

Explicit Thermodynamic Properties using Radial Basis Functions Neural Networks

*Olivier Adam** and *Olivier Léonard†*

Abstract

Gas turbine design, development, monitoring and maintenance are widely based on numerical simulations of the steady and transient engine performance. Most of the equations that are solved in the simulation programs involve the thermodynamic properties of the fluid flowing through the engine. These properties depend on temperature, pressure, humidity and fuel dosage. As the solution of chemical equilibrium is not compatible with real-time computations, a chemical solver is used off-line to generate a large database which neural networks are trained on.

These networks are built on radial basis functions such as multiquadrics. A forward selection approach is used to select data points from the training set as the centers of the transfer functions. The selection stops when the prediction error starts growing. The resulting networks for specific heat and enthalpy of the gas mixture are 3 orders of magnitude faster than the chemical solver.

In order to further increase the efficiency and the generalization capabilities of the model, an external optimization solver has been used to tune the shape of the transfer functions. Several solutions are proposed and preliminary results are presented.

*Assistant, University of Liège, Turbomachinery Group, Chemin des Chevreuils 1 (B52/3), 4000 Liège, Belgium. o.adam@ulg.ac.be

†Professor, University of Liège, Turbomachinery Group, Chemin des Chevreuils 1 (B52/3), 4000 Liège, Belgium. o.leonard@ulg.ac.be

1 Introduction

The design of gas turbine type propulsion systems strongly relies on an accurate modeling of their components. Aero-thermodynamic performance models are used during the design phase, for flight performance calculations of the airframe-propulsion system and also for engine test analysis during the development phase. In addition, modern engine monitoring systems also need reliable and fast, real-time engine performance prediction methods. This is valid for both steady and transient operations of the engine.

The design and analysis of jet engines deal with combustion of fuel in presence of air. Moreover the working gas inside the jet engine encounter strong local variations of temperature and pressure. The thermodynamic properties of these gas (enthalpy, entropy, specific heat, ...) depend on the combustion, compression or expansion processes occurring in the engine. They also depend on the presence of humidity in the air mainstream.

In a classical approach, exact determination of the thermodynamic properties of the combustion products requires the resolution of chemical equations for the equilibrium mass fractions. These mass fractions are used in combination with thermochemical reference tables [3] to obtain global properties of the gas, such as specific heat at constant pressure. However solving the chemical equations of combustion is time-consuming, and this solution is not suitable for real-time applications and iterative solvers. Getting a required value by means of a table lookup is also too expensive, because a valid database is likely to contain more than 10,000 points, given the variation intervals of the physical variables.

Therefore there is a clear need for thermodynamic properties as explicit functions of temperature, pressure, water-to-air and fuel-to-air ratios as far as engine performance prediction and real-time analyses are concerned. Hence the basic idea of this project is to model the properties of the gas traveling through a jet engine by means of neural networks previously trained on thermochemical data. In the present project, a separate chemical solver [4] was run on a large scale. The results led to the construction of a full thermodynamic database.

The modeling process can be seen as a regression problem. As the mathematical form of the thermodynamic functions is not *a priori* known, this regression is said to be nonparametric. The need for a reduction of the size of the problem (for maximum quickness) and the fact that the model must be valid for a wide range of input values has led to the selection of a neural network.

The regression aspect requires a local control of the network response. Radial basis functions networks were then preferred to multi-layer perceptrons. Moreover the former allow a linear optimization of the network parameters up to a certain point, where the latter invariably need a nonlinear optimization technique [2].

The thermodynamic explicit functions were then built on neural networks using radial basis functions. Each property is modeled by means of an equation using free parameters with no particular physical meaning but whose number must be sufficient to faithfully represent that function.

2 RBF neural networks

The radial basis functions (RBF) neural networks are a particular form of linear models, where the approximation function f takes the form of a linear combination of a set of defined functions. The flexibility of the model comes from the possibility to adapt the coefficients (or weights, w_j) of the linear combination, as in Fourier series or classical polynomial regression.

The general form of the model as a function of some input variables vector \mathbf{x} can be written as

$$f(\mathbf{x}) = \sum_{j=1}^m w_j h_j(\mathbf{x}) \quad (1)$$

The transfer functions $h_j(\mathbf{x})$ are combined using the free parameters w_j . The size of the resulting model is the number of implied transfer functions, m .

The regression problem fits quite well with radial functions, whose response decreases or increases monotonically with the distance from a central point. A one-dimensional example is given by a gaussian transfer function centered at c and scaled by a “radius” r :

$$h(x) = \exp\left(-\frac{(x-c)^2}{r^2}\right) \quad (2)$$

The equation (1), associated with the definition of a radial basis function, describes the principle of a RBF neural network. From a statistical point of view, it can be seen as an approximation model of a large number of sample data or observations. In the neural networks terminology, it also represents a single-layer network whose transfer functions are the chosen radial basis functions. The coefficients of (1) also stand for the network synaptic weights w .

This model is used within the framework of a supervised learning, where the large number of observations constitute a *training set*, i.e. a list of known input-output pairs. The size of the learning set is noted n ; the purpose of the neural network is to model the full learning set by means of m parameters, with $m \ll n$.

In the present work, only one hidden layer was used : the final functions always have the form of (1). Nevertheless, several variations of the free parameters w_j were tested. As long as the basis functions do not change size neither move, the problem remains linear and the optimal weights can be found through linear algebra operations. If parameters are affected to the position or shape of the basis functions, nonlinear optimization algorithms have to be introduced in order to find an optimal neural network.

Two kinds of basis functions were tested during the tuning of thermodynamic explicit functions : gaussian and multiquadric functions. In two dimensions, they can both be seen as “bells”. The gaussian type is limited in response and brings a local control of the model ; multiquadric shape functions allow better generalization capabilities. The position of the radial function is given by its center \mathbf{c} , and its shape is defined by the radii, which can be seen as scaling constants.

For a given point, say \mathbf{x} , the generalized distance z is

$$z^2 = (\mathbf{x} - \mathbf{c})^T \mathbf{R}^{-2} (\mathbf{x} - \mathbf{c}) \quad (3)$$

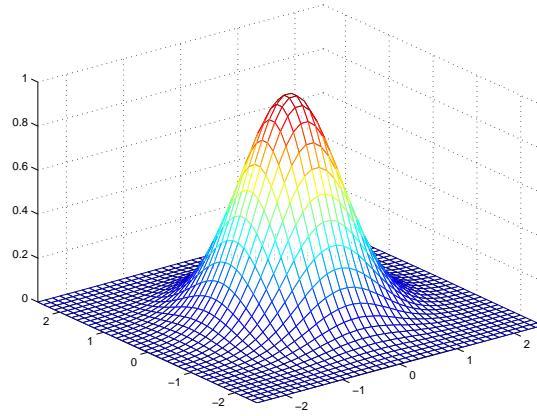


Figure 1. *Gaussian RBF*

The matrix of radii, \mathbf{R} , defines the metric in which the distances are evaluated. Still in 2D, one writes

$$\mathbf{R} = \begin{bmatrix} r_x & 0 \\ 0 & r_y \end{bmatrix} \quad (4)$$

This expression shows that only diagonal covariance matrices were used. All the variables are relevant in the determination of the thermodynamic properties and it was therefore not necessary to implement full covariance matrices [2].

With these notations, gaussian (h_g) and multiquadric (h_m) transfer functions are written as follows :

$$h_g(\mathbf{x}) = e^{-z^2} \quad (5)$$

$$h_m(\mathbf{x}) = \sqrt{1 + z^2} \quad (6)$$

Figures 1 and 2 outline their respective bounded and unbounded behaviors.

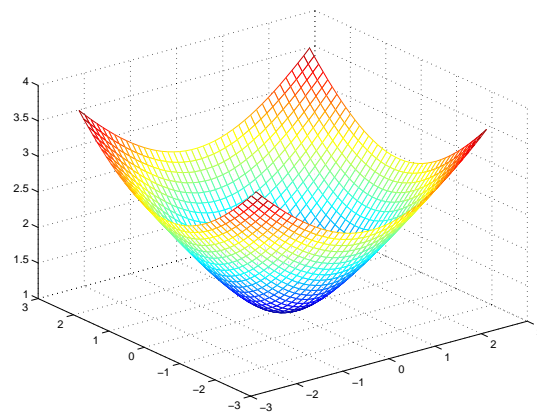


Figure 2. *Multiquadric RBF*

Ar	Argon
CO	Carbon monoxide
CO ₂	Carbon dioxide
H	Atomic hydrogen
H ₂	Molecular hydrogen
H ₂ O	Steam
N	Atomic nitrogen
N ₂	Molecular nitrogen
NO	Nitrogen oxide
O	Atomic Oxygen
OH	Hydroxide
O ₂	Molecular oxygen

Table 1. *Combustion products*

3 Physical aspects

Before going on with the construction of the RBF network, a few words must be said about the particular problem which is to be solved with the help of artificial neural networks.

Neural networks are introduced to replace explicit computation of the thermodynamic properties of the working gas inside a jet engine, in the cold parts (intake and compressor) as well as in the hot parts (combustion chamber, turbine, reheat duct and nozzle). However neural networks must be trained on a sufficiently large database filled with thermodynamic gas properties as functions of the flow variables.

These data are obtained from the tabulated properties of the numerous components of the gas [3]. If a combustion of fuel takes place, the gas composition is better found by solving the equations for chemical equilibrium analysis. The resulting mass fractions then yield the gas properties.

Four input variables define the state of the fuel/air/water mixture traveling through the jet engine, namely :

- the temperature, T
- the pressure, p
- the mass ratio of the water present in the air, i.e. the water-to-air ratio, or war
- the fuel-to-air ratio, or far

Given these 4 variables, the chemical program (CEA, see [4] for details) solves the chemical equations under the assumption of equilibrium and provides the mass fractions of the 12 distinct chemical species which are supposed to be present in the combustion gas. Each of these species has its own enthalpy and specific heat. The species are given for completeness in table 1.

The variation intervals of the physical variables (temperature, pressure, war , far) are given in table 2. The temperature ranges from high altitude inlet conditions (250 K) [1] to peak combustion temperature (2500 K). The pressure interval covers

Variable	Units	Min	Max
T	K	250	2500
p	Pa	$5 \cdot 10^3$	$6 \cdot 10^6$
war	—	0	0.01
far	—	0	0.06

Table 2. *Physical variable ranges*

more than observable flight or operating conditions. The maximum water and fuel ratios correspond to realistic values of 1 % and 6 %, respectively.

Because molecular dissociation occurs for high temperatures and low pressures, the graphs for specific heat such as figure 4 point out different behaviors of the thermodynamic functions. From 250 to 1400 K, the mass fractions of the species are influenced only by the temperature and the composition of the gas mixture, i.e. by the war and far . The specific heats for individual species are derived from the ideal gas assumption and consequently are functions of temperature only. As a result, the specific heat of the gas mixture is a function of temperature, war and far for temperatures up to 1400 K and is not a function of pressure. This can be seen in equation (12) : the second term vanishes in the absence of combustion.

Above 1400 K, chemical dissociation appears and pressure plays a growingly stronger role as temperature increases. This effect is clearly visible on the figure 3 where the temperature limit has been represented for a typical sample (war, far) couple.

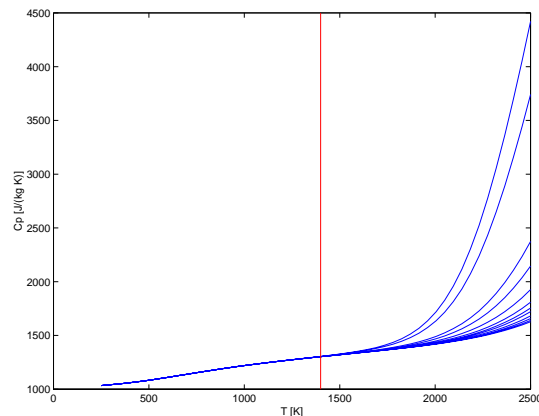


Figure 3. *Gas dissociation limit of temperature*

This limit of 1400 K has shown to be relevant for all (war, far) couples. It also holds for the enthalpy function. It allows the specific heat modeling problem to be divided into two subproblems :

$$T \leq 1400K \Rightarrow C_p = f(T, war, far) \quad (7)$$

$$T > 1400K \Rightarrow C_p = f(T, p, war, far) \quad (8)$$

A short matching zone is used between 1375 and 1425 K, where the returned values of the two models are blended to ensure continuity. The small overhead of operations to compute two models for a requested temperature is minimal and covers only about 2 % of the temperature range.

Two thermodynamic properties are herein concerned, namely the enthalpy and the specific heat of the engine working fluid. More specifically, the required enthalpy function is the global sensitive enthalpy. It can be seen as a quantification of the total energy held by the gas : therefore it needs to be defined relative to a chosen ground level. For the individual species, the reference at 298.15 K is the formation enthalpy $h_{f,j}$ at 298.15 K :

$$h_{s,j}(T) = \int_{298.15}^T C_{p,j}(T') dT' + h_{f,j}|_{298.15} \quad (9)$$

As a result of the chemical equations, the mass fractions ξ_j of these species are used as mixing weights for the global enthalpy h_g :

$$h_g = \sum_{j=1}^{12} \xi_j h_{s,j} \quad (10)$$

An additional hypothesis has to be introduced for the global specific heat : the gas are supposed to have reached *chemical equilibrium*. This is generally speaking not the case, but most of the reactions are fast and this assumption tends to be more accurate than the “frozen composition” assumption. Hence, the global specific heat at constant pressure reads :

$$C_p = \left[\frac{\partial h_g}{\partial T} \right]_p \quad (11)$$

$$= \sum_{j=1}^{12} \xi_j C_{p,j} + \sum_{j=1}^{12} h_{s,j} \left[\frac{\partial \xi_j}{\partial T} \right]_p \quad (12)$$

Taking the mass fraction derivatives with respect to temperature into account introduces strong variations of the specific heat in the case of high temperatures and low pressures ; it is an outward sign of the molecular dissociation occurring in such conditions. An example of specific heat curves is given on figure 4, for fixed *war* and *far*. Strong difficulties are expected to arise with the pressure variable, since it has absolutely no effect at low temperatures. At the opposite, the effect of pressure is exponential for hot gas. This issue will be discussed later in this paper.

It can be seen from equation (11) that the global enthalpy and the specific heat of the combustion gas are not independent of each other : the latter is the derivative, pressure held constant, of the first one. Nevertheless, differentiating an expression like the RBF model can be very expensive, keeping in mind that its size $m \propto 10^2$. Furthermore, equation (1) describes an approximation model built as an

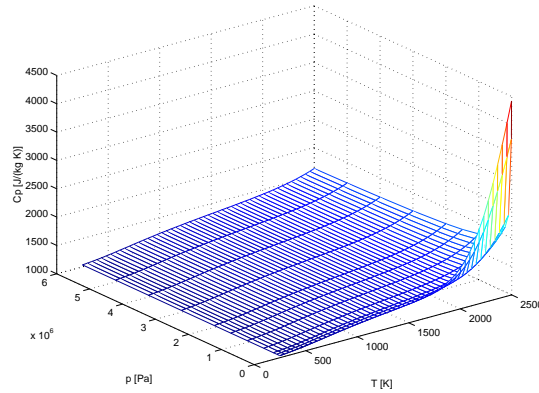


Figure 4. *Sample specific heat curves*

accumulation of “bumps”, as shown on figures 1 and 2. Differentiating a model of enthalpy would then lead to unwanted oscillations in the specific heat.

The retained solution is to build two neural networks : one for the specific heat, and one for the enthalpy. Both of them are based on an extensive database or *training set* of the general form

$$\mathcal{T} \equiv \{\mathbf{x}_i, \hat{y}_i\}_{i=1}^n \quad (13)$$

This expression indicates that there are n observations of both thermodynamic functions C_p and h_g . Both C_p and h_g depend on 4 independent variables combined into the input vector \mathbf{x} . The notation \hat{y} indicates the noise or numerical errors that could exist in the estimation of the thermodynamic properties through the chemical analysis program. These errors are due to program accuracy and truncation when the outputs are written into the database.

As one can expect more difficulties for the modeling of C_p than for h_g , the following sections detail the tuning process of artificial neural networks for the specific heat of the fuel/air/water gaseous mixture encountered inside a jet engine. As regards enthalpy, similar methods were applied and will not be explained. Only results will be shown for that function.

4 Network topology

Sections 2 and 3 tend to indicate that the RBF neural network has to model monotonic curves by means of nonlinear basis functions. There is indeed no exception to the fact that the specific heat and the enthalpy increase if the temperature increases, for example.

The “building bricks” of the specific heat model are the radial basis functions described in section 2. They simply belong to a class of transfer functions and could be used in any sort of model, linear or nonlinear. The selection process among the large number of data points, the clear need to define a compact model for fast execution purposes and the monotonic aspect of the thermodynamic functions have

led to the selection of single-layer neural networks. Equation (1) corresponds to this topology.

Linear and nonlinear aspects will be reviewed later in this paper. As already said, only the mixing weights are used as optimization parameters in linear models. Nonlinear models allow the modification of the shape and/or position of the radial basis functions.

Linear and nonlinear models can be sketched by figure 5 which represents a three-layer neural network whose hidden functions $h_j(\mathbf{x}_i)$ are the radial basis functions $h_j(\mathbf{x}_i)$ for completeness. A bias term is added to the graphical transcription of equation (1) for completeness. The bias term is a unit transfer function : it takes into account the possibility for the mean output to be different from zero, which is often the case. It increases the size of the resulting model by only one weight and has allowed better accuracy and increased generalization possibilities.

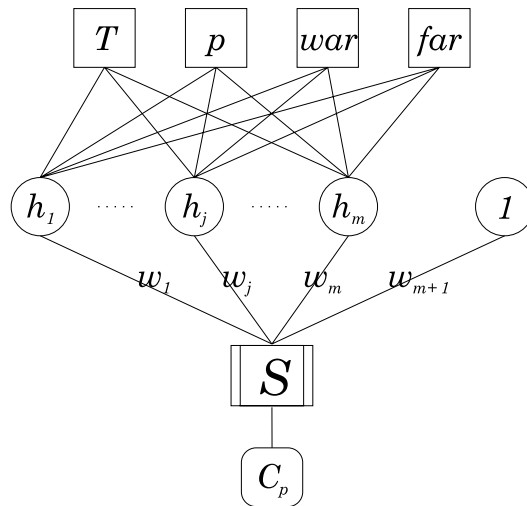


Figure 5. RBF neural network topology

5 Model accuracy

A simple measure of the fitting accuracy is given by the *sum-squared error*. According to the notations of equation (13), a given function value \hat{y}_i can be compared to the predicted value $f(\mathbf{x}_i)$. This value is returned by the model (1) for a vector of variables

$$\mathbf{x}_i = \{T_i, p_i, war_i, far_i\}^T \quad (14)$$

According to the *training set* notations, there are p values to compute, and the sum-squared error (SSE) is

$$S = \sum_{i=1}^n (\hat{y}_i - f(\mathbf{x}_i))^2 \quad (15)$$

A minimization of S with respect to the weights (see figure 5) keeps the problem linear. Choosing the radii or centers as additional parameters requires the introduction of nonlinear optimization methods. Both options use a simple least-squares optimization.

The ridge regression method adds a weight penalty term to the SSE [9]. It has proven to have a valuable smoothing effect on the resulting optimized models and has been used throughout the present project to ensure the mandatory monotonicity and smoothness of the models. Only one penalty parameter (λ) was introduced and its value was kept constant, so that the optimization process turned into the minimization of the modified SSE with a penalty term, called *cost function* :

$$C = \sum_{i=1}^n (\hat{y}_i - f(\mathbf{x}_i))^2 + \sum_{j=1}^{m+1} \lambda w_j^2 \quad (16)$$

Iterative re-estimation of the regularization parameter λ is possible but has not been implemented. In fact, this penalty term was added only to avoid too large weights during the minimization of the SSE (the terminology of neural networks names the ridge regression method the *weight decay*). In our problem all data points are equally meaningful. The smoothing effect of the regularization then tends to keep the model more realistic. The reduction of oscillation risks is satisfactory with one parameter only.

Once the shape and position of the radial basis functions have been chosen, the determination of the optimal weight vector reduces to a least-squares problem. The introduction of the full training set \mathcal{T} (13) into the model's equation (1) gives rise to the following vector/matrix notation which shows the linearity of the optimal weights determination problem :

$$\mathbf{H}\hat{\mathbf{w}} = \hat{\mathbf{y}} \quad (17)$$

This equation includes :

- the weights vector $\hat{\mathbf{w}}$; the hat denotes the optimized aspect of the weights
- the training set values $\hat{\mathbf{y}}$
- the transfer function matrix, also called *design matrix* written as

$$\mathbf{H} = \begin{bmatrix} h_1(\mathbf{x}_1) & h_2(\mathbf{x}_1) & \cdots & h_m(\mathbf{x}_1) & 1 \\ h_1(\mathbf{x}_2) & h_2(\mathbf{x}_2) & \cdots & h_m(\mathbf{x}_2) & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ h_1(\mathbf{x}_n) & h_2(\mathbf{x}_n) & \cdots & h_m(\mathbf{x}_n) & 1 \end{bmatrix} \quad (18)$$

The column $(m+1)$ is due to the bias term corresponding to the unit transfer function.

The system (17) is over-determined : the design matrix \mathbf{H} has a size of $[n \times (m+1)]$, as n training values are used to determine a much smaller number of $m+1$ optimal weights. The purpose of the artificial neural network is indeed to decrease the problem size from n to m or $m+1$ with a bias.

The least-squares problem reads from (17) :

$$\mathbf{H}^T \mathbf{H} \hat{\mathbf{w}} = \mathbf{H}^T \hat{\mathbf{y}} \quad (19)$$

$$\hat{\mathbf{w}} = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \hat{\mathbf{y}} \quad (20)$$

The solution of normal equations yields the weights making the sum-squared error minimal. If the ridge regression is taken into account, one has to modify the normal equations in (19) to introduce the penalty term :

$$\{\mathbf{H}^T \mathbf{H} + \mathbf{\Lambda}\} \hat{\mathbf{w}} = \mathbf{H}^T \hat{\mathbf{y}} \quad (21)$$

The matrix $\mathbf{\Lambda}$ is diagonal and contains the regularization parameter. The optimal weights vector becomes

$$\hat{\mathbf{w}} = \mathbf{A}^{-1} \mathbf{H}^T \hat{\mathbf{y}} \quad (22)$$

where \mathbf{A}^{-1} is the variance matrix :

$$\mathbf{A}^{-1} = \{\mathbf{H}^T \mathbf{H} + \mathbf{\Lambda}\}^{-1} \quad (23)$$

Equation (22) provides the vector of weights minimizing the cost function (16).

The comparison of the predicted values and the training set data introduces the projection matrix :

$$\hat{\mathbf{y}} - \mathbf{f} = \hat{\mathbf{y}} - \mathbf{H} \hat{\mathbf{w}} \quad (24)$$

The optimal weight vector comes directly from (22) :

$$\hat{\mathbf{y}} - \mathbf{f} = (\mathbf{I}_p - \mathbf{H} \mathbf{A}^{-1} \mathbf{H}^T) \hat{\mathbf{y}} \quad (25)$$

$$= \mathbf{P} \hat{\mathbf{y}} \quad (26)$$

The projection matrix \mathbf{P} is then useful to evaluate the sum-squared error or the cost function :

$$\begin{aligned} S &= (\hat{\mathbf{y}} - \mathbf{f})^T (\hat{\mathbf{y}} - \mathbf{f}) \\ &= \hat{\mathbf{y}}^T \mathbf{P}^2 \hat{\mathbf{y}} \end{aligned} \quad (27)$$

A vector transcription of (16) gives for the cost function

$$\begin{aligned} C &= (\hat{\mathbf{y}} - \mathbf{f})^T (\hat{\mathbf{y}} - \mathbf{f}) + \hat{\mathbf{w}}^T \mathbf{\Lambda} \hat{\mathbf{w}} \\ &= \hat{\mathbf{y}}^T \mathbf{P}^2 \hat{\mathbf{y}} + \hat{\mathbf{y}}^T (\mathbf{P} - \mathbf{P}^2) \hat{\mathbf{y}} \\ &= \hat{\mathbf{y}}^T \mathbf{P} \hat{\mathbf{y}} \end{aligned} \quad (28)$$

The sum-squared error (SSE) or the cost function are useful but not sufficient to lead an optimization process to a valuable solution. With a fixed set of centers and radii, i.e. with transfer functions fixed in number, position and shape, the normal equations (19) or (21) yield naturally the best weights for a given training set.

However when it comes to the selection of the centers, or to the choice of their position and shape, the solution that gives a minimal cost function or SSE leads to poor models. The reason is that too much added information in the model tends to over-fit it. In other words the neural network produces a good prediction of the training set data points but has very poor generalization capabilities.

To avoid such a behavior, one must estimate how well the neural network would perform on unknown data points. This estimation is called the *prediction error*. The best model is the one which produces the smallest prediction error.

The most obvious way to estimate it is to build the neural network with a part of the training set and to check the trained network with the rest of the data. Various techniques based on this principle exist and are part of the **cross-validation** methods. They consist of adjustments brought to the sum-squared error given in (27).

Leaving a part of the training set for future model validation is a bit crude and it would be preferable to use all the information available. The solution given by a basic cross-validation is always conditioned by the choice of the training set/test set partition.

A more elegant but expensive solution is to split the n data points into a training set of size $n-1$ and a test set of size 1 and to estimate the mean squared error over the n possible partitions. This is called leave-one-out cross-validation (LOO). In the case of linear models, LOO prediction error can be calculated analytically by

$$\hat{\sigma}_{LOO}^2 = \frac{\hat{\mathbf{y}}^T \mathbf{P} (\text{diag}(\mathbf{P}))^{-2} \mathbf{P} \hat{\mathbf{y}}}{p} \quad (29)$$

Generalized cross-validation (GCV) criterion is easier to handle mathematically ; its expression looks like a scaled SSE from (27).

$$\hat{\sigma}_{GCV}^2 = \frac{p \hat{\mathbf{y}}^T \mathbf{P}^2 \hat{\mathbf{y}}}{(\text{trace}(\mathbf{P}))^2} \quad (30)$$

The expressions of the cost function and the GCV prediction error can be combined to build the best possible model from a given training set. The forward selection method takes advantage of the linear aspect of the equations when the centers are fixed in position and shape. Some nonlinear tests are presented afterwards. Results for both methods (linear forward selection, nonlinear optimization) will be presented in the subsequent sections.

6 Forward selection principles

The basic strategy of the forward selection is : given a certain training set, build the model brick after brick, center after center, while controlling the balance between sum-squared error and prediction error [7].

The process starts with an empty subset. Basis functions are added one by one and are defined by the position of their center and a radius. The centers are chosen so that they coincide with data points from the training set. The radius related to a training set point (to a function center) is chosen before initiating the forward selection process. This *a priori* choice leads to considerations on the training set itself and to a trial-and-error strategy. This point will be discussed later.

Let us suppose that the radii are fixed. A selection process is iterative : each iteration selects a data point from the training set and builds a transfer function centered at this point. The center is selected so that the sum-squared error is most reduced. The selection process goes on as long as the sum-squared error keeps decreasing and is stopped when the GCV error (for example) starts increasing, so that over-fitting is avoided. Therefore the forward selection process results from

the combination of two complementary criteria : the diminution of the sum-squared error and the non-growth of the GCV error.

The forward selection process gives no *a priori* information on how many centers will be included in the final model. This is a drawback in the present case, because the designed neural network must be as fast as possible and therefore it would be preferable to keep control of the number of centers. However it is possible to build the training set to tune the selection process and indirectly to limit the size of the resulting neural network.

At each iteration of the forward selection, a data point must be selected as the center of a new basis function. The only way to choose the best data point is to examine all the remaining data in the pool of candidates which is the training set minus the already selected centers, and to evaluate what the SSE or cost function would be if this data point was selected as a center.

Fortunately it is possible to rewrite equations (27) and (28) to obtain only incremental values. Then the change in SSE or cost function can be evaluated for each data point without recomputing the entire error function. The full computation of the cost function, at each iteration and for all the remaining center candidates, would limit the domain of application of forward selection to uninteresting small problems [2].

When examining the data points as candidates for model centers, one chooses among the columns of the full design matrix the one that will be added to the model design matrix. An element (i, j) of the full design matrix \mathbf{F} is the value of the transfer function based on data point j and evaluated for the data points $i = 1 \dots n$. The size of \mathbf{F} is then $n \times n$ and may become prohibitive when it comes to training sets of several thousands of points.

Considering that the model already includes m centers, if the J -th basis function is added to the model, the corresponding column of \mathbf{F} , \mathbf{f}_J , is added to the model's transfer matrix \mathbf{H}_m . The column \mathbf{f}_J is renamed \mathbf{h}_{m+1} and the new design matrix is \mathbf{H}_{m+1} . The permutation matrix is also modified from \mathbf{P}_m to \mathbf{P}_{m+1} , according to

$$\mathbf{P}_{m+1} = \mathbf{P}_m - \frac{\mathbf{P}_m \mathbf{f}_J \mathbf{f}_J^T \mathbf{P}_m}{\mathbf{f}_J^T \mathbf{P}_m \mathbf{f}_J} \quad (31)$$

This equation provides the change in the sum-squared error or in the cost function when \mathbf{f}_J is added to the model. This implies some matrix manipulations ; see [7] for details.

In the most general case of ridge regression, the update reads for the sum-squared error :

$$S_m - S_{m+1} = \frac{2 \hat{\mathbf{y}}^T \mathbf{P}_m^2 \mathbf{f}_J \hat{\mathbf{y}}^T \mathbf{P}_m \mathbf{f}_J}{\lambda + \mathbf{f}_J^T \mathbf{P}_m \mathbf{f}_J} - \frac{(\hat{\mathbf{y}}^T \mathbf{P}_m \mathbf{f}_J)^2 \mathbf{f}_J^T \mathbf{P}_m^2 \mathbf{f}_J}{(\lambda + \mathbf{f}_J^T \mathbf{P}_m \mathbf{f}_J)^2} \quad (32)$$

$$C_m - C_{m+1} = \frac{(\hat{\mathbf{y}}^T \mathbf{P}_m \mathbf{f}_J)^2}{\lambda + \mathbf{f}_J^T \mathbf{P}_m \mathbf{f}_J} \quad (33)$$

After the update of the SSE (32) and of the permutation matrix (31), the GCV prediction error is deduced from equation (30). This completes the tools available to lead a forward selection process to the definition of an efficient neural network model.

Practically, the method can be further accelerated by using orthogonal least squares. In that context the design matrix is factorized into the product of a matrix with orthogonal columns and an upper triangular matrix. It follows that the variance matrix becomes diagonal and the projection matrix appears as a simple vectorial form. Computing the orthogonal sum-squared error requires a number of floating point operations of order n compared to n^2 for the non-orthogonal form (32). The columns of \mathbf{F} are maintained orthogonal to the columns of the growing design matrix and the upper triangular matrix is updated through simple expressions.

This method was used for the following results ; it allowed much faster computations. It was very helpful when it came to trial-and-error tests upon the radii of the radial basis functions.

7 Forward selection results

The subdivision of the modeling problem and the use of two separate neural networks is motivated by topological reasons, as it was explained in section 3. This choice is reinforced by pragmatic considerations. Two kinds of applications may require the use of explicit thermodynamic functions : one is related to the compressors whose mean working temperature is around 500 or 600 K. The other application is related to the combustion chambers or to the turbine for which temperature reaches 1800 K. There is consequently a little demand for function evaluations in the 1400 K zone. And above all the modeling of the high temperature zone of C_p may require much more centers than the low temperature zone, partly because of the pressure effect.

It may be interesting to outline that in the case of a unique neural network valid for the whole temperature range, the choice of the pressure radii becomes a real problem. The low temperature data require large radii to lower the influence of the pressure on that side of the graph. At higher temperatures however the pressure radii have to be significantly lower because of the strong variations observed in the pressure direction. The subdivision has been introduced for the sake of CPU time reduction and to ease the set up of the neural network.

All data points from the training set have the same importance and there is little or no indication about which specific radius to attach to a particular data. The two neural networks have been assigned sets of three and four radii respectively in order to take into account the evolution of C_p with regard to the different physical variables, instead of individual radii for each center or transfer function.

The model has to be fast. This consideration leads to rather large radii attached to rather spaced centers. The choice of the radii attached to T , war , far ,

and p were also guided by considerations on the sensitivity of the specific heat with regard to these variables. Varying a radius affects the network by bringing more or less stiffness in the direction of the variable. It means that a curve presenting rapid and strong variations requires peak transfer functions able to catch these bumps, namely small radii. For example, the rapid changes of slope for $C_p(T)$ curves suggested rather small radii for the temperature variable to avoid too large modeling errors. On the opposite a softly evolving curve will be better matched by large radii combined to a few centers : as the influence of war and far is very close to linear, large radii were allowed.

Several sets of radii were tested with the algorithm of forward selection and the best is presented hereafter. As said before the process of radii selection follows a trial-and-error approach because of its discrete aspect. One may indeed think of an optimization loop outside the forward selection but the final number of centers is not *a priori* known and evolves with the radii. Even if the modeling error may show a global minimum as a function of a single radius, the graph is extremely noisy.

Before presenting the neural network results, a word must be said about the size of the problem. The full database generated with the chemical solver contains a little less than 13,000 regularly spaced data following a regular meshing of the ranges presented in table 2. The subdivision into low and high temperature zones provides two training sets with less than 7,000 points. It is still far above the capabilities of an ordinary workstation to invert, orthogonalize or perform any n^2 operation on such a matrix within reasonable delays. The forward selection process, in particular, requires of the order of $2m^2 + mn + n^2$ floating point operations to add a single new pattern to the model and the optimal configurations include $m = 300$ centers.

The two training sets had to be shrunk down to a more reasonable size, by a manual selection of data points, but keeping the idea of regular meshing. The shrunk training sets contained respectively 1008 and 1296 data, allowing both a good matching of the model with the full dataset and acceptable CPU effort on a workstation.

Of course, after the forward selection process was done and the centers were selected, the neural network could be validated with the full data set. Its accuracy was measured by the local error and the RMS error.

For the low temperature model all the pressure points were included for a better catch of the near-divergence zone in the region of 1400 K. One radius per variable was used :

$$\begin{bmatrix} r_T \\ r_{war} \\ r_{far} \end{bmatrix} = \begin{bmatrix} 0.8 \\ 3.5 \\ 3 \end{bmatrix} \quad (34)$$

Before entering the forward selection process, all data points are mapped from their respective intervals onto the $[-1; +1]$ range and these radii correspond to normalized variables. They were picked by hand one the basis of numerous tests over realistic values for the radii. The following values correspond to the lowest errors (local and RMS) and to smooth model curves. As was already outlined, it

is almost impossible to lead an optimization algorithm to a satisfying result due to the discontinuous aspect of the forward selection.

The final “low T” neural network gives the very satisfying results written in table 3 with a total number of 47 neurons. The accuracy of the model tended to be not too dependent on the values of the radii.

Error	RMS	Local
Abs. [$J/(kg \cdot K)$]	0.3053	2.7467
Relative [%]	$9.80 \cdot 10^{-3}$	$8.86 \cdot 10^{-2}$

Table 3. *Low T model accuracy*

For the high temperature zone, a large number of networks were built using different radii in order to get a sufficient accuracy. Finally, the following set was used :

$$\begin{bmatrix} r_T \\ r_p \\ r_{war} \\ r_{far} \end{bmatrix} = \begin{bmatrix} 0.65 \\ 1.2653 \\ 4.697 \\ 3.13 \end{bmatrix} \quad (35)$$

It allowed the construction of a network of 308 centers. Its accuracy is given in table 4.

Error	RMS	Local
Abs. [$J/(kg \cdot K)$]	2.6082	37.9565
Relative [%]	$8.4 \cdot 10^{-2}$	1.21

Table 4. *High T model accuracy*

After the fusion of the two networks over a short mixing zone the final model includes two networks of size 50 and 300. Multiquadric functions were used for their better generalization capabilities. The resulting function establishes an explicit and straight relationship between the physical variables and the specific heat of the fuel/air/water mix in a jet engine, valid from the intake to the exhaust nozzle. The overall accuracy table shows that the neural network has a mean performance of about $10^{-2}\%$, which was the required objective. However the peak accuracy of 1 % is perfectible and justifies the introduction, in the following sections, of nonlinear optimization techniques.

The RMS errors presented in tables 3 and 4 were computed on the full data set. All the available thermochemical data were indeed used, which means that the training set contributed to the tabulated errors. It has of course no impact on the local error. As regards the RMS error, one must remember that the training set covers less than 20 % of the full data set. The presented values may then be

considered as representative. Furthermore, it was a project requirement to get an overall measure of the errors.

The comparison of the full test set with predicted values is presented for two (war, far) sample values on figures 6 and 7. The mixing zone is hardly visible on these figures. The C_p value shows very good agreement with the training set data even for the highest temperatures.

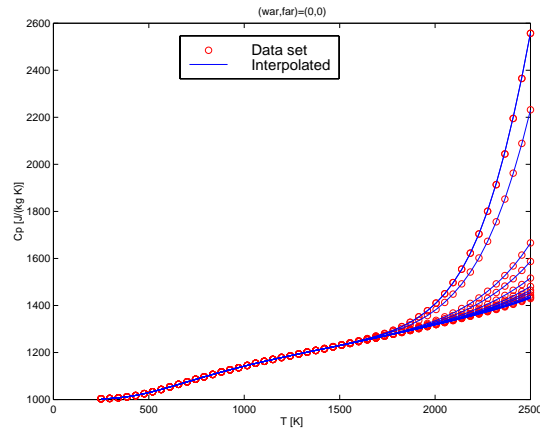


Figure 6. C_p prediction for dry air

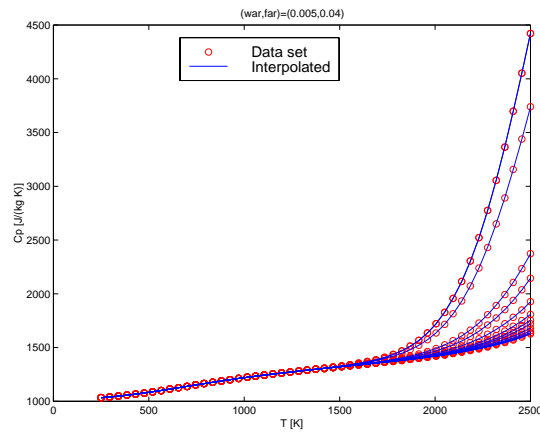


Figure 7. C_p prediction for sample (war, far)

The same methodology was applied to the enthalpy database obtained via CEA [4]. The transition region around 1400 K with a mixing zone for two different networks was adopted.

The global enthalpy varies smoothly, as it appears on figure 8. Picking up the “optimal” radii was therefore easier. The accuracy of the partial and global models is given in table 5. The same training set points were selected for the construction of the simplified database. Hence the same training set sizes, 1008 and 1296 for the “low T” and “high T” models.

The networks include 66 and 290 centers, which is very similar to the C_p network. Table 5 illustrates their performances in terms of accuracy. The absolute errors are written in $J/(kg K)$ and the relative errors in percent.

Network	Error	RMS	Local
Low T	Abs.	119.133	497.024
	Rel.	$5.88 \cdot 10^{-3}$	$2.45 \cdot 10^{-2}$
High T	Abs.	662.805	11760.7
	Rel.	$3.27 \cdot 10^{-2}$	$5.81 \cdot 10^{-1}$
Global	Abs.	480.172	11760.7
	Rel.	$2.37 \cdot 10^{-2}$	$5.81 \cdot 10^{-1}$

Table 5. *Enthalpy model accuracy*

As stated in this table 5, the global relative error is half the one measured on specific heat. Figure 8 shows a sample graph of predicted values of enthalpy through the global RBF neural network, built with multiquadric radial basis functions.

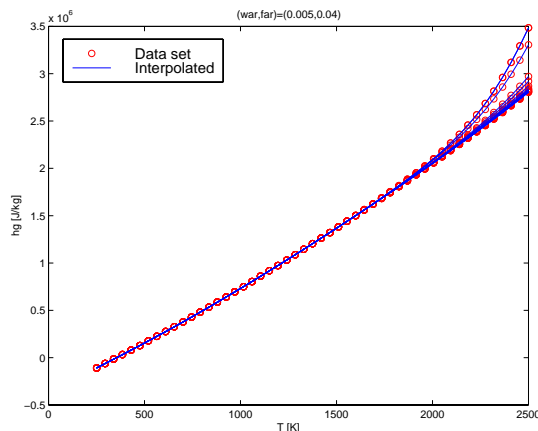


Figure 8. *h_g prediction for sample (war, far)*

Finally, as the neural networks were set up to bypass the full chemical computation while providing a sufficient accuracy, it must be checked whether the speed performances are satisfactory. Three kinds of execution times were investigated : intensive use of the low temperature network, as it would be done for a compressor,

intensive use of the high temperature network, as it would be done for a combustion or reheat chamber, and a span of the whole temperature domain.

From the point of view of the execution speed, multiquadric transfer functions are better than gaussian functions. Our tests have shown that the internal machine execution time of an exponential function is indeed twice as long as a square root. This advantage adds to their good generalization capabilities. Multiquadric also tend to produce somewhat less oscillatory behaviors since they don't show any inflexion point as the gaussians do, which turns to be a decisive advantage for smooth functions such as the enthalpy (figure 8).

Table 6 summarizes the execution times (in milliseconds) of the models on a Compaq DS10 workstation. The processor is an Alpha 21264 EV6 at 466 MHz and has 2 Mb internal cache. Its specfp95 index [6] is 47.1. The functions were written in Fortran 77 and were used on a large scale to get the tabulated values.

Network	Low	High	Mean
C_p	0.0058	0.0448	0.0254
h_g	0.0080	0.0424	0.0253

Table 6. *Model execution times (ms)*

8 Nonlinear optimization

The forward selection tests described in the previous section have outlined the need of adaptative basis functions, able to move or deform following the local needs of the neural network modeling process.

The forward selection phase makes use of radial basis functions centered at the training set points. These data points are added to the system iteratively so that the model is refined while keeping the method linear. But with scarce or badly defined databases appears the need to define centers outside the data points. Besides, the choice of the center location depends on the *a priori* fixed radii and there is no unbiased way to choose them prior to the selection process [2]. Furthermore when the centers are defined, one would like to tune these radii in an individual way to improve the modeling of the training set.

The basic idea of these further developments is to investigate the possibility to apply nonlinear optimization techniques to radii or center positions in the framework of thermodynamic modeling. The project developments rely upon three main steps.

- First, it would be interesting to investigate whether a “forward selected” model can be refined through a nonlinear optimization algorithm applied to the radii of the basis functions. Several radii could be used as optimization variables, in a number ranging from 4 (one per physical input variable) to $4 \cdot m$ (one per center and variable).
- The use of individual function shapes could lead to the definition of a unique neural network covering the whole temperature domain and including much less centers by cleverly tuning the radii.

- The position of the center of a transfer function could be optimized in addition to its radius. This would certainly lead to a large number of possible solutions and global techniques such as genetic algorithms or simulated annealing could become necessary to overcome the question of local minima.

At present only the first step has been achieved. Starting from the neural network defined for the specific heat in the high temperature region, the radii have been optimized to investigate whether the local error could be decreased under 1 %. A subspace-searching simplex algorithm for the unconstrained optimization of general multivariate functions [8] was introduced into the procedure. This method is known to have good global minimum finding capabilities even with a noisy objective function [5]. Moreover the number of function evaluations required for convergence typically increases only linearly with the problem size. This is a great advantage if a full optimization problem is undertaken, with a number of the order of one hundred optimization variables.

With only 4 radii - one per variable (temperature, pressure, *war* and *far*) - the optimization process led to unsatisfying results, whatever the objective function, namely the sum-squared error or the GCV prediction error. In both cases the optimization algorithm tried to decrease the objective function by using lower radii for the temperature. The 1296 data points of the training set were approximated accurately by the neural network, but the model appeared to be oscillatory and over-fitted.

As long as the centers are defined, no square $n \times n$ matrix is required to compute the optimal weights and the objective function. Therefore a larger training set was incorporated in the optimization loop, corresponding to the full high temperature database and containing around 7,000 points.

Two remedies were applied to keep the optimal temperature radii from being too small. Penalties were added to the objective function in order to guide the optimization towards a better and smoother solution, in the form of a positive term added to the SSE and growing as the radii decreased below a typical limit of 0.6.

The second solution consisted of individual radii for different temperature intervals, taking into account the specificities of the $C_p(T)$ curves without dramatically increasing the number of optimization variables.

Some tests were led with these two modifications, from different starting points, showing similar tendencies. Unfortunately the results depend on the scaling of the penalty terms. However the somewhat constrained optimization of the SSE observed on the full high temperature database gives rise to very different radii values for the temperature, ranging from 0.35 to 0.8. The other radii (pressure, *war* and *far*) nearly stuck on their initial value (35).

The behavior of the temperature radii fully justifies further developments. The previous neural networks were all built using the same radius for the whole range of each variable. The optimization process has showed the need to adapt the shape of the radial basis transfer functions for individual centers, or at least for some groups of centers.

Depending on the starting point, the optimization process allowed to decrease the RMS error from 2.6302 (see table 4) to 2.535 J/kg.K. But it increased the local error from 37.96 to 38.87 J/kg.K. This result is of course a little disappointing.

Extra work is under way for future developments.

9 Conclusions

This paper illustrates the application of neural networks using radial basis functions to fit thermodynamic data. The networks are used to bypass costly chemical computations by synthesizing a pre-computed database. The radial basis function neural networks are obviously well suited for data approximation. Several solutions were investigated as regards to the choice of the neurons, i.e. the number, positions and shapes of the transfer functions.

The forward selection technique is quite powerful due to the linear aspect of the computations. The choice of the centers is automatic and it allows the definition of the best neural network, given the database and the shape of the basis functions. As a result two explicit thermodynamic functions provide the specific heat and enthalpy of the gas traveling through a jet engine. The execution times were small enough to use the functions inside Newton-Raphson loops or real-time jet engine monitoring.

However the transfer functions should be able to vary in shape and/or position to provide increased generalization capabilities. This is possible if the set up of the neural network is supervised by an external optimization algorithm. This strategy has been investigated, but the size of the problem becomes rapidly prohibitive. Several solutions are still to be tested. Among them, the possibility to let the centers move and to optimize their position seems very promising. However it is likely to require more powerful optimization techniques, such as nonlinear gradient descent or genetic algorithms.

Another possibility that will be explored if the nonlinear optimization does not give satisfactory results is to use several hidden layers in the neural network, although it increases considerably the computation effort.

Bibliography

- [1] ANON., *Manual of the Standard ICAO Atmosphere*, SECOND EDITION, INTERNATIONAL CIVIL AVIATION ORGANIZATION, 1964
- [2] C.M. BISHOP, *Neural Networks for Pattern Recognition*, OXFORD UNIVERSITY PRESS, NEW YORK, 1995
- [3] M.W. CHASE JR., *NIST-JANAF Thermochemical Tables*, FOURTH EDITION, JOURNAL OF PHYSICAL AND CHEMICAL REFERENCE DATA, MONOGRAPH NO. 9, PARTS I & II, 1998
- [4] S. GORDON, B.J. MCBRIDE, *Computer Program for Calculation of Complex Chemical Equilibrium Compositions and Applications*, NASA REFERENCE PUBLICATION 1311, OCTOBER 1994
- [5] D.M. HIMMELBLAU, *Applied Nonlinear Programming*, MCGRAW-HILL, NEW-YORK, 1972
- [6] <http://www.specbench.org>
- [7] M.L.J. ORR, *Introduction to Radial Basis Function Networks*, CENTRE FOR COGNITIVE SCIENCE, UNIVERSITY OF EDINBURGH, APRIL 1996
- [8] T. ROWAN, *Functional Stability Analysis of Numerical Algorithms*, PH.D. THESIS, DEPARTMENT OF COMPUTER SCIENCES, UNIVERSITY OF TEXAS AT AUSTIN, 1990
- [9] A. TIKHONOV, V. ARSÉNINE, *Méthodes de Résolution de Problèmes Mal Posés*, MIR EDITIONS, MOSCOW, 1976