SOFTWARE TEST REPORT

OPTIMIZATION TOOLS TEST REPORT

Submitted to:

SPAWAR Systems Center San Diego
53560 Hull Street, Code 2206
San Diego, California 92152-5001

Attention:
Mr. Mark Tukeman, Code 2712
(619) 553-1510

In fulfillment of the requirements for:

Contract No. N66001-02-D-0039
Development of Port Related Transportation Technologies
to Advance Military Responsiveness to National Needs

Task Order No. 0011
Computational Fluid Dynamic (CFD) Design Tool Development and Validation

CDRL Data Item No. A002BA

Security Classification: Unclassified

Prepared and Submitted by:

Center for the Commercial Deployment of Transportation Technologies
California State University, Long Beach Foundation
6300 State University Drive, Suite 220 • Long Beach, CA 90815 • 562.985.7394

Approved for public release: distribution is unlimited

September 30, 2003

Software Test Report (STR)

Computational Fluid Dynamics (CFD) Tool Development

Deliverable 2

Optimization Tool development based on Neural Networks

Qualification and Validation

P/E Number: 2.20

Task Order Number: 11

POC

Hamid Hefazi, Professor & Chair

Mechanical and Aerospace Engineering Department

California State University, Long Beach

hefazi@csulb.edu

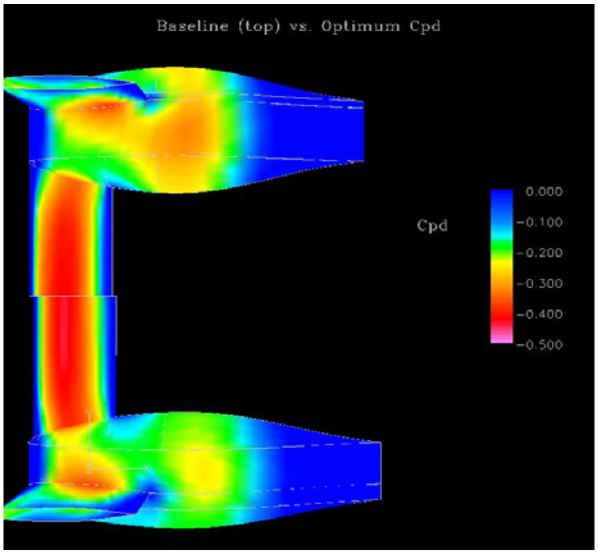

Table of Contents

This report presents the validation tests that are made on the software product described in the accompanying Computer Software Product End Item of Ref. 1. Section 1 describes the model problem used to validate the software and analyze the results of the method for generating the improvements discussed in Ref. 1. Section 2 presents the input and output data for the model problem and is followed by an analysis of the results.

1. Model problem description

The ability of using the Neural Network instead of the CFD method in the optimization process of Fig. 1 relies on the ability of the former to best represent the latter over the selected design space.

The objective of this report is, therefore, to evaluate –and validate– the theoretical characteristics of the method described in Ref. 1 on a model problem which, while being representative of a complex CFD optimization problem, is less demanding in computing resources and has a more focused scope. Such approach enables evaluating and refining the NN which is then be used for the complex three-dimensional CFD optimization of Refs. 2 and 3. Such CFD optimization model problem is defined in this section and its application is discussed in the following sections.

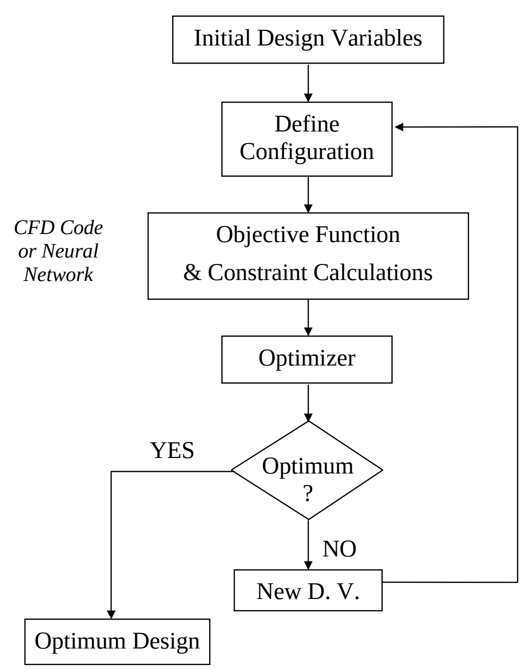

Fig. 1 Flowchart of the numerical optimization. In the NN approach, the CFD code is replaced by a NN evaluator which has been trained *a priori* – See Ref. 1 for details

The model problem consists of optimizing a two-dimensional hydrofoil shape for ***minimum drag*** with the following ***constraints***:

- $A > 0.9A_0$, where A is the area of the foil and A_o is the area of the initial hydrofoil. This constraint ensures that the buoyant lift is at least 90% of the initial hydrofoil.
- $C_p = (p - p_{atm})/q_\infty > -0.55$, which is equivalent to a cavitation free flow at 45 kts on a low aspect ratio body. Here, p denotes pressure, p_{atm} atmospheric pressure, and q_∞ the free stream dynamic pressure.
- Free stream velocity: 40 kts (20.6 m/s).
- Depth-to-chord ratio of 0.3.

The shape of the hydrofoil is represented by a base hydrofoil to which Hicks-Henne functions are added. As such, the foil upper and lower surfaces coordinates, y, using the shape functions $(f_i)_{1\le i\le s}$ become:

$$y(\bar{x}) = y_0(\bar{x}) + \sum_{i=1}^{s} x_i \, f_i(\bar{x}) \qquad (2.1)$$

where $\bar{x}$ is the coordinate along the airfoil chord, y_0 is a reference airfoil, e.g. a NACA 0012 airfoil, $(x_i)_{1\le i\le s}$ are the design variables. The Hicks-Henne functions are used for the present application and are given by [4]:

$$\begin{aligned} f_i(x) &= x^a(1-x)e^{-bx} \\ f_j(x) &= \sin\left(\pi x^{\frac{\ln(0.5)}{\ln(a)}}\right)^b \end{aligned} \qquad (2.2)$$

where a and b control the center and thickness of the perturbation, and x is the normalized coordinate along the chord. They have the advantage of being space-based functions, as opposed to frequency-based functions, and thus allow for greater local control of the design. This representation is well suited for aero- and hydrodynamic shapes and has been used successfully by several authors.[5-7] A total of eight design variables, four on the upper surface and four on the lower surface, were chosen to represent the parametric shape.

The training sets consist of databases with values for the objective function and constraints for each set of design variable (or “design point”). The data was generated by computing the flow with a two-dimensional Interactive Boundary Layer approach (IBL) [8] in which the free surface is modeled by negative images. [9] This approximation is valid for high Froude numbers, which is the case here. One such flow analysis typically requires a few seconds of CPU time, in contrast with a full three-dimensional viscous free-surface computation which can take up to several hours. The use of such "fast" model problem was key in delivering quick turn around time during the NN development phase, especially since the observations made on the model problem should be applicable to the problem with larger scope.

2. Validation and analysis

This section presents qualification and validation of the neural network program NN_OPT applied to the 2D Model Problem. It must be pointed out that the goal here is not to completely optimize the model problem. This has been done in the previous CCDoTT work and reported in Reference [10] and thus, it is not repeated here. The objective of this validation exercise is only to confirm that the neural networks and the new software perform properly. Therefore the model problem is simplified and we choose the objective function to approximate to be the "Lift" of the configuration (8 inputs, 1 output). The database files used for the Training Set, Validation Set and Generalization Set are respectively Optim33.db (size 486), Optim.31.db (size 97) and Optim35.db (size 1955). They have been generated by iSIGHT using the Latin Hypercube routine. Ten networks are build, with a pool of 7 candidate hidden units, the correlation formula S_{CASCOR} is used. No stopping criterion is used, so training will continue for each network until the maximum number of hidden units is reached (70) and the best number of hidden units for this network will be decided by the number of HUs corresponding to the minimum validation error. The best out of those ten networks is chosen by the program and its weights are saved in the file best_weights.data. This file is formatted as a C++ program and serves as main file for the Evaluator Program as described in [1]. For this example, the third network constructed leads to the best network, the squared error on the validation set divided by the size of the VS is 1.80512e-05. The network contains six hidden units. The time needed to build those ten networks is 403 minutes.

All file formats have been described in Reference 1, and will not be repeated here.
The following subsections show the listings for the input file (input.dat), one of the database files (Optim33.db), the main output file and the weights file (best_weight.data). The last subsection explains the results obtained for this example.

2.1. Input file

```
#****File input.dat for DBs created for 2D-Model problem (8
inputs)********************************
#inputs: cols 3 to 10 // outputs drag=col 12,lift= col 13,Surface=col 14,Cpmin=col 15,
Total lift (buoyant+dynamic)=col 16
#DataBase name for the Training Set (MAX 80 CHARACTERS) Include path if not in current
directory
        /disk6/mu/schmitz/Neural_net/CCDoTT_02/TS/Optim33.db
#DataBase name for the Validation Set
        /disk6/mu/schmitz/Neural_net/CCDoTT_02/TS/Optim31.db
#DataBase name for the Generalization Set (RM: if none enter 0)
        /disk6/mu/schmitz/Neural_net/CCDoTT_02/TS/Optim35.db
#InputArray
#ninputs        noutputs        TSsize  VSsize  GSsize  ntrys   ncandidates     Correlation
formula( 0 to 6 for SCasCor,S1,S2,S3,Sqrt_S1,Sqrt_S2,Sqrt_S3)
 8              1               486     97      1955    10      7                       0
# Stopping Criterion ( 0 to 12,
StopCrit={NONE,GL1,GL2,GL3,GL5,PQ005,PQ0075,PQ01,PQ02,PQ03,UP1,UP2,UP3})
0
#DBInputCol: input columns to be read from DB (ninputs)
3       4       5       6       7       8       9       10
#DBOutputCol: output columns to be read from DB (noutputs)
13
```

2.2. Database file

```
RunCounter      NbDV    DVu0    DVu1    DVu2    DVu3    DVl0    DVl1    DVl2    DVl3    Cm
        Drag    Lift    Surface CpMin   TotalLift       Objective       Penalty
        ObjectiveAndPenalty     Phase   Feasibility     Rules   Control Approach
        Analysis        Internal        Technique       TechniqueData
18N     18N     18N     18N     18N     18N     18N     18N     18N     18N     18N     18N
        18N     18N     18N     18N     18N     18N     18N     18N     18N     18N     18N
        18N     18N     18N     18N     18N
1       4       0.0     0.01476 0.002493        0.006794        -0.007295       0.01057 -
0.0003925       0.003387        -0.018316       0.005961        0.129000        0.083809
        -0.389126216960302      174671.7959952 0.0      0.0     0.0     3       9       none
        5       0       0       0       doe_Study0      {DOEStart(DOE;doe_Study.DOE)}
2       4       4.008e-05       0.007352        0.008217        0.002966        -0.0007616
        0.004081        0.01928 0.0004008       -0.028395       0.005915        0.147100
        0.079378        -0.425963819347567      199017.212247705        0.0     0.0     0.0
        3       8       none    5       0       0       0       none    {DOE(DOE;Exp:1,2)}
3       4       8.016e-05       0.01942 0.01255 0.002585        -0.01735        0.002157
        0.01534 0.001683        -0.028210       0.005904        0.148000        0.087590
        -0.432890813832097      200312.5469529 0.0      0.0     0.0     3       8       none
        5       0       0       0       none    {DOE(DOE;Exp:1,3)}
...
...
497     4       0.02    0.02    0.02    0.01    -0.001323       0.02    0.02    0.01    -
0.062336        0.005574        0.309500        0.082243        -0.453576703097717
        417882.004019475        0.0     0.0     0.0     3       8       none    5       0
        0       0       none    DOEResult(DOE,bestlevels)
```

2.3. Main output file

```
        Output file generated by NN_OPT_1

        Reading input data
        #****File input.dat for DBs created for 2D-Model problem (8
inputs)********************************
        #inputs: cols 3 to 10 // outputs drag=col 12,lift= col 13,Surface=col 14,Cpmin=col
15, Total lift (buoyant+dynamic)=col 16
```

```
        Number of Inputs=8     Number of Outputs=1
        DataBase file for Training Set:
        /disk6/mu/schmitz/Neural_net/CCDoTT_02/TS/Optim33.db  size=486
        DataBase file for Validation Set:
        /disk6/mu/schmitz/Neural_net/CCDoTT_02/TS/Optim31.db  size=97
        DataBase file for Generalization Set:
        /disk6/mu/schmitz/Neural_net/CCDoTT_02/TS/Optim35.db  size=1955
        Number of trys to be completed before choosing the best Network:    10
        Number of candidate units to try before adding new HU to network (<10 Opt, >10 Max
Cov method):   7
        Correlation formula 0 to 6 (SCasCor,S1,S2,S3,Sqrt_S1,Sqrt_S2,Sqrt_S3)0
        Stopping Criterion(0 to 12,
StopCrit={NONE,GL>1,GL>2,GL>3,GL>5,PQ>0.05,PQ>0.075,PQ>0.1,PQ>0.2,PQ>0.3,UP1,UP2,UP3}):
        0
        Columns to be read in DB files for input:    3      4      5      6      7
        8      9      10
        Columns to be read in DB files for output    13
        Finished reading input data
------------------------------------------------------------------------------------------
----------------------
-       Training Set   -
Reading database file: /disk6/mu/schmitz/Neural_net/CCDoTT_02/TS/Optim33.db


*********************************Non-
Dimensionalization*******************************************************
For the training set
               MinInput[j] =          0  MaxInput[j] =       0.02
               MinInput[j] =     -0.004  MaxInput[j] =       0.02
               MinInput[j] =     -0.004  MaxInput[j] =       0.02
               MinInput[j] =          0  MaxInput[j] =       0.01
               MinInput[j] =      -0.02  MaxInput[j] =          0
               MinInput[j] =     -0.004  MaxInput[j] =       0.02
               MinInput[j] =     -0.004  MaxInput[j] =       0.02
               MinInput[j] =          0  MaxInput[j] =       0.01
               MeanTarget[j] =   0.159774
------------------------------------------------------------------------------------------
----------------------
 New values for inputs and targets
 Inputs are rescaled between [0,1], and outputs are divided by the mean
 Following Columns are inputs(j,i) for j=1,n and targets(l,i) for l=1,m  with i=1,486
    0          0     0.781667     0.270542       0.6794      0.63525     0.607083
0.150312       0.3387     0.807389
    1     0.002004        0.473     0.509042       0.2966      0.96192     0.336708
0.97      0.04008     0.920674
    2     0.004008     0.975833     0.689583       0.2585       0.1325     0.256542
0.805833       0.1683     0.926307
    3      0.00601         0.94     0.965833       0.3768      0.77155     0.060125
0.567167       0.3367     0.952594
    4     0.008015     0.392833     0.254542       0.8898      0.99198     0.721667
0.35675       0.1463     0.925055
    5      0.01002     0.282583       0.7175       0.7254       0.2705     0.905833
0.6275       0.8958      1.45956
    6     0.012025     0.262542      0.93375       0.3507       0.2445      0.91375
0.695417      0.06613       1.0809
    7      0.01403     0.821667      0.36475       0.5792        0.032     0.585417
0.452958       0.5411      1.01643
    8      0.01603      0.76375      0.12025       0.3567       0.3105     0.346708
0.491       0.6673     0.861215
    9     0.018035     0.372792     0.967917       0.7475       0.7595     0.711667
0.336708       0.1403      1.05461
…
…
…

  485          1            1            1            1      0.93385            1
1            1      1.93711
------------------------------------------------------------------------------------------
----------------------
-       Validation Set -
Reading database file: /disk6/mu/schmitz/Neural_net/CCDoTT_02/TS/Optim31.db
```

```
-------------------------------------------------------------------------------------------
----------------------
 New values for inputs and targets
 Inputs are rescaled between [0,1], and outputs are divided by the mean
 Following Columns are inputs(j,i) for j=1,n and targets(l,i) for l=1,m  with i=1,97
    0          0     0.414083           0       0.3434      0.3435      0.88875
0.807917       0.1313     0.888128
    1     0.0101     0.727083      0.1313       0.6969      0.8585     0.535292
0.63625       0.7878      1.17603
    2     0.0202     0.979583    0.717083       0.8585      0.5656     0.787917
0.565583       0.8282      1.47208
    3     0.0303       0.1717    0.656667        0.505      0.6161       0.7575
0.979583       0.5454      1.33876
    4     0.0404       0.1515    0.424208       0.2323      0.0705    0.0404167
0.101            1     0.722269
    5     0.0505    0.0909167    0.929167       0.2424      0.8787     0.575708
0.89875       0.3838      1.16414
…
…
…
   96      0.596            1           1            1      0.6161            1
1            1      1.91896
-------------------------------------------------------------------------------------------
----------------------
-      Generalization Set    -
Reading database file: /disk6/mu/schmitz/Neural_net/CCDoTT_02/TS/Optim35.db
-------------------------------------------------------------------------------------------
----------------------
 New values for inputs and targets
 Inputs are rescaled between [0,1], and outputs are divided by the mean
 Following Columns are inputs(j,i) for j=1,n and targets(l,i) for l=1,m  with i=1,1955
    0          0     0.699167       0.735        0.818       0.288     0.974167
0.695417       0.6969       1.4896
    1    0.0005005     0.530458    0.869583       0.4518      0.6882     0.488917
0.265208       0.7725      1.08027
    2     0.001001     0.219167    0.786667       0.9496       0.147      0.28275
0.613333        0.858      1.34565
    3     0.002002    0.0390417     0.83125       0.9581     0.53555     0.220667
0.219167       0.4077     0.956975
    4    0.0025025     0.923333    0.877083       0.3067     0.55455     0.716667
0.257208        0.791      1.12221
    5     0.003003     0.133112    0.627083       0.4122      0.9044     0.146121
0.362292       0.4973     0.806137
…
…
…
1954       0.985            1           1            1           1           1
1            1      1.93523
************************************Try number 1***********************************
E/Pmax After Optimization- no HU: 2.87728e-05

 ....... #HU (E/Pmax)tr                (EMax)tr     (E/p)val        (EMax)val
       (E/p)gen        (EMax)gen                     GL[h]           PQ[h]          UPs[h]
.......    1   2.14338e-05      0.0225970    2.73458e-05      0.0203295    2.34744e-05
0.0247901          0.00000       0.00000              0
.......    2   1.87312e-05      0.0193056    2.31156e-05      0.0161325    2.07863e-05
0.0271225          0.00000       0.00000              0
.......    3   1.78384e-05      0.0192007    2.34628e-05      0.0181199    1.99869e-05
0.0255502          1.50186       0.00000              0
.......    4   1.63920e-05      0.0173375    2.29731e-05      0.0196657    2.03174e-05
0.0240866          0.00000       0.00000              0
.......    5   1.54422e-05      0.0154955    2.19982e-05      0.0202779    2.15115e-05
0.0264148          0.00000       0.00000              0
.......    6   1.48063e-05      0.0156305    2.30018e-05      0.0199623    2.18623e-05
0.0277272          4.56216       0.00000              0
.......    7   1.42903e-05      0.0165586    2.33096e-05      0.0211070    2.18068e-05
0.0263231          5.96160       0.00000              0
.......    8   1.36861e-05      0.0164403    2.43941e-05      0.0234750    2.25663e-05
0.0246117          10.8913       0.00000              0
.......    9   1.29327e-05      0.0163794    2.40383e-05      0.0227942    2.29819e-05
0.0233866          9.27435       0.00000              0
```

```
.......  10   1.24346e-05      0.0168899    2.46604e-05      0.0245917    2.37044e-05
0.0246065          12.1019       0.125883              1
…
…
…                0
.......  70   1.62640e-08     0.00116031    3.21549e-05      0.0188888    3.72486e-05
0.0286285          46.1709       0.120517              1

For this try, the optimum result was found for h=5 hidden units
Best mean squarred error Eval(tmin)= 2.19982e-05
Calculation Time        : 39 min 33 sec
Saving weight file in weights/weights_try_XX.data
**************************************Try number 2***********************************
E/Pmax After Optimization- no HU: 2.75729e-05

 ....... #HU (E/Pmax)tr                (EMax)tr     (E/p)val        (EMax)val
       (E/p)gen        (EMax)gen                    GL[h]          PQ[h]          UPs[h]
.......   1   2.15808e-05      0.0236005    2.78362e-05      0.0207328    2.38600e-05
0.0235745          0.00000        0.00000              0
.......   2   1.86741e-05      0.0200202    2.29628e-05      0.0165717    2.14191e-05
0.0262292          0.00000        0.00000              0
.......   3   1.72266e-05      0.0181260    2.31206e-05      0.0184325    2.19550e-05
0.0281701         0.686941        0.00000              0
.......   4   1.62708e-05      0.0188199    2.45549e-05      0.0204431    2.23136e-05
0.0290525          6.93328        0.00000              0
.......   5   1.52709e-05      0.0168774    2.36230e-05      0.0226784    2.26148e-05
0.0308795          2.87477      0.0173261              0
…
…
…

.......  70   1.20347e-08    0.000582638    2.90318e-05      0.0240310    3.37846e-05
0.0281474          32.7195      0.0803459              0

For this try, the optimum result was found for h=10 hidden units
Best mean squarred error Eval(tmin)= 2.18746e-05
Calculation Time        : 41 min 23 sec
Saving weight file in weights/weights_try_XX.data
**************************************Try number 3***********************************
…
…
…
**************************************Try number 10**********************************
E/Pmax After Optimization- no HU: 2.75994e-05

 ....... #HU (E/Pmax)tr                (EMax)tr     (E/p)val        (EMax)val
       (E/p)gen        (EMax)gen                    GL[h]          PQ[h]          UPs[h]
.......   1   2.17477e-05      0.0240674    2.83740e-05      0.0211502    2.42877e-05
0.0234197          0.00000        0.00000              0
.......   2   1.91375e-05      0.0197835    2.39364e-05      0.0161788    2.20001e-05
0.0215431          0.00000        0.00000              0
.......   3   1.75603e-05      0.0177309    2.31084e-05      0.0161258    2.21745e-05
0.0238413          0.00000        0.00000              0
.......   4   1.61779e-05      0.0175574    2.27406e-05      0.0175684    2.13327e-05
0.0237191          0.00000        0.00000              0
.......   5   1.52258e-05      0.0179309    2.21718e-05      0.0191528    2.16977e-05
0.0252443          0.00000        0.00000
…
…
…                0
.......  70   9.89560e-09    0.000641451    3.84180e-05      0.0227461    3.63721e-05
0.0286027          76.9552       0.191996              4

For this try, the optimum result was found for h=8 hidden units
Best mean squarred error Eval(tmin)= 2.17106e-05
Calculation Time        : 41 min 30 sec
Saving weight file in weights/weights_try_XX.data
Calculation Time for the 10 trys       : 403 min 28 sec
Best Result for try 3 ,Best Error = 1.80512e-05
Best Weight file found in ./weights/best_weights.data
```

2.4. Weight data file

```
#include "NN.H"
#include <iostream.h>
#include <fstream.h>
#include <stdlib.h>
//#include <time.h>
/*      Evaluator (NN1_Evaluator)
        Calculation of the NN output, an input vector being given
        The code looks for a dv.data file in the current folder, this
        file contains the values for the DV.
              PS: INPUTS-OUTPUTS ARE SUPPOSED DIMENSIONALIZED */


int main()
{
        int i;
        int j;
        double aux1,aux2;
       NN* Network;
//     Neural Network Generated by NN_OPT_1
//     Best Mean squarred Error = 1.8051229873897787e-05 for try number 3
       const int n1=8;
       const int m1=1;
       const int h1=6;
//     MinInput[j] & MaxInput[j] for j=1,n
       double MinInput[n1]= {0,-0.0040000000000000001,-0.0040000000000000001,0,-0.02,-
0.0040000000000000001,-0.0040000000000000001,0};
       double MaxInput[n1]= {0.02,0.02,0.02,0.01,0,0.02,0.02,0.01};
//     MeanTarget[j] for j=1,m
       double MeanTarget[m1] = {0.15977427983539072};
       double
vij[m1*(n1+h1+1)]={0.015197799563602054,0.044234941402215365,0.27661594929751776,0.358923
77081308724,0.021674581531789746,0.30118565596903785,0.48005152831000325,0.41559781790208
583,0.029300117654549498,0.012083128865297495,-
0.013724105693491207,0.0078422151426914104,-0.0048799849519052 79,-
0.0079812813099029607,0.0071903665759384255};
       double wij[h1*(n1+h1)]={-
2.5017233664398217,1.0007690473069926,10,9.9996810708647335,-
3.9647692188346242,3.181563425189561,7.461124200978726,4.3759185013226043,-
9.5280675762031937,0,0,0,0,0
       ,-1.2296481836682758,-0.92714831420241406,0.71530049716709532,-
4.9163493487291614,0.9351576156639283,10.000000000000002,9.9999999999999982,-
0.61244854544229643,-6.8082331523650286,-4.4994620624464083,0,0,0,0
       ,0.41724101942901221,9.2306495680340515,9.9999999999999982,-7.2243788491917487,-
2.3917482950017299,-1.519109174412453,-6.7083807330304115,-10,0.76958075246808444,-
2.7406873223440886,6.6223610459803686,0,0,0
       ,-2.7699168418671598,-8.7531223547793289,-10,-3.0671639002846365,-
6.7088894373450803,0.95062257041599973,-0.057724732766451514,-
6.130119888750019,9.9594558786709459,0.53282912087593892,-
2.5132985392836931,5.8242826351654085,0,0
       ,-3.9132198149245006,-
10,3.7479900279768974,2.3262756905540694,1.7388272906472357,-0.30192372522725019,-10,-
5.3865853741610037,8.101515110646794,-2.4836087713652395,-5.7395568560866108,-
2.5402040417081748,10,0
       ,-5.3791262995428157,-5.9257961829767929,-7.7132244648789126,-
10,3.138554652860047,-
2.2372199315006829,9.851949979517741,1.9807299463564656,9.2346039778454827,5.206787960098
5005,-4.6473794506033066,2.0739813459741572,9.3691825720179907,-1.2405487687207595
       };
        cout.setf(ios::fixed);
        char buffer[1000];
        Vector x(n1);
        Vector y(m1);
        cout<<"       output= ";
        ifstream fin("dv.data");
```

```
        fin.getline(buffer,1000);// this throws away the comment line;
        for(i=0;i<n1;i++)
        {
               fin>>aux1;
               //non-dimensionalize the inputs
               aux2=(aux1-MinInput[i])/(MaxInput[i]- MinInput[i]);
               x.initialize_element(i,aux2);
        }
        fin.ignore(80,'\n'); //this throws away the endline character
        Network= new NN(m1, n1, h1,vij,wij);
        y=Network->Evaluate3(x);
        for(j=0;j<m1;j++) cout<<y.element(j)*MeanTarget[j]<<endl;
        cout<<"        MinInput[j]= {"<<MinInput[0];
        for(j=1;j<n1;j++) cout<<","<<MinInput[j];
        cout<<"}"<<endl;
        cout<<"        MaxInput[j]= {"<<MaxInput[0];
        for(j=1;j<n1;j++) cout<<","<<MaxInput[j];
        cout<<"}"<<endl;
        cout<<"        MeanTarget= "<<MeanTarget[0];
        for(j=1;j<m1;j++) cout<<"      "<<MeanTarget[j];
        cout<<endl;
        delete Network;
}
```

2.5. Analysis of the results

Figure 1 shows the values obtained for the squared error, E, and the maximum error , Emax, for the Training (TS), Validation and Generalization (GS) Sets during training of the third network (the "best" one) . As expected, when more hidden units are added to the network, the errors on the TS decreases. The network converges to the data points of the TS. However, the error on the VS and GS, starts increasing after a while leading to overfitting. This shows the importance of a Validation set to decide when to stop training. This figure also shows that the VS (97 points) and the GS (1955 points) behave similarly and thus there is no need in practice to have a large set like the GS to define when to stop training. Indeed, the GS is only used in this test case to calculate a good approximation of the generalization error on the rest of the computational domain. In practice, having such a large set that is not used for training is not computationally efficient. The resulting network has six hidden units with a generalization error of 2.16e-6 and a maximum error on the GS of 2.6%. These results confirm that the neural networks are constructed properly and produce an accurate approximation of the objective function with reasonable number of hidden units.

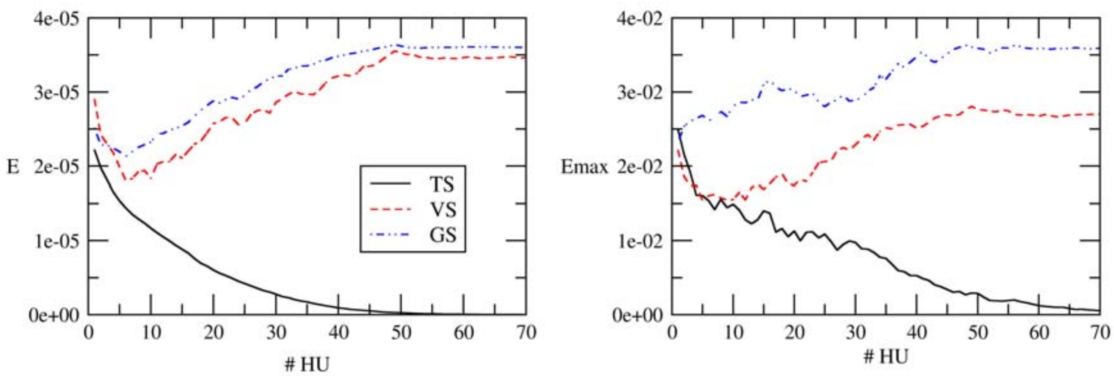

Fig. 1 Squared Error and Maximum Error on Training, Validation and Generalization Set for 2D Model Problem.

3. Summary/Conclusions

This report describes the qualification and validation of the optimization method based on Neural Networks that is outlined in the accompanying Computer Software Product End Item report of reference [1]. The method has been applied to a two dimensional hydrofoil, confirming that objective functions can be accurately approximated with this method. Full application of this method to a complex three dimensional shape optimization is reported in the reports on deliverable 3 of this task [2] and [3].

4. Glossary

Symbols

$C_p = \frac{p - p_{atm}}{q_\infty} = 1 - \left(\frac{V}{V_\infty}\right)^2 - \frac{2}{F_r^{\,2}}\left(\frac{z}{L}\right)$: Pressure coefficient

p_{atm} atmospheric pressure

q_∞ free stream dynamic pressure.

Acronyms

CFD: Computational Fluid Dynamics

NN Neural Networks

IBL Interactive Boundary Layer

TS Training Set

VS Validation Set

HU Hidden Unit

GS: Generalization Set

5. Acknowledgement:

The CCDoTT project is a collaborative effort involving several faculty and students at California State University, Long Beach. Staffs primarily responsible for this portion of the work are Dr. Eric Besnard, Associate Professor and Adeline Schmitz, Research Associate in the Mechanical and Aerospace Engineering Department.

6. References

1. Hamid Hefazi *et al.*, “Computer Software Product End Items, Deliverable 2, Optimization Tool development Based On Neural Networks DI-MCCR-80700” report, Center for the Commercial Deployment of Transportation Technologies, Long Beach, CA, Sept. 2003.
2. Hamid Hefazi *et al.*, “Computer Software Product End Items, CAD-Based CFD tool Development and Application, Deliverable 3, DI-MCCR-80700”, report Center for the Commercial Deployment of Transportation Technologies, Long Beach, CA, Sept. 2003.
3. Hamid Hefazi *et al.*, “Software Test Report, CAD-Based CFD tool Development and Application, Deliverable 3, DI-IPSC-81440A” report Center for the Commercial Deployment of Transportation Technologies, Long Beach, CA, Sept. 2003.
4. R. Hicks and P. Henne, “Wing Design by Numerical Optimization,” J. Aircraft, Vol. 15, No. 7, pp. 407-413, July 1978.
5. J. Reuther and A. Jameson, “A Comparison of Design Variables for Control Theory Based Airfoil Optimization,” Proceedings of the 6th International CFD Symposium, Lake Tahoe, NV, Sept. 1996.
6. S. Eyi, “Performance Evaluation and Improvements of CFD-based Aerodynamic Design Optimization,” Ph.D. Dissertation, University of Illinois, Urbana Champaign, 1995.
7. E. Besnard, A. Schmitz, E. Boscher, N. Garcia and T. Cebeci, “Two-Dimensional High Lift System Design and Optimization,” AIAA Paper No. 98-0123, Jan. 1998.
8. T. Cebeci, *An Engineering Approach to the Calculation of Aerodynamic Flows*, Horizon Pub., July 1999.

9. E. Besnard, A. Schmitz, K. Kaups, G. Tzong, H. Hefazi, H.H. Chen, O. Kural and T. Cebeci, "Hydrofoil Design and Optimization for Fast Ships," Proceedings of the 1998 ASME International Congress and Exhibition, Anaheim, CA, Nov, 1998.