i are the vacuum oscillation frequencies of the neutrinos, and $(\mu_0, \tilde{V}_0)$ are the multiplicative factors governing the neutrino-neutrino coupling potential $\mu(r)$ and matter potential $V(r)$. Parameter α is the mixing angle in vacuum. Neutrino 1 is initially x flavor and neutrino 2 is initially electron flavor.

In Fig. 2 we display the z components of the polarization vectors as functions of r for the parameters of table I. As the figure shows, the two neutrinos are initially in pure flavor states (electron and x). At some intermediate distance there is a large flavor transformation, and the two neutrinos swap flavors. As expected, with increasing distance, both matter and neutrino potentials become less relevant and we can observe vacuum oscillations.

III. MATTER POTENTIAL COUPLING AS AN UNKNOWN CONSTANT

In this section, we assume the matter potential coupling V_0 is an unknown parameter and ask whether the

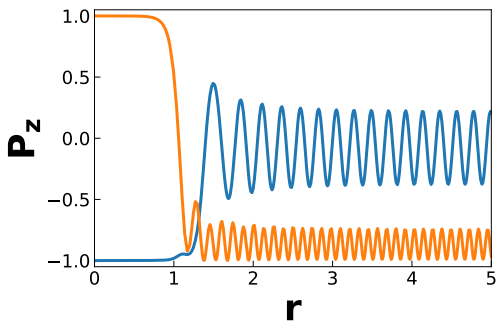

FIG. 2. The z components of the two neutrino polarization vectors as functions of the affine parameter r. The parameters used in the forward integration are shown in table I.

z components of the polarization vectors of the two neutrinos at the detector provide sufficient information to infer the true value $\tilde{V}_0$ provided in Table I. As this is a small system, and there is only one unknown parameter, we can plot the dependence of the cost function on the unknown parameter using repeated forward integration, as shown in Fig. 3. That is, the forward code was run several times with different parameter values V_0, and the corresponding values of P_z at the endpoint in each case were compared with the true values (that is, with $V_0 = \tilde{V}_0$) to generate the cost function using Eq. (2). In practice, physical systems contain many more particles, and more unknowns, rendering it infeasible to create the plot analogous to Fig. 3. We are considering a 'toy' problem, however, as a proof of concept for the approach we propose, and for the relative ease with which we may examine figures such as those displayed in this work.

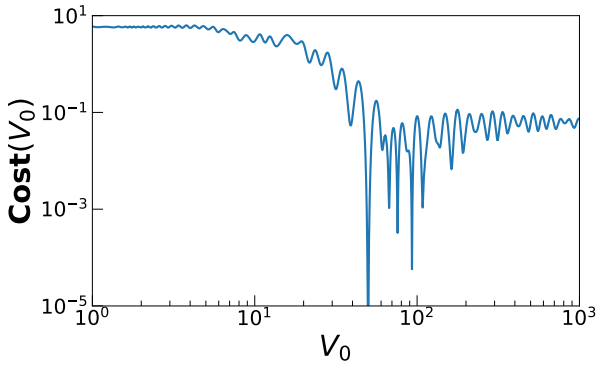

FIG. 3. Cost function dependence on the free parameter V_0. The global minimum is obtained at $V_0 = \tilde{V}_0$ (see table I), but there are many local miminma present, making this a very hard optimization problem.

From Fig. 3 we see that at $V_0 = \tilde{V}_0$ the cost function attains its global minimal value as expected. But, in addition, there are many local minima present which will make it hard for any local optimization algorithm to find the correct value $\tilde{V}_0$ if the initial guess is in the vicinity of a different local minimum. In addition, in the parameter range with small V_0 values, the cost function changes very quickly between large and small values. To understand how gradient and Hessian based optimizers would perform, in Fig. 4 we plot the first and second derivatives of the cost as function of V_0. As the figures

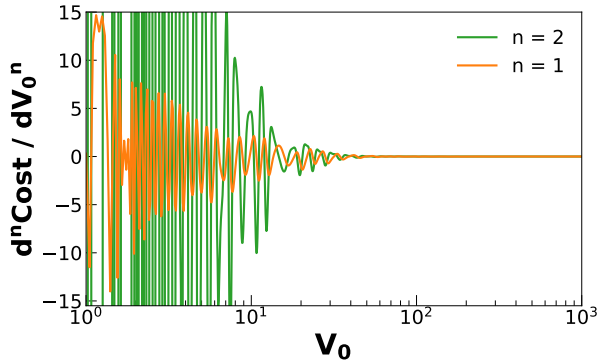

FIG. 4. Gradient and Hessian of cost function with respect to the free parameter $\theta = V_0$. There is a sudden transition from large oscillations to tiny ones near $V_0 = \tilde{V}_0$. For small V_0, gradient based methods will have very large parameter updates in either direction, and for large V_0 the updates will be very small.

show, there are large fluctuations in the derivatives of the cost function at small values of V_0, and many stable local minima at large V_0. On the one hand, if the initial guess is small, gradient-based optimization will change the value of the guess drastically, and on the other hand, if the initial guess is large, the changes will be miniscule. This does not bode well for gradient- (and Hessian-) based optimization. Even with just two neutrino modes, the problem is rather complicated. Hence our choice for global optimization and the wide range of optimization algorithms that we test.

In Fig. 5 we plot the cost function at the end of the iterations for each algorithm as a function of the initial guess for V_0. We sampled uniformly 100 initial values in the range $[0, 400]$. Not surprisingly, gradient and Hessian based algorithms have a final cost value much larger than the rest. To verify that small cost indeed translates to convergence to the global minimum, we also plot the final inferred values of the unknown parameter as function of the initial guess in Fig. 6.

As can be seen from both figures, most gradient-based methods have difficulties in converging to the optimal value, while gradient-free methods perform rather well. In particular, with the Jacobian/Hessian based methods like IPOPT and NTR, the final inferred value of V_0 is positively correlated with the initial guess, suggesting that the optimization procedure simply finds a local minimum close to the initial guess. This outcome agrees with expectations laid out by Figs. 3 and 4. In addition, the combination of a global (MLSL) and local method (MMA) seems successful.

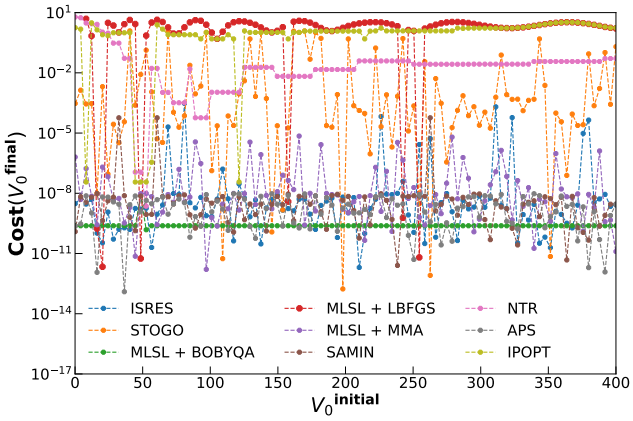

FIG. 5. The cost function at the final inferred values of V_0 as function of the initial guess V_0^{initial} for each optimization algorithm. Ideally this value should be 0, and in practice, the smaller it is the better the optimization.

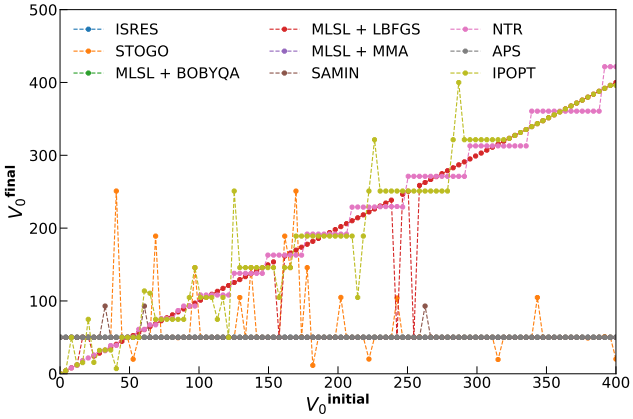

FIG. 6. The final inferred value of V_0 as function of the initial guess V_0^{initial} for each optimization algorithm. For a converged optimization this value should be $\tilde{V}_0 = 50$, independent of the initial guess (see table I).

IV. MATTER POTENTIAL COUPLING AS AN UNSPECIFIED FUNCTION OF POSITION

In the calculations presented in this section, we treat the numerator in Eq. (5), V_0, as a function of the affine parameter r. Note that the simulated "detector data" ($\boldsymbol{u}_D$) used is identical to the data used in the previous section. That is, it was generated using a constant matter potential coefficient $\tilde{V}_0$. Instead of selecting a specific functional form for this dependence, we represent the numerator by a two-layer feed-forward neural network, $V_0(r) = |\mathcal{N}_{\mathcal{A}}(\boldsymbol{\theta}, r)|$. Each layer has five neurons. The first layer has a hyperbolic tangent activation function, and the second is linear; for a total of 16 parameters denoted by $\boldsymbol{\theta}$. The depth, width, and activation functions of the neural architecture are hyper parameters. We choose these hyper parameters strictly, as our goal here is to merely explore whether a neural architecture can provide us with reasonable estimates for V_0 and the architecture chosen can represent a wide range of functions. Since only positive values are physically meaningful, the matter coupling parameter is taken to be the absolute value of the output of the neural architecture. We opted to perform global optimization in the range $\pm 10^3$ for each of the parameters. For each method we sampled uniformly 40 initial parameter sets for each optimization.

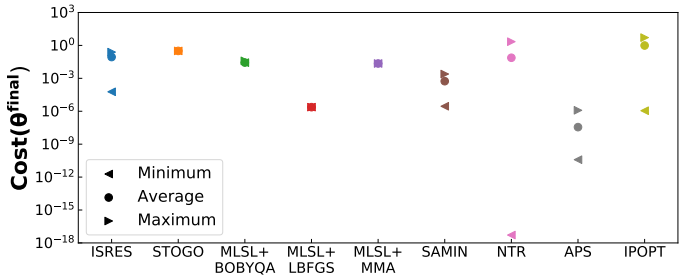

FIG. 7. The minimum, maximum and average values of the cost function at the final values of $\boldsymbol{\theta}$ for each optimization algorithm. Ideally, all these three values should be 0. The spread shows how dependent the optimization algorithms are on the initial guess of the unknown parameters.

In Fig. 7 we show the maximal, minimal and average values of the cost function at the end of the each optimization procedure. APS stands out from the rest: it performs quite well for all initial guesses and provides overall small cost values. While NTR achieves a near-zero cost value for a particular initial guess, its results are quite spread and have a strong dependence on initial conditions. Generally, dependence on the initial guess is to be expected. If a guess happens to be close to the optimal result, one would expect the optimization procedure to produce a final cost value close to zero. On the other hand, if the initial guess is quite far from the optimal value, the optimization might converge to local minimal nearby. An additional complication arises from the possibility of degeneracies as the number of unknown parameters increases. "Degeneracies" here refers to the possibility of multiple solutions $|\mathcal{N}_{\mathcal{A}}(\boldsymbol{\theta}, r)|$ that yield the same values of $P_z(r)$ at the endpoint.

To illustrate the behavior behind this remark, we plot the numerator $|\mathcal{N}_{\mathcal{A}}(\boldsymbol{\theta}, r)|$ for each method using the parameters that produced the smallest cost value in Fig. 8. A priori, we know that a constant function of r, namely $V_0(r) = \tilde{V}_0$, is one possible optimal solution.

As the plot shows, both ISRES and IPOPT converge to a constant function close to $\tilde{V}_0$. APS shows a sharp transition from a region of high density to the optimal $\tilde{V}_0$. A rather interesting result is displayed by NTR, where the matter density profile experiences two sharp transitions and yet the cost value is small ($\approx 10^{-18}$). Other methods that display sharp transitions are MLSL+LFBGS and SAMIN, which perform slightly better than ISRES. In

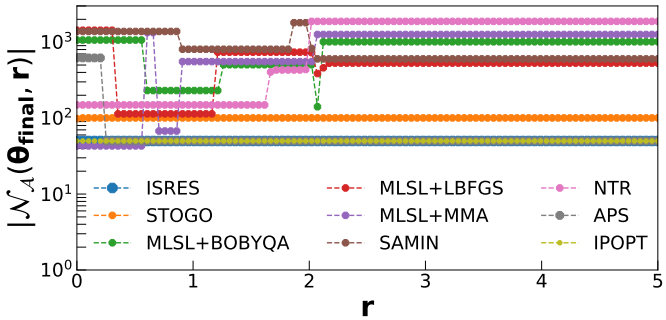

FIG. 8. Optimal matter potential coupling found by each method as function of the affine parameter r. Some algorithms find the coupling to be constant close to the value in table I, while others show sharp matter profile changes.

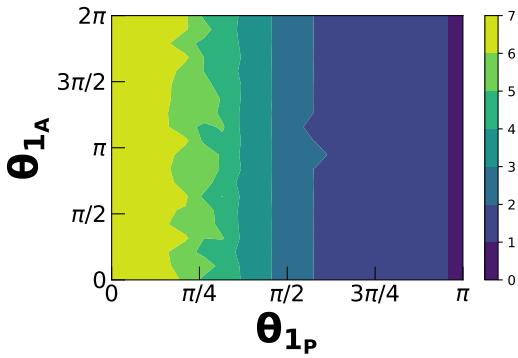

FIG. 9. The cost function, $\mathrm{Cost}(\theta_{1_P}, \theta_{1_A})$, dependence on initial conditions of the first neutrino. There is a relatively strong dependence on the polar angle in flavor space, and weak dependence on the azimuthal angle.

fact, many methods show sharp changes in the matter profile. For the model chosen, the matter density profile is inversely proportional to r^3, so the sharp changes in the numerator amount to small changes in the profile itself. In an actual core-collapse supernova environment, such sharp transitions may represent, for instance, a dense matter outflow in a lower density background, or alternatively the presence of a shock during the supernova explosion. In this manner, allowing for a variable numerator represented by the neural architecture can lead us to discovering other possible matter profiles consistent with the same detector measurement.

V. PARTIALLY UNSPECIFIED INITIAL CONDITIONS

In this section we study the influence of initial conditions on the inference of the neutrino flavor composition at the detector. As an illustration, we assume that Neutrino 1, which in the original setup was taken to be initially in the x flavor, decouples earlier and can potentially oscillate in flavor before the second neutrino is emitted. By assuming the matter and neutrino potentials of table I, and given detector measurements at $R = 5$ in Fig. 2 (i.e., the same simulated detector data as in the previous experiments), we investigate whether we can infer the initial polarization of this neutrino. The initial flavor polarization is normalized, so it can expressed by two free parameters, the azimuthal and polar angles in flavor space; that is, $u_0^{(1)} = \{\cos(\theta_{1_A})\sin(\theta_{1_P}), \sin(\theta_{1_A})\sin(\theta_{1_P}), \sin(\theta_{1_P})\}$, where $\theta_{1_A} \epsilon [0, 2\pi]$, $\theta_{1_P} \epsilon [0, \pi]$. Here, $\boldsymbol{\theta} = \{\theta_{1_A}, \theta_{1_P}\}$ are the unknown parameters to optimize. As we assume coherent evolution, the polarization is normalized throughout the evolution.

In Fig. 9 we plot the cost function dependence on these two parameters. Given that the second neutrino is initially of electron flavor, the optimal value for the first neutrino is to be an x flavor (i.e., $P_z = -1$, or equivalently, $\theta_{1_P} = \pi$), as the figure confirms. In addition, we can see that the polar angle plays a major role in determining the value of the cost function. The figure shows the cost value decrease as the polar angle changes from 0 to π, which is the optimal value. There is also minor dependence on the azimuthal angle for a fixed polar angle. Overall, this is to be expected, as for $\theta_{1_P} = \pi$, the neutrino is x flavor regardless of θ_{1_A}.

We have performed 100 optimization experiments with different initial guesses of $\boldsymbol{\theta}$, from a uniform grid of initial values within the allowed range. The statistics for the final values are summarized in table II. APS and

Method	$(\mu_{\theta_P}, \mu_{\theta_A})/\pi$	$(\sigma_{\theta_P}, \sigma_{\theta_A})/\pi$
ISRES	$(0.996, 1.05)$	$(0.001, 0.57)$
STOGO	$(0.87, 1.52)$	$(\approx 0, \approx 0)$
MLSL + BOBQYA	$(1.0, 1.4)$	$(\approx 0, \approx 0)$
MLSL + LBFGS	$(0.484, 1.0)$	$(0.31, 0.64)$
MLSL + MMA	$(0.997, 1.011)$	$(0.00013, \approx 0)$
SAMIN	$(0.997, 0.996)$	$(0.002, 0.615)$
NTR	$(0.47, 0.98)$	$(0.31, 0.65)$
APS	$(1.0, 1.0)$	$(\approx 0, 0.9)$
IPOPT	$(0.9, 1.1)$	$(0.2, 0.4)$
Optimal Values	$(1, -)$	$(0, -)$

TABLE II. Sample average and standard deviation of the inferred angles in flavor space, for the initial polarization of the first neutrino at the end of the each optimization procedure. The optimal values shown at the end of the table; there is no preferred azimuthal angle.

MLSL + BOBQYA reach the optimal value of θ_{1_P} for most of the initial guesses, as shown by the tiny variances in the table. Many other algorithms (ISRES, MLSL + MMA, SAMIN, IPOPT) converge quite close to the optimal value and have small variances. All methods converge to large values of the azimuthal angle. To understand this behavior, we computed the gradients of the cost function and found one major gradient flow toward $\theta_{1_P} = \pi$, as expected. We also found a rather small flow toward $\theta_{1_A} = 2\pi$. But interestingly, the average in-

ferred values of the azimuthal angle seem to be clustering around π rather than 2π. This is an unexpected result, and a priori hard to guess, as one would need to solve to the flavor evolution equations for all initial conditions to notice this secondary flow.

In Fig. 10 we display the maximal, minimal and average cost value obtained from each method. APS and MLSL + BOBQYA result in small cost values for all initial guesses, in agreement with table II. Most methods do

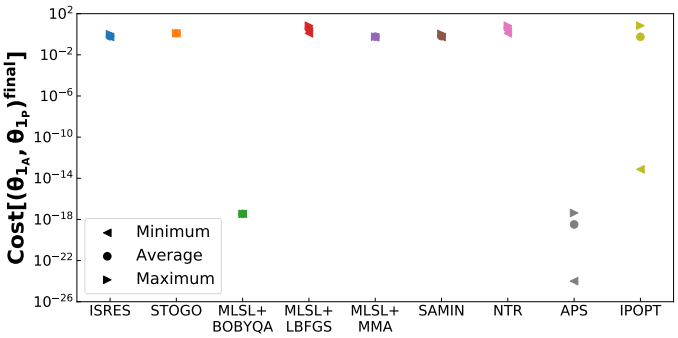

FIG. 10. The minimum, maximum and average values of the cost function at the final values of $\boldsymbol{\theta}$ for each optimization algorithm. Ideally, all these three values should be 0. The spread shows how dependent the optimization algorithms are on the initial guess of the unknown parameters.

not show any spread in the final values of the cost function, apart from APS and IPOPT. In these two cases, for some initial conditions, the methods achieve very low cost function values.

VI. COMPLETELY UNSPECIFIED INITIAL CONDITIONS

In this section, we make no assumptions about the initial polarizations. Instead, given the detector data generated by the parameters in table I, we optimize the cost function for the azimuthal and polar angles in flavor space for both neutrinos. We pick 5 uniformly-spaced values for each of the 4 angles for a total of 625 experiments for each optimization method.

As Fig. 11 shows, most methods converge to sub-optimal solutions. There is convergence for initial conditions close to optimal values, but this does not happen for initial guesses farther away. APS is the only method that performs well for all initial guesses. On the hand, STOGO does not provide a small cost value even for initial guesses close to the optimal one.

As an additional check, in tables III and IV we summarize the statistics for the final values obtained from each method for each neutrino respectively. The sample standard deviation shown is an additional indication of the dependence on the initial guess. Ideally, this deviation should be zero as the methods should converge to the optimal value regardless of the initial guess for the

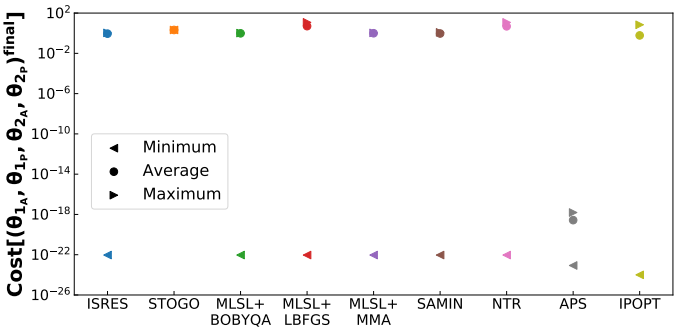

FIG. 11. The minimum, maximum and average values of the cost function at the final values of $\boldsymbol{\theta}$ for each optimization algorithm. Ideally, all these three values should be 0. The spread shows how dependent the optimization algorithms are on the initial guess of the unknown parameters.

unknown parameter, but this is not the case in practice.

Method	$(\mu_{\theta_P^{(1)}}, \mu_{\theta_A^{(1)}})/\pi$	$(\sigma_{\theta_P^{(1)}}, \sigma_{\theta_A^{(1)}})/\pi$
ISRES	$(0.89, 0.98)$	$(0.07, 0.53)$
STOGO	$(0.73, 1.0)$	$(\approx 0, \approx 0)$
MLSL + BOBQYA	$(0.81, 1.88)$	$(0.04, 0.27)$
MLSL + LBFGS	$(0.5, 1.0)$	$(0.35, 0.71)$
MLSL + MMA	$(0.81, 0.14)$	$(0.21, 0.04)$
SAMIN	$(0.87, 0.99)$	$(0.07, 0.59)$
NTR	$(0.5, 1.0)$	$(0.5, 1.0)$
APS	$(1.0, 1.04)$	$(\approx 0, 0.54)$
IPOPT	$(0.45, 0.89)$	$(0.26, 0.57)$
Optimal Values	$(1, -)$	$(0, -)$

TABLE III. Sample average and standard deviation of the inferred angles in flavor space, for the initial polarization of the first neutrino at the end of the each optimization procedure. The optimal values shown at the end of the table; there is no preferred azimuthal angle.

As the tables show, most methods tend toward a large polar angle for the first neutrino and a small value for the second one. In other words, most methods expect the first neutrino to be mostly x flavor and the second to be mostly electron flavor. This result indicates that, for the two neutrino system, the final polarization values can provide information on initial conditions, under the assumption that we know the matter density profile. In addition, we have identified APS as a method that works quite well in understanding the initial flavor composition of the system.

VII. DISCUSSION AND CONCLUSION

In this work, we have combined recent developments in deep learning with data assimilation, to examine what information is contained within a detected neutrino signal regarding complex astrophysical environments such as supernovae.

Method	$(\mu_{\theta_P^{(2)}}, \mu_{\theta_A^{(2)}})/\pi$	$(\sigma_{\theta_P^{(2)}}, \sigma_{\theta_A^{(2)}})/\pi$
ISRES	$(0.17, 0.98)$	$(0.10, 0.53)$
STOGO	$(0.43, 0.99)$	$(\approx 0, \approx 0)$
MLSL + BOBQYA	$(0.15, 0.18)$	$(0.03, 0.26)$
MLSL + LBFGS	$(0.5, 1.0)$	$(0.35, 0.71)$
MLSL + MMA	$(0.21, 0.33)$	$(0.04, 0.22)$
SAMIN	$(0.19, 1.0)$	$(0.07, 0.59)$
NTR	$(0.35, 0.71)$	$(0.35, 0.71)$
APS	$(\approx 0, 1.03)$	$(\approx 0, 0.64)$
IPOPT	$(0.19, 0.89)$	$(0.21, 0.59)$
Optimal Values	$(0, -)$	$(0, -)$

TABLE IV. Sample average and standard deviation of the inferred angles in flavor space, for the initial polarization of the second neutrino at the end of the each optimization procedure. The optimal values shown at the end of the table; there is no preferred azimuthal angle.

By exploiting existing knowledge of solving differential equations within the layers of the neural architecture, we have avoided potential errors associated with domain discretization, while automatically satisfying physical constraints for the problem under investigation. This framework has allowed us to focus on the prediction error (that is, the cost function), and greatly reduced the number of free parameters to be optimized.

In addition, we have tested nine optimization algorithms that cover a wide range of techniques, and we have identified the "Adaptive Particle Swarm Algorithm" (APS) as the best suited one for our purposes. This algorithm, through its four evolutionary stages, is able to move out of local minima, and thus has a high chance of finding a global minimum.

The study conducted here has focused on a small system, primarily as a first testing ground for our framework. We expect the computational complexity to inevitably increase with larger systems, and many degeneracies to be present in the parameter space. Thus, when the particle number is greatly increased, we might combine evolutionary algorithms such as APS for a wide parameter search with a follow-up gradient- and Hessian-based methods such as IPOPT as a secondary search within smaller optimal regions that are found by the first search. We might also need to transition to distributed ordinary differential equation solvers, which can take advantage of computer clusters.

We intend to maintain a level of modeling complexity lower than that of three dimensional supernovae simulations (which take months for a single run to complete), and provide a computational service that is complementary to simulations and can function as a bridge between them and earth-based neutrino detection. A more realistic setting, however, would require more than one affine parameter, for instance both temporal and spatial flavor evolution. In this case, the grid discretization of previous work can be applied to spatial dimensions and combined with the new framework developed here.

ACKNOWLEDGMENTS

We thank Anthony Mezzacappa for the useful discussions. E. And thanks from my wife tooR., A. V. P., and G. M. F. acknowledge the NSF (grant no. PHY-1630782) and the Heising-Simons Foundation (2017-228), and the N3AS NSF Hub. E. A. acknowledges an Institutional Support for Research and Creativity grant from New York Institute of Technology. Additionally, G. M. F acknowledges NSF Grant No. PHY-1914242 and Department of Energy Scientific Discovery through Advanced Computing (SciDAC-4) grant register No. SN60152 (award number de-sc0018297). The work of A. V. P was also supported in part by the U.S. Department of Energy under contract number DE-AC02-76SF00515. The authors acknowledge the Minnesota Supercomputing Institute (MSI) at the University of Minnesota for providing resources that contributed to the research results reported within this paper. See http://www.msi.umn.edu. In addition, HSL, a collection of Fortran codes for large scale scientific computation was used in conjunction with IPOPT. See http://www.hsl.rl.ac.uk.

[1] A Burrows, "Neutrinos from supernova explosions," Annual Review of Nuclear and Particle Science **40**, 181–212 (1990).

[2] Alessandro Mirizzi, Irene Tamborra, Hans-Thomas Janka, Ninetta Saviano, Kate Scholberg, Robert Bollig, Lorenz Hudepohl, and Sovan Chakraborty, "Supernova Neutrinos: Production, Oscillations and Detection," Riv. Nuovo Cim. **39**, 1–112 (2016).

[3] Gabriel Martínez-Pinedo, Tobias Fischer, Karlheinz Langanke, Andreas Lohs, Andre Sieverding, and Meng-Ru Wu, "Neutrinos and their impact on core-collapse supernova nucleosynthesis," in *Handbook of Supernovae*, edited by Athem W. Alsabti and Paul Murdin (Springer International Publishing, Cham, 2017) pp. 1805–1841.

[4] B. Müller, "Neutrino emission as diagnostics of core-collapse supernovae," Annual Review of Nuclear and Particle Science **69**, 253–278 (2019).

[5] Huaiyu Duan, George M. Fuller, J Carlson, and Yong-Zhong Qian, "Simulation of Coherent Non-Linear Neutrino Flavor Transformation in the Supernova Environment. 1. Correlated Neutrino Trajectories," Phys. Rev. D **74**, 105014 (2006), arXiv:astro-ph/0606616.

[6] Huaiyu Duan, George M. Fuller, and Yong-Zhong Qian, "Collective neutrino flavor transformation in supernovae," Phys. Rev. D **74**, 123004 (2006), arXiv:astro-ph/0511275.

[7] Huaiyu Duan, George M. Fuller, J. Carlson, and Yong-Zhong Qian, "Coherent Development of Neutrino Flavor in the Supernova Environment," Phys. Rev. Lett. **97**, 241101 (2006), arXiv:astro-ph/0608050.

[8] Steen Hannestad, Georg G. Raffelt, Gunter Sigl, and Yvonne Y.Y. Wong, "Self-induced conversion in dense neutrino gases: Pendulum in flavour space," Phys. Rev. D **74**, 105010 (2006), [Erratum: Phys.Rev.D 76, 029901 (2007)], arXiv:astro-ph/0608695.

[9] John F. Cherry, J. Carlson, Alexander Friedland, George M. Fuller, and Alexey Vlasenko, "Halo Modification of a Supernova Neutronization Neutrino Burst," Phys. Rev. D **87**, 085037 (2013), arXiv:1302.1159 [astro-ph.HE].

[10] Masamichi Zaizen, John F. Cherry, Tomoya Takiwaki, Shunsaku Horiuchi, Kei Kotake, Hideyuki Umeda, and Takashi Yoshida, "Neutrino halo effect on collective neutrino oscillation in iron core-collapse supernova model of a 9.6 $M_\odot$ star," JCAP **06**, 011 (2020), arXiv:1908.10594 [astro-ph.HE].

[11] Sajad Abbar, Huaiyu Duan, Kohsuke Sumiyoshi, Tomoya Takiwaki, and Maria Cristina Volpe, "On the occurrence of fast neutrino flavor conversions in multidimensional supernova models," Phys. Rev. D **100**, 043004 (2019), arXiv:1812.06883 [astro-ph.HE].

[12] Huaiyu Duan and James P Kneller, "Neutrino flavour transformation in supernovae," J. Phys. G **36**, 113201 (2009), arXiv:0904.0974 [astro-ph.HE].

[13] Huaiyu Duan, George M. Fuller, and Yong-Zhong Qian, "Collective Neutrino Oscillations," Ann. Rev. Nucl. Part. Sci. **60**, 569–594 (2010), arXiv:1001.2799 [hep-ph].

[14] Alessandro Mirizzi, Irene Tamborra, Hans-Thomas Janka, Ninetta Saviano, Kate Scholberg, Robert Bollig, Lorenz Hudepohl, and Sovan Chakraborty, "Supernova Neutrinos: Production, Oscillations and Detection," Riv. Nuovo Cim. **39**, 1–112 (2016), arXiv:1508.00785 [astro-ph.HE].

[15] Sovan Chakraborty, Rasmus Hansen, Ignacio Izaguirre, and Georg Raffelt, "Collective neutrino flavor conversion: Recent developments," Nucl. Phys. B **908**, 366–381 (2016), arXiv:1602.02766 [hep-ph].

[16] Georg G. Raffelt and Alexei Yu. Smirnov, "Self-induced spectral splits in supernova neutrino fluxes," Phys. Rev. D **76**, 081301 (2007), [Erratum: Phys.Rev.D 77, 029903 (2008)], arXiv:0705.1830 [hep-ph].

[17] Georg G. Raffelt and Alexei Yu. Smirnov, "Adiabaticity and spectral splits in collective neutrino transformations," Phys. Rev. D **76**, 125008 (2007), arXiv:0709.4641 [hep-ph].

[18] Basudeb Dasgupta and Amol Dighe, "Collective three-flavor oscillations of supernova neutrinos," Phys. Rev. D **77**, 113002 (2008), arXiv:0712.3798 [hep-ph].

[19] Lucas Johns and George M. Fuller, "Strange mechanics of the neutrino flavor pendulum," Phys. Rev. D **97**, 023020 (2018), arXiv:1709.00518 [hep-ph].

[20] Arka Banerjee, Amol Dighe, and Georg Raffelt, "Linearized flavor-stability analysis of dense neutrino streams," Phys. Rev. D **84**, 053013 (2011), arXiv:1107.2308 [hep-ph].

[21] Georg Raffelt, Srdjan Sarikas, and David de Sousa Seixas, "Axial Symmetry Breaking in Self-Induced Flavor Conversion of Supernova Neutrino Fluxes," Phys. Rev. Lett. **111**, 091101 (2013), [Erratum: Phys.Rev.Lett. 113, 239903 (2014)], arXiv:1305.7140 [hep-ph].

[22] Sovan Chakraborty and Alessandro Mirizzi, "Multi-azimuthal-angle instability for different supernova neutrino fluxes," Phys. Rev. D **90**, 033004 (2014), arXiv:1308.5255 [hep-ph].

[23] Huaiyu Duan and Shashank Shalgar, "Flavor instabilities in the neutrino line model," Phys. Lett. B **747**, 139–143 (2015), arXiv:1412.7097 [hep-ph].

[24] Sajad Abbar and Huaiyu Duan, "Neutrino flavor instabilities in a time-dependent supernova model," Physics Letters B **751**, 43–47 (2015), arXiv:1509.01538 [astro-ph.HE].

[25] S. Chakraborty, R. S. Hansen, I. Izaguirre, and G. G. Raffelt, "Self-induced flavor conversion of supernova neutrinos on small scales," JCAP **2016**, 028 (2016), arXiv:1507.07569 [hep-ph].

[26] R. F. Sawyer, "Multiangle instability in dense neutrino systems," Phys. Rev. D **79**, 105003 (2009), arXiv:0803.4319 [astro-ph].

[27] R. F. Sawyer, "Neutrino Cloud Instabilities Just above the Neutrino Sphere of a Supernova," Phys. Rev. Lett. **116**, 081101 (2016), arXiv:1509.03323 [astro-ph.HE].

[28] Basudeb Dasgupta and Alessandro Mirizzi, "Temporal instability enables neutrino flavor conversions deep inside supernovae," Phys. Rev. D **92**, 125030 (2015), arXiv:1509.03171 [hep-ph].

[29] Ignacio Izaguirre, Georg Raffelt, and Irene Tamborra, "Fast Pairwise Conversion of Supernova Neutrinos: A Dispersion-Relation Approach," Phys. Rev. Lett. **118**, 021101 (2017), arXiv:1610.01612 [hep-ph].

[30] Francesco Capozzi, Basudeb Dasgupta, Eligio Lisi, Antonio Marrone, and Alessandro Mirizzi, "Fast flavor

conversions of supernova neutrinos: Classifying instabilities via dispersion relations," Phys. Rev. D **96**, 043016 (2017), arXiv:1706.03360 [hep-ph].

[31] Basudeb Dasgupta and Manibrata Sen, "Fast neutrino flavor conversion as oscillations in a quartic potential," Phys. Rev. D **97**, 023017 (2018), arXiv:1709.08671 [hep-ph].

[32] Sajad Abbar and Maria Cristina Volpe, "On fast neutrino flavor conversion modes in the nonlinear regime," Physics Letters B **790**, 545–550 (2019), arXiv:1811.04215 [astro-ph.HE].

[33] Shashank Shalgar and Irene Tamborra, "On the Occurrence of Crossings between the Angular Distributions of Electron Neutrinos and Antineutrinos in the Supernova Core," Astrophys. J. **883**, 80 (2019), arXiv:1904.07236 [astro-ph.HE].

[34] Francesco Capozzi, Basudeb Dasgupta, Alessandro Mirizzi, Manibrata Sen, and Günter Sigl, "Collisional Triggering of Fast Flavor Conversions of Supernova Neutrinos," Phys. Rev. Lett. **122**, 091101 (2019), arXiv:1808.06618 [hep-ph].

[35] Milad Delfan Azari, Shoichi Yamada, Taiki Morinaga, Wakana Iwakami, Hirotada Okawa, Hiroki Nagakura, and Kohsuke Sumiyoshi, "Linear analysis of fast-pairwise collective neutrino oscillations in core-collapse supernovae based on the results of Boltzmann simulations," Phys. Rev. D **99**, 103011 (2019), arXiv:1902.07467 [astro-ph.HE].

[36] Hiroki Nagakura, Taiki Morinaga, Chinami Kato, and Shoichi Yamada, "Fast-pairwise Collective Neutrino Oscillations Associated with Asymmetric Neutrino Emissions in Core-collapse Supernovae," Astrophys. J. **886**, 139 (2019), arXiv:1910.04288 [astro-ph.HE].

[37] Lucas Johns, Hiroki Nagakura, George M. Fuller, and Adam Burrows, "Neutrino oscillations in supernovae: angular moments and fast instabilities," Phys. Rev. D **101**, 043009 (2020), arXiv:1910.05682 [hep-ph].

[38] Madhurima Chakraborty and Sovan Chakraborty, "Three flavor neutrino conversions in supernovae: slow & fast instabilities," JCAP **01**, 005 (2020), arXiv:1909.10420 [hep-ph].

[39] Taiki Morinaga and Shoichi Yamada, "Linear stability analysis of collective neutrino oscillations without spurious modes," Phys. Rev. D **97**, 023024 (2018), arXiv:1803.05913 [hep-ph].

[40] John F. Cherry, George M. Fuller, Shunsaku Horiuchi, Kei Kotake, Tomoya Takiwaki, and Tobias Fischer, "Time of Flight and Supernova Progenitor Effects on the Neutrino Halo," (2019), arXiv:1912.11489 [astro-ph.HE].

[41] Babak Abi *et al.* (DUNE), "Deep Underground Neutrino Experiment (DUNE), Far Detector Technical Design Report, Volume II DUNE Physics," (2020), arXiv:2002.03005 [hep-ex].

[42] K. Abe *et al.* (Hyper-Kamiokande), "Hyper-Kamiokande Design Report," (2018), arXiv:1805.04163 [physics.ins-det].

[43] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).

[44] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le, "Learning transferable architectures for scalable image recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2018).

[45] Boukaye Boubacar Traore, Bernard Kamsu-Foguem, and Fana Tangara, "Deep convolution neural network for image recognition," Ecol. Informatics **48**, 257–268 (2018).

[46] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria, "Recent trends in deep learning based natural language processing," CoRR **abs/1708.02709** (2017), 1708.02709.

[47] Daniel W. Otter, Julian R. Medina, and Jugal K. Kalita, "A survey of the usages of deep learning in natural language processing," CoRR **abs/1807.10854** (2018).

[48] Binhua Tang, Zixiang Pan, Kang Yin, and Asif Khateeb, "Recent advances of deep learning in bioinformatics and computational biology," Frontiers in genetics **10**, 214–214 (2019).

[49] Ryad Zemouri, Noureddine Zerhouni, and Daniel Racoceanu, "Deep learning in the biomedical applications: Recent and future status," Applied Sciences **9**, 1526 (2019).

[50] Eve Armstrong, Amol V. Patwardhan, Lucas Johns, Chad T. Kishimoto, Henry D. I. Abarbanel, and George M. Fuller, "An optimization-based approach to calculating neutrino flavor evolution," Phys. Rev. D **96**, 083008 (2017).

[51] Eve Armstrong, Amol V. Patwardhan, Ermal Rrapaj, Sina Fallah Ardizi, and George M. Fuller, "Inference offers a metric to constrain dynamical models of neutrino flavor transformation," Phys. Rev. D **102**, 043013 (2020).

[52] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud, "Neural ordinary differential equations," in *Advances in Neural Information Processing Systems 31*, edited by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Curran Associates, Inc., 2018) pp. 6571–6583.

[53] A. Tarantola, *Inverse Problem Theory and Methods for Model Parameter Estimation*, Other Titles in Applied Mathematics (Society for Industrial and Applied Mathematics, 2005).

[54] BP vanMilligen, Tribaldos, and Jiménez, "Neural network differential equation and plasma equilibrium solver." Physical review letters **75 20**, 3594–3597 (1995).

[55] I. E. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," IEEE Transactions on Neural Networks **9**, 987–1000 (1998).

[56] Jens Berg and Kaj Nyström, "A unified deep artificial neural network approach to partial differential equations in complex geometries," Neurocomputing **317**, 28–41 (2018).

[57] Martin Magill, Faisal Qureshi, and Hendrick de Haan, "Neural networks trained to solve differential equations learn general representations," in *Advances in Neural Information Processing Systems 31*, edited by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Curran Associates, Inc., 2018) pp. 4071–4081.

[58] M. Raissi, P. Perdikaris, and G.E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," Journal of Computational Physics **378**, 686 – 707 (2019).

[59] Ernst Hairer and Gerhard Wanner, "Stiff differential equations solved by radau methods," Journal of Computational and Applied Mathematics **111**, 93 – 111 (1999).

[60] J. Revels, M. Lubin, and T. Papamarkou, "Forward-mode automatic differentiation in Julia," arXiv:1607.07892 [cs.MS] (2016).

[61] Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind, "Automatic differentiation in machine learning: a survey," Journal of Machine Learning Research **18**, 1–43 (2018).

[62] Adrian Sandu, Dacian N. Daescu, and Gregory R. Carmichael, "Direct and adjoint sensitivity analysis of chemical kinetic systems with kpp: Part i—theory and software tools," Atmospheric Environment **37**, 5083 – 5096 (2003).

[63] Yang Cao, Shengtai Li, Linda Petzold, and Radu Serban, "Adjoint sensitivity analysis for differential-algebraic equations: The adjoint dae system and its numerical solution," SIAM Journal on Scientific Computing **24**, 1076–1089 (2003).

[64] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B. Shah, "Julia: A fresh approach to numerical computing," SIAM Review **59**, 65–98 (2017).

[65] Christopher Rackauckas and Q Nie, "Differentialequations.jl – a performant and feature-rich ecosystem for solving differential equations in julia," Journal of Open Research Software **5**, 15 (2017).

[66] Michael Innes, Elliot Saba, Keno Fischer, Dhairya Gandhi, Marco Concetto Rudilosso, Neethu Mariya Joy, Tejan Karmali, Avik Pal, and Viral Shah, "Fashionable modelling with flux," CoRR **abs/1811.01457** (2018).

[67] Christopher Rackauckas, Mike Innes, Yingbo Ma, Jesse Bettencourt, Lyndon White, and Vaibhav Dixit, "Diffeqflux.jl - A julia library for neural differential equations," CoRR **abs/1902.02376** (2019).

[68] Léon Bottou, "Online algorithms and stochastic approximations," in *Online Learning and Neural Networks*, edited by David Saad (Cambridge University Press, Cambridge, UK, 1998).

[69] John Duchi, Elad Hazan, and Yoram Singer, "Adaptive subgradient methods for online learning and stochastic optimization," J. Mach. Learn. Res. **12**, 2121–2159 (2011).

[70] Diederik P Kingma and Jimmy Lei Ba, "Adam: A method for stochastic gradient descent," in *ICLR: International Conference on Learning Representations* (2015).

[71] Robert Malouf, "A comparison of algorithms for maximum entropy parameter estimation," in *Proceedings of the 6th Conference on Natural Language Learning - Volume 20*, COLING-02 (Association for Computational Linguistics, USA, 2002) p. 1–7.

[72] William Goffe, Gary Ferrier, and John Rogers, "Global optimization of statistical functions with simulated annealing," Journal of Econometrics **60**, 65–99 (1994).

[73] Goffe William L., "Simann: A global optimization algorithm using simulated annealing," Studies in Nonlinear Dynamics & Econometrics **1**, 1–9 (1996).

[74] T. P. Runarsson and Xin Yao, "Stochastic ranking for constrained evolutionary optimization," IEEE Transactions on Evolutionary Computation **4**, 284–294 (2000).

[75] T. P. Runarsson and Xin Yao, "Search biases in constrained evolutionary optimization," Trans. Sys. Man Cyber Part C **35**, 233–243 (2005).

[76] John A. Nelder and Roger Mead, "A simplex method for function minimization," Computer Journal **7**, 308–313 (1965).

[77] Z. Zhan, J. Zhang, Y. Li, and H. S. Chung, "Adaptive particle swarm optimization," IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **39**, 1362–1381 (2009).

[78] Andreas Wächter and Lorenz T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," Math. Program. **106**, 25–57 (2006).

[79] Jorge Nocedal and Stephen J. Wright, *Numerical Optimization*, 2nd ed. (Springer, New York, NY, USA, 2006).

[80] Kaj Madsen and Julius Žilinskas, "Parallel branch-and-bound attraction based methods for global optimzation," in *Stochastic and Global Optimization*, edited by Gintautas Dzemyda, Vydūnas Šaltenis, and Antanas Žilinskas (Springer US, Boston, MA, 2002) pp. 175–187.

[81] A. H. G. Rinnooy Kan and G. T. Timmer, "Stochastic global optimization methods part i: Clustering methods," Mathematical Programming **39**, 27–56 (1987).

[82] Sergei Kucherenko and Yury Sytsko, "Application of deterministic low-discrepancy sequences in global optimization," Computational Optimization and Applications **30**, 297–318 (2005).

[83] M. J. D. Powell, "The bobyqa algorithm for bound constrained optimization without derivatives," technical report NA2009/06 (2009).

[84] Krister Svanberg, "A class of globally convergent optimization methods based on conservative convex separable approximations," SIAM Journal on Optimization **12**, 555–573 (2002).

[85] Jorge Nocedal, "Updating quasi-newton matrices with limited storage," Mathematics of Computation **35**, 773–782 (1980).

[86] Dong C. Liu and Jorge Nocedal, "On the limited memory bfgs method for large scale optimization," Mathematical Programming **45**, 503–528 (1989).

[87] David G. Luenberger and Yinyu Ye, *Linear and Nonlinear Programming* (Springer Publishing Company, Incorporated, 2015).

[88] P. A. Vikhar, "Evolutionary algorithms: A critical review and its future prospects," in *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)* (2016) pp. 261–265.

[89] Steven G. Johnson, "The nlopt nonlinear optimization package," ab-initio.mit.edu/nlopt.

[90] Patrick Kofod Mogensen and Asbjørn Nilsen Riseth, "Optim: A mathematical optimization package for Julia," Journal of Open Source Software **3**, 615 (2018).