SAND2022-0633J

# Characterizing Tradeoffs in Memory, Accuracy, and Speed for Chemistry Tabulation Techniques

Elizabeth Armstrong[a,*], John C. Hewson[b], James C. Sutherland[a]

[a] *Department of Chemical Engineering, University of Utah, SLC, UT, USA*
[b] *Fire Sciences and Engineering Department, Sandia National Laboratories, Albuquerque, NM, USA*

**Abstract**

Chemistry tabulation is a common approach in practical simulations of turbulent combustion at engineering scales. Linear interpolants have traditionally been used for accessing precomputed multidimensional tables but suffer from large memory requirements and discontinuous derivatives. Higher-degree interpolants address some of these restrictions but are similarly limited to relatively low-dimensional tabulation. Artificial neural networks (ANNs) can be used to overcome these limitations but cannot guarantee the same accuracy as interpolants and introduce challenges in reproducibility and reliable training. These challenges are enhanced as the physics complexity to be represented within the tabulation increases. In this manuscript, we assess the efficiency, accuracy, and memory requirements of Lagrange polynomials, tensor product B-splines, and ANNs as tabulation strategies. We analyze results in the context of nonadiabatic flamelet modeling where higher dimension counts are necessary. While ANNs do not require structuring of data, providing benefits for complex physics representation, interpolation approaches often rely on some structuring of the table. Interpolation using structured table inputs that are not directly related to the variables transported in a simulation can incur additional query costs. This is demonstrated in the present implementation of heat losses. We show that ANNs, despite being difficult to train and reproduce, can be advantageous for high-dimensional, unstructured datasets relevant to nonadiabatic flamelet models. We also demonstrate that Lagrange polynomials show significant speedup for similar accuracy compared to B-splines.

*Keywords:* chemistry tabulation, artificial neural networks, interpolation, nonadiabatic flamelet

## 1. Introduction

Chemistry tabulation is a common approach utilizing ideas of multi-scale modeling for increasing computational efficiency of chemical kinetics calculations with separation of time-scales between fluid mixing rates and fast chemical rates. This enables practical simulations of challenging applications such as turbulent fires, which involve hundreds of chemical species participating in thousands of reactions at very small timescales relative to the flow field. One

*Corresponding author
*Email addresses:* `elizabeth.armstrong@chemeng.utah.edu` (Elizabeth Armstrong), `jchewso@sandia.gov` (John C. Hewson), `james.sutherland@chemeng.utah.edu` (James C. Sutherland)

*Preprint submitted to Combustion Science and Technology* *January 19, 2022*

representative approach to incorporating detailed chemical kinetics into combustion or fire simulations is the flamelet method where one-dimensional flamelets are precomputed to provide additional information for the flow evolution (Peters, 1984). We use a nonadiabatic flamelet model (Domino et al., 2021) to facilitate our assessment of tradeoffs in memory, accuracy, and evaluation cost for various tabulation strategies.

In traditional chemistry tabulation methods, a range of thermochemical states that can possibly be seen during a simulation are tabulated *a priori* over a reduced set of reaction coordinates corresponding to scalars that are relevant to the flow evolution. This reduced set of coordinates, or some function of them, is then transported in simulation and used to update the original properties through table queries. In general, the procedure for creating property tables involves two steps. First, a reduced set of reaction coordinates $\vec{\eta}$ that result in desirable manifold topology (Armstrong and Sutherland, 2021) must be chosen for parameterizing any properties $\vec{\phi}$ needed by the simulation such as density, temperature, viscosity, source terms, *etc*. Second, a method for storing/querying the data is selected. In order to query these tables, a functional representation, $\mathcal{F}$, must be chosen such that $\vec{\phi} = \mathcal{F}(\vec{\eta})$. A simple and common approach is to use linear interpolation. However, a large number of data points are often needed for constructing linearly interpolated tables of sufficient accuracy; and increasing the dimensionality of these tables can lead to significant memory requirements. Linear interpolants also have discontinuous derivatives which can cause problems with robustness, for example, of projection methods used in low-Mach number combustion.

Smooth derivatives are also needed for scenarios with a mismatch between table input variables $\vec{\eta}$ and those transported in simulation which require root-finding methods such as Newton's method to resolve the inconsistency. For example, if a density weighted variable, $\rho\vec{\eta}$, is transported, a nonlinear solve is required to find $\rho$ and $\vec{\eta}$ given $\rho\vec{\eta}$ from simulation and a table for $\rho = \mathcal{F}(\vec{\eta})$ (McConnell et al., 2020). A difference in table input variables and simulation transport variables is also commonly encountered after structuring unstructured combustion data for faster query times. In these scenarios, a straightforward relation between the transported variables and the structured table inputs does not always exist. This happens, for example, when choosing a heat loss coordinate for tabulating non-adiabatic flamelet data (Domino et al., 2021; Armstrong et al., 2019), where a Newton solve is also performed to determine $\vec{\eta}$.

It is therefore of interest to explore smooth-derivative alternatives to linear interpolation that address memory-accuracy tradeoff concerns. We focus on comparing methods that compute and tabulate states prior to simulation rather than assessing *in situ* tabulation techniques which both construct and query tables during simulation according to the states observed. Higher-degree interpolants guarantee continuous derivatives and can improve accuracy compared to linear interpolants to some degree, but are still limited in practice by memory to five-dimensional tabulation, which can be exceeded in turbulent combustion scenarios. Clustering data for tabulation around areas requiring high accuracy, while reducing the resolution elsewhere in the domain, is one method of reducing table memory. This is commonly done with the mixture fraction grid in the context of combustion. An approach for reducing the memory requirements of B-spline interpolants was proposed by Bode et al. (Bode et al., 2019). These optimized B-splines were shown to maintain accuracy compared to a linear interpolant while reducing the amount of stored data in tabulation over a filtered mixture fraction, mixture fraction variance, and progress variable coordinate. The tradeoff in using interpolants of higher degree is the cost of evaluation, the impact of which may vary depending on the cost breakdown of a simulation. More time will be spent performing table queries when the tables require nonlinear solves, as in the previously mentioned example scenarios. Assessments of Lagrange interpolants for tabulation in combustion, which are simpler than B-splines, are largely missing as well.

Artificial neural networks (ANNs) have also been considered in the context of combustion as a replacement for interpolation due to their ability to provide smooth representations of variables with considerable memory savings. ANNs generally provide a global fit to the data, in contrast to the local interpolation approaches. While the number of parameters needed for a global representation may be greater than those of local interpolation, the large number of local interpolants being stored for accuracy leads to large memory requirements. ANNs were first employed for tabulation of combustion chemistry by Christo et al. (Christo et al., 1996). Since then, many more groups have incorporated ANNs into tabulation for combustion (Blasco et al., 1998, 1999; Chen et al., 2000; Kempf et al., 2005; Ihme et al., 2008, 2009; Sen and Menon, 2009, 2010; Emami and Eshghinejad Fard, 2012; Chatzopoulos and Rigopoulos, 2013; Franke et al., 2017; Owoyele et al., 2020), each with their own training methods and implementation. Even the simplest ANNs have been shown to reduce memory compared to traditional tabulation. Additional memory savings can also be achieved through using a single ANN for predicting multiple outputs as demonstrated by Blasco et al. (Blasco et al., 1998). Accuracy, on the other hand, is more difficult to guarantee since ANNs regress instead of interpolate data. While the large number of trainable ANN parameters can aid in representing complex nonlinear functions, they can also make it difficult to find an optimal network configuration. For example, the ANNs for Chatzopoulos and Rigopoulos (Chatzopoulos and Rigopoulos, 2013) behaved well for major species but had issues in accurately predicting minor species. Additionally, in the study by Kempf et al. (Kempf et al., 2005), the ANNs used for tabulation in LES showed overall good agreement with experimental data, but exhibited difficulties in certain regions of the flame. Optimal artificial neural networks were introduced to remove some of the randomness from optimizing ANNs for each variable and problem (Ihme et al., 2008, 2009), but these can take days to train and cannot guarantee bounds on prediction errors. Even with these advancements in ANN literature there is a lack of standardized training and new methods are still being developed in the search for better accuracy such as through partitioning training domains (Owoyele et al., 2021). The reliability of ANNs is still, therefore, a concern in many combustion scenarios requiring the accuracy standards of interpolation.

ANNs have been used in flamelet tabulation approaches involving a mixture fraction and dissipation rate or progress variable coordinate (Kempf et al., 2005; Ihme et al., 2009; Emami and Eshghinejad Fard, 2012) as well as, most recently, in higher-dimensional tabulation of spray combustion (Owoyele et al., 2020; Honzawa et al., 2021). However, they have not been extended to higher dimensional nonadiabatic flamelet modeling where the data is typically structured for building interpolated tables. The smaller memory footprint of ANNs should show increasing benefit with increasing dimensionality of these flamelet models compared to interpolated tables. The memory savings of ANNs also allow for tabulation in more than the practical five-dimensional limit of interpolated tables, which can facilitate flamelet modeling in scenarios such as turbulent combustion with multiple mixture fractions or dissipation rates. Another unexplored benefit of ANN tabulation for combustion is how they show no particular advantage when structuring data beforehand, as commonly done with interpolated flamelet tables, so ANNs can be built directly for unstructured combustion data. Therefore, ANNs for the thermochemical properties can be constructed over the exact variables being transported in simulation, bypassing the need for a Newton solve. This could lead to additional speedups when using ANNs compared to structured interpolation. This advantage extends generally to structured tabulation scenarios involving a mismatch between table input variables and those transported in simulation, not just in flamelet modeling. It is therefore of interest to compare ANNs to Lagrange polynomial and B-spline interpolants in tabulating chemistry.

This manuscript performs *a priori* comparisons of methods for storing/querying tables while addressing some of the aspects missing from the combustion literature described above. An *a posteriori* assessment is not included as implementations and simulation environments may vary widely in practice. In this paper, we first present details on methods used to query tables in §2. Results for analyzing the accuracy, evaluation cost, and memory requirements of interpolants and ANNs in the context of nonadiabatic flamelet tabulation are then presented in §3.

## 2. Tabulation techniques

Once a reduced set of table coordinates are chosen to represent the original data, a method for storing and querying the data is selected. In this section, we present details on Lagrange polynomial interpolants (§2.1), B-spline interpolants (§2.2), and artificial neural networks (§2.3) for tabulation.

### *2.1. Lagrange polynomial interpolants*

Lagrange polynomial interpolants (Späth, 1995) provide a convenient form for interpolating $p+1$ points with an $p$th-degree polynomial. In one dimension, a Lagrange polynomial interpolant for a set of points $\{(x_k, f(x_k))\}$ is

$$f(x) = \sum_{k=0}^{p} f(x_k) L_k(x), \quad L_k(x) = \prod_{\substack{i=0 \\ i \neq k}}^{p} \frac{x - x_i}{x_k - x_i} \; . \tag{1}$$

The set of points $\{(x_k, f(x_k))\}$ is obtained, for example, from a flamelet simulation or some other method that is computationally cheaper than the simulation of interest. Equation (1) can be extended to an arbitrary number of dimensions, $d$, as well through

$$f(\vec{x}) = \sum_{k_1=0}^{p} \cdots \sum_{k_d=0}^{p} f(\vec{x}_{\vec{k}}) L_{k_1}(x_1) \cdots L_{k_d}(x_d). \tag{2}$$

Analytic derivatives for (2) can be derived, which is important when using Newton's method for table queries as described in §1. In 1D tabulation, for example, the derivatives of a Lagrange polynomial interpolant are

$$f'(x) = \sum_{k=0}^{p} f(x_k) L'_k(x), \quad L'_k(x) = \sum_{\substack{i=0 \\ i \neq k}} \frac{1}{x_k - x_i} \prod_{\substack{j=0 \\ j \neq i,k}}^{p} \frac{x - x_j}{x_k - x_j} \; . \tag{3}$$

We will explore $1 \leq p \leq 4$ which minimizes overshooting errors that can be experienced with high-degree polynomial interpolants (Runge's phenomenon). This is also a typical range of degrees for interpolants and splines used in combustion.

When using Lagrange interpolants for tabulation, the data to be interpolated is tabulated in memory. This means when the interpolant is queried at a new location, the correct neighboring points from the table have to be found in order to evaluate the Lagrange polynomial, $L_k$, and their corresponding dependent variables. For non-uniform grids, this step requires a nonlinear solve for the index of each independent variable bracketing the new location. In this study, the `lower_bound` function from the C++ standard library, employing bisection, is used for determining these indices. In the case of uniform grids, the index search simplifies to a few arithmetic operations based on $(x - x_{\min})/\Delta x$. The Lagrange polynomials themselves can also be simplified for constant grid spacing.

### 2.2. B-spline interpolants

B-splines (Piegl and Tiller, 1997) offer more control over the resulting interpolant shape than Lagrange polynomials, but also have a more complex formulation. A $p$-degree one-dimensional B-spline curve for interpolating a set of $n+1$ points $\{(x_k, f(x_k))\}$ is given by

$$f(x_k) = \sum_{i=0}^{n} N_{i,p}(\bar{u}_k) P_i \tag{4}$$

$$N_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \leq u < u_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \tag{6}$$

where $P_i$ are the control points, $N_{i,p}(u)$ are the B-spline basis functions, $\bar{u}_k$ are specified parameter values, and $u_i$ are elements of a $(n+p+2)$ large knot vector, $U$. In order to use (4), parameter values are assigned to each $x_k$, a knot vector is chosen, and the control points that satisfy the resulting $(n+1) \times (n+1)$ system of linear equations are computed. Training and evaluating B-splines can be costly and challenging for large systems due to the explicit calculations required for control points and the recursive nature of the basis functions. Guidelines for choosing parameters and knots are outlined in (Piegl and Tiller, 1997); for this study, $\bar{u}_k$ are computed using a simple scaling of the independent variables and the knots are computed using an averaging technique so they reflect the distribution of $\bar{u}_k$. The B-spline formulation in (4) can be extended to multiple dimensions using tensor product B-splines as

$$f(\vec{x}_{\vec{k}}) = \sum_{i=0}^{n_1} \dots \sum_{j=0}^{n_d} N_{i,p_1}(\bar{u}_{k_1}) \dots N_{j,p_d}(\bar{u}_{k_d}) P_{i,\dots,j} \tag{7}$$

with more details shown in (Piegl and Tiller, 1997). Analytic derivatives for (7) can also be derived and are useful where Newton's method is needed in querying tables as described in §1.

Continuity of B-splines is determined by the basis functions, which means control points can be modified (by changing the knots, for example) to obtain different curves without sacrificing continuity. B-splines also have local support since the basis functions are nonzero on a limited number of knot subintervals; only $p + 1$ control points per dimension will have an influence on the evaluation. The freedom in choosing $\bar{u}_k$ and $U$ offers more flexibility in creating the interpolants than Lagrange polynomials. However, one must solve a linear system of equations for the B-spline control points and the evaluation of B-splines are not as easily expanded or simplified in implementation as Lagrange interpolants. An advantage of B-splines is that they possess the convex hull property to avoid Runge's phenomenon.

When using B-splines for tabulation, the set of knots and control points for each dependent variable are tabulated in memory as opposed to the independent and dependent variable points. This leads to storing an additional $p + 1$ values per dimension for each dependent variable compared to Lagrange interpolants. Methods exist for reducing memory while maintaining accuracy (Bode et al., 2019) but are not considered here. When the table is queried at a new location, the corresponding parameter for the B-spline is first calculated. For non-uniform grids, a nonlinear solver is then used to find the indices for the neighboring knots in the table in order to evaluate the B-spline basis functions. The method for this index search along with the simplifications for uniform grids are the same as described in §2.1.

### 2.3. *Artificial neural networks*

ANNs (Blasco et al., 1998; Franke et al., 2017) provide smooth representations of dependent variables as nonlinear functions of independent variables. Many parameters can be tuned in a neural network including the number of layers and neurons, the activation functions, and the optimization parameters. The general form for computing the output of the $i^{\text{th}}$ neuron in the $L^{\text{th}}$ layer ($O_i^L$) is

$$O_i^L = \mathcal{G}_L\left(\sum_{j=1}^{N_L} \omega_{ij}^L O_j^{L-1} + \beta_i^L\right), \quad i = 1, \ldots, N_L \tag{8}$$

where $\omega_{ij}^L$ is the weight for connecting the $j^{\text{th}}$ neuron of the $(L-1)^{\text{th}}$ layer and the $i^{\text{th}}$ neuron of the $L^{\text{th}}$ layer, $\beta_i^L$ is the bias neuron, $N_L$ is the total number of neurons in the $L^{\text{th}}$ layer, and $\mathcal{G}_L$ is the activation function for layer $L$. Training an ANN finds the set of weights and biases that minimizes a specified error measure. The large number of possible parameters enable ANNs to represent nonlinear data well, but also make it non-trivial to find optimal configurations. In this sense, ANNs require the greatest amount of upfront effort, at least for global fits where the complexity of the network is non-trivial. We also center and scale inputs and outputs between -1 and 1 for ANN training, increasing efficiency of ANN training (Christo et al., 1996; Ihme et al., 2008; Sen and Menon, 2010; Emami and Eshghinejad Fard, 2012).

When using an ANN for tabulation, only the network architecture needs to be saved after training the coefficients. This enables efficient storage of ANNs for higher than five-dimensional tabulation, unlike interpolated tables. In contrast to the Lagrange polynomial and B-spline interpolants, an index lookup is not needed before querying the model at a new location; instead, appropriate centering/scalings are applied to the table inputs and outputs.

In this paper, ANNs are built using the open-source library Keras (Chollet et al., 2015). They are trained with the Adam (Kingma and Ba, 2014) optimization algorithm at a learning rate of $10^{-3}$. The network weights are randomly initialized from a Glorot uniform distribution (Glorot and Bengio, 2010) and the model is trained to minimize a mean squared error loss function computed on predictions for a given dependent variable of a training dataset over $2{\cdot}10^5$ epochs. Each epoch handles the full training dataset without shuffling or splitting into smaller batches. The model with the lowest error evaluated on a testing dataset during training is saved for tabulation. Controlling and minimizing the error during training over multiple parameter sets is challenging and a significant drawback of ANNs. Many parameters used here are either common choices from literature or are chosen after simple explorations to reduce training errors with our data. More details on ANN architecture are found in the following section.

Since we only consider simple feedforward ANNs, we opted for a simple implementation of ANN evaluation in C++ rather than using existing libraries, such as TensorFlow (Abadi et al., 2016) or the header-only frugally-deep (Hermann, 2021), that more generally support training/evaluating ANNs on various platforms. Our implementation supports five activation functions and an arbitrary number of layers and neurons per layer. This allows for straightforward evaluation using pointers to standard vectors of inputs rather than requiring conversions to tensor types that other existing C++ libraries use. This design performs the operations in (8), including centering/scaling inputs and outputs, using coefficients trained in python, minimizing overhead associated with supporting generality and data-type conversions which can pollute timings from existing C++ libraries. We acknowledge that our implementation sacrifices generality for speed and does not consider other platforms such as tensor or graphics processing units.

## 3. Results

We will now assess accuracy (§3.1), evaluation cost (§3.2), and memory (§3.3) of interpolants and ANNs in the context of nonadiabatic flamelet tabulation. The three-dimensional laminar nonadiabatic flamelet data considered in this section was generated using Spitfire (Hansen et al., 2020) for ethylene (Luo et al., 2012) in air, both at 298K and one atm. The generation and processing of flamelet libraries follows the description in Domino et al. (Domino et al., 2021). The data uses a mixture fraction ($Z$) grid of 321 points clustered around stoichiometric, a stoichiometric dissipation rate ($\chi_{\mathrm{st}}$) grid of 41 points uniformly spaced on a logarithmic scale over $[10^{-3}, 10^{2}]$ $\mathrm{s}^{-1}$, and a stoichiometric enthalpy defect ($\gamma_{\mathrm{st}}$) grid of 121 points uniformly spaced over [0, -2.52] MJ (representing heat loss). This data is subsampled into a training dataset, taking every other point in each dimension starting with the boundary values to leave 161, 21, and 61 points in the $Z$, $\chi_{\mathrm{st}}$, and $\gamma_{\mathrm{st}}$ coordinates, respectively. This provides the data to be interpolated using Lagrange polynomials or B-splines and regressed using ANNs, and is referred to as the laminar “training” data. A complimentary laminar “testing” dataset was similarly composed by subsampling the simulated data, taking every other point in each dimension starting with the first point inside the boundary to leave 160, 20, and 60 points in the $Z$, $\chi_{\mathrm{st}}$, and $\gamma_{\mathrm{st}}$ coordinates, respectively. We additionally consider $\gamma_{\mathrm{st}} \geq -1$ MJ for the laminar testing space, which is representative of heat loss seen for nonadiabatic laminar flames of interest to our three-dimensional table.

We additionally consider four-dimensional turbulent flamelet data constructed by Favre filtering the laminar datasets through convolution of each unfiltered property with the joint probability density function (PDF) of the table inputs. A clipped-Gaussian-PDF was used for $Z$, which is constructed with a mean and variance; and a Dirac delta PDF was used for $\chi$ and $\gamma$. The final turbulent "training" dataset has 9 points for the mixture fraction variance logarithmically spaced from 0.001 to 1 and 81 points for the mixture fraction mean subsampled including the boundaries with every other value from the 161 mixture fraction values in the laminar training dataset; the remaining dimensions are the same size as the laminar version. A turbulent "testing" dataset is similarly constructed from the laminar testing dataset, but does not include the boundary mixture fraction variance values seen in the turbulent training dataset.

Lagrange polynomial interpolants of degree $p = 1 \ldots 4$ will be represented as LP1, LP2, LP3, and LP4 while B-spline interpolants will be represented as BS1, BS2, BS3, and BS4. These interpolants tabulate the laminar flamelet data over $(Z, \ln(\chi_{\mathrm{st}}), \gamma_{\mathrm{st}})$ to maintain a structured format. In comparison, the ANNs tabulate over the unstructured $(Z, \ln(\chi), \gamma)$ coordinates rather than their stoichiometric values, with both $\chi$ and $\gamma$ showing variation over $Z$. This is because ANNs do not need structured data for training and $(Z, \ln(\chi), \gamma)$ more directly corresponds to the variables transported in simulation. While these variables are not the only option for nonadiabatic tabulation, they are common and enable a comparison among tabulation techniques in the more general context of interpolated table input variables not matching those transported in simulation. The turbulent data is similarly tabulated over the filtered coordinates with an additional input of the mixture fraction variance.

An assortment of architectures are tested to provide sample ranges on accuracy, speed, and memory of ANNs, which can be constructed arbitrarily in practice. The ANNs are trained following §2.3. Hyperbolic tangent activation functions are used for all ANN hidden layers, being a traditional choice in the combustion examples of §1. Output layers have a single neuron with linear activation for predicting a single property. The number of hidden layers is kept fairly low as dramatically increasing the ANN depth can increase chances of overfitting. The number of neurons per hidden layer is limited to twenty for similar reasons. If overfitting was present, it would be seen in errors from predicting the "testing" datasets described above. ANNs are represented as A followed by the number of hidden layers and number of neurons per layer separated with a dash. For example, A2-5 represents an ANN with 2 hidden layers and 5 neurons per layer.

### *3.1. Accuracy*

We first assess accuracy of Lagrange and B-spline interpolants and ANN tabulation strategies in reconstructing the laminar testing data described above. Specifically, we compute the relative difference between table predictions and the testing data for density. The norm of this relative error is computed for all $\gamma_{\mathrm{st}} \geq -1$ MJ, which is representative of heat loss seen for nonadiabatic laminar flames of interest to our three-dimensional table as mentioned earlier. These errors are shown in Figure 1, computed for evaluating interpolated tables at $(Z, \chi_{\mathrm{st}}, \gamma_{\mathrm{st}})$ of the laminar testing data and evaluating ANNs at $(Z, \chi, \gamma)$ of the laminar testing data. An $\mathrm{L}_\infty$ norm (solid bars) and $\mathrm{L}_1$ norm scaled by the number of observations (hatched bars) of the relative errors are shown in Figure 1.

Figure 1 shows an exponential decrease in interpolant error with increasing polynomial degree, as expected. There is no difference in terms of accuracy between LP1 and BS1, but for $p > 1$, B-splines have the lowest error for a given degree and dimensionality. An advantage of using Lagrange interpolants is they can be easier to construct than B-splines for similar errors, though B-splines offer more flexibility in their construction.

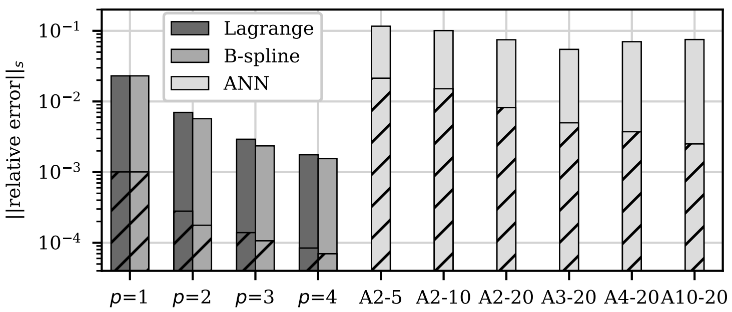
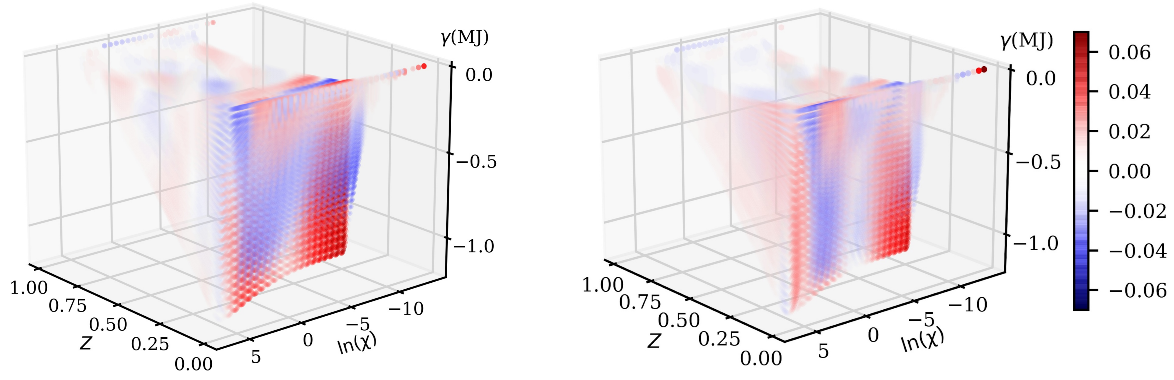



Figure 1: Relative error norms for predicting density compared to the laminar testing dataset. The solid bars represent the $L_\infty$ error norm and the hatched bars represent the $L_1$ error norm scaled by the number of observations.

While the interpolants guarantee no error for returning the points seen in training and so should accurately predict neighboring points if the data is well-resolved, the ANNs as regressors do not have this guarantee. Figure 1 shows how increasing the complexity of an ANN, while to some degree aids in capturing nonlinearities of the data, does not guarantee better accuracy. We see the scaled $L_1$ norm decreasing with increasing ANN complexity, but the $L_\infty$ norm starts to increase as more layers are added after A3-20. As the ANN complexity increases further, the chances of overfitting increase. More importantly, the maximum ANN errors in Figure 1 are > 5% while $p > 1$ interpolant errors are < 0.7%. Even the lowest ANN scaled $L_1$ norm of 0.2% is higher than that of linear interpolation at 0.1%.

The ANN errors can also be very sensitive to the training procedure and parameters. For example, initializing the ANN weights differently can result in a factor of 2-10 higher or lower error in some cases. This sensitivity makes it difficult to provide general claims on ANN accuracy and is why much of the combustion literature iterates on finding acceptable ANNs rather than adopting a single architecture. This is also one of the reasons why reproducibility of ANNs from literature is so difficult. The information needed to reproduce an ANN is largely missing from combustion literature demonstrating ANNs in simulation. This difficulty in reproducing ANNs is another unmentioned drawback of using them.

We also assess where the errors lie in the table input space since, in practice, the input space covered during a simulation may not extend over the entire input space used for training. Figure 2 shows the three-dimensional input space of the laminar testing data colored by the relative error for predicting density using A3-20 and A4-20. While the errors for A3-20 cover a slightly larger area of the testing data input space than for A4-20, the highest errors come from A4-20 which show up in the slightly larger $L_\infty$ norm in Figure 1. The largest errors for A4-20 happen at $\gamma$ near adiabatic with small values of $Z$ and $\chi$. In comparison, the dominant errors for A3-20 show up across a range of $\gamma$ for moderate $\chi$ and small $Z$. A4-20 also shows errors in this region over a smaller range of $\chi$. Therefore, these ANN errors would not be problematic if these regions were never accessed. Unfortunately, small $Z$ and $\chi$ as well as $\gamma$ near zero are commonly accessed in simulation. We therefore expect A3-20 to outperform A4-20 in simulation, but the errors in comparison to interpolants still might not be acceptable depending on the context. A10-20 shows similar trends to A4-20 in locations for the largest error suggesting that accuracy in these regions is being sacrificed to lower the average error over the laminar testing input space. This highlights one of the difficulties in using globally-trained ANNs to represent quantities with complex local features.




(a) A3-20

(b) A4-20

Figure 2: Relative errors of Figure 1 over the input space.

Unlike interpolants, ANNs also lack guarantees of increased accuracy with increased resolution of training data. Figure 3 demonstrates trends in the error computed for evaluating tables built with the laminar training dataset subsampled in the mixture fraction coordinate on the original laminar testing dataset. The largest grid size for $Z$ is the 161-point grid of the original laminar training dataset, with relative errors equivalent to those shown in Figure 1. This training data is then subsampled, taking every $2^{\text{nd}}$, $4^{\text{th}}$, $8^{\text{th}}$, and $16^{\text{th}}$ data point in $Z$ (keeping the boundary points) to create training datasets with 81 points, 41 points, 21 points, and 11 points in $Z$, respectively.

We see in Figure 3 how adding resolution to the laminar training data for the Lagrange interpolants decreases the error in predicting points within the table bounds, as expected. Adding more information for the interpolants will continue to decrease the error up to machine precision. This additional accuracy comes with a cost of increasing the memory footprint. We also see the errors for LP3 decreasing faster than LP1, both at expected order once sufficient resolution is reached. In contrast, increasing resolution of data for training the ANNs will only aid in decreasing the error until the dependent variable behavior is well enough resolved for the architecture to capture it as well as possible. Adding further resolution will not significantly change the ANN results. We see both A2-5 and A3-20 performing as well as the LP3 interpolant for smaller grid sizes, and the increased resolution of the training data has no effect on the memory footprints of the ANNs unlike the interpolants. However, the ANN errors start to stagnate after the 81-point

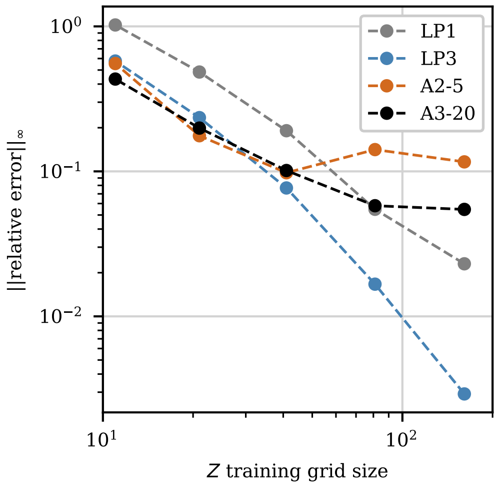



Figure 3: Relative $L_\infty$ error norm computed for the laminar testing data over subsampling the $Z$ grid of the laminar training data.

grid while both interpolant errors continue to decrease. Once this happens, a change of architecture is needed to attempt a better representation of the nonlinearities in the data, as seen in the improved errors of A3-20 compared to A2-5. However, as mentioned before, increasing the ANN architecture does not necessarily improve the predictions; training parameters can also impact results significantly. It is much harder to control errors in ANN predictions compared to interpolants.

Similar trends are seen from building ANNs with the turbulent training and testing data described in §3. Although the scaled $L_1$ error norm slightly decreases in moving from A3-20 to A4-20 to A10-20, the $L_\infty$ error norm increases. The scaled $L_1$ error norm for A3-20 is around 4% which is close to that seen with an LP3 interpolated table, while the $L_\infty$ error norm is closer to 20%. The errors seen for A4-20 are very similar. The turbulent interpolant errors show a larger increase over the laminar table errors in Figure 1 than the turbulent ANNs show over the laminar ANNs. However, much of the sensitivity and unreliability in ANN training is exacerbated by increasing dimensionality.

Therefore, controlling ANN accuracy is much more difficult in comparison to interpolants. Guarantees about convergence of error with interpolant degree and resolution are not offered by ANNs. While increasing the complexity of an ANN can increase accuracy on average, we have demonstrated how the maximum errors seen over the table input space also increase. For much larger ANN architectures, average errors will also increase as overfitting becomes an issue. These ANN errors are also very sensitive to training parameters, including random initializations. Iterating on training procedures/parameters is one avenue for improving ANN accuracy, but is tedious and difficult to know what will significantly improve errors. Newer approaches attempt to gain accuracy by partitioning the domain and training multiple specialized ANNs for each subdomain (Owoyele et al., 2021). Some, such as partition of unity networks (Lee et al., 2021), also address ANN accuracy concerns by incorporating polynomial representations with trainable coefficients. These are possible avenues for leveraging the advantages of ANNs while addressing the shortcomings associated with training and accuracy. It is not surprising that applying ANNs more locally results in improved accuracy over globally-fitted ANNs in representing functions with strong local nonlinearities. As mentioned earlier, there is still a lack of standard training methods for ANNs in combustion. More work is certainly required to reliably produce accurate ANNs, especially for higher-dimensional datasets such as the nonadiabatic flamelet datasets demonstrated here.

### 3.2. *Evaluation cost*

We will now assess table query performance. In order to make more general conclusions about querying tables in multiple dimensions, we start our analysis with synthetic datasets before moving to details of querying the nonadiabatic flamelet tables which involve a conversion between simulation and table input variables for interpolants. The specifics of a dependent variable do not impact the time for directly querying a table, which only depends on the independent variable grids. The synthetic datasets are tabulated using a 50-point independent variable grid with the same range, clustering point, and clustering factor as $Z$ of the laminar nonadiabatic flamelet table. This grid is repeated for each independent variable to create datasets for 1D to 5D tabulation. All reported timings, performed with a 2.4 GHz Skylake Intel processor, are an average of five trials for querying $10^6$ random points within the table bounds and dividing by $10^6$ to compute the time for a single query.

Figure 4 shows timings for querying Lagrange (right columns labeled LP) and B-spline (left columns labeled BS) interpolated tables of the synthetic datasets. The time spent performing the interpolant index lookups, outlined in §2.1 and §2.2, is labeled IL. This plot also shows markers for LP1 and LP3 that have been optimized (labeled opt LP1 and opt LP3) by unrolling the coded loops corresponding to the summation and product shown in (1). Implementing a similar optimization for B-splines is not as straightforward and not considered.

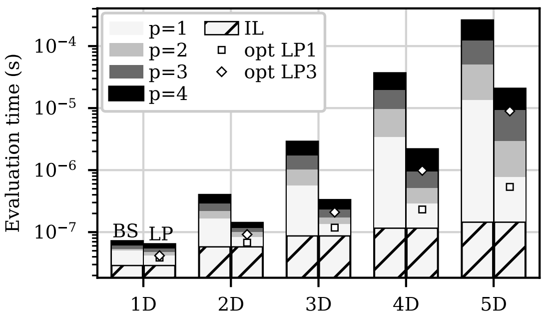



Figure 4: Evaluation time for a single query using Lagrange (LP, right columns) and B-spline (BS, left columns) interpolants for tabulating synthetic datasets (§3.2). IL refers to the interpolant index lookup time. opt LP1 and opt LP3 are optimized implementations of LP1 and LP3.

We see in Figure 4 that B-splines are more expensive than Lagrange interpolants of a given degree and dimensionality. The B-splines require more than double the evaluation time of Lagrange polynomials in 2D tabulation and become around 13-17 times more expensive in 5D tabulation for increasing degree. The cost of the B-splines increases by an order of magnitude more than the Lagrange polynomials when increasing dimensionality from one to five. The higher evaluation cost of B-splines, especially in higher dimensions, is due to their hierarchical nature and difficulty to optimize in implementation. Optimizing the Lagrange polynomial implementation has a greater effect on lower than higher dimensional timings. In 1D tabulation, optimization leads to a factor of 1.3 and 1.1 reduction in the LP3 and LP1 times, respectively. In 4D tabulation, any optimization for LP3 becomes negligible, possibly due to the length of instructions being unrolled for the compiler. However, additional speedup for Lagrange interpolants is achievable with minimal effort, at least for low $p$ and dimensionality, which is an advantage over B-splines.

While the cost of both interpolant basis functions scales exponentially with dimensions, the IL time scales linearly with dimensionality of structured tables. The required iterations for convergence on the index depend on the size and nonlinearity of the grid. As the grid becomes more uniformly spaced or becomes clustered around the middle of the domain, the bisection algorithm will require the same time to compute an index for querying both sides of the domain. For our synthetic dataset, the algorithm favors queries made on the right half of the domain which has fewer points than the left half. The percentage of time required for the index lookup is larger for smaller $p$ and dimensionality. For example, this percentage decreases from 69% in 1D tabulation to 19% in 5D tabulation with LP1 and decreases from 54% in 1D tabulation to 1.5% in 5D tabulation with LP3. These percentages are smaller for B-splines as they are more expensive than Lagrange polynomials.

Since the IL times depend on the nonlinearity of the grid and can be optimized for uniform grid spacing, it can be advantageous to use uniform grids when possible. For example, rescaling a table input, such as log-scaling $\chi_{\text{st}}$, can maintain accuracy on a uniform grid. Changing the 1D synthetic data grid to use uniform grid spacing decreases the IL time by a factor of 3.7. A uniform grid can also be created by increasing the number of stored data points, maintaining accuracy at the cost of increasing memory (which is not always feasible). For example, increasing the number of grid points to 168 uniformly spaced in the 1D synthetic table reduces the LP3 time by 58% at a cost of increasing the number of stored values by a factor of 3.36 to maintain the smallest grid spacing. This tradeoff becomes less favorable for higher dimensions. If we did this for each dimension in our 3D table, we could potentially see a decrease in the evaluation time of 39%, but the number of stored values would increase by a factor of 38 higher compared to the nonuniform table. Only changing one of the dimensions of the 3D table would increase the memory by a factor of 3.36 for only a 10% change in evaluation time. This tradeoff is important to consider, especially for small dimensionality, when a significant portion of the total time is spent performing the index lookup. Further optimizations for the Lagrange polynomials could also be made to simplify (1) for a constant grid spacing, but were not explored here.

Next, we compare timings for evaluating the ANNs in Figure 5. The time to center/scale the query point and ANN prediction, as described in §2.3, is included in these ANN results. We see that the ANN timings for a given architecture are relatively constant over dimensionality. This is because increasing dimensionality adds an input neuron, which only affects the operations done for the first hidden layer. There is a slight increase in the evaluation time moving from 1D to 5D tabulation for the ANNs, but it is insignificant next to the range of interpolant timings. Doubling the neurons in each ANN layer roughly doubled the evaluation time (comparing A2-10 to A2-5), which suggests the query time is dominated by the activation function for simple architectures. Adding a layer to the two- and three-layer (twenty neuron) ANNs for 3D tabulation roughly increased evaluation time by 65% and 40%, respectively, also suggesting high impact of the activation function with slightly more influence from network size.

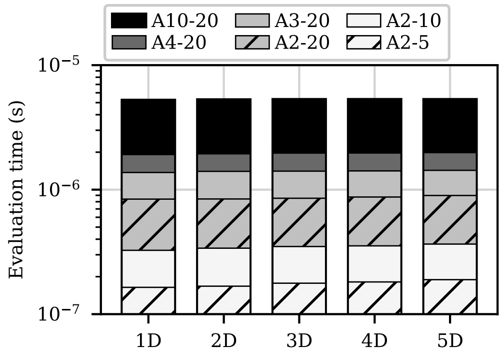



Figure 5: Evaluation time for a single query using ANNs trained on synthetic datasets (§3.2).

It is interesting to see the strong influence of architecture on timings compared to large, relatively similar errors across ANNs in Figure 1. Balancing gains in ANN accuracy for larger architectures with slower query times is important to consider, especially in higher dimensions as accurate ANN training becomes more difficult. We note that timings collected with existing ANN software (e.g., (Abadi et al., 2016; Hermann, 2021)) did not show much sensitivity to architecture and were a factor of 10-100 slower, suggesting overhead calculations dominated query times. For example, A4-20 and A2-5 timings became nearly indistinguishable when using available software. This emphasizes why we used our own implementation for querying ANNs, despite limiting the generality.

For 3D, 4D, and 5D tabulation, A2-5, A2-20, and A10-20 become faster than LP3, respectively, comparing Figures 4 and 5. All are faster than BS3 for 4D tabulation, and A3-20 is faster than BS3 for 3D tabulation. This suggests a potential speedup for using ANNs over interpolants in 3D tabulation, at least for simple architectures, with increasing significance in higher dimensions. This is especially true compared to B-spline interpolants.

We now reframe the synthetic data tabulation results in the context of the nonadiabatic flamelet tabulation. As previously mentioned, structured tabulation of nonadiabatic flamelet data can lead to differences between the variables transported in simulation and those required for table queries. For some variables, the inconsistency is resolved through algebraic manipulation, but for others, such as the heat loss dimension, a Newton solve is required. We again note that this scenario is not specific to nonadiabatic flamelet tabulation and its effect on the query time analyses can be of interest to more general scenarios such as those mentioned in §1.

In structuring the laminar flamelet data, we chose tabulation over $(Z, \ln(\chi_{\mathrm{st}}), \gamma_{\mathrm{st}})$, but the variables being transported are $(Z, \chi, \gamma)$ [1]. We used a presumed form (Peters, 2000) for $\chi$ over $Z$ in generating the flamelet data, so this function

[1] Total and adiabatic enthalpies are transported from which $\gamma$ is computed.

can be used to compute $\chi_{\mathrm{st}}$ for querying the table. There is no presumed form of $\gamma$, however, which is why a nonlinear solve is needed to query the heat loss dimension. We use Newton's method for this nonlinear solve, which computes the $\gamma_{\mathrm{st}}$ for which the $\gamma$ from simulation matches a tabulated $\gamma$. Therefore, evaluating an interpolated table for a property of interest involves first querying a $\gamma$-table and its derivative each Newton iteration followed by querying the property of interest table at the computed $(Z, \ln(\chi_{\mathrm{st}}), \gamma_{\mathrm{st}})$. This increases the time associated with evaluating interpolated tables for properties of interest compared to Figure 4. In comparison, since ANNs show no advantage to structuring data beforehand, we constructed the ANNs using $(Z, \ln(\chi), \gamma)$ so no conversions are needed aside from logarithmically scaling $\chi$.

We first assess the evaluation time for an interpolated table of density as our property of interest. As explained above, this first involves querying a $\gamma$-table and its derivative each Newton iteration to correctly determine $\gamma_{\mathrm{st}}$. An LP3 interpolant and derivative query for the laminar flamelet table are both around $0.21\mu$s. Evaluated over the laminar testing input space, around three Newton iterations are needed on average to converge to the correct $\gamma_{\mathrm{st}}$. We then query the tabulated density at the converged $(Z, \ln(\chi_{\mathrm{st}}), \gamma_{\mathrm{st}})$. We ignore the time to algebraically convert $\chi$ to $\chi_{\mathrm{st}}$ which should be negligible in comparison to the nonlinear solve for $\gamma_{\mathrm{st}}$. This leads to a total query time of approximately $1.47\mu$s. Therefore, while the A3-20 ANN is more than six times slower than a single LP3 query in 3D tabulation, it is 4% faster to evaluate in the context of the nonadiabatic flamelet tabulation. This is due to the additional queries needed by interpolated tables to resolve the mismatch between table input and transport variables. The A4-20 ANN for 3D tabulation is still around 34% slower to query than the interpolated laminar flamelet table.

An LP3 interpolant and derivative query for the turbulent flamelet table are around $0.59\mu$s and $0.64\mu$s, respectively. Around three Newton iterations are also needed on average to correctly find $\gamma_{\mathrm{st}}$ over the turbulent testing space. This leads to a total query time of approximately $4.27\mu$s, not including the additional time to compute $\tilde{\chi}_{\mathrm{st}}$. Therefore, while the A10-20 ANN is still slower to evaluate for 4D tabulation, the A4-20 ANN becomes more than two times faster to evaluate than an LP3-interpolated table in the context of turbulent nonadiabatic flamelet tabulation.

Larger-architecture ANNs can become advantageous in the context of evaluation times for nonadiabatic flamelet tabulation where they might not in the context of general 3D tabulation. While only A2-5 was faster than LP3 for the synthetic 3D table evaluation, A3-20 is a faster option in the context of nonadiabatic flamelet tabulation. Similarly, larger than two-layered ANNs were slower than LP3 for the synthetic 4D table evaluation, but the four-layered ANN was faster for turbulent nonadiabatic flamelet tabulation. In general, evaluation times for larger ANNs will be favorable compared to interpolated tables requiring nonlinear solves, especially as dimensionality increases and/or more table input variables need to be determined with the nonlinear solve.

While relatively large ANNs may be faster to evaluate than interpolated tables requiring nonlinear solves, it is important to consider the tradeoffs with accuracy. Figure 6 shows the errors from Figure 1 for the 3D laminar flamelet tabulation plotted against evaluation times from Figures 4 and 5 for 3D tabulation strategies. Note that the interpolant timings act as an approximation for evaluating the laminar flamelet table since the index lookup times for the synthetic data are not equivalent to those for the flamelet table. We see the large increase in evaluation times over the ANN architectures for a relatively small change in the $L_\infty$ norm, as previously mentioned. We also see the much larger evaluation times of B-splines compared to Lagrange polynomials for a similar accuracy. Also included in Figure 6 is the evaluation time for LP3 involving a nonlinear solve. As described above, the larger ANN architectures become faster to evaluate than interpolated tables requiring nonlinear solves, whereas Lagrange interpolants are otherwise the fastest option. However, while these ANN architectures become faster to evaluate, they still have much larger errors than interpolants. Accuracy is still the main concern when using ANNs for tabulation, as discussed in §3.1.

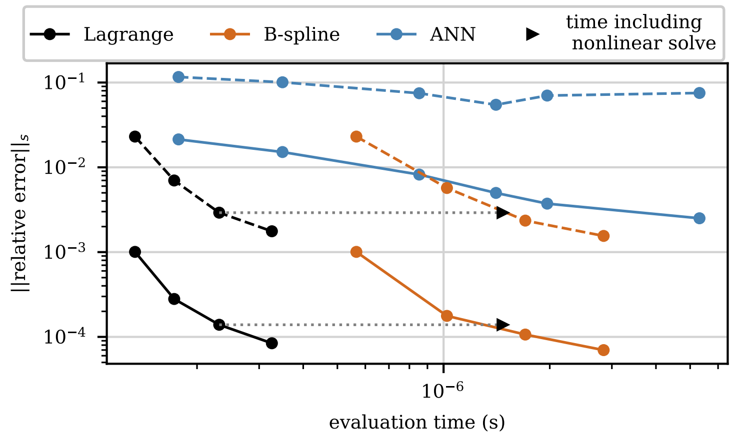



Figure 6: Relative errors of Figure 1 plotted over the evaluation times from Figures 4 and 5 for 3D tabulation. The dashed lines represent the $L_\infty$ error norm and the solid lines represent the $L_1$ error norm scaled by the number of observations. Also included is the approximate time to evaluate LP3 when a nonlinear solve is required. Note that the interpolant timings act as an approximation for evaluating the laminar flamelet table since the index lookup times for the synthetic data are not equivalent to those for the flamelet table.

### 3.3. *Memory*

We analyze theoretical memory as measurements depend on implementations and tools used. The memory of a single Lagrange table will be proportional to the independent variable grids as well as their tensor product (number of dependent variable values). We can write this memory estimate, $M$, for a single structured table with $N_{x_i}$ points in the $i^{\text{th}}$ independent coordinate as

$$M_{\text{Lagrange}} \propto \sum_{i=1}^{d} N_{x_i} + \prod_{i=1}^{d} N_{x_i}. \tag{9}$$

B-spline memory is similar, but with $(p+1)$ extra points stored per dimension. While each local evaluation of an interpolated table may involve fewer data and therefore smaller memory than a global ANN evaluation, we focus on assessing the global memory of interpolated tables as it is relevant to the memory accessed during simulation. Methods aimed at reducing these memory footprints also exist, as mentioned in §1; however, an exponential increase with dimensionality is still expected. The ability of mesh refinement/coarsening to reduce memory is also limited for tensor-product interpolants, especially in higher dimensions. The memory for storing an ANN is dominated by its coefficients including weights, biases, and scaling/centering factors. For a total number of layers, $L_t$, (including input and output layers) we can write the ANN memory estimate as

$$M_{\mathrm{ANN}} \propto (2N_1 + 2N_{L_t}) + \sum_{L=2}^{L_t} (N_L N_{L-1} + N_L) \tag{10}$$

where $N_1$ will be determined by dimensionality $d$ and $N_{L_t}$ will be determined by the number of dependent variables per ANN. ANNs show significant advantages for memory since the number of stored coefficients is generally smaller than the size of data required for accurate interpolation, especially in higher dimensions. This enables ANNs to extended past 5D tabulation where interpolants are restricted by practical memory considerations.

Using these approximations, the theoretical memory of the 3D Lagrange-interpolated laminar nonadiabatic flamelet table is around 150 times larger than that of A4-20 and more than 3000 times larger than that of A2-5. The memory for the 3D A4-20 table is around 21 times larger than that of A2-5. The 4D interpolated turbulent nonadiabatic flamelet table is around 4.5 times larger than the laminar table. Compared to the turbulent A4-20 and A2-5, the turbulent interpolated table is around 670 times and more than 13000 times larger, respectively. The tables in these memory comparisons may have different levels of accuracy, but even the LP3 table built over an 11-point mixture fraction grid from Figure 3 requires approximately 200 times more memory than A2-5 for a similar accuracy. This demonstrates how dramatic the difference in memory can become in higher dimensions, especially for large data sets, and the advantages of using ANNs in these scenarios.

One could also build a single ANN with multiple outputs for the quantities needed during a CFD simulation, as demonstrated by Blasco et al. (Blasco et al., 1998). This only increases the output layer coefficients for the ANNs compared to a linear dependence of memory with number of properties when using interpolants. However, predicting multiple dependent variables of varying behaviors with one model can lower accuracy. When training a single ANN for multiple predictions, it is important to consider properties with similar features, especially given the errors seen from using a variety of complex architectures to represent a single property containing local nonlinearities. The lower memory requirements are an advantage of using ANNs, especially in the context of turbulent combustion where the dimensionality can exceed five dimensions.

## 4. Conclusions

Chemistry tabulation methods enable high fidelity combustion simulations for challenging applications such as turbulent fires with significant heat loss. Various methods for constructing property tables were presented in this study including first- to fourth-degree Lagrange polynomial and B-spline interpolants, and ANNs with varying architectures. Each of these models have tradeoffs when considering accuracy, evaluation cost, and memory.

B-splines were marginally more accurate than Lagrange polynomials and provide more flexibility, such as choosing knot locations, in obtaining accurate solutions. However, Lagrange polynomials were the fastest to evaluate for low dimensionality and/or low degrees and their simple construction allows for query optimization that can lead to significant further speedup. The larger ANNs showed comparable evaluation times in four or more dimensions for general table queries and were faster than higher-degree interpolants at this dimensionality. However, in the context of table queries involving extra conversions, including a nonlinear solve, to compute interpolated table inputs from simulation variables, such as demonstrated with both three- and four-dimensional nonadiabatic flamelet queries, the larger ANNs became faster than cubic Lagrange interpolated tables. In general, as dimensionality increases and/or more interpolated table input variables need to be determined with a nonlinear solve, larger-architecture ANN evaluation times will be favorable. ANNs also often have lower memory requirements than interpolants, which is advantageous for high-dimensional problems relevant to turbulent combustion.

ANN accuracy is highly sensitive to training options and does not offer the same guarantees as interpolants. While increasing the degree of an interpolant guaranteed an increase in accuracy, increasing the architecture of an ANN had no such guarantees and lead to larger errors after a fourth layer was added. Similarly, increasing training data resolution guaranteed an increase in accuracy for interpolant predictions within the training bounds, but only assisted ANN accuracy until prediction errors stagnated for a given architecture. Iterating on training ANNs is tedious and, alongside challenges in reproducing them from literature, is a significant drawback of using them. Newer approaches such as training multiple ANN models over domain partitions may be the answer in addressing accuracy concerns of ANNs. ANNs, therefore, present a competitive option for high-dimensional flamelet tabulation in combustion and should be explored further with a focus on accuracy requirements and establishing standard methods for training.

**Funding**

This work was supported by the Sandia National Laboratories [Contract Purchase Agreement number 2156884].

**Acknowledgments**

The authors gratefully acknowledge the support of Sandia National Laboratories. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government. The authors also thank Robert C. Knaus and Michael A. Hansen from Sandia for their many helpful discussions and suggestions.

**Disclosure Statement**

No potential conflict of interest was reported by the author(s).

## References


Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X., 2016. TensorFlow: Large-scale machine learning on heterogeneous systems, arxiv:1603.04467v2 [cs.dc].

Armstrong, E., Hansen, M. A., Knaus, R. C., Hewson, J. C., Sutherland, J. C., 2019. Constructing and accessing tabulated chemistry for fire scenarios. Western States Section of the Combustion Institute Fall Technical Meeting, paper 918-0040.

Armstrong, E., Sutherland, J. C., 2021. A technique for characterising feature size and quality of manifolds. Combust. Theor. Model. 25, 646–668.

Blasco, J., Fueyo, N., Dopazo, C., Ballester, J., 1998. Modelling the temporal evolution of a reduced combustion chemical system with an artificial neural network. Combust. Flame 113, 38 – 52.

Blasco, J. A., Fueyo, N., Larroya, J. C., Dopazo, C., Chen, Y. J., 1999. A single-step time-integrator of a methane-air chemical system using artificial neural networks. Comput. Chem. Eng. 23, 1127 – 1133.

Bode, M., Collier, N., Bisetti, F., Pitsch, H., 2019. Adaptive chemistry lookup tables for combustion simulations using optimal B-spline interpolants. Combust. Theor. Model. 23, 674–699.

Chatzopoulos, A., Rigopoulos, S., 2013. A chemistry tabulation approach via rate-controlled constrained equilibrium (RCCE) and artificial neural networks (ANNs), with application to turbulent non-premixed $CH_4/H_2/N_2$ flames. Proc. Combust. Inst. 34, 1465 – 1473.

Chen, J.-Y., Blasco, J., Fueyo, N., Dopazo, C., 2000. An economical strategy for storage of chemical kinetics: Fitting in situ adaptive tabulation with artificial neural networks. Proc. Combust. Inst. 28, 115 – 121.

Chollet, F., et al., 2015. Keras. `https://keras.io`.

Christo, F., Masri, A., Nebot, E., 1996. Artificial neural network implementation of chemistry with pdf simulation of $H_2/CO_2$ flames. Combust. Flame 106, 406 – 427.

Domino, S. P., Hewson, J., Knaus, R., Hansen, M., 2021. Predicting large-scale pool fire dynamics using an unsteady flamelet- and large-eddy simulation-based model suite. Phys. Fluids 33, 085109.

Emami, M., Eshghinejad Fard, A., 2012. Laminar flamelet modeling of a turbulent $CH_4/H_2/N_2$ jet diffusion flame using artificial neural networks. Appl. Math. Model. 36, 2082 – 2093.

Franke, L. L., Chatzopoulos, A. K., Rigopoulos, S., 2017. Tabulation of combustion chemistry via artificial neural networks (ANNs): Methodology and application to LES-PDF simulation of sydney flame L. Combust. Flame 185, 245 – 260.

Glorot, X., Bengio, Y., 2010. Understanding the difficulty of training deep feedforward neural networks. Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, PMLR 9, 249–256.

Hansen, M., Armstrong, E., Sutherland, J., McConnell, J., Hewson, J., Knaus, R., 2020. Spitfire. `https://github.com/sandialabs/Spitfire`.

Hermann, T., 2021. frugally-deep. `https://github.com/Dobiasd/frugally-deep`.

Honzawa, T., Kai, R., Hori, K., Seino, M., Nishiie, T., Kurose, R., 2021. Experimental and numerical study of water sprayed turbulent combustion: Proposal of a neural network modeling for five-dimensional flamelet approach. Energy and AI 5, 100076.

Ihme, M., Marsden, A. L., Pitsch, H., 2008. Generation of optimal artificial neural networks using a pattern search algorithm: Application to approximation of chemical systems. Neural Comput. 20, 573–601.

Ihme, M., Schmitt, C., Pitsch, H., 2009. Optimal artificial neural networks and tabulation methods for chemistry representation in les of a bluff-body swirl-stabilized flame. Proc. Combust. Inst. 32, 1527 – 1535.

Kempf, A., Flemming, F., Janicka, J., 2005. Investigation of lengthscales, scalar dissipation, and flame orientation in a piloted diffusion flame by LES. Proc. Combust. Inst. 30, 557 – 565.

Kingma, D. P., Ba, J., 2014. Adam: A method for stochastic optimization, arxiv:1412.6980v9 [cs.lg].

Lee, K., Trask, N. A., Patel, R. G., Gulian, M. A., Cyr, E. C., 2021. Partition of unity networks: deep hp-approximation, arxiv:2101.11256v1 [cs.lg].

Luo, Z., Yoo, C., Richardson, E., Chen, J., Law, C., Lu, T., 2012. Chemical explosive mode analysis for a turbulent lifted ethylene jet flame in highly-heated coflow. Combust. Flame 159, 265–274.

McConnell, J. T., Saad, T., Sutherland, J. C., 2020. An explicit low-mach projection method for modeling flows with finite-rate chemistry. AIAA AVIATION 2020 FORUM, paper AIAA 2020-3035.

Owoyele, O., Kundu, P., Ameen, M. M., Echekki, T., Som, S., 2020. Application of deep artificial neural networks to multi-dimensional flamelet libraries and spray flames. Int. J. Engine Res. 21, 151–168.

Owoyele, O., Kundu, P., Pal, P., 2021. Efficient bifurcation and tabulation of multi-dimensional combustion manifolds using deep mixture of experts: An a priori study. Proc. Combust. Inst. 38, 5889–5896.

Peters, N., 1984. Laminar diffusion flamelet models in non-premixed turbulent combustion. Prog. Energy Combust. Sci. 10, 319 – 339.

Peters, N., 2000. Turbulent Combustion. Cambridge University Press, Cambridge, U.K.

Piegl, L., Tiller, W., 1997. The NURBS Book. Springer-Verlag, Berlin, Germany.

Sen, B. A., Menon, S., 2009. Turbulent premixed flame modeling using artificial neural networks based chemical kinetics. Proc. Combust. Inst. 32, 1605 – 1611.

Sen, B. A., Menon, S., 2010. Linear eddy mixing based tabulation and artificial neural networks for large eddy simulations of turbulent flames. Combust. Flame 157, 62 – 74.

Späth, H., 1995. One Dimensional Spline Interpolation Algorithms. A K Peters, Ltd., Wellesley, Massachusetts, U.S.A.