Mathematically Reduced Chemical Reaction Mechanism Using Neural Networks

Technical Progress Report

Period ending: 09/30/2004

Prepared by

Nelson Butuk

Principal Investigator

Department of Mathematics
Prairie View A & M University
Prairie View Texas. 77446-4189

December, 2004

DOE Grant Number: DE-FG26-03NT-41913

Office of Sponsored Programs

Prairie View A & M University
P. O. Box 667
Prairie View Texas 77446-0667

Disclaimer

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

Abstract

This is an annual technical report for the work done over the last year (period ending 9/30/2004) on the project titled "Mathematically Reduced Chemical Reaction Mechanism Using Neural Networks." The aim of the project is to develop an efficient chemistry model for combustion simulations. The reduced chemistry model will be developed mathematically without the need of having extensive knowledge of the chemistry involved. To aid in the development of the model, Neural Networks (NN) will be used via a new network topology known as Non-linear Principal Components Analysis (NPCA).

We report on the development of a procedure to speed up the training of NPCA. The developed procedure is based on the non-parametric statistical technique of kernel smoothing. When this smoothing technique is implemented as a Neural Network, it is known as Generalized Regression Neural Network (GRNN). We present results of implementing GRNN on a test problem. In addition, we present results of an in-house developed 2-D CFD code that will be used throughout the project period.

TABLE CONTENTS

Disclaimer

Abstract

Introduction 5

Results and Discussions 9

Conclusion 17

References 17

Introduction

This is an annual technical report for the work done over the last year (period ending 9/30/2004) on the project titled "Mathematically Reduced Chemical Reaction Mechanism Using Neural Networks." The aim of the project is to develop an efficient chemistry model for combustion simulations. The reduced chemistry model will be developed mathematically without the need of having extensive knowledge of the chemistry involved. To aid in the development of the model, Neural Networks (NN) will be used via a new network topology known as Non-linear Principal Components Analysis (NPCA).

Many Combustion systems are modeled by very high-dimensional systems of non-linear differential equations. These equations often exhibit solutions which are un-evenly distributed in phase-space, and which may exist as circles, tori or other manifolds. It is desirable to approximate these isolated regions of the phase-space by a mathematical model of lower dimension than the dimension of the original ambient space. We propose to develop NPCA to accomplish this task using NN.

Given a data set $X \in \mathbb{R}^n$ NPCA determines a reduction mapping

$$G: X \rightarrow Y$$

where the set $Y \in \mathbb{R}^m$ has reduced dimension $m < n$. Y is then said to be a reduction of X . The inverse mapping reproduces the original data X from the reduced data set Y as

$$H: Y \rightarrow X$$

Hence, NPCA is a composition of mappings which is the same as the identity mapping, i.e.

$$H \circ G: X \rightarrow X$$

The NPCA is a NN topology with five layers, in which the input layer has the same number of nodes as the output layer. This allows the input values to be used as target output values of the network during training. The first part of the network approximates the mapping G . It contains the mapping layer. The middle layer is the bottle-neck layer consisting of the m nodes (i.e. desired reduced dimension m). The last part of the network implements the mapping H and contains the de-mapping layer. This layer takes the output of the middle layer and maps it onto the output layer. It is on the bottle-neck layer, that the reduced manifold of the chemistry is reconstructed.

The Objectives of this research are therefore:

1. Improve the training rate of the NPCA-NN algorithm developed previously.
2. Develop a method to determine optimum trajectory data of reaction mechanisms needed to use NPCA-NN.
3. Apply the NPCA-NN algorithm to the reduction of Dimethyl ether (DME) mechanism.
4. Couple the developed NPCA-NN model to the KIVA CFD code.
5. Test the CFD code on a few other simple sample mechanisms
6. Student Education in Computational Applied Mathematics

Over the past year we have concentrated on the first objective and developed our own in-house CFD code in preparation for objective 4. The last objective has also been addressed as two students (Ken Johnson and Owen Clarke) have been and continue to be trained in computing under the direct support of the project. This report will address what has been accomplished so far.

Among the different implementation of Neural Networks (NN), the network structure that posses the fastest training time, is the Generalized Regression Neural Networks (GRNN). Over the past year, we have studied this network with the aim of incorporating its desirable properties into our NPCA-NN data reduction algorithm. Our results so far including current investigations (to be reported later) are promising. Below we will give a brief description of GRNN, present results of actual implementation on a test problem and highlight some of its advantages.

Generalized Regression Neural Network:

The mathematical background for GRNN is based on the statistical kernel smoothing techniques. Kernel smoothing provides a simple way of finding structure in data set without the imposition of a parametric model (i.e models based on basic functions such as logs, exponents, sines, and cosines). This smoothing without using known models is known as non-parametric regression.

Consider a parametric regression in which you want to fit data to a linear function. Given a data set

$$\{(X_i, Y_i)\}_{i=1}^n$$

The linear fit to the data is of the form:

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i \quad (1)$$

β_0 and β_1 are model parameters determined from data using appropriate numerical fitting techniques. ε_i are the errors of the approximation assumed to be symmetric random variables having mean zero.

Call the developed approximating function $m(X)$. The function m for which

$$E\{Y - m(X)\}^2$$

is minimized, is the conditional mean Y given X :

$$m(X) = E(Y | X) \quad (2)$$

in non-linear regression theory, $E(Y|X)$, the expected value of Y given X is:

$$E[Y | X] = \frac{\int_{-\infty}^{\infty} Y f(X, Y) dY}{\int_{-\infty}^{\infty} f(X, Y) dY} \quad (3)$$

where $f(X, Y)$ is the joint probability density function (pdf) of X and Y .

Now if we remove the restriction that m belongs to a particular family of models (i.e let the data speak for themselves) then you now have what is known as non-parametric regression. The kernel smoothing approach of GRNN is based on “Parzen’s Method of Density Estimation” first proposed by Parzen in 1962.

The method to be described below is a way of estimating the pdf, $f(X,Y)$ above, given only a sample data set $\{(X_i, Y_i)\}_{i=1}^n$.

Suppose we have an unknown sample whose value of X variable is x . We want to know the estimate y of Y given this value of X . Compute:

$$D_i^2 = (x - X_i)^T (x - X_i)$$

$$h_i = \exp\left[-\frac{D_i^2}{2\sigma^2}\right] \quad i = 1 \dots n \quad (4)$$

Several methods of choosing σ are available and can be different for each data point X_i . In this work we have used a constant σ .

The approximation for y given x will now be given by:

$$y = \frac{\sum_{i=1}^n h_i Y_i}{\sum_{i=1}^n h_i} \quad (5)$$

GRNN, implements the above theory in a Neural Network, which consist of four layers (Figure 1). The first layer is the input layer, which is fully connected to the next layer and is just used to distribute the input vector to each node in the next layer. The second layer is the first hidden layer (also called Pattern Layer). This layer consists of N Processing elements or nodes, where N is the number of samples within a training data set and each node represents the input vector, X_j , associates with the j -th sample (or observation) in training data. In each node, each input vector is subtracted from the vector assigned to the node, X_j . This difference is then squared by the node. The result is fed into a non-linear Kernel, which is usually an exponential function as shown in equation 4 above.

Figure 1. Neural Network Implementation of Generalized Regression

The choice of the smoothing parameter of the Kernel, σ , is important. It has the effect of smoothing the training examples. Small values of σ tend to make each training point distinct, whereas large values will force a greater degree of interpolation between the training observations. The output of the Kernel function is a weighted average of the outputs of each training observation. The weighting is related to σ and the distance of the input vector from X_j the node vector. Therefore, points nearby to the node vector contribute most heavily to the overall output of the Kernel. Choosing, σ , determines the width of an area in the input space to which each node responds.

The third layer, is the 2nd hidden layer (or Summation layer) and consists of only two nodes. The input to the first node, is the sum of the 1st hidden layer outputs (or activations) each weighted by the observed output y_j corresponding to X_j . In other words, the input is simply a dot product between the 1st hidden layer outputs and the desired network output vector, y . The input to the 2nd node, is the summation of the 1st hidden layer activations. The 1st node represents the numerator of the above equation 5, and the 2nd node represents the denominator.

The fourth layer is the output layer. It receives the two outputs from the 2nd hidden layer and divides them according to equation 5, above to yield an estimate for y . Note that the 2nd hidden layer always has exactly one more node, than the output layer. The 2nd hidden-to-output layer weights are just the target values, y , so that the estimated network output, is simply a weighted average of the target values, y , of the training observations close to a given input case.

When you need a multidimensional (vector) output, the only change to the network, is to add one additional node to the 2nd hidden layer, plus an additional node in the output layer for each element of the output vector.

Overall, the GRNN has only a single parameter, σ , that needs to be determined during calibration (training), and therefore can be trained fast. σ is optimized during training of the network. Available data is split into two sets-training and test sets. Cross-validation with the test set is then used to determine the optimum value of σ . Alternatively, σ , can be computed directly from data. For each 1st hidden layer node, find the RMS distance between its vector, X_j , and the center of its M , nearest neighbors. Assign this value to σ_j . Then assign the average of all σ_j 's to σ . Another characteristic of GRNN, which makes it fast in training, is that there is only the linear output layer beyond the 1st hidden layer, this guarantees that the network will converge to a global minimum. The linearity is with respect to the weights of the network and not the input variables x . Finally for large scale problems the GRNN can be efficiently parallelized and implemented in a super computer with many computing nodes. This is because the GRNN consists of a large number of simple processes that can be executed mostly in parallel.

Results and Discussions

The project aim as indicated above is to develop an algorithm for dimension reduction of chemical systems. The problem may be posed mathematically as

$$\begin{aligned} f_1(x_1 \dots x_n) &= y_1 \\ f_2(x_1 \dots x_n) &= y_2 \\ &\vdots \\ f_n(x_1 \dots x_n) &= y_n \end{aligned} \quad (7)$$

If we consider only f_i , then $x \in \mathbb{R}^n$ and $y \in \mathbb{R}$. The dimension of the problem in this case is n , which can be quite large for most chemical systems. The dimension reduction problem is to reduce the dimension of the input variables from n to m , where m is much smaller than n . If the reduced variables are represented as $z \in \mathbb{R}^m$ then our problem can be modeled in terms of z as

$$g(z_1 \dots z_m) = \hat{y} \quad (8)$$

Our objective then is to be able to use GRNN to determine z and g , so that $\|y - \hat{y}\|$ is minimized. Here we discuss and present results of work accomplished so far.

Our first step was to implement and thoroughly understand the GRNN. The algorithm to implement GRNN was written and implemented in MatLab. In order to test the algorithm, we have used the following non-linear function.

$$f(x_1, x_2, x_3) = 4x_1^2 + 2x_2^2 + x_3^2 - x_1x_2 + x_2x_3 - 5x_1 - 9x_2 + x_3$$

(9)

This is 3-dimensional function and our objective is to reduce it to a one and two dimensional function. A MatLab m-file was written to implement the function, which was then used to automatically generate N, observations or data points for $(x, x_2, x_3) \in [-10, 10]$. The N observations formed the set of training data. For test data only 20 points were generated again at random in a similar way as the training data. The results for generating 500 and 1500 data points is presented below. The MatLab files are also attached below.

```
>>[xt ft yt]=testing2(500)
```

```
function [xt,ft,yt] = testing2(n)
[xd yd]=testf(n);
c=xd;
gama=1.4;
H = grnn2(xd,c,gama);
w=yd;
[xt yt] = testf(20);
Ht = grnn2(xt,c,gama);
ft = (Ht*w)./sum(Ht,2);
x=[1:1:20];
hold on
plot(x,yt,'r-',x,ft,'*');
ylabel('function value, f');
xlabel('Observation');
title('GRNN Function Prediction 500 Points');
legend('Actual','GRNN Estimate',1);
hold off
```

```
function [xd,y]=testf(nd)
% this will generate nd data points x and y for regression analysis
% of a particular function
id=3;
xd=10*rand(id,nd);
xd = 2*xd - 10;
xd = sign(xd).*min(10,abs(xd));
y=zeros(nd,1);
x=zeros(1,id);
    for j=1:nd
        x = xd(:,j);
        y(j)= 4*x(1)^2 + 2*x(2)^2 + x(3)^2 - x(1)*x(2) + x(2)*x(3) - 5*x(1) -
9*x(2) + x(3);
    end
```

```
function H = grnn2(X, C, R)
% Function to implement Generalized Regression Neural Network
% It Calculates the Matrix of 1st Hidden Layer Activations
[d,p] = size(X);
if size(C,1) ~= d
    error([prog ': Dimensions of C and X do not Match'])
end
```

```

m = size(C,2);
% Calculate the H Matrix
H = zeros(p,m);
for j = 1:m % m is total nodes in Hidden Layer
    c = C(:,j);
    A = X - c(:,ones(1,p));
    Z = sum(A.^2)./(2*R^2);
    H(:,j) = exp(-Z');
end

```

Figure 2 GRNN Results Using 500 Points

Figure 2 GRNN Results Using 1500 Points

Current efforts are to develop the GRNN as NPCA-NN to be used for dimension reduction. The results of this task will be reported later. The figure below is a pre-view of ongoing work.

Figure 3. Pre-view of Ongoing Work

Discussion of CFD Results:

In preparation for the 4th objective, we have implemented our own simple 2-D Euler solver. This code will be interfaced to CHEMKIN and used to test our NPCA-NN mathematical model prior to interfacing the model to the popular KIVA CFD code.

The code chosen was the 2-D Euler solver described in the book by Michael Lobo, titled; Time-Marching a step-by-step guide to a flow solver, Ashgate Publishers, 1997. The book contained a detailed description of a two dimensional flow solver with the necessary background mathematics explained in details. The complete code in FORTRAN programming language is now running successfully. The code was tested on a simple problem, which was to solve for steady flow of air through a plane symmetrical diffuser. In this problem the viscosity and density of air were taken to be 1.91×10^{-5} kg/ms and 1.21 kg/m^3 , respectively. The flow profile at the inlet was assumed to be flat with a bulk velocity of 160 m/s. The flow at the outlet was assumed to be fully developed. The results of the computation visualized in Tecplot are attached. The results are for the grid profile, the Mach number, and static pressure distribution.

Grid Dimensions in M

Mach Number

Conclusion

Preliminary results for the development of a fast training method for NPCA-NN has been presented. GRNN was implemented and found to be a fast Neural Network implementation with many desirable properties that can be used to speed up NPCA-NN training. Methods of incorporation GRNN features to NPCA-NN are currently being addressed. In addition preliminary CFD results have been presented for a 2-D Euler solver. Work will proceed to incorporate chemistry into this code.

Reference

1. Caudill M., "GRNN and Bear It" , AI Expert, May 1993, pp 28-33
2. Kramer, M.A., "Nonlinear Principal Component Analysis Using Autoassociative Neural Networks", AIChE Journal, No 2, Vol, 37, 1991, pp 233-243
3. Specht, D.E., "A Genral Regression Neural Network", IEEE Trans. Neural Networks, No 5, Vol. 2, 1991, pp 568-892.