Master Thesis

Enhanced Finite Element Analysis with ANN Techniques

Kuan Zhang

Supervisor: Prof. Eugenio Oñate
Prof. Francisco Zárate

CIMNE
Technical University of Catalonia

25 June 2009

Acknowledgments

I would like to thank my supervisors, Professor Eugenio Oñate and Professor Francisco Zárate, for their guidance, support, and inspiration through my study at CIMNE.

I am sincerely grateful to Dr. Robert Lopez for his help during the implementation of Artificial Neural Network program.

This thesis is one part of my study in Erasmus Mundus Master in computational mechanics. I really appreciate this chance to study in Swansea University and UPC in Barcelona. I would like to thank Professor Pedro Diaz for his help my study and living during these 2 years.

Contents

Chapter 1

General introduction

1.1 Introduction.

Finite element method (FEM) is a powerful technique for numerically solving complex problems in structural mechanics. In the FEM, the structural system is usually modeled by a set of appropriate finite elements, which are interconnected at points called nodes.
The classical procedure of solving a structure problem is shown as follows. The theory of FEM for structural analysis can be presented via the virtual work principle.

The principle of virtual displacements expresses the mathematical identity of external virtual work and internal virtual work:

$$\text{External virtual work} = \int_V \delta\boldsymbol{\epsilon}^T \boldsymbol{\sigma}\, dV \qquad (1\text{-}1)$$

The right-hand-side of the above equation shows the internal virtual work. It may be derived by summing up the virtual work in the individual element. Eq. (1-1) leads to the following

governing equilibrium equation for the system:

$$\mathbf{R} = \mathbf{K}\mathbf{r} + \mathbf{R}^{o} \tag{1-2}$$

R is vector of external nodal forces.
r is vector of system's nodal displacements. Displacements at any point of the finite element mesh can be yielded by interpolation with **r**.
K is system stiffness matrix, which will be established by assembling the elements' stiffness matrices.

The nodal displacements are found by solving the system of linear equations (1-2), symbolically:

$$\mathbf{r} = \mathbf{K}^{-1}(\mathbf{R} - \mathbf{R}^{o}) \tag{1-3}$$

During the above process of solving matrix equation, iterative solvers with initial value 0 might be always chosen as a proper tool. Generally, there are several iterative solvers:

Jacobi Iteration Method,
Gauss-Seidel Iteration Method,
Conjugate Gradient Method,
Quasi-Newton Methods,
GMRES Method.

1.2 Iterative solvers

An iterative solver attempts to solve a matrix equation by finding successive approximations to the solution starting from an initial guess.
Iterative solvers are often useful to solve linear matrix equations involving a large number of variables, where direct methods (such as Gauss elimination) would be prohibitively expensive.
To solve equations as:

$$A \cdot x = b \quad \Rightarrow \quad \sum_{j=1}^{n} a_{ij} \cdot x_j = b_i \quad (i=1, 2, 3...n) \tag{1-4}$$

1) Jacobi Iteration Method:

Jacobi method is for solving a matrix equation on a matrix that has no zeros along its main diagonal. In Jacobi iteration, each equation of the system is solved for the component of the solution vector associated with diagonal element. This procedure is repeated until some convergence criterion is satisfied. The Jacobi algorithm for the general iteration step (k):

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \cdot \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)}\right) \quad (i=1, 2, 3...n) \tag{1-5}$$

2) Gauss-Seidel Iteration Method:

Gauss-Seidel method is an improved version of Jacobi method. For the general iteration step (k), the algorithm is:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \cdot \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)}\right) \quad (i=1, 2, 3...n) \tag{1-6}$$

The Gauss-Seidel method is applicable to strictly diagonally dominant, or symmetric positive definite matrices.

3) GMRES Method:

The generalized minimal residual method (GMRES) is another iterative method, which approximates the solution by the vector in a Krylov subspace with minimal residual.
The Arnoldi iteration is used to find this vector. The algorithm for Arnoldi iteration is as follows:

Start with vector q = b with norm 1.

- $q_k \leftarrow A q_{k-1}$
- **for** j from 1 to $k-1$
  - $h_{j,k-1} \leftarrow q_j^* q_k$
  - $q_k \leftarrow q_k - h_{j,k-1} q_j$
- $h_{k,k-1} \leftarrow \|q_k\|$
- $q_k \leftarrow \frac{q_k}{h_{k,k-1}}$

$q_1, q_2 \ldots q_n$ forms a basis for K_n.

The vector $x_n \in K_n$ can be written as $x_n = Q_n y_n$ with $y_n \in R_n$
Hence, x_n can be found by minimizing the norm of the residual:

$$r_n = \tilde{H}_n y_n - e_1. \qquad (1\text{-}7)$$

4) Conjugate gradient method:

The conjugate gradient method is an algorithm for the numerical solution of particular systems of linear equations, especially those whose matrix is symmetric and positive-definite. The typical idea of the method is that it does not repeat advance directions.

Iterative scheme: $x^{k+1} = x^k + \alpha_k p^k$ (1-8)

α_k is determined by solving a minimization problem in the advance direction:

$$\alpha_k = -\frac{\langle \mathbf{p}^k, \mathbf{r}^k \rangle}{\langle \mathbf{p}^k, \mathbf{A}\mathbf{p}^k \rangle} \quad (1\text{-}9)$$

The advance directions in each iteration are chosen to be A-conjugate and are defined as:

$$\mathbf{p}^k = \begin{cases} \mathbf{r}^0 & k = 0 \\ \mathbf{r}^k + \beta_k \mathbf{p}^{k-1} & k > 0 \end{cases}$$

$$\beta_k = -\frac{\langle \mathbf{r}^k, \mathbf{A}\mathbf{p}^{k-1} \rangle}{\langle \mathbf{p}^{k-1}, \mathbf{A}\mathbf{p}^{k-1} \rangle} \quad (1\text{-}10)$$

1.3 Difficulty of two type problems.

With the help of iterative solvers, solution for the system of linear equations (1-2) can be derived after definite iteration steps. At the same time, the structure problem can be valued by such a run. Comparing with the classical procedure, there are many applications that many runs may be needed.

1) Stochastic Analysis:

Instead of dealing with only one possible 'reality' of how the process might evolve under time, in a stochastic or random process there is some indeterminacy in its future evolution described by probability distributions. This means that even if the initial condition (or starting point) is known, there are many possibilities the process might go to, but some paths are more probable and others less.
For stochastic analysis, inputs are series of random variables. Each input needs to run the whole procedure to get the output and the whole calculations might come to be very expensive.

2) Optimization:

Optimization is the process of modifying a system to make some aspect of it work more efficiently or use fewer resources. In optimization problems, Genetic Algorithm and Evolution Strategy might be involved. A genetic algorithm is a search technique used in computing to find exact or approximate solutions to optimization and search problems. As a result, several cases need to be run in optimization problems.

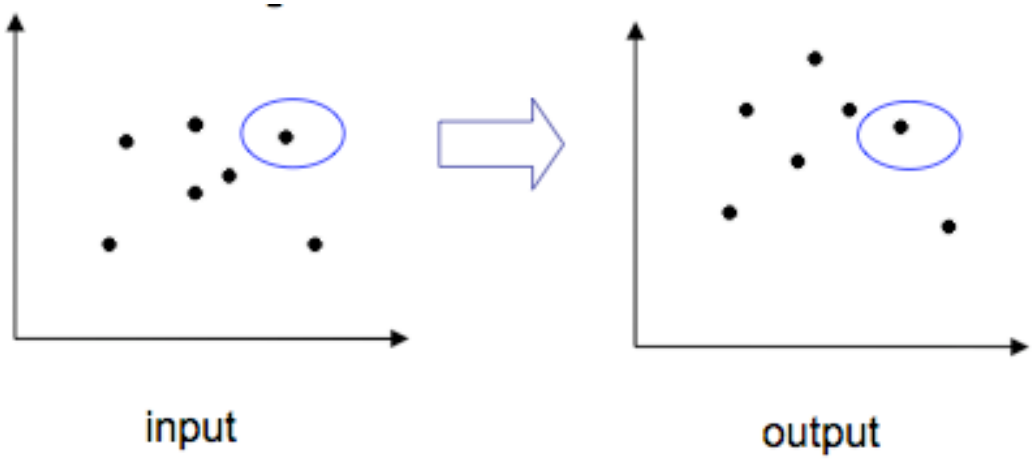


Figure 1-1

Both for stochastic analysis and Optimization, the total process might be time-consuming, which is the reason to introduce ANN Techniques.

1.4 ANN.

An Artificial Neural Network (ANN) is an information-processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems.

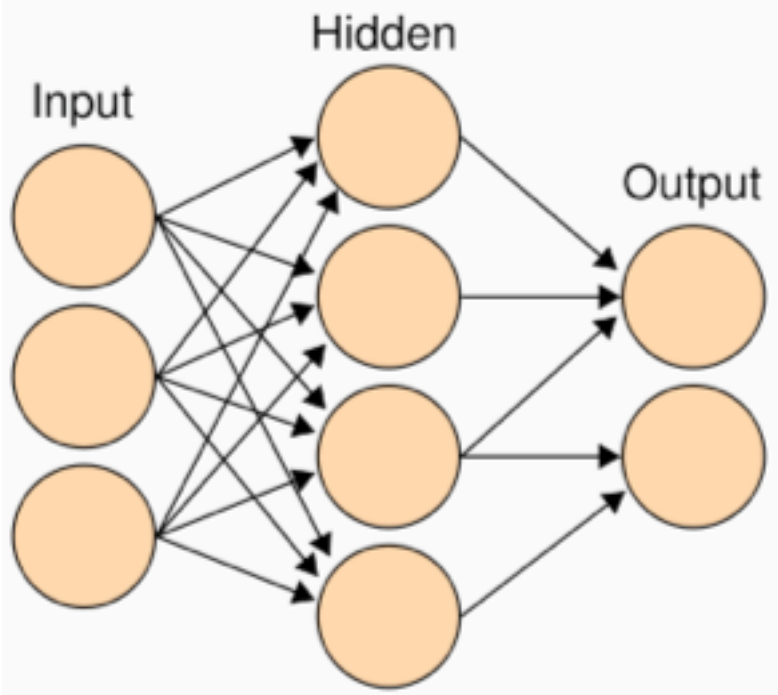


Figure 1-2: ANN´s scheme

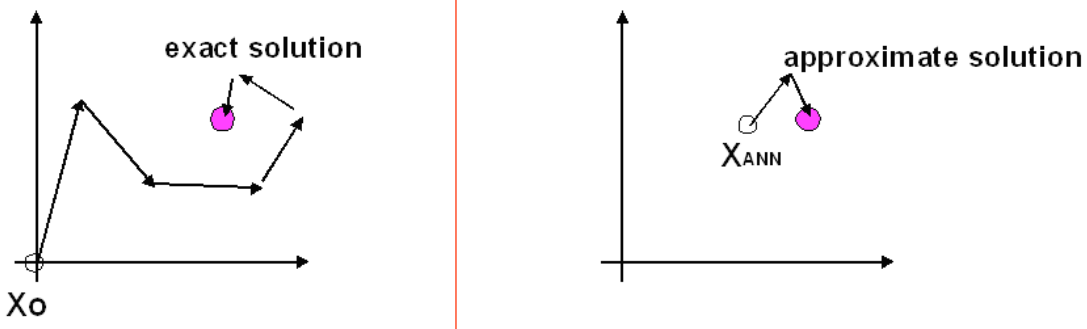
To filfull the function of an ANN, there are two processes needed to run: learning process and operative process. What has attracted the most interest in neural networks is the possibility of learning, which in practice means the following:

Given a specific task to solve and a class of functions F, learning means using a set of observations, in order to find $f^* \in F$ which solves the task in an optimal sense.

This entails cost function $C : F \rightarrow \mathbb{R}$
Such that $C(f^*) \leq C(f) \quad \forall f \in F$

The cost function C is an important concept in learning, as it is a measure of how far away we are from an optimal solution to the problem that we want to solve.

1.5 Using ANN as initial solution.



classical iterative solution ANN solution

Figure 1-3

The left graph shows that classical iterative solution can be derived by taking several steps from initial value 0 to the exact solution. In a more advanced way, ANN produce an initial value instead. From this ANN initial value, much fewer steps needed to be taken to arrive the approximate solution. In this way, time of computer work can be reduced.

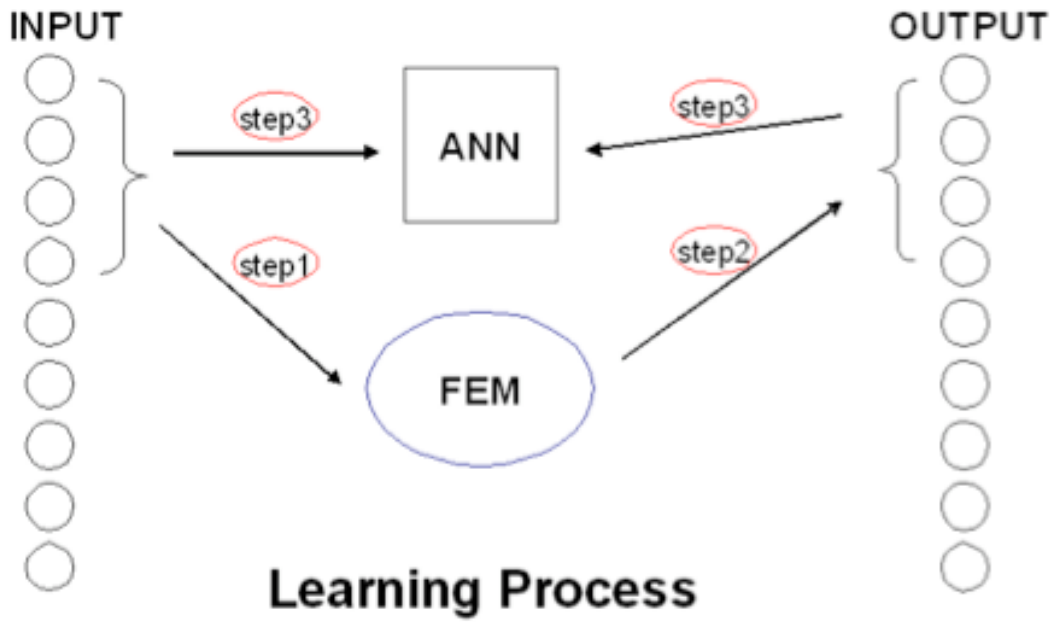


Figure 1-4: Learning Process of ANN

The graph above shows the process in learning of ANN.

Step1 and step2: send input test values to FEM code. Then the corresponding output values are produced.
Step3: ANN's Learning Process can be fulfilled with the help of both input test values and corresponding output values.
In step2, the initial value for iterative solver is 0.

Figure 1-5: Operative Process of ANN

In operative process, ANN gives initial value for FEM code in step2.
The number of rest input values should be much larger than the number of the test inputs during learning process. This is to ensure saving time of the scheme.

1.6 Tools needed to implement

STAC is a CIMNE code to produce random variables with Monte Carlo method. Program for ANN is needed to establish learning and operative processes.
Program iterative solvers in CALTEP and build the relationship between FEM from CALTEP and ANN.

Chapter 2

Application in Stochastic Analysis

2.1 Objective.

Stochastic Mechanics is a rapidly growing area of research, whose importance is being recognized not only in academic circles but also in industrial practice. In probability theory, a stochastic process is the counterpart to a deterministic process. A deterministic process deals with only one possible root of process, but in a stochastic or random process there is some indeterminacy in its future evolution described by probability distributions. This means that even if the initial condition is defined, there are still many possibilities of the process. The more possibilities a stochastic process has, the more expensive the calculation by finite element technique should be.
The purpose of the test example is trying to reduce time of computer work. Firstly, implement the computational platform and analyze the behavior of the process using different iterative solvers. At last, introduce ANN (Artificial Neural Network) into the platform and research on the effect of reducing time. The most important thing is the measurement of the speed-up of the process.

2.2 Introduction of the test example.

In our test example, there are three rooms in such a box with different conductivity in each room.
The temperature on the left hand side is 0 and right hand side's temperature is 10. One possible solution of T is like the graph below.

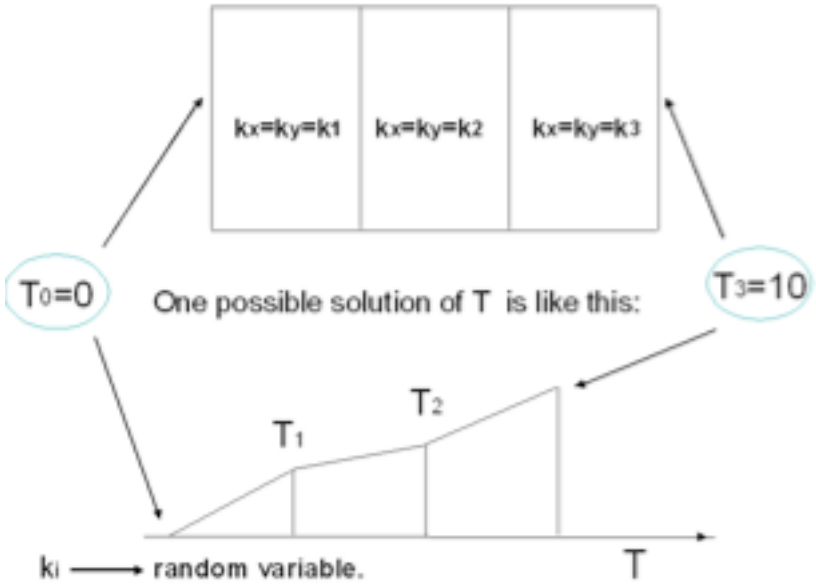


Figure 2-1: description of the example

The result depends on different values of conductivity k_i, which are random variables with certain density functions, conducing this test example to be a stochastic analysis case. Random variables are produced by Monte Carlo method. Monte Carlo method is a computational algorithm that relies on repeated random sampling to compute their results. Monte Carlo method is very useful in studying systems for modeling phenomena with significant uncertainty in inputs, such as the different conductivities in these three rooms. In our case, we use STAC to produce random input data set for conductivities.

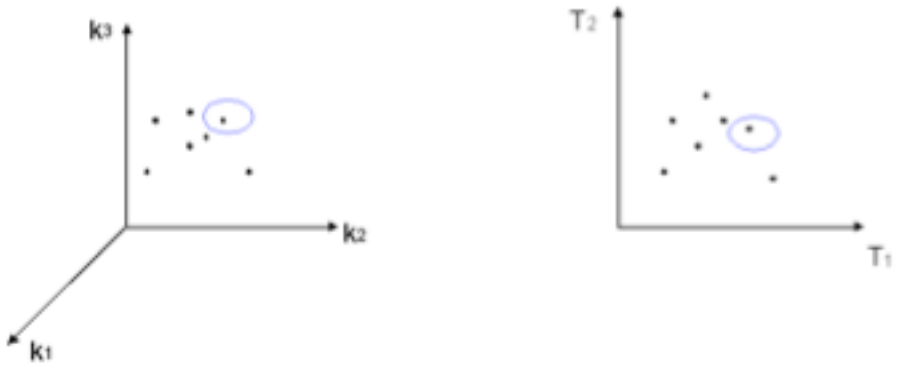


Figure 2-2: random input data

Build the geometry of three rooms in a box and generate the mesh by GID as below, in which 148 nodes and 246 triangle elements exist:

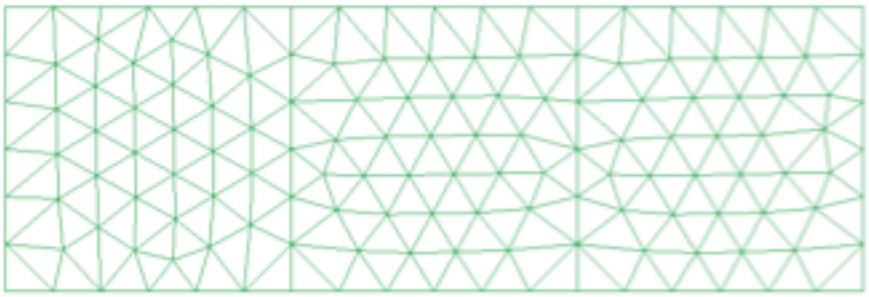


Figure 2-3: mesh of the example

CALTEP is used to solve this problem. In CALTEP, for matrix equation: $K \cdot d = f$, three iterative solvers are considered to use: Jacobi Iteration Method, Gauss-Seidel Iteration Method and Conjugate Gradient Method. For each iterative solver, Firstly, set three conductivities as chosen values K_i and run the code with initial value 0. The output is the result of temperature on each node of the mesh. Then using the output as the initial value to test 500 examples with random conductivities around the chosen value of K_i. In CALTEP, the convergence rate is set to be 1e-5.

During the process for each iterative solver, the effect of choosing different initial values will be examined.

The conductivities will be chosen as K_1= 4.2e+11, K_2= 2.1e+11 ,K_3= 8.1e+11 $[J/m^2 s]$. The corresponding result of temperature on each node of the mesh is as followed:

Figure 2-4: result of temperature by GID

2.3 "Intelligent" Finite Element Method with Jacobi Iteration Method:

Run the code in CALTEP once with initial value $X_0 = 0$. Number of iterations is 606. The output values are temperatures on the nodes of the mesh. Put these values into 2.vet file as the initial value that is $x_0 = x_{exact}$ for 500 examples. Analyze number of iteration for each example.

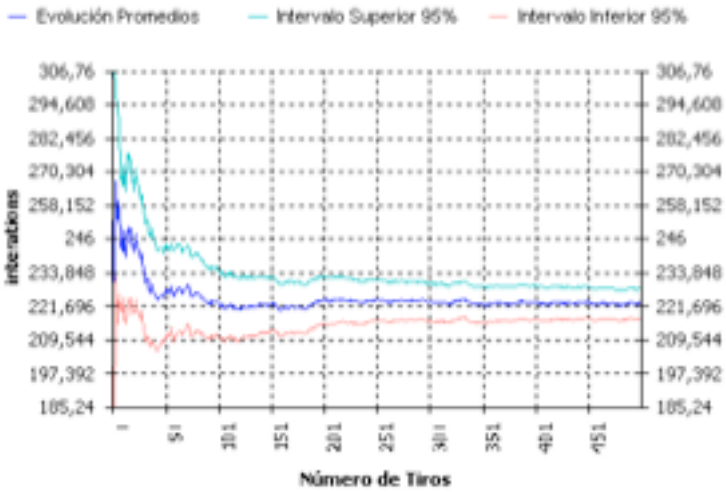


Figure 2-5: result of iteration numbers by STAC

Media=2.2308e+002 Var=3.9234e+003

Number of iterations is 223.08. The total time is 4m14s. From these, we can see that the output of temperature on each node corresponding to the chosen conductivities is a better initial value than 0, because it can reduce number of iterations and save time. Plot shot number versus iteration number.

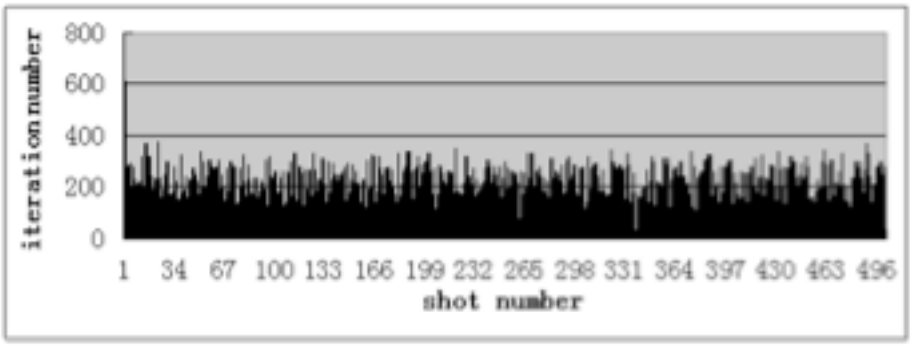


Figure 2-6: Plot of shot number versus iteration number

Run the code by STAC for 500 examples again, using Jacobi solver, with initial value equal to 0. The average number of iteration is 607.84. The total time is 5m35s.
Then calculate the speedup between choosing initial value as 0 and as temperature on each node:
$Speedup_{time}$= time1/time2=5m35s/4m14s=1.3189
$Speedup_{iter}$= number1/number2=607.84/233.08=2.6079

In order to testify the time-up, we change initial value from 0 to αX_{exact}. Examine the time-up's rate corresponding to different α.

α=0.25	time=8:13	media=5.8356e+002	var=3.3375e+002
α=0.75	time=7:36	media=4.8669e+002	var=1.9062e+002
α=0.95	time=7:27	media=3.4802e+002	var=9.6793e+001
α=0.99	time=6:27	media=2.1305e+002	var=2.4899e+001
α=0.999	time=6:10	media=6.0146e+001	var=2.1311e-001
α=1	time=6:01	media=1	var=0

Plot α versus iteration number and time.

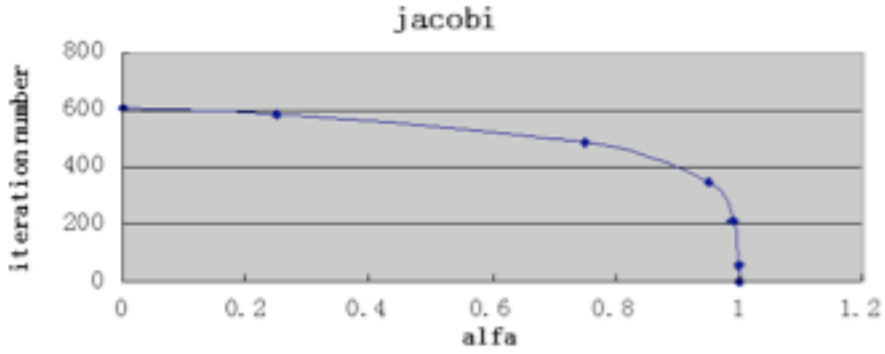


Figure 2-7: Plot of α versus iteration number

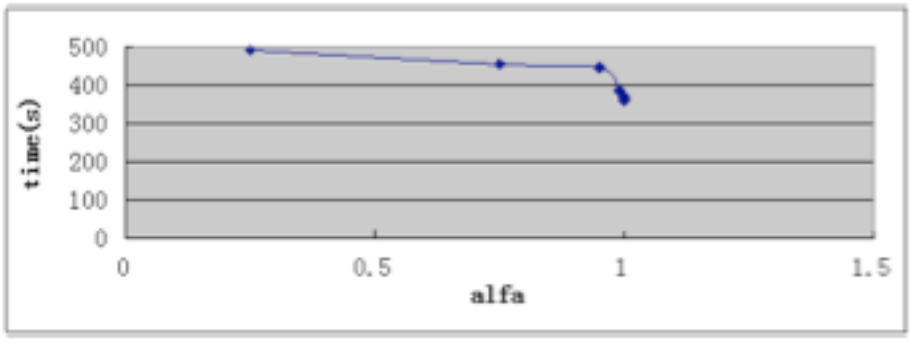
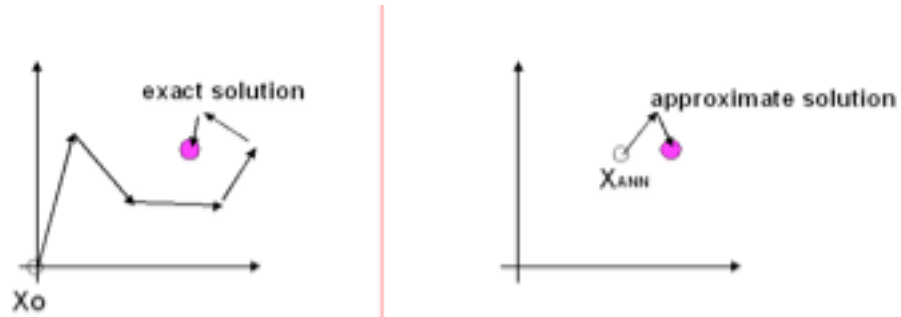


Figure 2-8: Plot of α versus iteration number

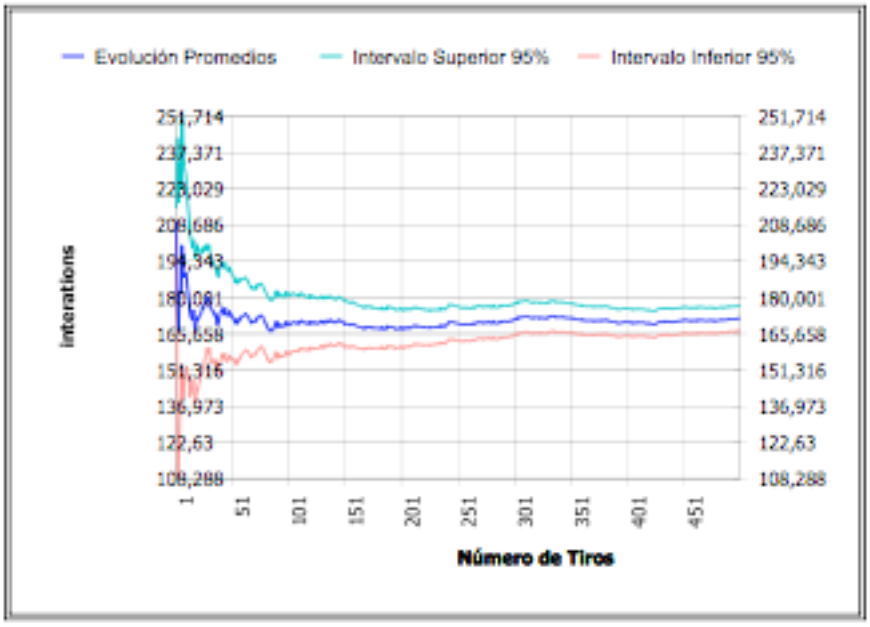
At last, introduce an Artificial Neural Network (ANN) into the platform. To fulfill the function of an ANN, there are two process needed to be run: learning process and operative process. The ANN program's built in the platform of Flood.
During the learning process, 100 examples are needed to run by STAC to produce Training Data Set, which is the input test value for the neural network. The number of hidden neurons is set to be 20. The purpose of the learning process is to produce MultilayerPerceptron.dat file, which is an essential part in operative process.
During the operative process, with the help of MultilayerPerceptron.dat file produced by the learning process, initial values for matrix equations mentioned above can be obtained corresponding to different conductivities. These initial values can help to reduce the number of iteration during the process of solving matrix equations. The idea is shown as below:



classical iterative solution ANN solution

Figure 2-9: steps taken from initial value to the solution

After introducing ANN, 500 examples are run by STAC. Numbers of iteration are reduced as:

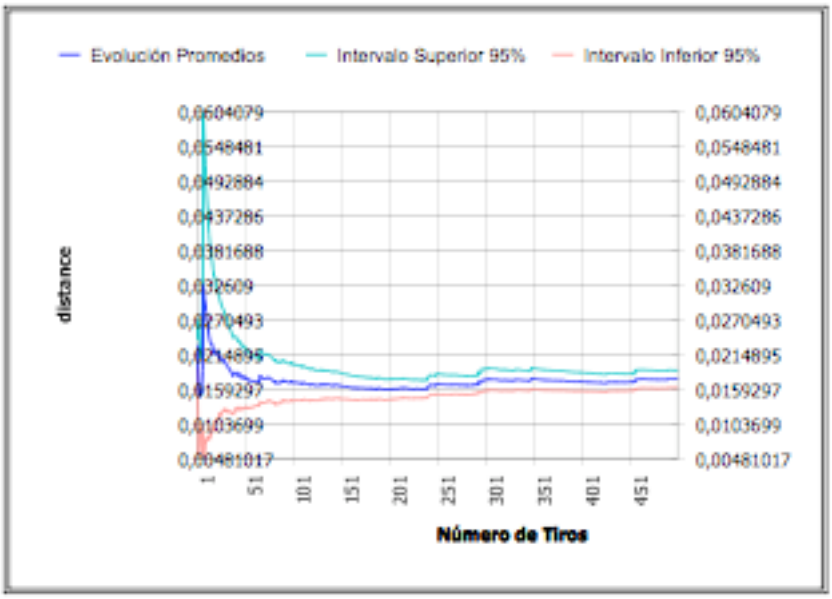


Media= 171.63 var= 3.1690e+3

Figure 2-10: result of iteration numbers by STAC

Comparing the initial values produced by ANN with

approximate results obtained from FEM code in CALTEP, the distances are shown as follows:

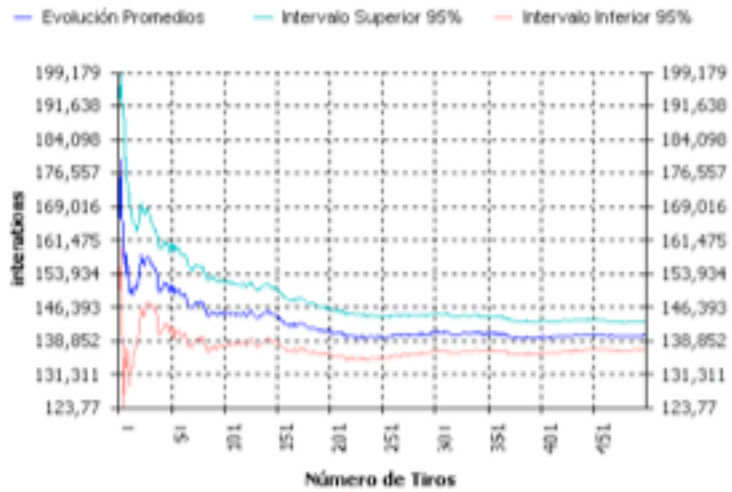


Media= 1.7576e-2 var= 2.5158e-004

Figure 2-11: distance between initial values by ANN and results

2.4 "Intelligent" Finite Element Method with Gauss Seidal Method:

Run the code once with initial value $X_0 = 0$. Number of iterations is 340. The output values are temperatures on the nodes of the mesh. Put these values into 2.vet file as the initial value, which is $x_0 = x_{exact}$ for 500 examples, to solve matrix equations. Analyze number of iteration for each example.



Media=1.4018e+002 Var=1.2813e+003

Figure 2-12: result of iteration numbers by STAC

Number of iterations is 140.18. The total time is 3m27s. From these, we can see that the output of temperature on each node corresponding to the chosen conductivities is a better initial value than 0, because it can reduce number of iterations and save time. Plot shot number versus iteration number.

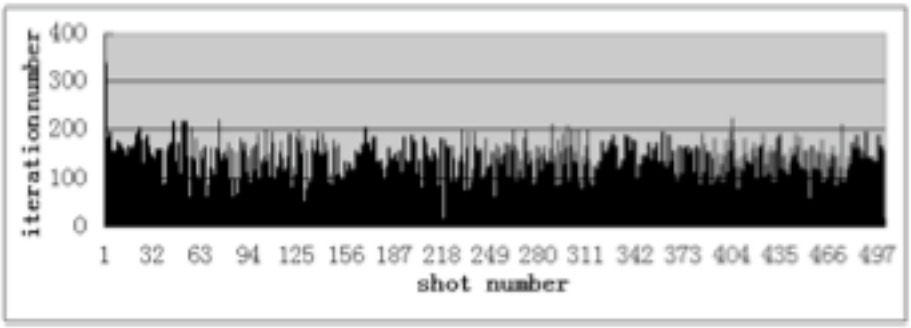


Figure 2-13: Plot of shot number versus iteration number

Run the code by STAC for 500 examples again, using Gauss Seidal solver, with initial value equal to 0. The average number of iteration is 339.65. The total time is 4m24s.

Then calculate the speedup between choosing initial value as 0 and as temperature on each node:
$Speedup_{time}$= time1/time2=4m24s/3m27s=1.2754
$Speedup_{iter}$= number1/number2=339.65/140.18=2.4230

In order to testify the time-up, we change initial value from 0 to αX_{exact}. Examine the time-up's rate corresponding to different α.

α=0.25	time= 5:34	media= 3.2802e+002	var= 1.0506e+002
α=0.75	time= 5:21	media= 2.8000e+002	var= 6.0876e+001
α=0.95	time= 5:18	media= 2.1020e+002	var= 3.4832e+001
α=0.99	time= 5:5	media= 1.4056e+002	var= 1.2748e+001
α=0.999	time= 4:59	media= 5.1370e+001	var= 3.7485e-001
α=1	time= 4:44	media=1	var=0

Plot α versus iteration number and time.

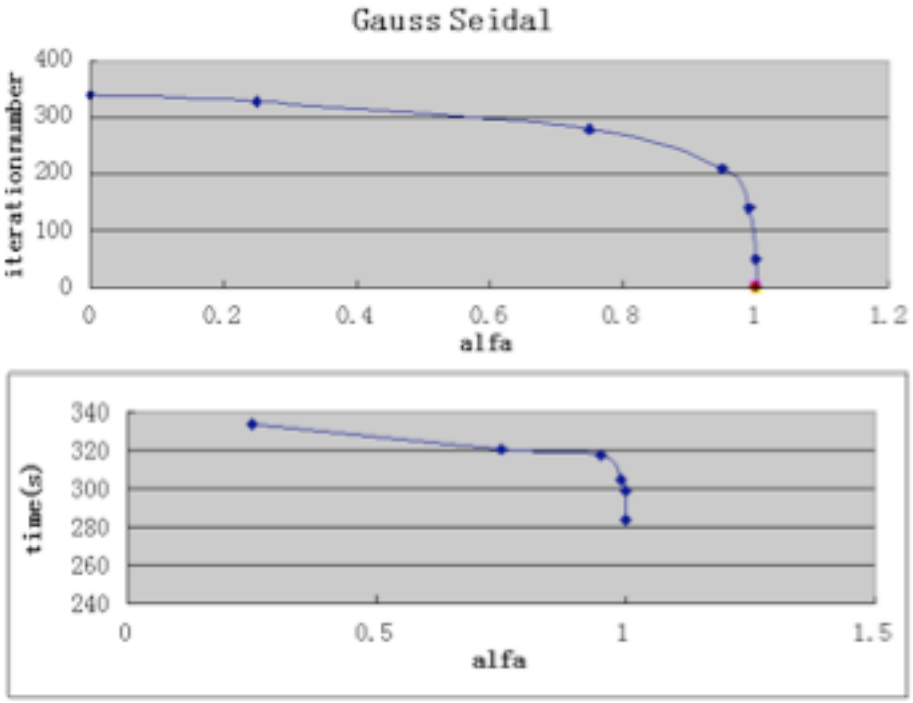
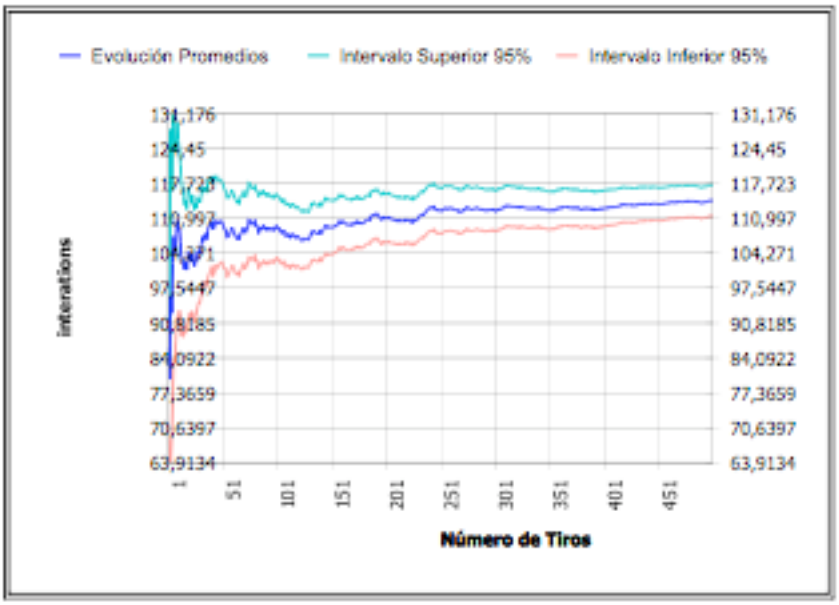


Figure 2-14: Plot of α versus iteration number and time

At last, introduce an Artificial Neural Network (ANN) into the platform. To fulfill the function of an ANN, there are two process needed to be run: learning process and operative process. The ANN program's built in the platform of FLOOD.

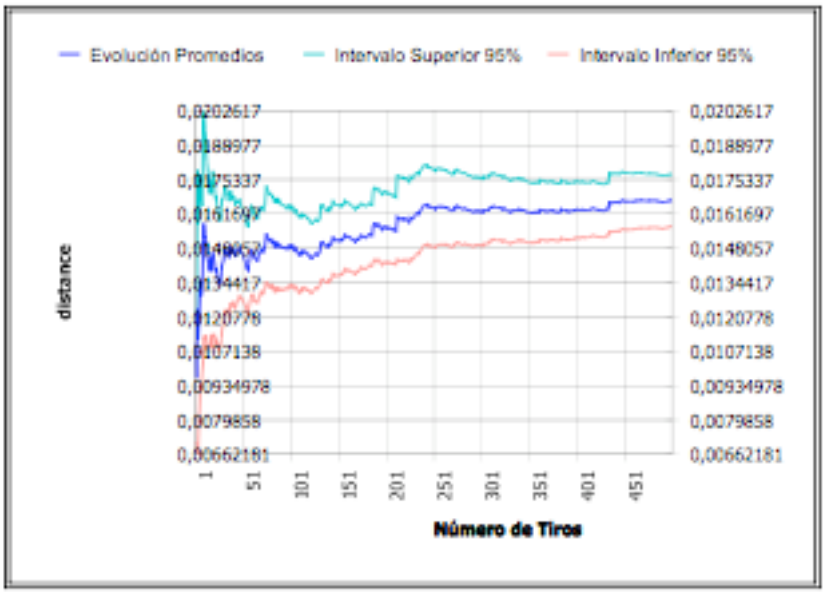
During the learning process, 100 examples are needed to run by STAC to produce Training Data Set, which is the input test value for the neural network. The number of hidden neurons is set to be 20. The purpose of the learning process is to produce MultilayerPerceptron.dat file, which is an essential part in operative process.
During the operative process, with the help of MultilayerPerceptron.dat file produced by the learning process, initial values for matrix equations mentioned above can be obtained corresponding to different conductivities. These initial values can help to reduce the number of iteration during the process of solving matrix equations.
When introducing ANN, 500 examples are run by STAC. Number of iteration are reduced as:



Media= 114.35 var= 1.1528e+3

Figure 2-15: result of iteration numbers by STAC

Comparing the initial values produced by ANN with approximate values obtained from FEM code in CALTEP, the distances are shown as follows:

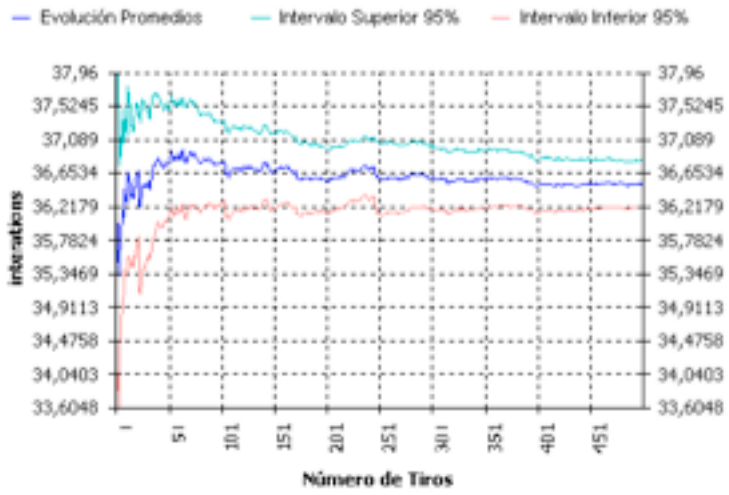


Media= 1.6674e-2 var= 1.3886e-004

Figure 2-16: distance between initial values by ANN and results

2.5 "Intelligent" Finite Element Method with Conjugate Gradient Method:

Run the code once with initial value $X_0 = 0$. Number of iterations is 54. The output values are temperatures on the nodes of the mesh. Put these values into 2.vet file as the initial value, which is $x_0 = x_{exact}$ for 500 examples, to solve the matrix equations. Analyze number of iteration for each example.



Media=3.6524e+001 Var=1.1532e+001

Figure 2-17: result of iteration numbers by STAC

Number of iterations is 36.52. The total time is 3m2s. From these, we can see that the output of temperature on each node corresponding to the chosen conductivities is a better initial value than 0, because it can reduce number of iterations and save time. Plot shot number versus iteration number.

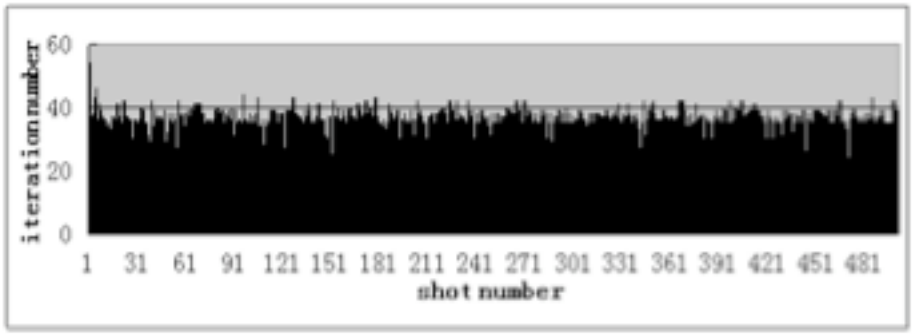


Figure 2-18: Plot of shot number versus iteration number

Run the code by STAC for 500 examples again, using conjugate gradient solver, with initial value equal to 0. The average number of iteration is 55.106. The total time is 3m5s.

Then calculate the speedup between choosing initial value as 0 and as temperature on each node:
$Speedup_{time}$= time1/time2=3m5s/3m2s=1.0165
$Speedup_{iter}$= number1/number2=55.106/36.52=1.5089

In order to testify the time-up, we change initial value from 0 to αX_{exact}. Examine the time-up's rate corresponding to different α.

α=0.25	time=4:10	media=5.4414e+001	var=5.7822e+000
α=0.75	time=4:18	media=5.1534e+001	var=4.4497e+000
α=0.95	time=4:22	media=4.6766e+001	var=4.5644e+000
α=0.99	time=4:6	media=3.8956e+001	var=3.2325e+000
α=0.999	time=4:15	media=3.5094e+001	var=2.4140e+000
α=1	time=4:6	media=1	var=0

Plot α versus iteration number and time.

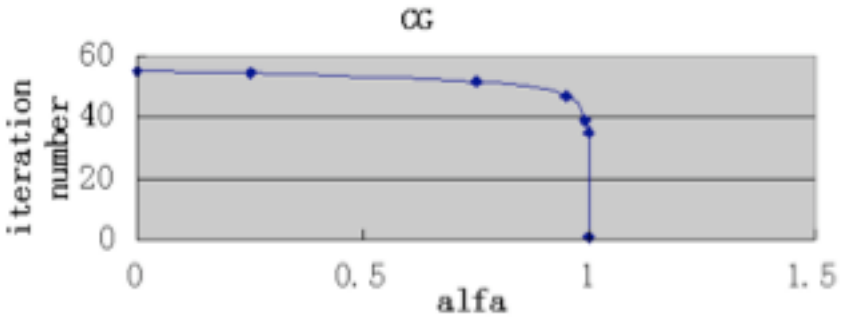


Figure 2-19: Plot of α versus iteration number

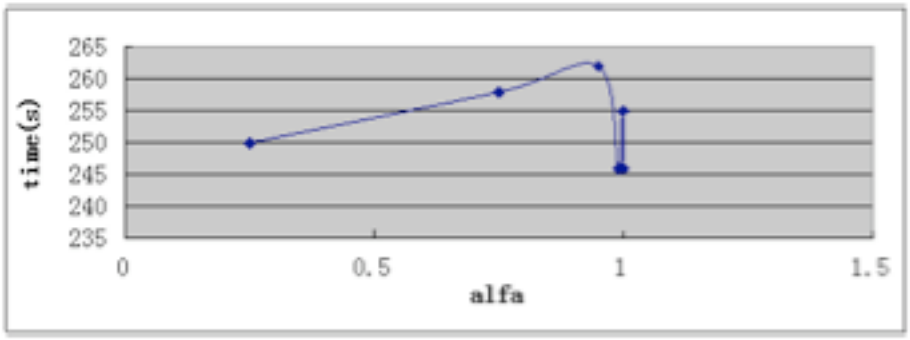


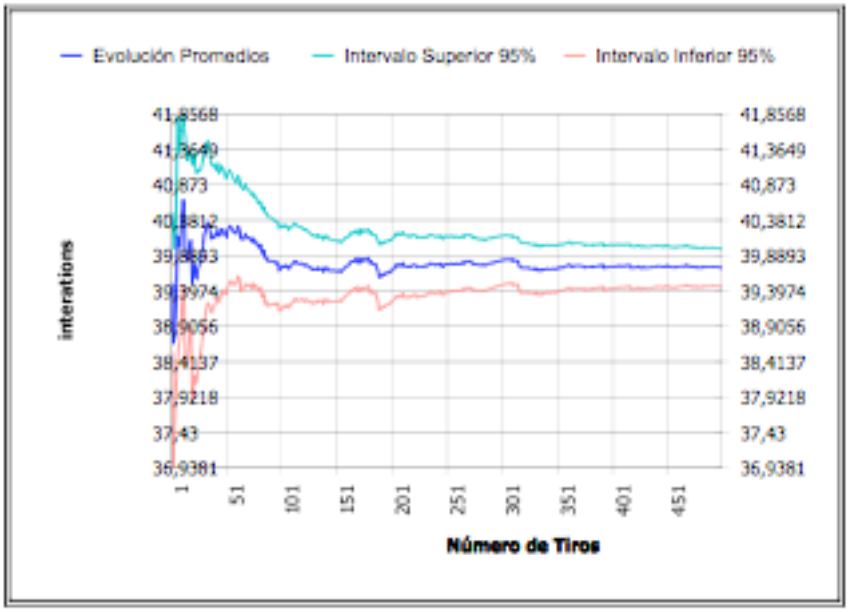
Figure 2-20: Plot of α versus iteration number

At last, introduce an Artificial Neural Network (ANN) into the platform. To fulfill the function of an ANN, there are two process needed to be run: learning process and operative process. The ANN program's built in the platform of FLOOD.

During the learning process, 100 examples are needed to run by STAC to produce Training Data Set, which is the input test value for the neural network. The number of hidden neurons is set to be 20. The purpose of the learning process is to produce MultilayerPerceptron.dat file, which is an essential part in operative process.
During the operative process, with the help of MultilayerPerceptron.dat file produced by the learning process, initial values for matrix equations mentioned above can be obtained corresponding to different conductivities. These initial values can help to reduce the number of iteration during the

process of solving matrix equations.

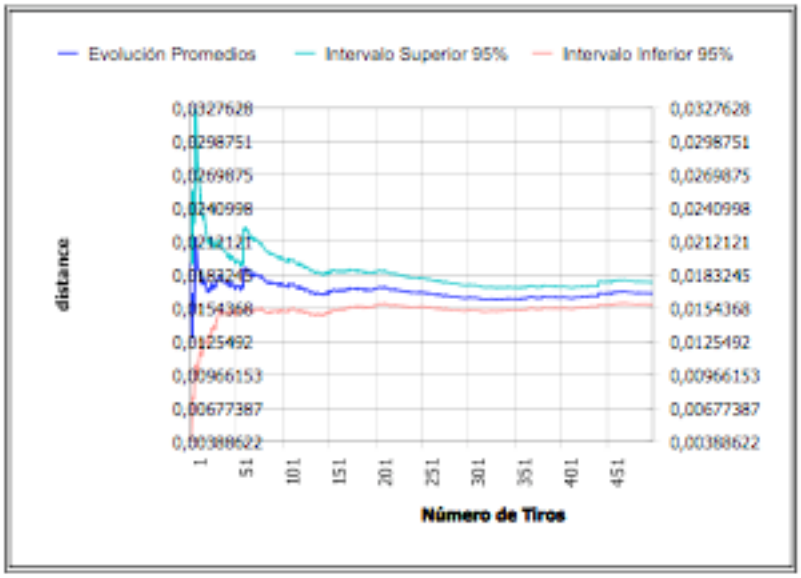
When introducing ANN, 500 examples are run by STAC. Number of iteration are reduced as:



Media= 39.718 var= 8.9444e+0

Figure 2-21: result of iteration numbers by STAC

Comparing the initial values produced by ANN with approximate values obtained from FEM code in CALTEP, the distances are shown as follows:



Media= 1.6756e-2 var= 1.1935e-004

Figure 2-12: distance between initial values by ANN and results

2.6 Conclusion:

After testing three iterative solvers with ANN technique, the effect of reducing computer time has been proved. The measurement of the speed-up of the process is shown as followed:

MEASUREMENT OF THE SPEED-UP OF ITERATIONS

Iterative solvers	Initial value as 0/ Initial value produced by ANN
Jacobi method	3.5416
Gauss Seidal method	2.9703
Conjugate gradient method	1.3874

Figure 2-13: Measurement of the speed-up of iterations

From the above graph, we can find that the effect of the speed-up of introducing ANN into Jacobi method is most tremendous. The time consumed by computer during Jacobi iteration process when choosing initial value as 0 is more than 3 times as the time taken by Jacobi iteration process when setting initial value with the help of ANN technique.

Chapter 3

Application in Optimization

3.1 Objective.

Optimization is the process of modifying a system to make some aspect of it work more efficiently or use fewer resources. In optimization problems, Genetic Algorithm and Evolution Strategy might be involved. A genetic algorithm is a search technique used in computing to find exact or approximate solutions to optimization and search problems. Genetic algorithms are implemented in a computer simulation in which a population of abstract representations of candidate solutions to an optimization problem evolves toward better solutions. As a result, more cases need to be run in optimization problems.

An airfoil is the shape of a wing. An airfoil-shaped body moved through a fluid produces a force perpendicular to the motion called lift. Flight airfoils have a characteristic shape with a rounded leading edge, followed by a sharp trailing edge. We simulate a set of similar airfoils with different shapes, and try to optimize the best shape of airfoil.

The NACA airfoils are airfoil shapes for aircraft wings developed by the National Advisory Committee for Aeronautics (NACA). The shape of the NACA airfoils is described using a series of digits following the word "NACA." The parameters in the numerical code can be entered into equations to precisely generate the cross-section of the airfoil and calculate its

properties. Equation for a symmetrical 4-digit NACA airfoil is shown as followed:

$$\frac{t}{0.20}\left[0.2969\times\sqrt{\frac{x}{c}}-0.1260\left(\frac{x}{c}\right)-0.3516\left(\frac{x}{c}\right)^{2}+0.2843\left(\frac{x}{c}\right)^{3}-0.1015\left(\frac{x}{c}\right)^{4}\right]$$

where:

c is the chord length,
x is the position along the chord from 0 to c,
y is the half thickness at a given value of x. (3-1)

We simulate a set of similar airfoils with different shapes, and try to optimize the best shape of airfoil with optimal force of lift and drag. More important task is trying to reduce time of computer work by introducing ANN techniques.

3.2 Introduction of the test example.

This example simulates the fluid flow on a domain with an airfoil inside. For this case, there will be both Neumann and Dirichlet conditions imposed on the boundaries. The geometry of the airfoil´s model is build as followed:

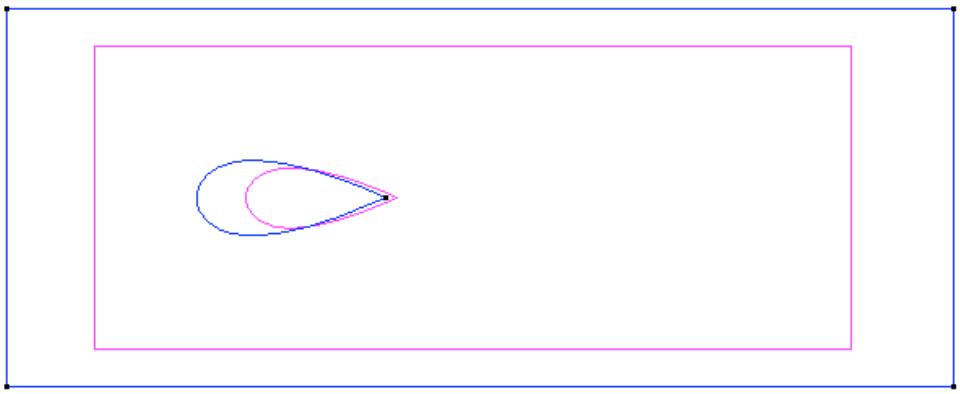


Figure 3-1: geometry of the test example in GID

The outer flow field for airfoils is one application of potential flow. In fluid dynamics, potential flow describes the velocity field as the gradient of a scalar function: the velocity potential.

$$u = \frac{\partial \varphi}{\partial x} = \frac{\partial \psi}{\partial y} \tag{3-1}$$

$$\Delta \varphi = \frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} = 0 \tag{3-2}$$

Because of similarity between equation (3-2) and heat conduction equation, we can use CALTEP to solve this problem. Now in CALTEP, the meaning of the function is no longer temperature.

The conditions are:

On the left of domain: $\frac{\theta}{n}=10$

On the right of domain: $\theta=0$

On the top and bottom of domain, and the boundary of airfoil: $\frac{\theta}{n}=0$

Select the problem type as CALTEP2000, and mesh:

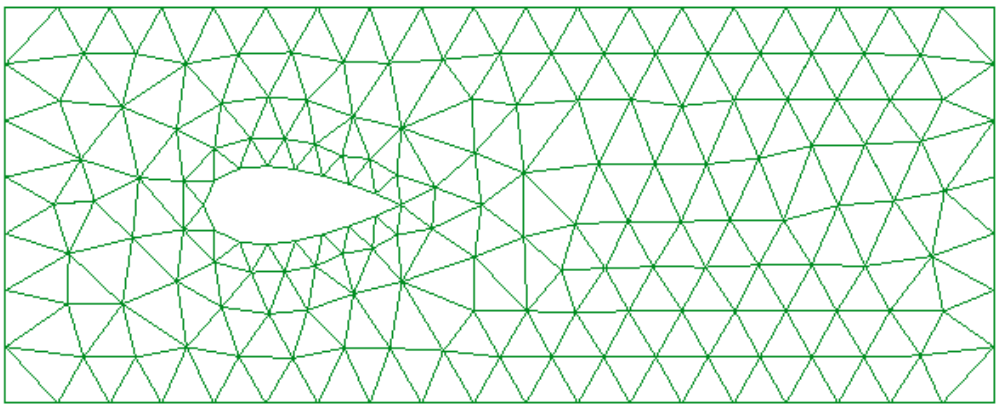


Figure 3-2: mesh of the example in GID

At the same time, .cal file, the input file for CALTEP has been received. Calculate with iterative solver in CALTEP. Number of iterations is 9195. The result of pressure is shown as followed:

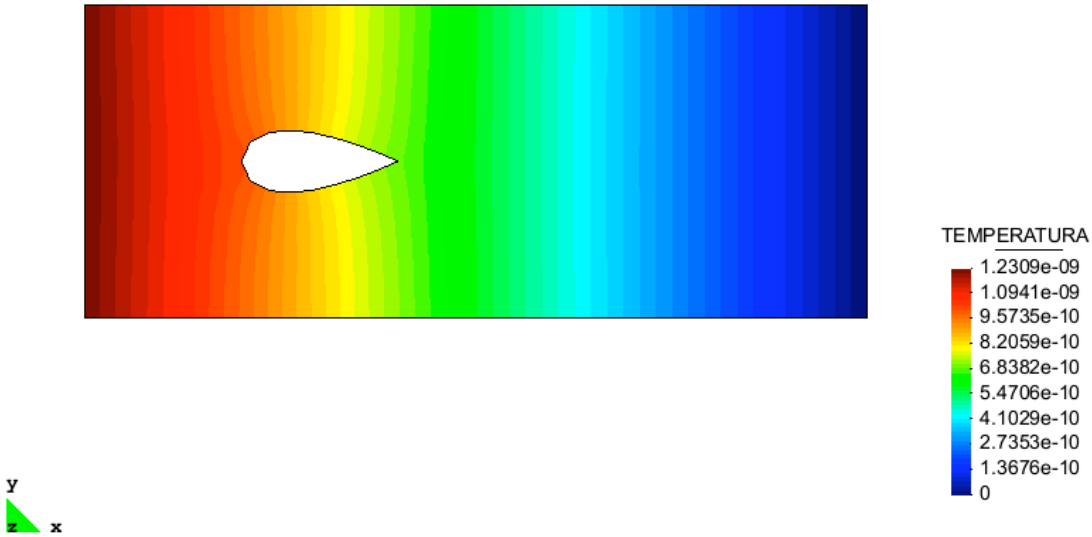


Figure 3-3: result of pressure by GID

Plot the pressure vs. x and y coordinates of the boundary of the airfoil respectively.

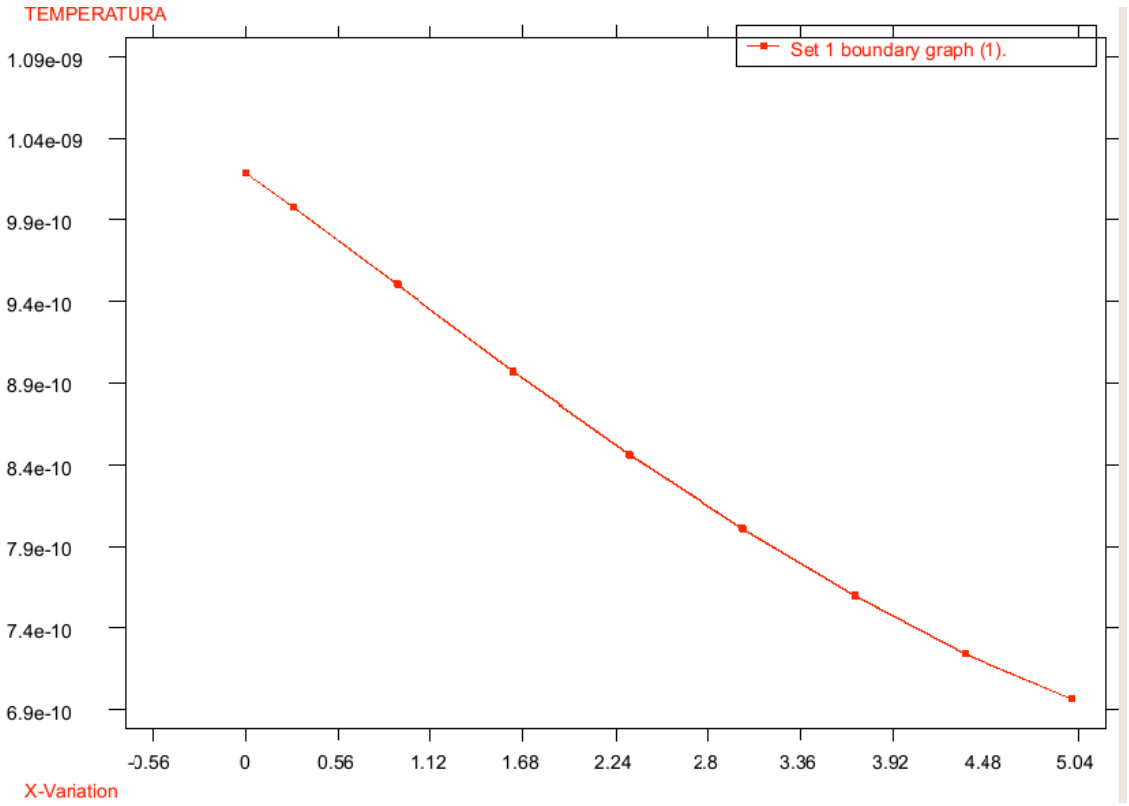


Figure 3-4: plot of pressure versus x coordinate of nodes on the boundary of airfoil by GID

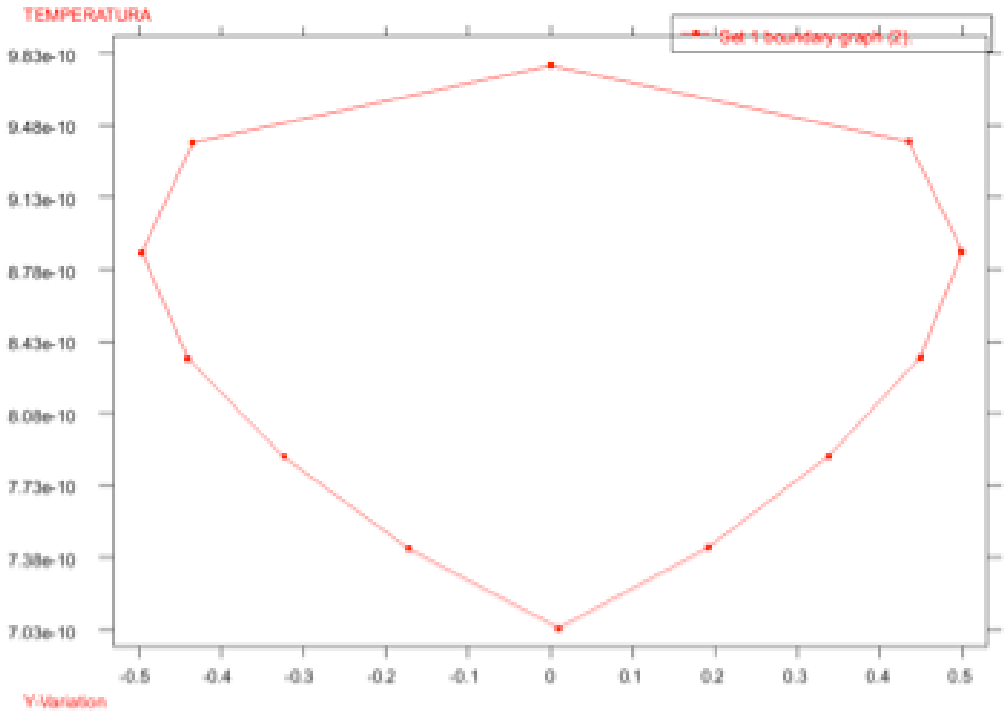


Figure 3-5: plot of pressure versus y of nodes on the boundary

With the result of pressure of each node on the airfoil´s boundary, lift and drag force of airfoil by flow can be calculated.
The airfoil´s lift:
-1.822497942270840E-012
The airfoil´s drag:
1.815381621501750E-010

3.3 Optimization of airfoils.

Introducing all the input and output files into STAC. In STAC, 16 variables of data and 1 variable of result have been defined:

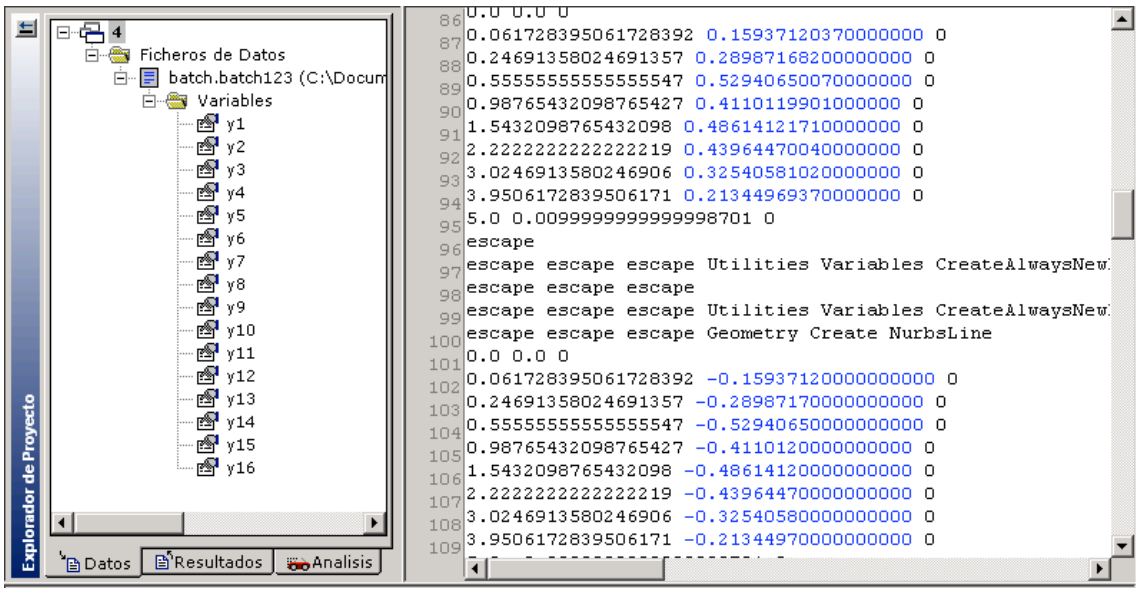


Figure 3-6: the setting of variables of input

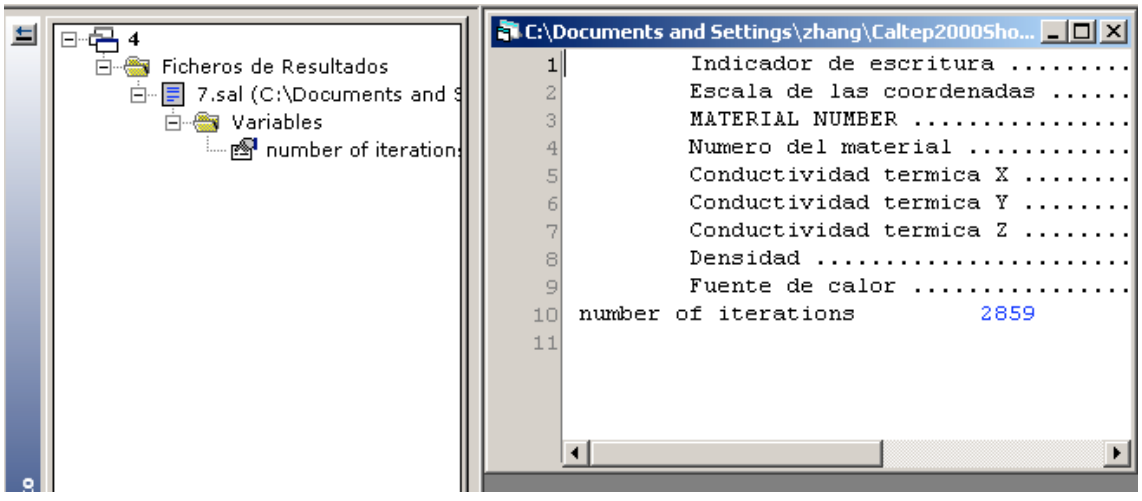


Figure 3-7: the setting of variables of output

Simulation of different airfoil can be achieved by changing those 16 variables of data and the variable of result can show the

number of iterations for each case.

For example, run 200 cases in STAC with different shape of airfoils:

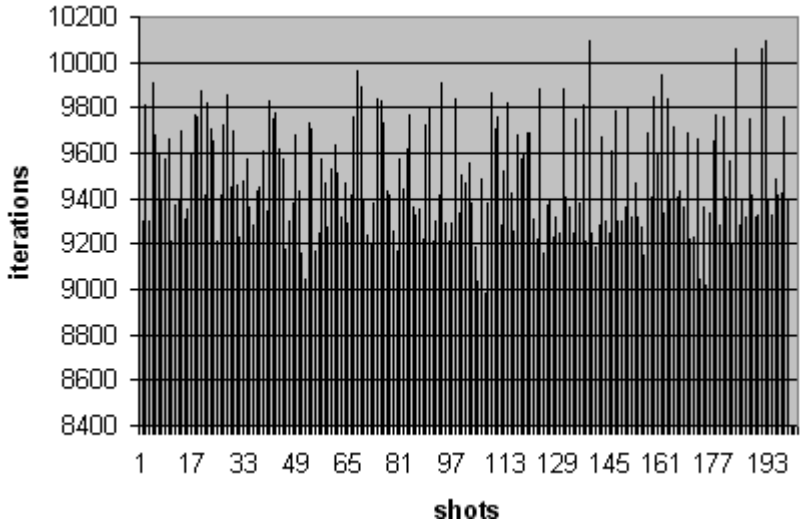


Figure 3-8: result of iteration numbers

The mean value of number of iterations for Jacobi solver is 9489.3.

IN STAC, reset the variable of result as two variables: force of lift and force of drag. Run 200 cases in STAC with different shape of airfoils again and try to find the best shape of airfoil:

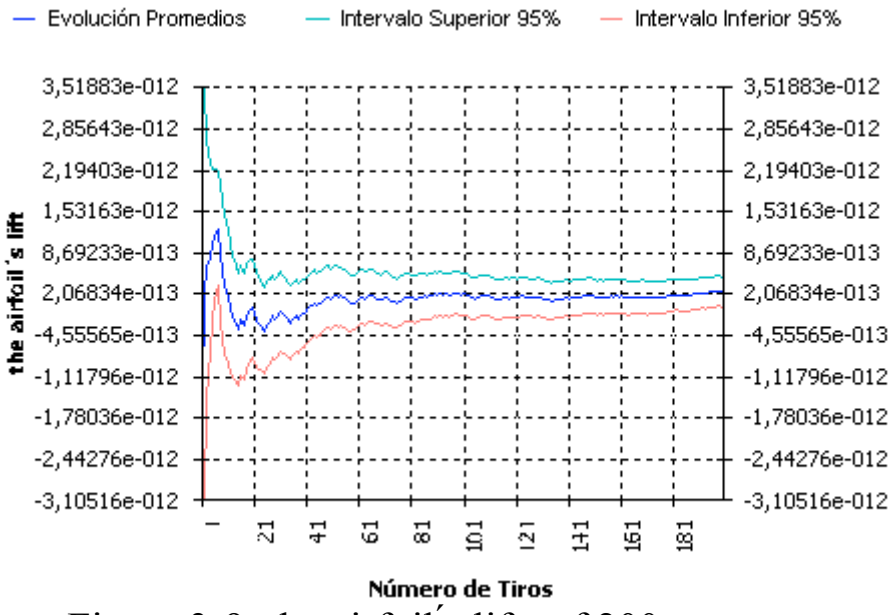


Figure 3-9: the airfoil´s lifts of 200 cases

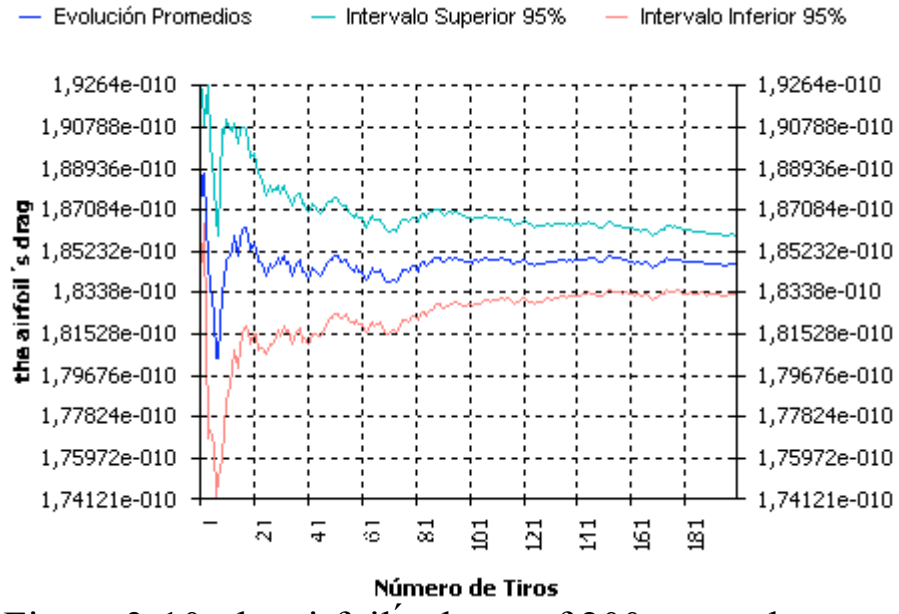


Figure 3-10: the airfoil´s drags of 200 examples

Among 200 shapes of airfoil, the best shape with least force of lift and biggest force of drag is obtained by setting the 16 variables of data as:

1.758051708E-01 3.090463331E-01 4.013961822E-01 5.261907726E-01 5.554660255E-01 5.270220612E-01 4.236484830E-01 2.560271772E-01 -1.758052e-001 -3.090463e-001 -4.013962e-001 -5.261908e-001 -5.554660e-001 -

5.270221e-001 -4.236485e-001 -2.560272e-001 1.522219039099205E-012
2.089909148799025E-010

These 16 variables of data mean y coordinate value of nodes on the boundary of airfoil. They can together define the shape of airfoil as:

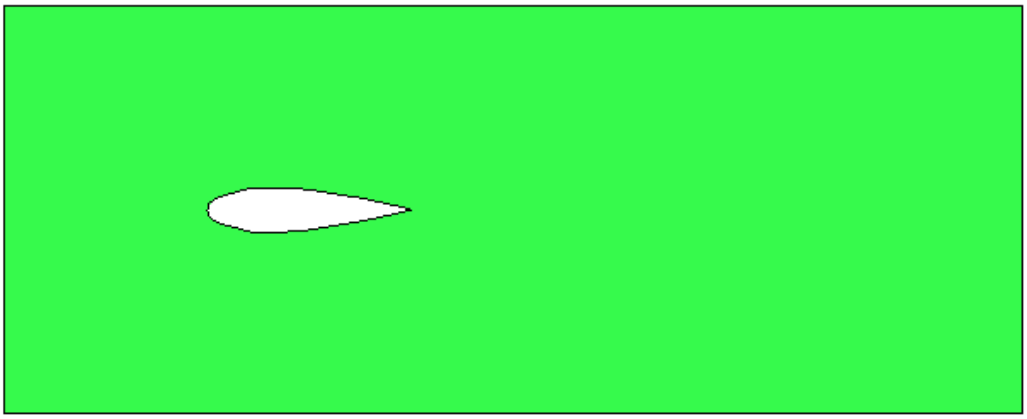


Figure 3-11: the optimal shape of airfoil

The optimal design of airfoil produces the least force of lift as 1.522219039099205E-012, and the best force of drag as 2.089909148799025E-010.

3.4 Introducing ANN for reducing time.

In previous work, 200 cases with different shapes of airfoil have been run in STAC. With 200 cases´ results, the learning process of ANN can be fulfilled.

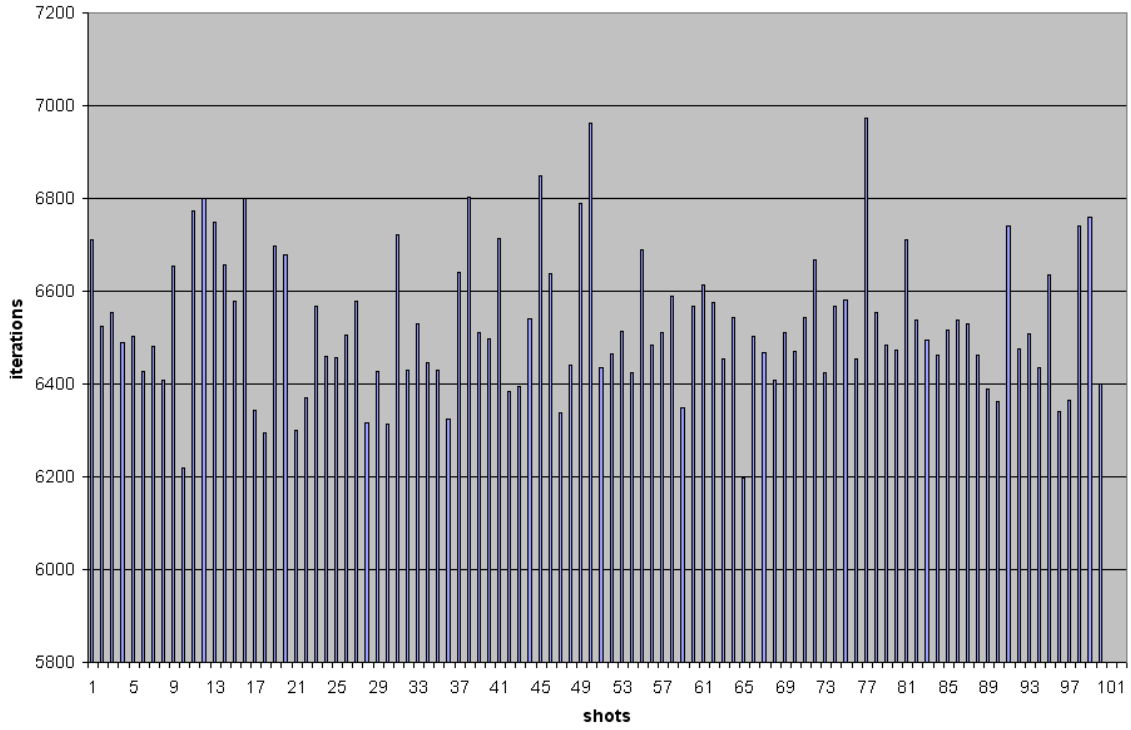


Figure 3-12: iteration number of each shot after introducing ANN

The mean value of number of iterations for Jacobi solver is 6529.1. Comparing with the result obtained by setting initial value as 0, number of iterations deduced obviously. Calculate the speedup:

$Speedup_{iter}$= number1/number2=9489.3/6529.1=1.4534

3.5 Conclusion.

In this chapter, we testify the speed-up of introducing ANN technique into an application in optimization. The purpose is to calculate the force of lift and drag for 200 airfoils with different shapes and to try to find the airfoil with optimal shape. During calculating the governing equilibrium matrix equations, we use iterative solvers. As it is mentioned before, for Jacobi iterative solver, normally we set the initial value as 0. The mean number of iterations for Jacobi solver is 9489.3. Then ANN method has been introduced. The learning and using processes have been implemented. With the help of ANN, a proper initial value is provided. Using this initial value, the mean number of iterations for Jacobi solver can be reduced to 6529.1. Decreasing of number of iteration means the speed-up of computer work.

Chapter 4

Conclusions

CALTEP is a calculus program to solve heat conduction problems of 2D models. In this thesis, two problems respectively in stochastic analysis and optimization have been solved in the platform of CALTEP.
During the process, three iterative solvers have been implemented in CALTEP: Jacobi iteration method, Gauss-Seidal iteration method, and conjugate gradient method.
After solving the problems in CALTEP, ANN technique has been introduced into the computational platform. Flood is a comprehensive implementation of the multilayer perceptron neural network. It has been modified to fulfill the process of learning and using.
The traditional way to solve the problem using iterative solvers is to set the initial value as 0. After introducing ANN, Flood predicts an approximate solution to the problem. If using this approximate solution as an initial value, number of iterations can be reduced.
In stochastic analysis and optimization, usually many runs are needed. If number of iterations is reduced in each run, a lot of time of computer work should be saved as a total.
The thesis is aiming to develop an ¨intelligent¨ finite element method of solving a specific class of problems in quasi real time using ANN techniques and iterative solvers. From the results discussed in previous chapters, the purpose is well fulfilled.

REFRENCES

[1] O.C. Zienkiewicz and R.L. Taylor, The Finite Element Method, McGraw-Hill (vol 1) (1989)
[2] J.E. Hurtado, F. Zarate and E. Onate, The Monte Carlo method. Application to the stochastic analysis of sheet stamping processes, Publication CIMNE, 2000