Intelligent Optimum Design with the support of Internet Techniques

Nariman Amin

A report submitted in partial fulfilment of the requirements of The Nottingham Trent University for the degree of Doctor of Philosophy

This research programme was carried out in the School of Engineering, The Nottingham Trent University, Nottingham, UK.

January 2003

Abstract

The aim of the research is to investigate the feasibility of an efficient implementation of Internet to engineering design, with particular reference to gear design optimisation and electronic cataloguing. The research has contributed a number of originalities into this area, as listed below:

- Implementation of a large size and complicated design software package over the Internet, in a multi-user environment.
- Dynamic creation of 2-D design drawing in the form of DXF (Data Exchange Format).
- Estimation of the output of a genetic algorithm, using Artificial Neural Networks (ANN) to improve the speed of the execution of the program.
- Improvement of the learning process of the backpropagation learning algorithm.

A gear optimisation software is used as an application area and research is carried out to implement this design software over the Internet in an efficient way. This would enable the geographically dispersed companies or remote clients to have full access to the design software at any time. As one of the key issues of the system, a method to remotely execute a large size software package over the Internet has been developed.

This optimisation program utilises Genetic Algorithm (GA) to search for the best possible solution. It works in a cascade fashion, comprising of two tiers. GA is implemented in both these tiers. The output of tier one represents the initial values to tier 2, which in turn carries out a fine-tuning task. ANN is integrated into the system to estimate the output of tier 1. This improves the speed of execution of the optimisation program dramatically, since the ANN execution time is negligible.

An investigation was also carried out to improve the performance of ANN backpropagation learning algorithm. It is proposed to re-scale the output data to lie slightly below and above the two extreme values of the full range neural activation function. This allows a more weight change for the output neurons that have reached the saturation.

The final area of this research project is the creation of a prototype online design catalogue for a gear manufacturing company. A novel feature of this catalogue is its ability to dynamically create the drawings in DXF format. These drawings can then be imported into CAD software that can accept DXF format, such as AutoCAD or ProEngineer.

Acknowledgement

I would like to express my sincere gratitude to my supervisor, Professor Daizhong Su for his guidance and for having confidence in me throughout the course of this research.

Special thanks to all my colleagues and friends who made these years memorable for me. Uwe Ruedel, Geza Nagy, Pawel and Ola Sobczak, Lorenzo Brignone, Hossein and Angela Jalali, Shiva Bonsaei and my true friend Oliver Butz. No words can describe my gratitude to Sara Otero Barros and Kunthavi (Kay) Nagendra for their true friendship and support. Without whom I would have been lost.

With special thanks to my brother for his support and guidance, whose endless encouragement inspired confidence in me and greatly smoothed the way through my study.

Deep love and respect to my sisters Froozan and Pooran, to my brother Ali and to my parents Hadi and Belgheis for their immense love and understanding. I am deeply indebted to them.

Acknowledgement

تقدیم به پدر و مادر گرامیم

با سپاس فراوان از زحماتتان

مژگان فیاشپور

و همچنین

بیاد کیوان و مهران عزیزمان

Nomenclature

AI	Artificial Intelligence
API	Application Programming Interface
ANN	Artificial Neural Networks
CAD	Computer Aided Design
CGI	Common Gateway Interface
DXF	Data Exchange Format
GA	Genetic Algorithm
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
JDBC	Java Database Connectivity
ODBC	Open DataBase Connectivity
RMI	Remote Method Invocation
SQL	Structured Query Language
TCP/IP	Transmission Control Protocol / Internet Protocol
URL	Uniform Resource Locator
WWW	World Wide Web
εt	Training error for ANN
εg	Generalisation error for ANN
%facewidth	Weight factor for the objective function of *Facewidth* in the gear optimisation program
%CD	Weight factor for the objective function of *Centre Distance* in the gear optimisation program
%slide	Weight factor for the objective function of *Equal Slide/Roll* in the gear optimisation program
%CR	Weight factor for the objective function of *Contact Ratio* in the gear optimisation program
%stress	Weight factor for the objective function of *Equal Stress* in the gear optimisation program

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

In recent years, Internet has started to revolutionise the world. It has affected the economy, society and specially the technological systems. The extent in which it has integrated into our system and its contribution to the progress of a company or a country as a whole, is undeniable. No enterprise can afford to underestimate its importance.

A detail research carried out by the United States Internet Council (ITTA, 2000) into the state of the Internet provides some interesting points. It emphasises that the Internet continues to grow at a phenomenal pace. Internet users have soared from 171 million in 1999 to 304 million as of March 2000. Business and Industry in every sector have begun to adopt the Internet to attract customers, facilitate communications and reduce costs. The report by the United States Internet Council concludes that today, companies are investing aggressively to keep pace with changing technologies to ensure success on the net. Countries and companies around the world are rapidly plugging into the Internet.

The globalisation is a worldwide trend for today and tomorrow's industrial practice. Manufacturers are facing more severe competition in the global market than at any time before. Manufacturers must work hard to achieve not only high quality, productivity and reduced cost, but also the ability to react quickly, responsively and effectively to the market which is becoming more international, dynamic and customer-driven. Many companies now are spread around the world. The design team could be based in one continent, manufacturing team in another and the marketing in a different continent. By implementing Internet technology into every aspect of the work, the teams are able to achieve exchange of information in a more robust and convenient manner. Users, customers, designers, developers and sales representatives across the globe must have a unified and quick access to their product and services.

In order to maintain and improve the competitiveness of their business in global market, manufacturing companies have to continuously improve their manufacturing environment through applying new technologies such as new design and manufacturing tools, quality inspection systems, control facilities, post-process monitoring and other systems which may benefit their business operations. This often results in a common situation in which all the computer-based manufacturing tools within the companies are made by different vendors or manufacturers. Such a heterogeneous manufacturing environment will inevitably not only result in problems in management, communication and production, but will also hamper the systematic upgrading of design and manufacturing tools and thus affect the competitiveness of a company.

Internet technology is a promising technology particularly in the engineering field to overcome such problems in the global competition. It has the potential of extending the traditional plant information system and eliminating barriers to integration. Its platform-independent architecture allows manufacturers to deploy applications across virtual organisations, which can include various teams within the organisation, business partners or customers whose computing environments may be completely different.

It was this urge towards Internet that triggered this research project. The success of a company and more important, the progress of a country relies very much on how advance they are in utilising the potential of Internet in their communication, exchange of information, design and manufacture. In this research project, attempt is made to develop an Internet-based system with the aid of Artificial Intelligence (AI) technologies for design optimisation; and gear design is used as a vehicle to illustrate the approach developed. The system provides a comprehensive design support environment for the users and to achieve gear design optimisation and gear selection by utilising the advantages of Internet technology.

1.2 Overview

This section presents an overview of the research project. The aim of the research is to investigate the feasibility of an efficient implementation of Internet for engineering design. The approach is developed and implemented through the integration of Artificial Intelligence (AI), Internet technologies and electronic database with gear design technique. The structure of the research project has been constructed graphically as shown in figure 1-1.

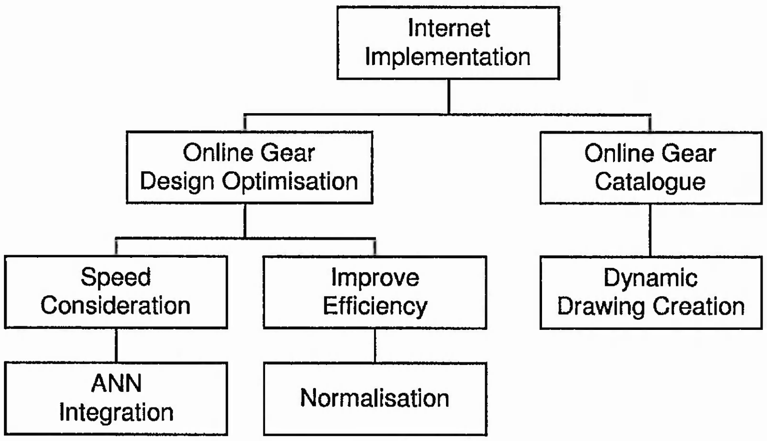


Figure 1-1 Overview of the proposed research

As can be seen in the figure, two main areas have been investigated: online gear design and online gear catalogue. A gear design optimisation is used as an application area and research is carried out to implement this design software over the Internet in an efficient way. This would enable the geographically dispersed companies or remote clients to have full access to the design software. The multi-user environment has been taken into consideration, where multiple users will be accessing the software package simultaneously. This gear optimisation program utilises genetic algorithm (GA) to search for the best possible solution. One drawback of GA is that it is computationally expensive. It takes a long time for the program to complete a given task. This is a disadvantage for an Internet environment where speed is a crucial factor. Consequently, research was carried out to address this problem.

The original version of the optimisation program works in a cascade fashion, comprising of two tiers. GA is implemented in both these tiers. The output of tier 1 represents the warm values to tier 2, which in turn carries out a fine-tuning task. In order to speed up the optimisation process, artificial neural networks (ANN) has been integrated into the system. ANNs are trained to estimate the output of tier 1, replacing the GA of this tier, which leads to eliminating the iterative operation of the genetic algorithm for the first stage. This would dramatically reduce the speed of execution of the optimisation program, due to the fast execution time of ANN.

An investigation was also carried out to improve the performance of ANN backpropagation learning algorithm. It re-scales the output data to lie slightly below and above the two extreme values of the full range neural activation function. This allows a more weight changes for the output neurons that have reached the saturation and hence results in a more accurate prediction. Data rescaling allows the neural networks learn faster and perform better.

The second area of this research project is the creation of an online design catalogue for a gear manufacturing company. The catalogue would be accessible from any part of the world. The clients will have access to the up-to-date data at any time. An important feature of this catalogue is its ability to dynamically create the drawings in DXF format. These drawings can then be imported into CAD software that can accept DXF format, such as AutoCAD or ProEngineer.

To achieve these goals a number of specific objectives are required to be met. These are listed below:

- Review current work associated with application of Internet to engineering design and manufacturing.
- Develop a methodology for an Internet-based design system and investigate its associated issues.

- Develop a methodology to implement a gear design optimisation program over the Internet.
- Investigate the speed issue of Genetic Algorithm implemented in the gear design optimisation program and propose a solution to accelerate the process.
- Investigate the possibility of improving the learning process of backpropagation learning algorithm of the Artificial Neural Networks.
- Review current work associated with electronic catalogues for engineering design.
- Develop an approach to create a dynamic online electronic catalogue for a gear manufacturing company, to be able to retrieve the appropriate data and generate the drawing dynamically for the selected gear.

A number of publications have arisen from this research, which are listed in appendix G.

1.3 Structure of thesis

This thesis consists of 9 chapters. A brief description of each chapter is given here.

1.3.1 Chapter 2 – Literature Review

Research and literature relating to the major topics of the project are investigated and reviewed. Examples and issues of the current research into the application of Internet within the different fields of engineering and manufacturing are presented and discussed. It shows the extent in which the Internet has integrated into engineering design and manufacture, and the variety of methods used to accomplish their tasks. This chapter also outlines the AI concept, mainly genetic algorithm and artificial neural networks. A brief description of these two techniques as well as their application is presented. In the final section of this chapter, a historical review of electronic catalogue is given, including examples of the current online electronic catalogues. The methods for remote access of databases are introduced and discussed.

1.3.2 Chapter 3 – Internet Approach for Gear Design Optimisation

In this chapter, the functionality and the structure of the original gear design optimisation software are reviewed first. The advantages of the selected technique for executing a standalone software over the Internet are then explained. The efficiency of the system in the Internet environment depends on the appropriate allocation of the tasks to server and client. This issue is discussed thoroughly, hence a methodology is presented based on a combination of HTML, JavaScript, Java Applet and Common Gateway Interface (CGI) to accomplish the task. The features of Java applets are presented, illustrating their usage in the graphical user interface. In the last section, problems that could arise in a multi-user environment are discussed and a method is proposed to overcome these problems.

1.3.3 Chapter 4 – Integration of Artificial Neural Networks into the GA Optimisation

Genetic algorithm used in the gear design optimisation software has the disadvantage of being slow to execute. This is a drawback specially for an Internet environment. This problem is addressed in this chapter. To overcome this problem, a method of integrating ANN into the system is developed. The ANN training method is explained, illustrating the important features that need to be taken into consideration for improving the performance of the network, such as the learning rate, momentum term, size of the network and the initial weights. The problem area is analysed in depth, working out the design parameters needed for the input/output training patterns. In addition, a number of ANNs are introduced to tackle the problem rather than using a single network. At the end of the chapter, the results are discussed, illustrating the improvement in the speed of the execution of the optimisation program.

1.3.4 Chapter 5 – Improvement of Performance of Backpropagation Learning Algorithm

A review of the backpropagation learning algorithm is presented in this chapter. A problem area is studied where the output derivative is near the saturation levels. In such situation, the generalisation error of the network increases. An approach is demonstrated that re-scales the data by changing the data pattern to allow the neural network to solve the relationship between input and output more easily. The method is tested on the function approximation and gear deisgn optimisation software. The results are presented at the end of the chapter, showing an improvement in the efficiency and speed of learning of the ANN.

1.3.5 Chapter 6 – Internet Techniques for Online Gear Catalogue

A historical review of the catalogues in the engineering field is provided, illustrating the trend toward the electronic format, particularly in the form of online catalogues. A background of the databases is also presented. The structure of Java Database Connectivity (JDBC) is discussed, explaining the features of the different drivers and the reasons for selecting type 3 driver for this project. The approach used for making a connection to the database using JDBC is discussed in detail. The performance is improved by pre-configuring the connection and statements. The development of the graphical user interface using Java applet is also illustrated.

1.3.6 Chapter 7 – Dynamic Creation of DXF Files

In this chapter, a method is presented that improves the functionality of the online gear catalogue. It generates the drawing of the selected gear dynamically in the form of DXF files. The approach is based on the combination of Java servlet, Java applet and C++ programming. The structure of the Java servlet is discussed and the problem associated with multi-user environment are presented and addressed. The important features of a DXF file are discussed. In addition, the chapter provides a detailed description of the DXF creation problem specially in hatching section. A parametric solution is provided which fulfils the task.

1.3.7 Chapter 8 - Discussion and Conclusion

In this chapter conclusions are drawn to the work described in the previous chapters and a summary of the results obtained are presented. Further work that could be carried out to build on the present knowledge is also included in this chapter. It mentions the further improvement to the methodologies presented in the previous chapters.

Chapter 2

Literature Review

2.1 Introduction

The rapid advances of information technology has affected every aspects of today's life. Not a single engineering field can be found that has not been touched by it, ranging from product design to highly complex manufacturing processes. In the current design environment, design teams as well as the tools and facilities are usually distributed and located at different geographical sites (Wang, et al, 1998). The success of the design work relies on the level of interaction between these agents. Miscommunication would result in the elongation of the production time, which could be devastative in today's highly competitive market. Internet is an ideal media that facilitates high level of interaction and information exchange, which is crucial for the design team.

The advent of the Internet and its associated World Wide Web (WWW) and Intranet, has sparked new activities among researchers into the development of the systems that would allow collaboration from remote sites by sharing ideas and information. This is particularly important for a geographically dispersed engineering groups which request for improved communication and cooperation among these groups in different locations.

The use of Internet demands for efficient system tools that are able to cope with the huge volume of requests by the end users. That is specially crucial when dealing with design solutions of iterative nature (i.e. genetic algorithm). In such tools the response time to every user should be as short as possible.

Artificial Intelligence (AI) is another rapidly developing technique that has gained much attention in the recent decades. AI involves the capability of a system to perform functions normally associated with human intelligence, such as reasoning, learning and self improvement. Developments in AI have had an increasing impact on engineering design and manufacturing, and almost all aspects of the area have been greatly affected and benefited by AI.

In this chapter, selected research works related to the theme of the project are reviewed. In the first section, research carried out on the application of Internet to engineering design and manufacture is presented. The issues of gear design optimisation and the application of Artificial Intelligence, mainly Artificial Neural Networks, are then presented. The final section deals with the product electronic catalogues and the current development status of on-line catalogues.

2.2 Application of Internet to engineering design and manufacture

National Institute of Standards and Technology (NIST) reported that it is envisioned that to compete successfully in the increasingly dynamic and global manufacturing environment characterised by short product life-cycle, demand for user customised products, and time based competition, manufacturing companies would use the Internet to form, coordinate and dissolve supply-webs in a dynamic manner (NIST, 1994). In this environment the companies that are not able to smoothly interface with others, and make rapid and effective decisions will find it challenging to remain competitive (Veeramani et al, 1998). Internet is a vital media in communicating between the different agents. In this section, a review of the selected works is given explaining their architecture and the methods they have used.

2.2.1 Network based collaborative CAD

In the past decade, concurrent engineering has been employed to shorten product development cycles, improve product quality and reduce product development cost by resolving product, process and organisation issues at the early stage (Qiang et al, 2001). This requires a seamless communication channel. Internet offers the potential for improved communication and co-operation among the geographically dispersed work-groups. Harvey et al (2001) have performed research in the area of concurrent engineering. They have developed an Internet-based CAD tool that can be used to access the existing CAD systems. CAD information from local disks and remote servers can be gathered on the company's Internet. They can then be assembled, displayed and modified on local screens.

Not all designers from different companies have the same CAD software and operating systems. In addition to this, expertise of product developers may be distributed worldwide and having a facility that is both platform independent and has the ability to remotely view and modify the design is very beneficial to them. The architecture of the methodology proposed by Harvey et al (2001) is based on CORBA and Java 3D. CORBA stands for Common Object Request Broker Architecture, defined by the Object Management Group (OMG). It defines a framework for developing object oriented, distributed applications (CORBA, 1997). Java 3D is a high level graphical tool and is a part of Java tools providing an application programming interface (API) for writing three dimensional graphical applications or web-based 3D applets (Java 3D, accessed March 2002). The CAD model can be represented in a variety of formats including DXF and VRML. The program can read a file with a particular format, e.g. DXF and then converts it to a Java 3D scene.

Same issue of transmitting and viewing 3D CAD models and engineering information has previously been addressed by Kim et al (1998). They developed a prototype system called *CyberView,* which can provide support for members of a distributed concurrent engineering team to share 3-D shape information. The functionality of the system is very similar to the one proposed by Harvey et al (2001), however the architecture of CyberView is based on CGI and Java for communicating between client and server rather than CORBA, and the 3D drawing is done solely using VRML, whereas in the system of Harvey et al (2001) Java 3D was used for this purpose. In CyberView, the CAD models are stored in STEP format. Upon their request, these are converted into VRML file format which can then be viewed by other users.

In addition to providing access to 3D models, Cyberview also deals with the issue of communicating between team members. It provides another facility that allows a client to write a markup for one or more components. The component is stored in a database on the server which can then be retrieved by other servers.

One of the early web applications in design and manufacturing is rapid prototyping. Smith et al (1996) developed a system called *Cybercut* that assists the designer to access a rapid prototype service. It allows client-designers to quickly design and manufacture prototypes for mechanical parts. When the design is completed, the part specifications are sent directly to a computer controlled three axis milling machine and fabrication can then begin immediately (Cybercut, accessed February 2002). Cybercut capitalizes on Java's client-side processing to give the CAD designer real-time manipulation of the part in progress, while at the same time the more complex programming and computer-intensive tasks are assigned to the server. The internal structure of a single agent and some of the information flow is shown in figure 2-1.

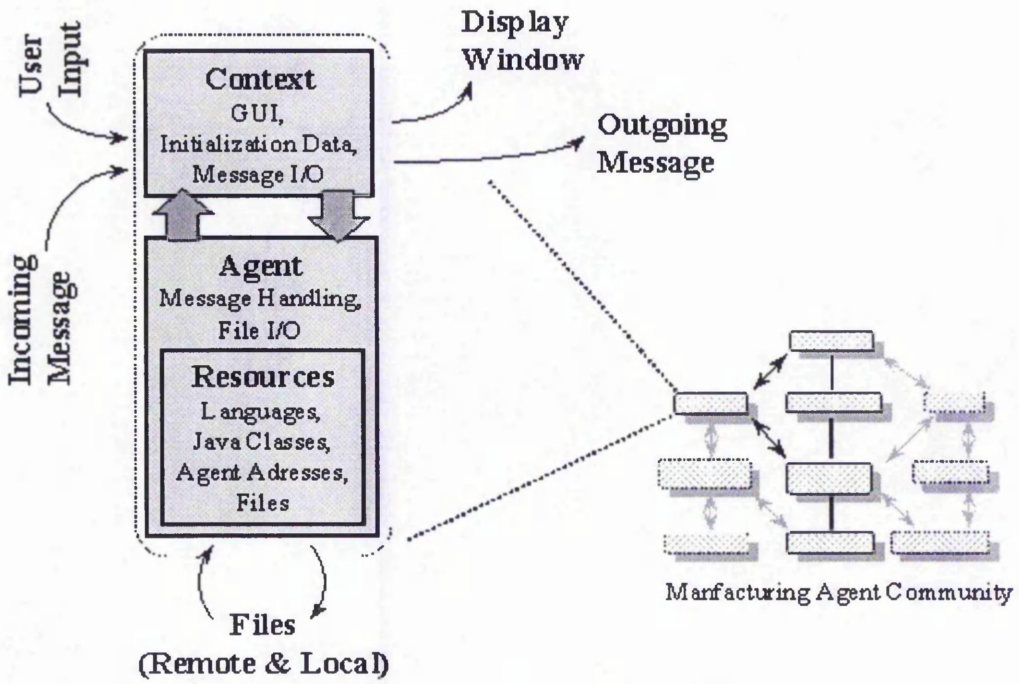

User Input
Incoming Message
Context
GUI, Initialization Data, Message I/O
Agent
Message Handling, File I/O
Resources
Languages, Java Classes, Agent Adresses, Files
Display Window
Outgoing Message
Files (Remote & Local)
Manfacturing Agent Community

Figure 2-1 Structure of Cybercut, (Wang, 1998)

Cybercut constitutes of a tool called WebCAD-3D that facilitates the design process for producing machinable parts (Wang, 1998). WebCAD-3D is developed using Java applet and is available online via WWW connection. (WebCAD, accessed February 2002). The design tool has a 2D editing interface with a 3D wireframe display capability. The WebCAD-3D design applet creates a communication socket and binds it to a dedicated port on the server machine through the Java communication server.

2.2.2 Internet-based Manufacturing system

Jiang et al (1999) have developed software called *TeleRP*, for servicing and maintaining rapid prototyping Tele-manufacturing. A Java-enabled solution based system is adopted for implementing the remote part submitting, queuing and monitoring. Java applets are used to create the interface between users and manufacturers and also to implement the client-side computing. Java Servlets are used for client server communication. It acts as the core role for server-side computing. The main functions of this system are: Submitting and queuing a part, monitoring manufacturing tasks, managing manufacturing queues, maintaining manufacturing sites and starting a collaborative work if necessary. Karn et al (1998) have also used Java-based method for their system architecture, called *Web-It-Man*(Web-based Integrated Manufacturing). It allows users to invoke a particular tool to design a specific task. The portability is achieved by using Java programming environment for implementation. The architecture is based on Java applet, Remote method Invocation (RMI) and Java Database Connectivity.

Paidly et al (1996) have carried out research on the application of Internet for manufacturing control. They have developed a system that takes order inquiry from the remote clients via World Wide Web for their Computer Integrated Manufacturing System called *CAMCELL*. CAMCELL is an integrated flexible manufacturing system that is capable of machining parts that can be made with milling and turning. Customers are presented with the details of the parts that can be made in CAMCELL. Once an order is accepted, the parts are made automatically. The structure is based on a combination of HTML, CGI, Java and JavaScript. Server side processing has been implemented using CGI script. Java and JavaScript have been used for client side processing. Mok (1998) has also used similar structure based on Java Applet and CGI for controlling manufacturing processes remotely. It allows the controller to communicate with factory engineers via the local area network.

Internet technology has also entered the domain of embedded systems in industrial installations. Web pages from the embedded system need to be generated dynamically to convey the current device state to the user. End-users can also use Web browsers to send information to the embedded system for configuration and control of the device (Agranat, 1998). Itschner et al (1998), have designed and developed *GLASS* (Global Access for Service and Support), for remote monitoring of devices. Through the Internet, GLASS provides access to embedded systems in order to service and support a wide range of facilities and installations, from unmanned substations to large plants or devices on mobile platforms like trains or ships.

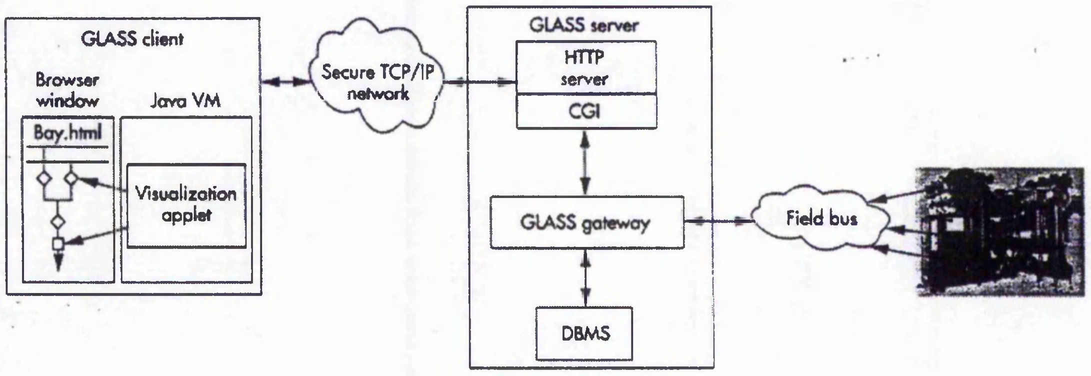


Figure 2-2 GLASS system architecture, (Itschner et al, 1998)

The GLASS remote monitoring system has a typical three-tier architecture, as shown in figure 2-2. It consists of a Web browser with Java applets as clients and a server that shields the embedded systems. Java applets on the client's machine communicate with the GLASS server and retrieve and visualise live data from the plant. The GLASS server consists of three applications. An off-the-shelf HTTP server, a database management system and the GLASS gateway. The GLASS gateway is the core of the system. It retrieves raw data from the actual devices in the plant through a field-bus connection. It pre-processes this data and stores it in a database system. Clients access the gateway using CGI program.

Jiang et al (2001) have carried out research into the Internet based design system for globally distributed manufacturing. They have developed a three-tier client/server structure on which a tool is built to visualize CAD-neutral formats. This tool allows users to gather CAD models from local disks, WWW servers, and others on the company's Intranet, and then assemble and display them on the local screen. Through this facility, users can modify and save a CAD model into a database so that all project participants can work on the same set of information. The tool is platform independent with superior local 3D manipulation capabilities and supports DXF, OBJ, and VRML formats. They have utilised CORBA and Java 3D for development of their system. CORBA is used to link different systems over the Internet or an Intranet. Java3D is used to enable users to interact with a design model irrespective of time and distance. It is suitable for geometrical object manipulation.

The Virtual Reality Laboratory at the University of Michigan has explored the revolutionary technology of virtual reality for collaborative applications over the Internet in support of concurrent engineering and distributed work over distances (Beier, K. P., 2000). The laboratory has explored the usefulness of VRML in design and manufacturing through a variety of studies and projects. They have applied this technology to a number of design and manufacturing tasks including ship motion simulation, modelling of robot operations, virtual prototyping of sailing yachts, simulation of ship production processes, virtual factory planning, and others. As an example, in the area of robotics, they have modelled functional robots using VRML's scripting capability. This paves the way for a future Web environment where robots (or other machinery) can be downloaded from the supplier's Web site and placed in the virtual model of a factory. Ultimately, the downloaded equipment would be fully functional and would allow for the simulation of a production process before any hardware is being bought and installed.

2.2.3 Collaborative design

A research project jointly undertaken at Leeds Metropolitan University and Glasgow Caledonian University, presents an approach to develop an Internet-based design support system for rolling bearings (Cheng et al, 2001 and Cheng et al 2000). The system aims at providing bearing selection and design support for engineering community via Internet. An authorised client can remotely log in and utilise the facilities provided. The system presents detailed design information on a number of bearing mounting methods for supporting rotational machinery. It also provides access to manufacturing database and electronic catalogue. It makes use of Artificial Neural Networks and Fuzzy Logic for intelligent selection of bearing.

The system is developed using HTML, JavaScript and Java programming with AI techniques. Remote Method Invocation (RMI) manager is used to control the access to remote design facilities through Java Database Connectivity. The proposed structure is shown in figure 2-3.

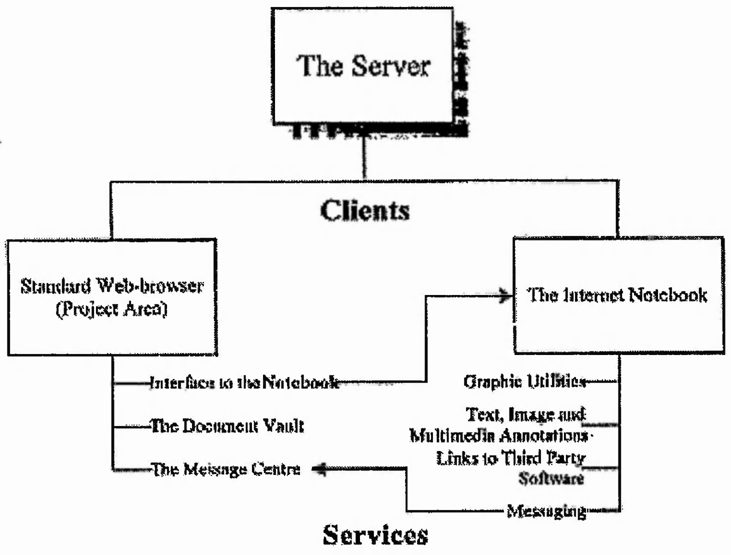

The Server
Clients
Standard Web-browser (Project Area)
The Internet Notebook
Interface to the Notebook
The Document Vault
The Message Centre
Graphic Utilities
Text, Image and Multimedia Annotations
Links to Third Party Software
Messaging
Services

Figure 2-3 The three-tier architecture of integrating information resources, (Cheng et al, 2001)

Another area that has attracted research is Computer Supported Collaborative Work (CSCW). Design generally proceeds by cooperating specialists who achieve common understanding of a design model and who reach a consensus of design through a series of discussions and negotiations. Since many team members and design experts are now geographically dispersed, it would be very beneficial to conduct the meeting over the Internet. CSCW provides a basis for people to hold meetings in which the participants are geographically distributed and the computer provides the medium for communication. There are two types of possible applications: asynchronous and synchronous. In an asynchronous situation, different users do not share decisions or decision-making tasks, whereas a synchronous application allows multiple users of the same project to share decision in a common workspace (Cheng et al, 2001).

Nidamarthis et al (2001) have developed an Asynchronous Collaborative Software (ACS) tool. An Internet notebook is installed on each client's local hardware. The notebook is a multimedia, asynchronous environment for collaborative authoring that allows the team to sketch, paste images and annotate with text, graphics, audio and video. The basic architecture is shown in figure 2-4. Team members can access the server using one of the two routes: standard web browser or the notebook. The routes provide different services as shown in the figure. The architecture uses VRML to help the design team visualise the solid model over the Internet.

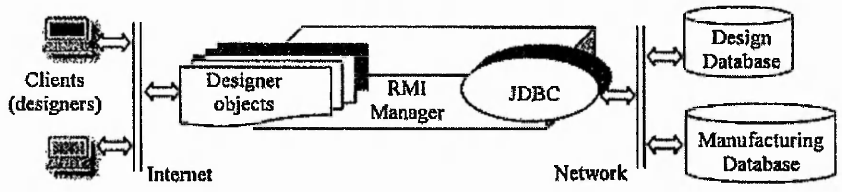


Figure 2-4 Basic architecture of the ACS, (Nidamarthis et al, 2001)

Maher et al (1997) conducted research in synchronous collaborative design. They outline the requirements for a collaborative design environment, i.e. a shared workspace, an application domain and data management and propose a framework for building a collaborative environment based on these requirements. Another example of research in this area is the work conducted by Duke et al (1999). Their research is based on telepresence environment. Telepresence is the facility that enables collaborating parties to be virtually located within a given (3D) environment, in which they are able to interact with one another or with virtual objects that are present in that environment. These visual representations will provide access to underlying project data (drawings, schedules, etc.) and to the people themselves via the integrated communication channels. The environment contains a 3D representation of the construction project. It is multi-user and contains embodiments of the other users.

2.2.4 Summary

In the previous sections, research in the implementation of Internet in different aspects of engineering design and manufacture were reviewed. This is one of the fast growing technologies and the research in this area is advancing rapidly. There are various techniques available, which were described in the previous sections. CGI and Java and JavaScript have widely been used for the process of communication between the client and server. CORBA is also gaining popularity as seen in the work of Harvey et al (2001). Many have implemented VRML for displaying the three-dimensional graphical images.

For the ease of observation and comparison, a number of design works, that were described in this chapter, are presented in table 2-1. As can be seen in the table, CGI and Java are the most used techniques for the designs. This is because these two methods are both freely available and both are implemented in a familiar environment that are very similar to the C++ software tools; i.e. CGI can be written in C++ and java classes are similar to C++ object oriented implementation. The familiarity of the developers with the C++ environment and the efficiency issues has lead to the development of C++-like tools for the Internet and will continue to do so in the future implementation (or improvement) of new tools.

	Communication	**Interface & 3D**
CAMCELL	CGI	Java,CGI
Cheng	JDBC/RMI	Java/JavaScript
Cybercut	Java	Java
Cyberview	Java, CGI	VRML
GLASS	Java/CGI	Java
Harvey	CORBA	Java3D
Mok	CGI	Java/CGI
Nidamarthis	JDBC/RMI	VRML
TeleRP	Java servlet	Java
WebItMan	Java database	Java

Table 2-1 Comparison of different Internet based designs

2.3 Application of AI to gear design

Artificial Intelligence is the study of how to make computers do things which, at the moment, people do better. It is the science concerned with the creation of machine intelligence which is able to perform tasks formerly performed only by people (Bernold, 1987). AI has experienced a rapid growth in industrial application during the past few years. The types of problems which AI attempts to solve are non-linear and complex (e.g. planning/scheduling, image understanding, etc.). The impact of their being non-linear is that there do not exist algorithms which will provide optimal solution to an application while catering for various conditions and parameter changes. Gear design optimisation is one application where AI has been used successfully.

2.3.1 Gear Design

Gears are used in a wide range of engineering design to transmit power from one shaft to another. Examples of power gearing are the transmission and differential gears in automobiles and the gears in electronic hand grills. There are a wide variety of types of gears in existence, each serving a range of functions. The types in general use may be grouped under four headings: spur gears, helical gears, bevel gears and worms gears. In almost every application, the major requirement is for the smoothest possible transfer of force and motion. Thus the basic criterion for gear tooth profiles is the ability to transmit motion in such a manner that the angular-velocity ratio is constant at all times.

In most cases, the design of gears is a highly complicated task involving the satisfaction of a number of design constraints and the quality of the design depends on the designer's knowledge and experience. It involves taking into account large number of parameters and conditions which is done by experts who have gained knowledge in this area due to years of experience. Neglecting any of these could result in the failure of the design, therefore the design assessment requires many compromises (Wang et al, 1994). Several approaches for gear design have been proposed. Among those, the use of optimisation techniques has received much attention (Wang et al, 1994 and Savage et al, 1982 and Houser et al, 2000). Optimisation techniques usually require the minimisation of an objective function that is usually a combination of the various parameters (Houser et al, 2000). If there are many design parameters in the objective function, it is difficult for the designer to assess the importance of each one. This scenario exists in gear design. Slight changes in the objective function of gears would result in an entirely different design.

Su et al (2000) have developed a gear design optimisation software package, called *OptMMGear*. The software utilises genetic algorithm to perform the search for the design configuration that will give the optimum performance for spur and helical gears. The optimisation process can modify up to 9 parameters of the gear design including:

- Facewidth
- Module
- Pressure angle
- Helix angle
- Rack tip radius
- Addendum coefficient
- Addendum modification coefficients for both Pinion and Wheel
- Number of teeth on the Pinion

Selection of these parameters will include them in the optimisation process. Non selection will freeze the values to those defined in the initial design. The user should also provide information about the genetic algorithm which includes the fitness functions, population size and the number of tests.

There are five optimisation objectives including minimising facewidth, minimising centre distance (variable centre distance only), reducing the difference in tooth root bending stresses between the pinion and wheel, increasing contact ratio, and reducing the difference in tooth tip sliding speed between the pinion and wheel. Three design constraints are considered during the optimisation, including that tooth root bending stress cannot exceed the allowable stress, tooth contact stress cannot exceed the allowable stress, and sliding/rolling speed ratio cannot exceed 3.

The basic configuration of the gear design must be provided by the client which includes geometry, performance and material information. Once the user has entered the initial design, then the optimisation process can proceed. Genetic algorithm conducts an adaptive search of various configurations of gear, derived from an initial design. The numerical analysis program is invoked by the genetic algorithm to calculate the tooth strength. The results of the genetic algorithm optimisation are displayed to the user, giving both the current performance and relative performance to the initial design.

The genetic algorithm optimisation process is applied in a cascade fashion. The procedure comprises of two tiers, see figure 2-5. The cascade procedure requires an initial, rough starting design to base the optimisation process upon. The initial values of the rough design form the starting positions and limiting conditions for the parameters that are to be optimised. The first tier of the optimisation is invoked using the rough design or warm values, provided by the user, to adjust the parameters in search of a global optimum. The optimisation process continues until a limiting percentage of the genome population are identical. At this point the information encoded within the converged genome is decoded forming the solution to this tier and the intermediate warm values for the next tier.

In tier 2, the genetic algorithm optimiser is initiated again with the solution from the previous optimiser, applying a narrower, more accurate band to the search. Again the search is repeated until the limiting percentage of the population are identical, at which point the converged genome is decoded to form the final solution.

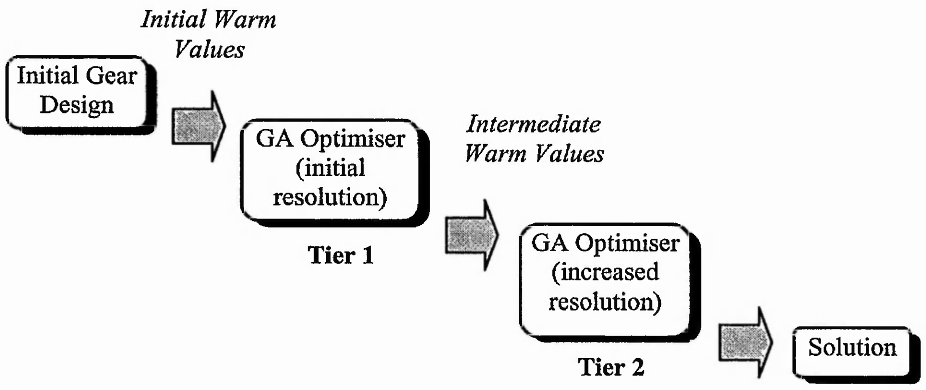


Figure 2-5 The cascade procedure of GA optimisation, (Su et al, 2000)

2.3.2 Genetic Algorithm

Genetic Algorithm has been used for complex problems such as machine learning. Advantage of GAs is in their parallelism. GA is travelling in a search space with more individuals so they are less likely to get stuck in a local extreme. Disadvantage of GAs is in their computation time. To get an idea about problems solved by GA, here is a short list of some applications:

- Nonlinear dynamical systems – predicting, data analysis
- Robot trajectory

- Strategy planning
- Finding shape of protein molecules
- Functions for creating images

GA offers significant potential to produce solutions to difficult problems. GAs are based upon natural biological evolution. Just as living organisms evolved to cope with their environments by reproduction of the strong through generations, the genetic algorithm develops strings of parameter (called chromosomes), which characterise performance, by combining features of successful (or fit) chromosomes within a population to create the next generation. Unfit chromosomes are discarded from the population. As the generations progress the fitness of the chromosome should increase to a maximum.

To ensure that a local maximum is not obtained a mutated chromosome is introduced into the population periodically, increasing the search area. This is achieved by applying two evolutionary processes known as mutation and cross over, to produce alternative new string (Rzevski et al, 1994). Mutation occurs when an existing binary string has one of its bit values selected at random changed, either from 0 to 1 or 1 to 0. This process is illustrated in figure 2-6.

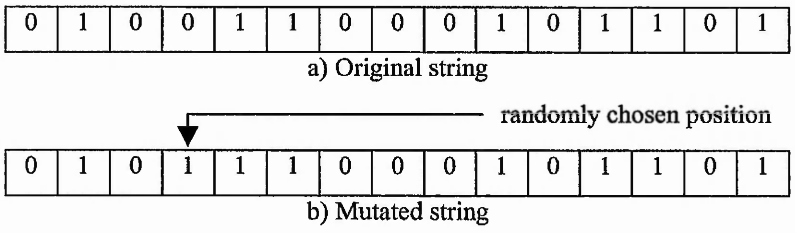
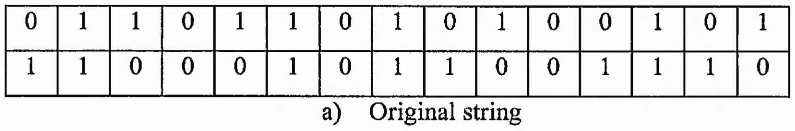


Figure 2-6 Mutation process in GA

Cross over occurs when step changes solutions are obtained by taking a pair of binary strings, cutting them at the same randomly chosen point and then swapping fragments, as shown in figure 2-7.



0	1	1	0	1	1	0	1	0
1	1	0	0	0	1	0	1	1

1	0	0	1	0	1
0	0	1	1	1	0

b) split strings

0	1	1	0	1	1	0	1	0	0	0	1	1	1	0
1	1	0	0	0	1	0	1	1	1	0	0	1	0	1

c) New string

Figure 2-7 Cross over process in GA

2.3.3 Artificial Neural Networks

There are large classes of problems that appear to be more manageable to be solved by ANN than by other available techniques. These tasks often involve ambiguity. Problems of this sort are difficult to tackle with conventional methods, in part because the metrics used by the brain may not be very closely related to those chosen by an engineer designing the system.

Perhaps the most important advantage of ANN is their adaptivitiy. ANNs can automatically adjust their parameters (weights) to optimise their behaviour as character recognisors. Self-optimisation allows the ANN to design itself. The system designer first defines the ANN architecture, determines how the neural connects to the other parts of the system, and chooses a training methodology for the network. The ANN then adopts to the application. Adaptivity allows the ANN to perform well even when the environment or the system being controlled varies over time.

An Artificial Neural Networks is a "black box" that accepts inputs and produces outputs. It is an information processing network consisting of a large number of interconnected processing elements, called neurons. The mathematical relationship between input and output is informed through an iterative adjustment of a set of weights during the learning process. The ability to solve highly complex problems by learning from examples is the outstanding feature of ANNs (Lippmann, 1987).

Multi-layer perceptron consists of layers of interconnected neurons. The knowledge of a multi-layer neural networks is distributed over the nodes and their connecting links. Patterns are presented to the network via the input layer, which communicates to one or more hidden layers where the actual processing is done. The hidden layers then link to an output layer where the answer is output. Multi-layer perceptron networks trained by backpropagation algorithm are capable of achieving the effective result for the problem in hand. Backpropagation is an abbreviation for the backwards propagation of error. Backpropagation consists of three segments, feed-forward, weight-adaptation, and feed-back (or error-backpropagation).

Once a neural network is trained to a satisfactory level, the user no longer specifies any training runs and instead allows the network to work in forward propagation mode only. New inputs are presented to the input pattern where they filter into and are processed by the middle layers as though training were taking place, however, at this point the output is obtained and no backpropagation occurs.

Artificial Neural Networks are often used for statistical analysis and data modelling, in which their role is perceived as an alternative to standard nonlinear regression or cluster analysis techniques (Cheng et al, 1994). Thus they are typically used in classification or forecasting problems. This type of problem also falls within the domain of classical artificial intelligence so that engineers and computer scientists see neural nets as offering a style of parallel distributed computing, thereby providing an alternative to the conventional algorithmic techniques that have dominated in machine intelligence (Gurney, 1997). The parallelism refers to the fact that each node is conceived of as operating independently and concurrently (in parallel with) the others, and the "knowledge" in the network is distributed over the entire set of weights, rather than focused in a few memory locations as in a conventional computer.

Applications of ANN have been developed for problems in all major engineering disciplines, including image and speech recognition, textual character recognition, signal processing, machining control, design optimisation, etc (1994). Rzevski et al (1994) have provided a list of research applications in different fields of engineering, including chemical and process engineering, civil and structural engineering, electrical and electronic engineering, manufacturing and mechanical engineering, and systems and control engineering. Some of the given applications in the area of manufacturing and mechanical engineering are shown below:

- Predicting chip breakability and surface finish in metal machining (Yao et al, 1993).
- Optimising machining parameters (Wang et al, 1993).
- Classifying machine faults (Engelbrecht et al, 1999).
- Designing the configuration of aircraft wing box structures (Widrow et al, 1994).

Research has also been conducted in the attempt to improve the performance of ANN. An example is the work done by Engelbercht et al (1999). They have presented an algorithm to prune feed forward neural networks architecture. Based on their given results, the algorithm has reduced the architecture of the neural networks, resulting in an improvement in the rate of learning of ANN.

2.4 Electronic catalogue

Standard components or parts make up a large proportion of engineering assemblies and systems. The clients can select the required standard part from the catalogues in the form of hard copy that is produced by the manufacturer. Large sum are spent on the publication and distribution of this information. This method is inefficient because it is time consuming for the user to get the required information and by the time the user is searching the catalogue, the information could be out-dated (Tumkor, 2000). Studies have shown that eighty percent of an engineer's time is spent not on design, for which they are trained for, but on communications, such as obtaining information about components comprising the design and coordinating with other engineers about design issues (Coutinho et al, 1998). Thus, the speed and quality of product development is heavily dependant on designer's ability to configure the suitable components from the product catalogue.

With the development of computing techniques, product catalogues in electronic forms, or so-called electronic catalogues have been increasingly emerging, for example (RS, 1998 and Lee Spring, 1997). The advent of the CD-ROM technology means that useful package of information can be delivered cost effectively to a wide variety of customers. In comparison to the traditional product catalogues, i.e. the ones in the forms of hard copies, electronic catalogues have obvious advantages, such as compact size, large amount of information storage and data handling by the means of computers.

The manufacturers of the standard components still deliver the majority of information to designers in the form of printed catalogues. Currently there are a number of electronic catalogues available on the Web, however many of these catalogues are static. They are made up of a number of simple HTML pages with hard coded tables of data and images. If any changes need to be done to the catalogue, the HTML code needs to be modified (Mentornet, Fesca, Mullins, SAIA-Burgess, accessed March 2002) In this section, a review of some of the available online design electronic catalogue is given. It is then followed by description of the databases, pointing out the techniques used for Internet implementation.

2.4.1 A review of the online design electronic catalogues

Automec and Oakes Tools (Automec, March 2002) are sister companies based in Buckinghamshire, England who specialise in the manufacture and distribution of quality automotive components and tools. They specialise in brake pipes, brake pipe sets and silicone brake fluid. Their catalogue is still in the traditional hard copy format containing the descriptions and ordering information for over 5000 items. The company states that they will shortly be available on-line, however this will be in the static PDF format, allowing the client to view and print their own copy.

Live Steam Models (LiveSteamModels, accessed March 2002) produce traction engines. They have two catalogue: *Models* catalogue offers a range of drawings and parts for complete traction engines and *Supplies* catalogue offers the materials needed for the model steam. However both these catalogues are static. The former is in HTML format displaying the picture of the model with its general dimension. The latter is in the form of word document, which could be downloaded.

Frame World Structural Aluminum Framing Components (Frame-world, accessed March 2002) provides a system that allows quick and easy design and assembly of machine bases and frames, stands, guard assemblies, material handling fixtures and tooling supports. The company provides a catalogue of the product components online. For each component, the user can download the specification for that part, or the drawing in the form of DXF or DWG.

Sherline Products (Sherline, accessed March 2002) manufactures a complete line of miniature lathes and vertical milling machines. They produce lathes, lathe accessories, vertical milling machines and mill accessories. The company claims to be *the world's most complete line of small precision machine tools and accessories.* However even though it provides online list of the products, it still depends on the traditional hard copy for the instruction and catalogue of the products.

Quality Transmission Components (QTC, accessed March 2002) provides an online catalogue for different types of gears. In addition to displaying the table of data, the user has the option of downloading the drawing in Data Exchange Format (DXF) for a few components. Both the data and the DXF file are hard coded. The DXF has already been created by the developer and the client could download this by clicking on a link.

S.H.M. On-line Catalogue (SHM) makes use of Acrobat Reader to display the data. The information is presented efficiently, but once again it is static and does not offer further features to the client. SKF (SKF, accessed March 2002) presents an outstanding catalogue. It uses Java applet to display the menu for user's selection and also for displaying the table of data, which is scrollable. It does not however create any drawing for the user.

2.4.2 Databases

A study on the growth of digitised databases shows that over the time period from 1975 to 1998, the databases have grown by a factor of 38, producers have grown by a factor of 18 and vendors have grown by a factor of 23 (Williams, 1999). In terms of form of representation, 71% of databases in 1998 were of the word-oriented type (bibliographic, full-text, patent/trademark, directory, and dictionary), 14% were number-oriented, 11% were image or picture-oriented, 3% were audio or sound oriented and the remaining 1% included various electronic services and software. In terms of the application area, the largest number of databases is in the business field (26%). Second is science/technology/engineering with 17%. Third is general news with (15%) followed by health and life sciences each with 11% and the other miscellaneous fields make up the remaining 20%. These statistics show that electronic databases have entered all the fields with outstanding growth in the field of business and engineering. Today's statistic could even show a much higher rate in these two fields. In this section, selected research in the area of application of online database to engineering design will be discussed, describing the techniques they have utilised.

2.4.2.1 CGI based technique

Many web-based database applications access databases through CGI (Gulesian, 1998). CGI is a standard for external gateway programs to interface with information servers such as web servers. Upon user's request, a gateway program known as CGI script, is executed at server site. Using the parameters received from the client, the gateway program accesses the databases through the underlying DBMS (Database Management Systems) and sends the result back to the server, which are then sent to the user. The CGI interface has many advantages, including portability of server software and wide availability of public domain gateway programs and development tools (Xia et al, 1999).

Roy et al (1998) use CGI program for their online database. They have defined a framework called *INDEMAND*, that utilises Internet technology to facilitate the maintenance of the supplier database for the Original Equipment Manufacturer (OEM). The system includes three distinct components: INDATA (Indemand DATa Acquisition system), ENGINE (Engineering design using the web technology) and the INDEMAND Database. ENGINE extracts supplier capability information from the structured web pages created by INDATA, and populates the INDEMAND database at the server machine. ENGINE is a CGI program written in Visual Basic for the website server. The system in integrated with an ACCESS database (INDEMAND database) located on the server machine.

CGI however is not the best solution for online databases. Most gateway programs are external to the server. A process has to be created for each request with database accesses. Creating a process is time-consuming and expensive in terms of the server's main memory, and can also exhaust resources available to the server applications.

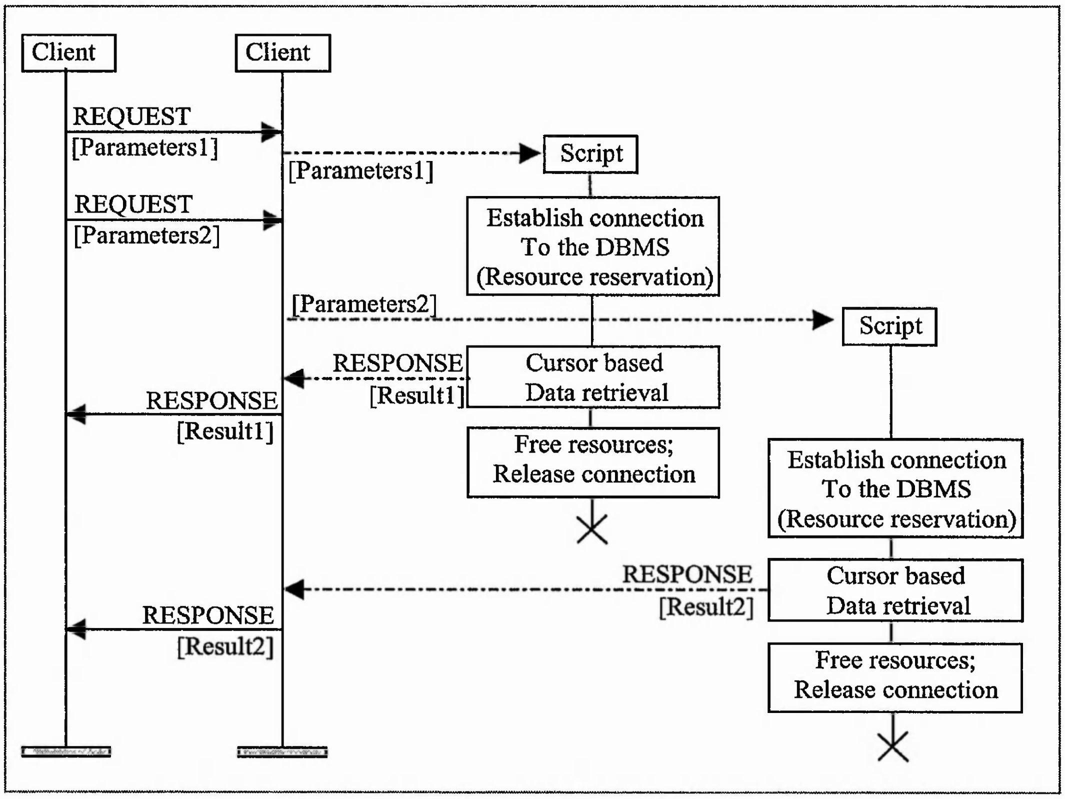

Client
Client
REQUEST
[Parameters1]
[Parameters1]
Script
REQUEST
[Parameters2]
Establish connection To the DBMS (Resource reservation)
[Parameters2]
Script
RESPONSE
[Result1]
Cursor based Data retrieval
RESPONSE
[Result1]
Free resources; Release connection
Establish connection To the DBMS (Resource reservation)
RESPONSE
[Result2]
Cursor based Data retrieval
RESPONSE
[Result2]
Free resources; Release connection

Figure 2-8 Activation of CGI processes, (Hadjiefthymiades et al, 1997)

Figure 2-8 provides an overview of the activation of CGI process (Hadjiefthymiades et al, 1997). Client requests are transmitted to the WWW server (HTTPd) using the HyperText Transfer Protocol in conjunction with the URL encoding scheme. The two requests shown in figure 2-8 have direct connection to the same script. Upon reception of the first request, the server spawns the first instance of the designated CGI script. While the latter performs the required processing (i.e. database access) a new request for the same script arrives at the server. Despite the existence of a process instance, a new one is forked independently. As soon as both instances complete the required processing and pass their results to the WWW server they are terminated.

The establishment of connections to the database management system causes the reservation of resources (memory, processes or threads, etc) and requires a significant amount of time for its completion. When multiple instances of the CGI script are simultaneously active, this resource consumption becomes noteworthy and response times increase (Hadjiefthymiades et al, 1997).

In addition to the problem of the CGI mentioned above, most gateway programs are specific, application dependent and written in languages such as C, C++ or Visual Basic. There are often the cases that new gateway programs have to be developed for new applications (Xia et al, 1999).

2.4.2.2 Java based technique

Due to the problems associated with using CGI for implementing online databases, the focus has been shifted towards using Java programming language. Java was originally proposed by Sun Microsystems (JavaSun, accessed March 2002). It has been promoted as a programming language, which allows users to develop applications (applets) which can be embedded into web pages. It is a robust, object-oriented, high level language with full programming features. In addition to this, Java gained its popularity due to its platform-independent characteristic, making it an efficient language for web programming. Java is used to develop thin clients which are small applets that are downloaded from the Internet and run locally on the user's computer (via the user's web browsers) (Beck et al, 1998). These clients are *thin* in that they delegate most of the work to other applications and databases running on servers. The applet's main purpose is to generate the user interface.
Java however cannot directly communicate with the database. In order to facilitate the development of web-based database applications using Java, a standard API, JDBC, has been defined (Xia et al, 1999). JDBC stands for Java Database Connectivity. The first version of JDBC was released in the summer of 1996. It was Sun's attempt to provide a vendor-independent interface to any relational database system (Hunt, 1998). JDBC is a Java API for executing SQL statements, consisting

of a set of classes and interfaces written in Java. Through JDBC, it is possible to establish a connection with a database, send SQL statement and receive results. The combination of Java and JDBC allows information held in databases to be published on the web easily and quickly via an applet. It also provides a bridge that supports the Open DataBase Connectivity (ODBC) standard. ODBC is a database access standard developed by Microsoft that provides a common set of application interfaces (API) calls to manipulate databases. Application developers can write applications that make ODBC calls and will work with many databases, instead of writing programs specifically for a particular database using its native database APIs (Huang et al, 2001).

To use JDBC with a particular database management system (DBMS), a JDBC driver is required to mediate between JDBC and the database. JDBC drivers are a group of Java classes which act as the interfaces between the JDBC and the database. The JDBC passes the programmer's SQL to the database via the driver (Hunt, 1998). There are four different types of JDBC drivers available:

1. *JDBC-ODBC bridge driver*
2. *Native-API partly Java driver*
3. *Network-protocol, pure Java driver*
4. *Native-protocol pure Java driver*

Sood (1998) has carried out a research into the performance differences between these drivers in terms of connection, result-retrieval, stored-procedure execute, transaction-execute and concurrent-query execute time. The experiments have proved that type 3 drivers work well for Internet based applications with a large concurrent users. It is usually suitable for applications that require access to multiple databases, have stringent performance requirements, and need security features. Type 4 drivers were found to have the largest footprint and large download times, which is a significant drawback for Internet environment. In terms of performance they found to be faster than type 3.

The JDBC API supports both two-tier and three-tier models for database access (Guan et al, 1998). In a two-tier model, the applet talks directly to the data source, whereas in the three-tier, the middle-tier processes the request from the client, and then sends the SQL to the database. The retrieved data is also sent to the middle tier, before it is passed to the client. In the three-tier model, the middle-tier enables the control over the data access. It also simplifies the deployment of applications and provides performance advantages (White et al, 1999). Figure 2-9 shows the structure of a three-tier model.

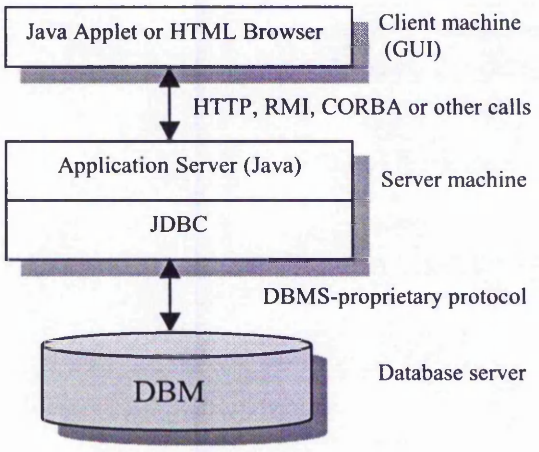

Java Applet or HTML Browser
Client machine (GUI)
HTTP, RMI, CORBA or other calls
Application Server (Java)
JDBC
Server machine
DBMS-proprietary protocol
DBM
Database server

Figure 2-9 JDBC three-tier model, (White et al, 1999)

2.5 Concluding remarks

This chapter has presented a review of the fields that are relevant to subject of this research project. In the first section, a few examples of research undertaken in the area of application of Internet in engineering and manufacture were presented. A quick glance at these examples shows how far Internet has penetrated into the engineering and manufacturing field. Different techniques used for the Internet implementation were also mentioned. There are various methods currently available, each having their own strength and weaknesses. In many cases, there might not be a single ideal approach for that particular problem, and many different methods could equally produce the same desired result. The literature review revealed that no significant work has been carried out in implementing the Internet in the area of gear design and manufacturing. There is a need to explore the potential of Internet in this field.

A brief description of application of AI, mainly ANN and GA, to gear design was given. An overview of a gear design optimisation software was presented that uses GA to search for the best possible solution. The problem with GA is that it is computationally expensive and time consuming. This would be a disadvantage in an Internet environment, when speed is a crucial factor. In this research, attempt is made to integrate ANN into the architecture to improve the speed of the execution.

The last section focused on the area of online electronic catalogue for engineering design application. A brief overview of the current online electronic catalogue was presented. Catalogues have been in the market for some time now and the tendency is to make these available over the Internet. However, many of these electronic catalogues, whether online or offline, are just replications of their paper catalogue. Thus to use a catalogue effectively, the user requires to have a prior knowledge of the product. It was also indicated that most of these sites only present a static data. Some provide the user with a few drawings in DXF format, such as Quality Transmission Components (QTC, accessed March 2002). If it is required to provide DXF for all the possible dimensions, it will take up a lot of space on the server, specially if the data is large. Great deal of productive time would also be wasted to create these individual drawings. It would be of great importance to create these dynamically.

The electronic design catalogues act as expert systems that help the designers in selecting some parts by retrieving the required information from the databases and doing the necessary calculation. A variety of techniques are being used to develop such catalogues in order to make possible the easy use and re-use of both the part and the experience associated with the use and manufacturing of that part. Although this kind of offline electronic catalogue is more efficient than the traditional printed catalogues and saves time in the design process, the information given to the user might still be obsolete. The optimum solution is to integrate this into Internet so that the remote user could have access to the up-to-date information at any time. This will also eliminate the needs for designing different application interfaces across different platforms.

Chapter 3

Internet Approach for Gear Design Optimisation

3.1 Introduction

Gears are a means of changing the rate of rotation of a machinery shaft. They can also change the direction of the axis of rotation and can change rotary motion to linear motion. They are one of the earliest inventions of the mankind and still one of the most important parts of mechanical system. Gear design to professional standards, e.g. ISO (International Standard Organisation), AGMA (American Gear Manufacturer Association) and British Standards is a complicated task, since many factors need to be taken into considerations. Thus making gear design optimisation is very difficult. Optimisation of gear design is an important issue for improving the performance of the machines or systems. It is very beneficial for the engineers, since it encapsulates the expert knowledge that an experienced designer has gained during years of designing.

This chapter describes the development of an Internet-based system for geographically dispersed teams to collaborate over the Internet for the purpose of integration in design and manufacture. As one of the key issues of the system, a CGI-based method to remotely execute a large size software package over the Internet has been developed. It combines the techniques of CGI, HTML, JavaScript and Java in a multi-user environment. A gear design optimisation package was chosen as the vehicle for the development of the method.

In this chapter, a review of the gear design optimisation software is given first, followed by a detailed description of the methods used to implement this software over the Internet. In the last section, problems that could arise in a multi-user environment are discussed and a method is proposed to overcome these problems.

3.2 Overview of the original optimisation software package

The gear optimisation software package (Su et al, 2000) used as an application area for this project has been developed by the Mechanical Design and Concurrent Engineering Research Group at The Nottingham Trent University. The package consists of three parts:

- a graphical user interface
- a genetic algorithm program
- a numerical analysis program for gear strength calculation to BS 436.

The Graphical User Interface (GUI) was developed using Visual Basic and the other two were written in C++. The GUI is used for design data input, setting-up optimisation specifications (goals, weight factors, population size and number of tests) and display of results. The numerical analysis program is invoked by the GA program in the optimisation process to calculate the tooth strength. All the three parts are fully integrated into a single software environment.

The package is used to optimise the design of external spur and helical gears with involute tooth profile. Up to nine gear design parameters can be optimised, including tooth facewidth, module, pressure angle, helix angle, rack tip radius, addendum coefficients, pinion and wheel addendum modification (tooth profile shift) coefficients, and number of pinion teeth. As was mentioned earlier, the optimisation process performs genetic algorithm. Genetic algorithm is known to be computationally expensive. One of the main factors which affects the execution time is the population. Population needs to be of adequate size to ensure that the search area is comprehensively covered. The optimisation process may be time consuming depending on the size of the population defined, the number of tests and the number of parameters selected.

Figure 3-1 shows the structure of the optimisation software in a simplified form. The user is required to input via the GUI the necessary information including design specification (power to be transmitted, speed ratio, etc.) and GA parameters (population size, number of tests, etc.). These values are then written into two input files, '*Genome*' and '*OptGear*'. The former contains the information specific to the optimisation process, i.e. what parameters needs to be optimised and also the GA setting, such as number of generations, number of tests and the fitness functions that need to be processed. The latter file stores the basic configuration of gear design provided by the user, including parameters that define the geometry, material and also the type of the application. An example of these two files are given in appendix A.

There is no data flow between the processing and the visual interface. The processing element is simply invoked by Visual Basic once the input files have been created. The optimisation program will first read the data from the input files and starts the genetic algorithm. After the execution is completed, the results are written into output files which are then displayed to the user.

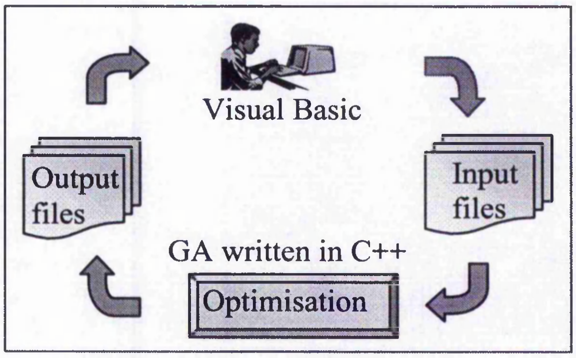


Figure 3-1 Structure of the original gear design optimisation software

Upon completion, the package provides the information of the optimised design as follows:

- The performance of the optimisation process can be displayed in graphical form. The search paths for the goals are provided to allow the user to ensure that the solution has repeatability.

- The final design is displayed giving its major parameters and its performance, including the improvements on the initial design and the stresses acting upon the design.

3.3 Internet solution

One of the techniques required for developing the Internet-based system is how to remotely execute a computationally expensive software package over the Internet. A particular site within the collaborative team may have a relatively large-size or computationally expensive software package owned by the site only and other partners may need to access it. The problem is that the software package is standalone. The Internet environment was not taken into consideration during the design and development of the package. Now the challenge lies in how to convert the system into web-based counterparts. Methods have to be developed to transform with minimum efforts such a standalone system into reusable web-based ones.

Three techniques are described here for developing a method to run a standalone software over the Internet.

- Download the program from the host web site to the client's web site. Then the program is executed in the client's site entirely. In this case the web is only used as an information retrieval system. This is a relatively easy method, but there are several issues to concern: the host would lose control over the program, the client must obtain permission to have the program, the program must be platform-independent, and, for a large size program, there may be problems of download time and the client's computer space and processor speed.

- Recode the system completely in an Internet programming language such as Java Applet. The converted systems are accessible on the Internet/Intranet through the web browsers. But there are a number of limitations associated with this approach. First recoding is time consuming. It might not be feasible in terms of cost and time. Second, it would be difficult to fully understand the logic of the existing system, specially after the original developers have left the project. A slight mistake could prove to be very costly. In addition to this, Java implementation of neumerical computation is known to be slower than the C/C++ implementation. Therefore, by converting it into Java, the execution time will be increased, since the optimisation program contains considerable amount of calculations.

- Encapsulate the existing systems so that they can be made accessible and reusable on the Internet through the web browsers. In other words, let the program reside within the host site and the clients from their sites access the program via a server side programming language. This method can overcome the problems of the above two, but more development work is required compared to the first method.

The main concern of implementing the design software in Internet is the ability of the client to run the software from any machine, whether it is UNIX, Windows, etc, from any part of the world. This is one of the fundamental criteria for the rapid success of the Internet. The optimisation program (i.e. GA) is written in C++ and compiled in windows environment. If the user is working on the UNIX machine, downloading this software would not be compatible with its environment and it cannot be executed.

The first technique is definitely not recommended, due to the reasons mentioned. The second method is also rejected, since it is very time consuming and costly. Moreover the large size of the applet would slow down the downloading time, which would be a disadvantage. To meet the requirements of the Internet-based gear design system, the third method is adapted.

This method is based on distributed computing. In the distributed computing model, the user interface runs locally on the client's computer and is used to call various software components residing on a remote computer. Distribution across multiple computers allows more flexibility, more efficient resource allocation/sharing, more modularity, and more computing power. The Internet is the ideal medium for distributed computing. Such application allows a user to have the processing power and storage capacity of any resource on the Internet (Rou, 1997).

3.4 Fat or thin clients?

It is very essential to determine which tasks should be implemented on the server and which on the client. Generally speaking, there are two typical combinations: '*Fat clients + Thin servers*' and '*Thin clients + Fat servers*' (Huang et al, 2001). As the names imply, a '*Fat*' component is responsible for the majority of the computation, and a '*Thin*' component is doing little computation. For example, a simple HTML web page as a client may not be involved in computation at all. The web browser is only responsible for rendering HTML to display the inputs to, and outputs from, the server component. In contrast, a fat client maybe responsible for all the computaion tasks and no servers are involved at all.

Client side processing reduces the amount of data and traffic going to the web server, which improves response time for the user and reduces the bandwidth and processing requirements for the web server. Because of the daily heavy burden and working load on the web, taking account of these aspects is needed in developing a web-based support system. Client side processing gives CAD designer real time manipulaiton of part in progress, while at the same time relying on remote, server-side processing for less frequent, but far more computationally-intensive task of process planning. It has been known well that a dynamic balance between server-side and client-side computing, and local applications should be reached.

For developing the web-based gear design optimisation, the following measures were carefully planned:

- Using client-side script, i.e. JavaScript for user input validation. This would reduce the interaction and the number of the times that communication is made with the server.
- Using Java applet for displaying the progress of the program execution and also for displaying the results graphically. This also makes the site more dynamic and user friendly.
- Locate the executable GA optimisation on the server. Processes that need access to data located on the server which perform some calculations, or tasks that require high performance from the server machine, belong to the server.

This will also maintain the copyright and makes the system platform-independent.

3.5 Structure of the proposed method

To accomplish the web-based gear design optimisation, a system based on a combination of HTML, Common Gateway Interface (CGI), JavaScript and Java applets were used. Figure 3-2 shows the structure of the proposed system. The HTML, JavaScript and Java applet are located and interpreted locally, i.e. on the client's machine, whereas the CGI and the optimisation program are executed remotely, i.e. on the host server.

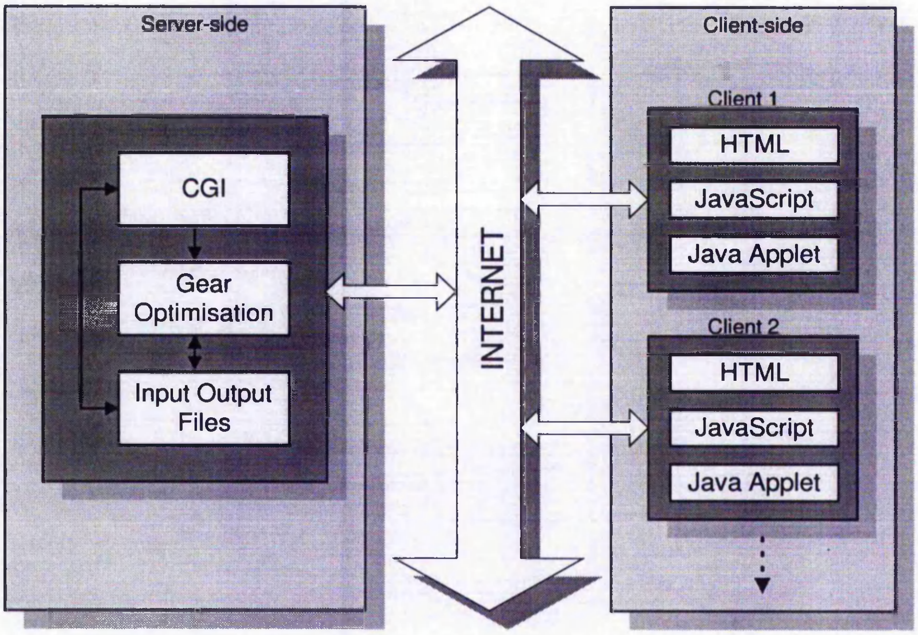


Figure 3-2 Structure of the proposed web-based gear design optimisation system

As discussed earlier, the user Interface of the original software has been created using Visual Basic (VB). Visual Basic is an object-oriented language which is used to create graphical windows-based application. Typically graphical programs created on one platform may be completely incompatible with other systems. Because VB is platform dependent, it cannot be simply used for the Internet application. Therefore the user Interface needs to be created using an alternative language, i.e. HTML, which is platform independent and the codes are interpreted locally, i.e. on the client's machine.

3.5.1 Designing the Web site

As mentioned previously, HTML is used to develop the graphical user interface (GUI). HTML stands for HyperText Markup Language. It is the document format used on the Web and is one of the reasons why the web has become so popular. The documents are hypermedia, meaning that they are connected through links. By clicking a link, the user can move from one document to another. It is easy to make documents in the HTML format and they can be displayed on just about any platform.

HTML has a feature called *form* which makes it possible to obtain information from the user. HTML *form* elements range from simple text input boxes that take a single line of input, to radio buttons and drop down menus. Each form element is given a name, and the CGI scrip that processes the form is presented with several *NAME=VALUE* strings that contain the data that the user has entered.

Once the user inserted all the required information and pressed the submit button, all the data within the form will be transferred to the CGI program located on the server. The CGI program can then perform the necessary calculation on the data and send the results back to the user.

In the development of the GUI, special consideration has been given in the design process. It is important to know who the clients are and what is their main purpose for visiting this site. The user of the online gear optimisation could be a customer, a designer, a developer, a tester, a vendor, a manufacturer or a sales representative. All of these working in an engineering environment. Most engineers, do not look to web sites for entertainment. They want effective information delivery, and “they do not have time for Java-powered dancing balloons” (Braham, 1997).

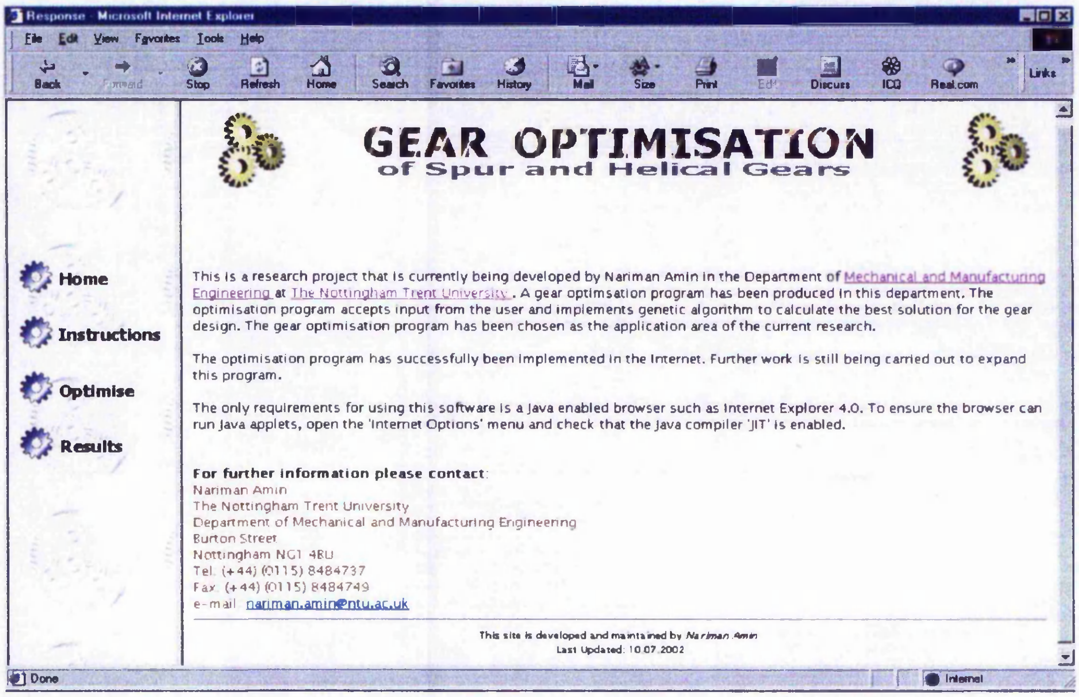
Attempt was made to keep the size of the documents as low as possible. Limited number of graphics were used and the size of the images in pixels were specified within the image tags, as shown below:

```
<IMG SRC="gear.gif" WIDTH="200" HEIGHT="300">
```

Using this method, the browser will leave space for the image, quickly fills in all the remaining text, and then goes back to fill in the graphics.

Background image also contributes to the download time. A combination of solid colours and background images were used for the creation of the online gear design optimisation web site. Solid colours are defined using *<BGCOLOR>* tag and they have no download cost. For background graphics, a small gear image was used. The browser will automatically replicate it to cover the whole screen.

Figure 3-3 shows the screen shot of the first page of the online gear design optimisation software. As can be seen, vertical frame has been used in the construction of these web pages. The presence of the frame, makes the selection of the menu easier for the user, and the user can easily move through the available pages.


Response - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Size Print Edit Discuss ICQ Real.com Links
GEAR OPTIMISATION
of Spur and Helical Gears
Home
Instructions
Optimise
Results
Done
Internet

This is a research project that is currently being developed by Nariman Amin in the Department of Mechanical and Manufacturing Engineering at The Nottingham Trent University . A gear optimsation program has been produced in this department. The optimisation program accepts input from the user and implements genetic algorithm to calculate the best solution for the gear design. The gear optimisation program has been chosen as the application area of the current research.

The optimisation program has successfully been implemented in the Internet. Further work is still being carried out to expand this program.

The only requirements for using this software is a Java enabled browser such as Internet Explorer 4.0. To ensure the browser can run Java applets, open the 'Internet Options' menu and check that the Java compiler 'JIT' is enabled.

For further information please contact:
Nariman Amin
The Nottingham Trent University
Department of Mechanical and Manufacturing Engineering
Burton Street
Nottingham NG1 4BU
Tel: (+44) (0115) 8484737
Fax: (+44) (0115) 8484749
e-mail: nariman.amin@ntu.ac.uk

This site is developed and maintained by *Nariman Amin*
Last Updated: 10.07.2002

Figure 3-3 The first page of the gear design optimisation software

3.5.2 JavaScript

HTML documents are static, plain-text files and their function is totally informative. They do not have the ability to perform any calculations or validations on the data submitted to the server. To make the web page more dynamic and user interactive, JavaScript is used in this project.

JavaScript is a cross-platform, object-based scripting language for client and server application. JavaScript supports a runtime system based on a small number of data types representing numeric, Boolean, and string capabilities. It also supports functions without any special declarative requirements. Functions can be properties of objects, executing as loosely typed methods (Friesenhahn, 1998). It is a powerful language that allows the documents to interact with the browser and the user in a number of ways. It makes creating intelligent dynamic HTML pages possible. JavaScript is written as text which is embedded in the HTML document and is interpreted and run by the web browser whenever the user retrieves the web page and it does not need to be compiled into a program.

The function of JavaScript in online gear design optimisation is to validate the form input before passing it to the server. This is beneficial because:

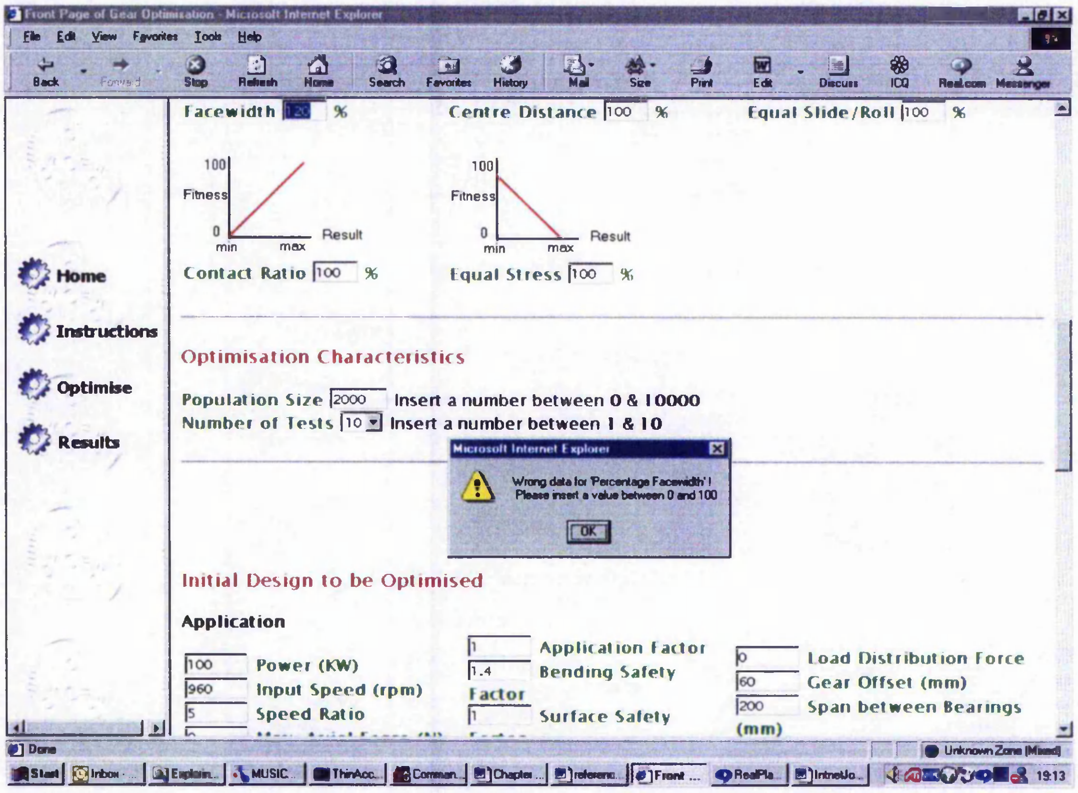
- It reduces load on the server. JavaScript processes the form locally without contacting the server, which reduces the time dramatically. This means the server will receive *clean* data since wrong data have been filtered out on client's machine.
- It reduces delays in case of user error. Validation otherwise has to be performed on the server, so data must travel from client to server, be processed and then returned to client for valid input.

All the data are checked using *onClick* event handler on the *Submit* button. Once the user presses on the *Submit* button, before passing the data to server, JavaScript would check if any of the data is wrong, i.e. out of the permitted range. If there is an error, then JavaScript would prevent transferring the data to the server and will display a message box informing the user what the error is. If the data is correct, then the values are passed onto the CGI program located on the server. An example of JavaScript that checks the value of the population size is shown below:

```
<script language = Javascript>
function validate()
{
   if (document.forms[0].population.value<0  ||
                        document.forms[0].population.value>10000  ||
                        (isaNumber(document.forms[0].population.value)==false))
   {
```

```
                document.forms[0].population.focus( );
                document.forms[0].population.select( );
                alert("Wrong data for 'Population Size' !\n Please insert a value between 0
                                                                        and 10000");
                return false;
        }
    }
    </script>
```

Figure 3-4 shows the screen dump of the situation when JavaScript has detected an error in the input. Value of 120 has been entered for percentage Facewidth and JavaScript has detected this, highlighted the relevant text box and displayed the error message. The optimisation cannot proceed unless this error is fixed.


Front Page of Gear Optimisation - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Size Print Edit Discuss ICQ Real.com Messenger
Home
Instructions
Optimise
Results
100
Fitness
0
Result
min
max
Done
Unknown Zone (Mixed)
Start Inbox Explain... MUSIC... ThinAcc... Comman... Chapter... referenc... Front ... RealPla... IntnetJo...
19:13

Facewidth 120 % Centre Distance 100 % Equal Slide/Roll 100 %

Contact Ratio 100 % Equal Stress 100 %

Optimisation Characteristics

Population Size 2000 Insert a number between 0 & 10000
Number of Tests 10 Insert a number between 1 & 10

Microsoft Internet Explorer

Wrong data for 'Percentage Facewidth' !
Please insert a value between 0 and 100

OK

Initial Design to be Optimised

Application

100 Power (KW)
960 Input Speed (rpm)
5 Speed Ratio
1 Application Factor
1.4 Bending Safety Factor
1 Surface Safety
0 Load Distribution Force
60 Gear Offset (mm)
200 Span between Bearings (mm)

Figure 3-4 JavaScript message box informing an error in the input.

3.5.3 Common Gateway Interface (CGI)

The Common Gateway Interface is a standard for interfacing external applications with information servers, such as HTTP or Web servers. A plain HTML document that the Web daemon retrieves is static, which means it exists in a constant state: a text file that does not change. A CGI program, on the other hand, is executed in real time, so that it can output a dynamic information.

CGI defines a protocol for communication between the server and the program (Adida, 1997) :

- The web server receives input from the browser request.
- It forks a new process in which it sets environment variables corresponding to connection information such as browser data, platform and presence of proxy.
- It runs the CGI program in that process, feeding it the HTML form data through standard input.

The server executes the CGI script program when the browser passes the program file's URL to the server as a result of a user action. A CGI script's URL contains the path and name of the script file on the server or an alias that the server can translate. The server uses the path specification to recognise that the URL is a CGI script specification and the file name extension to determine what kind of CGI script to run (Stenvens, 1997).

The standard output of the program is returned directly to the server, which usually passes it straight out to the web browser. This means the CGI program could be written in any programming language as long as it follows the convension of getting its input from the environment variables and sending its output to *standard out*. This could be Perl, C/C++, Java, etc. CGI scripts are powerful enough to execute external programs. In this project, the C++ language was used to write the CGI program, due to the fact that the author was familiar with this programming language. Moreover, it is more secure compare to script languages such as Perl, since the compiled version will be located on the server.

CGI is reasonably secure, specially since permission can be extremely strict in the process used to run the program and furthermore, this process is separate from those of the web server. This separation allows the web server to have any level of acces to the file system, while limiting the permissions on CGI programs.

In this research, a user interface is created for the CGI application using input forms. Forms are created in HTML and are displayed within the client's browser. A user would fill out the individual fields in the form and press a Submit button. All fields would then be sent via the CGI protocol to the server.

Figure 3-5 shows how the information contained within these fields would be sent via the CGI protocol to the application referenced in the form residing on the server.

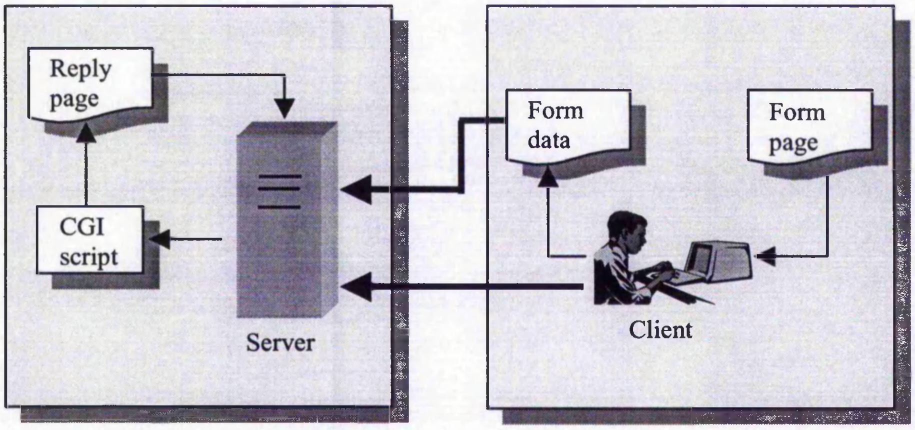


Figure 3-5 Data flow during the execution of a CGI script

The CGI specification describes how user-data is to be passed to the server-side program. There are two different methods of sending the data to the server: *GET* and *POST*. The GET method sends its data as command line input, i.e. as one long string assigned to an environment variable, whereas the POST method sends all data values as one line of text to a standard input device which must be read within the program (Levy, 1998). The amount of data that can be retrieved from a GET is considerably smaller than that from a POST. Because in the online design optimisation software package, the number of data that needs to be obtained from the user is considerably large, the POST mechanism was preferred for transmitting the data.

In this web-based gear design optimisation system, CGI programming is used to:

- Process information regarding the basic configuration of the gear design submitted in a query form by the client.
- Provide an interface with the optimisation program

- Process the output of the optimisation program and transfer the result back to the client.

The CGI program first parses the data and after applying some calculations on the data, it will write the data into the input files. These calculations are done in the original optimisation program on some of the data, therefore they need to be applied here as well to make sure that the input data files produced, are exactly the same as the original optimisation program. CGI program will then call the optimisation program which is a C++ executable file. This program will then start running on the server. This works the same way as it was explained in section 3.2. The optimisation program will read the data from the input files and starts the genetic algorithm. It will take some time, depending on the values given, before the results are out. In this situation, if for any reason the connection is lost between the client and the server, it will not create any problem, since the program is running on the server independent of the client. When the execution of the optimisation program is completed, it will write the results into the output files. The structure of the method is shown in figure 3-2.

A flag has been used to check if the optimisation is completed. When the user clicks the *Result* button, the CGI program checks the value of the flag. If the execution is not completed, it will inform the user, otherwise it will display the results on user's screen.

3.5.4 Java applet

Java is a tool and programming environment that supports the distributed processing for manufacturers via Internet/Intranet. Java is a robust, general-purpose, high level programming language and a powerful software platform. It is portable, object-oriented, distributed and multi-threaded. It enables the creation of standalone applications or applets that run on virtually any computer on the Internet. Java stand-alone applications run directly on the Java platform. They should be delivered and installed on each client machine before it is run from the command line.
Java applets run within a Java-enabled browsers for client-side processing. Java applets are located on a server. They depend on web browsers for their installation (downloading) and execution in the client's machine. The whole process is

automatically accomplished by the web browser after connection to the appropriate web server page. No client configuration is needed and the maintenance and upgrading are automatic from the user's viewpoint.

The purpose of utilising Java applet for the web-based gear design optimisation system is to enhance the user interface. They are used to construct the progress bar and the graph of the results. The progress bar displays how far the execution of the optimisation program has reached. Since optimisation program is computationally expensive and depending on the data inserted by the user, it might take some time before the results are given out, the presence of a method which informs the user about the progress of the program is very beneficial.

Java applets are defined in HTML language using the <applet> tag. When the user loads the page, the applet is downloaded from the server onto the user's machine. Once downloaded, it is executed wholly on the client's machine. Since applets can be downloaded from any site on the World Wide Web and run on user's system, some security issues have been taken into consideration. Some restrictions have been implemented to prevent malicious applets that contain viruses or Trojan horses, which could cause system damage. The restrictions on applets include the following: (Lemay, 1996)

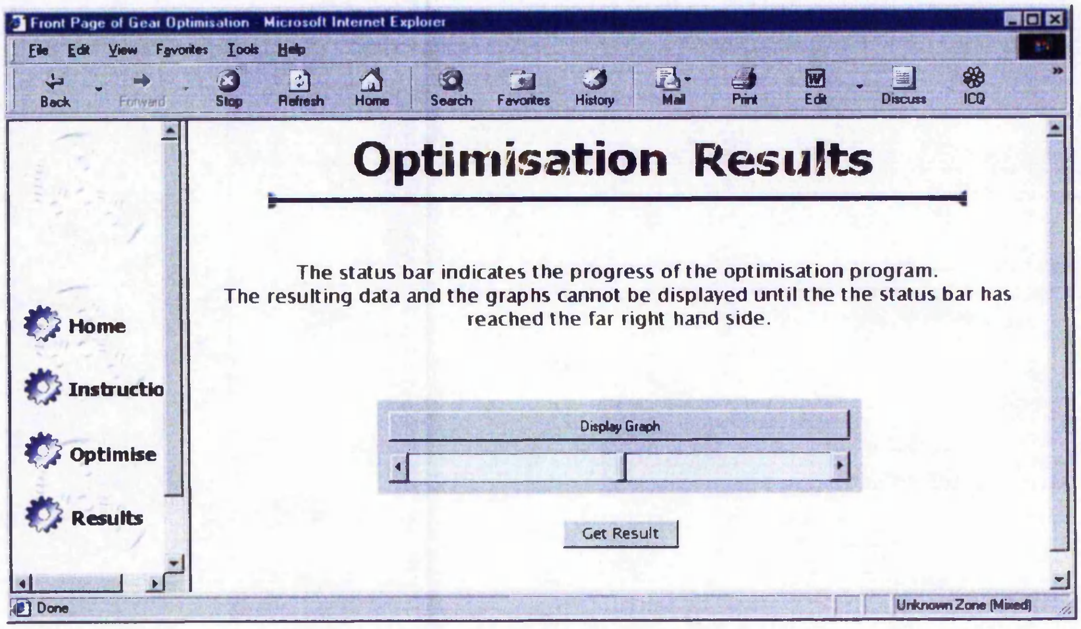
- Applets cannot read or write to the client's file system, which means they cannot delete files or test to see what programs you have installed on the hard drive.
- Applets cannot communicate with any network server other than the one that had originally stored the applet, to prevent the applet from attacking another system from the client's system.
- Applets cannot run any programs on the client's system.
- Applets cannot load programs native to the local platform, including shared libraries such as DLLs.

These restrictions do not limit our process. Applets are needed to communicate with the server and to read the data from a number of files. This is permissible, since the files are located on the server where the applets have been downloaded from.

3.5.4.1 The progress bar

As explained earlier, the time taken to perform the optimisation process varies and is dependent upon the population size and the number of times that the process is to be repeated. The results will only be released when the optimisation process which performs genetic algorithm is completed. In this situation, if for any reason the connection is lost between the client and the server, it will not create any problem, since the program is running on the server independent of the client. A flag has been used to check if the optimisation is completed.

In such an environment, the users know that it will take some time before the results are out. This can however be very inconvenient, since they have to click on the *Result* button repeatedly to check if the optimisation program is completed. It was decided to build a structure, which would check the progress of the execution continuously and would display the progress graphically on user's screen. Java applet has the capability of accomplishing the desired task.


Front Page of Gear Optimisation - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss ICQ
Home
Instructio
Optimise
Results
Display Graph
Get Result
Done
Unknown Zone (Mixed)

Optimisation Results

The status bar indicates the progress of the optimisation program.
The resulting data and the graphs cannot be displayed until the the status bar has reached the far right hand side.

Figure 3-6 Optimisation Result page showing the progress bar applet

A data file is created by the optimisation package on the server. Data within this file indicates the progress of the optimisation in converging on a solution. While the optimisation program is being executed, the data within this file is continuously being updated. The data is utilised along with the number of tests to estimate the time for the process to converge on a solution. They must however be scaled down to the size of the progress bar, before the object is redrawn.

A feature in Java called *thread* has been employed to control the accessing of the data file every second, each time reading and updating data into the progress bar. Anything that runs continuously should run in its own thread. This would help in the reduction of the processing time.

In order to use the thread, the applet must be defined as runnable, by adding *implements Runnable* to the applet class. An instance variable must be defined to hold the applet's thread object. The *start()* method will create a new method and start it running. The actual activities occur in the *run()* method. This is where the data file is read, the necessary calculations to determine the progress of the status bar is done and the status bar is repainted. As was mentioned earlier, the content of the progress bar is updated every second, i.e. the content of the data file is read every second. To make the activity wait for 1 second before accessing the data file another time, the *Thread.sleep()* method is called within the run() method, as shown here:

```
try{ Thread.sleep (1000);  }
catch (InterruptedException e) {}
```

In the *stop()* method, the thread is stopped from executing. The stop() method is activated whenever the user leaves the page. If the user returns to the page the start() method would restart the thread.

If for any reason, the connection is lost with the server or the power is cut off, there would not be any disturbances in the overall functionality of the progress bar. This is due to the fact that the optimisation program is running on the remote server, and the value of the data file that determines the progress of the optimisation is being updated, since it is working independent of the power on the client's machine or the connection between the client and the server. When the connection is established again, the progress bar would jump ahead into the new position. Figure 3-6 shows the screen shot of the progress bar.

3.5.4.2 Data retrieval

All the data files that are produced by the optimisation program are saved on the server. Java applet running on the client's machine needs to access these files whether it is for the progress bar or the graphs. The *URL* class has been used to encapsulate a uniform resource locator. This allows the applet to quickly and easily access the file system on the remote server. The URL class specifies the TCP/IP protocol to use either 'http' or 'ftp', the port number (usually 80 for the web servers), and the exact location of the remote object.

Relative URLs have been used rather than the hard coded URL address to locate the output files on the server. By using the *getDocumentBase()* method of the URL class, the file is searched for within the original location of the web page that contained the applet. This is very useful if for any reason the site needs to be moved, then the Java code will not need to be recompiled.

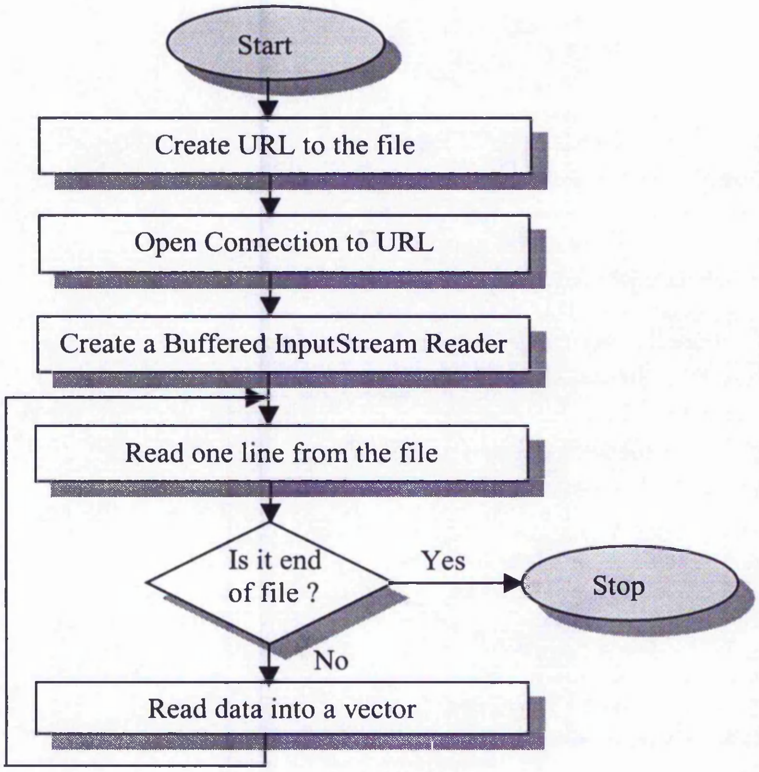


Figure 3-7 Data retrieval process.

Figure 3-7 shows the process of retrieving the data from the files located on the server. First a new instance of the URL is created given the relative path as was described above. Then the connection is made to the specified file. *InputStreamReader* class has been implemented to open a stream for reading data from the individual files. Each stream is opened by a *BufferedReader* so the data can be temporarily stored in a buffer to avoid networking problems.

```
BufferedReader buf = new BufferedReader(new
                              InputstreamReader(con.getInputStream()));
```

where *con* is an instance of the *URLConneciton* class. The *readLine()* method of *BufferedReader* could then be used to pass the string data into a vector. The size and contents of each file before it is read is unknown, consequently vectors are ideally suited to this application because their size is not fixed, and the data is treated as objects. The vector class implements the growing array of objects.

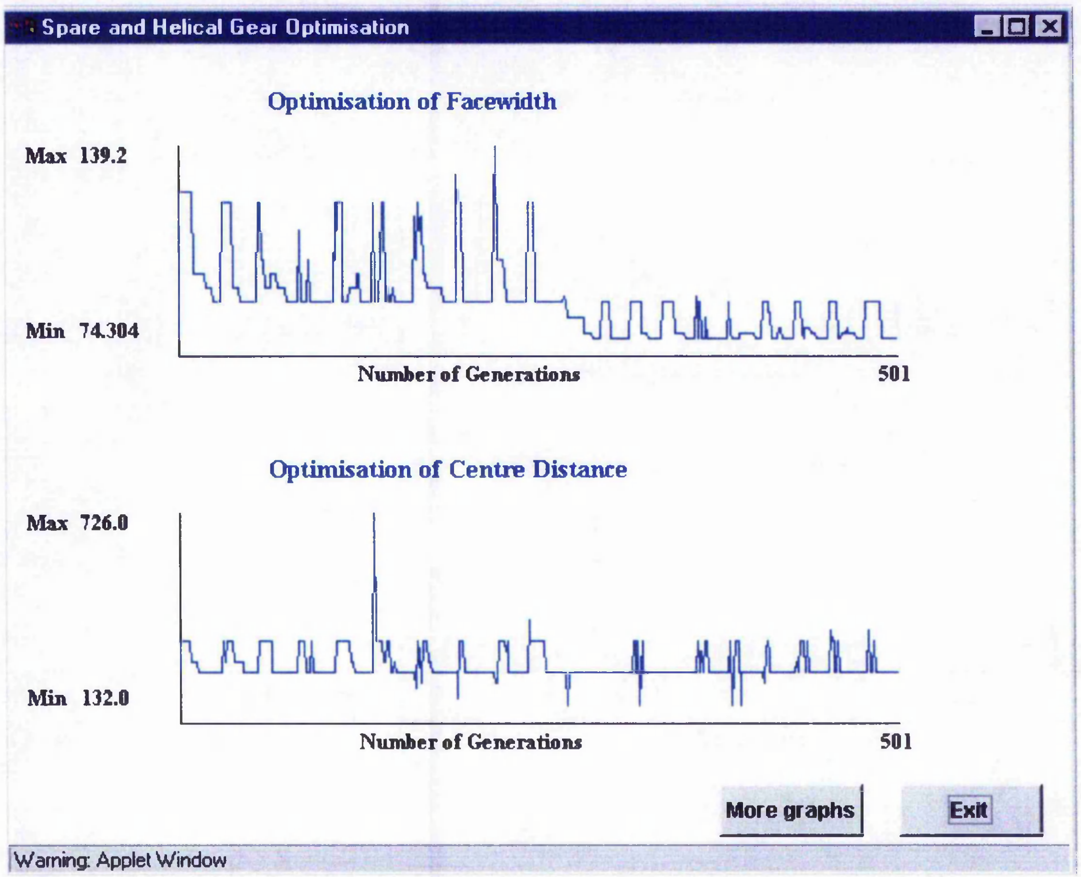


Figure 3-8 Optimisation process performance.

3.5.4.3 Optimisation graph

When the execution of the optimisation program is completed, in addition to accessing the resultant data the user has the option of displaying the graph of the optimisation. The performance of the designs is given in the graphs, indicating the trend of the search and the levels of performance. The result traces also provide a means of evaluating the repeatability of the genetic algorithm to achieve a solution to this design and thus indicates whether the fitness functions and population size are set correctly. Each spike represents the beginning of a test and as the trace levels out convergence is displayed. An example of the graph for *Facewidth* and *Centre Distance* is shown in figure 3-8.

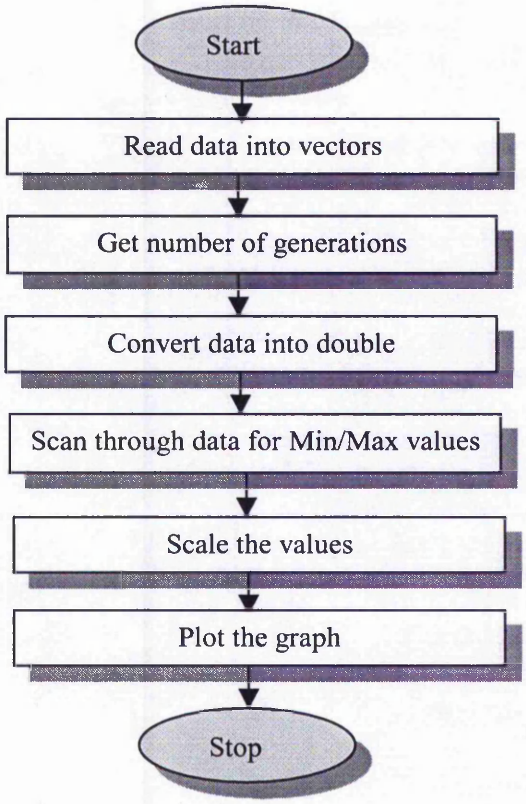


Figure 3-9 Optimisation graph flow diagram

While the optimisation program is being executed, it writes the information about *Centre Distance, Facewidth, Bending Stress Difference* and *Contact Ratio*, for each generation in four different files. The data within these files are used to plot the graphs. For each data in the file, a point is plotted on the graph after being scaled.

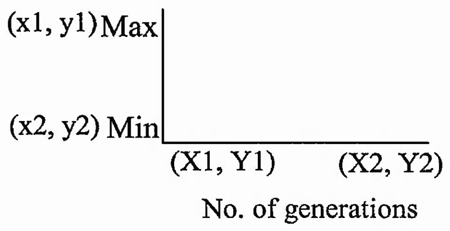
The graphs have been plotted using the *frame* feature of Java. Frames are windows that are independent of an applet and of the browser that contains it. They are separate windows with their own titles, resize handles, close boxes and menu bars (Lemay, 1996). The first step is to read the data from the relevant file and save them into a vector. As was mentioned earlier, vector is an array of object, therefore it needs to be converted into an array of double in order to be used for scaling and plotting the graph. These data are then scanned through and the minimum and maximum values are found. The number of generation is also retrieved from one of the input files. The data is then scaled to be able to be drawn on the screen. Refer to figure 3-9 for the flow diagram.

3.5.4.4 Data scaling

Before the charts can be drawn, all the data must be scaled to the size of the graph within the frame. This is achieved by searching through the data for maximum and minimum values. Once found a suitable scale is calculated and for every generation, a value is scaled against the vertical axis using the following equations:

Scaler = (y2-y1) / (Max-Min)

Xincrement = (X2-X1) / No.of generations

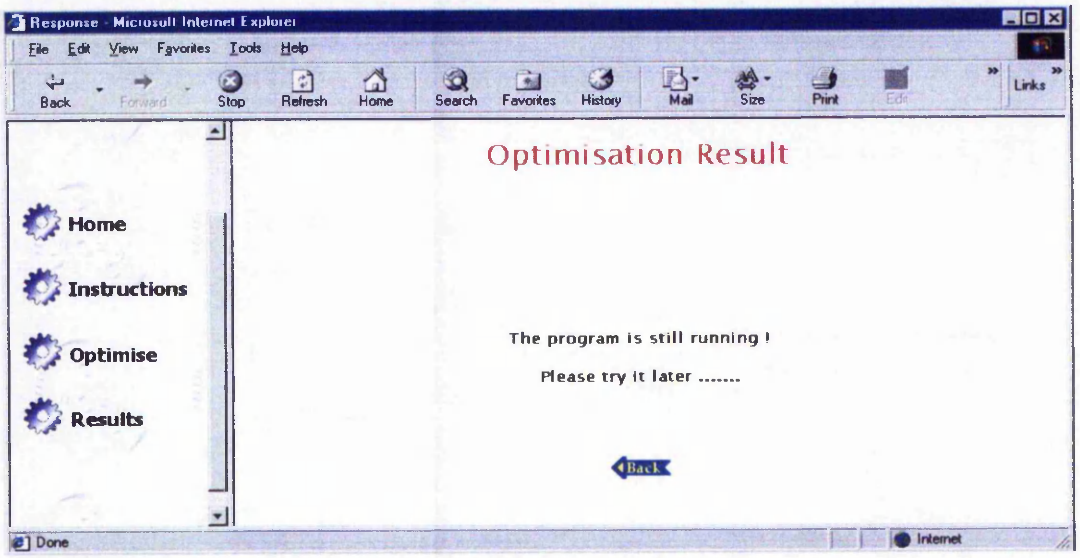


Therefore:

For the number of generations, repeat the following process:

```
    pointX2 = pointX1 + Xincrement
    pointY2 = data * Scaler
    Draw the line (pointX1, pointY1, pointX2, pointY2)
    pointX1 = pointX2
    pointY1 = pointY2
```

The graph is then plotted using the graphics class of the Abstract Windowing Toolkit (AWT).


Response - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Size Print Edit Links
Home
Instructions
Optimise
Results
Back
Done
Internet

Optimisation Result

The program is still running !

Please try it later ....

Figure 3-10 Optimisation Result page indicating that the execution is not complete

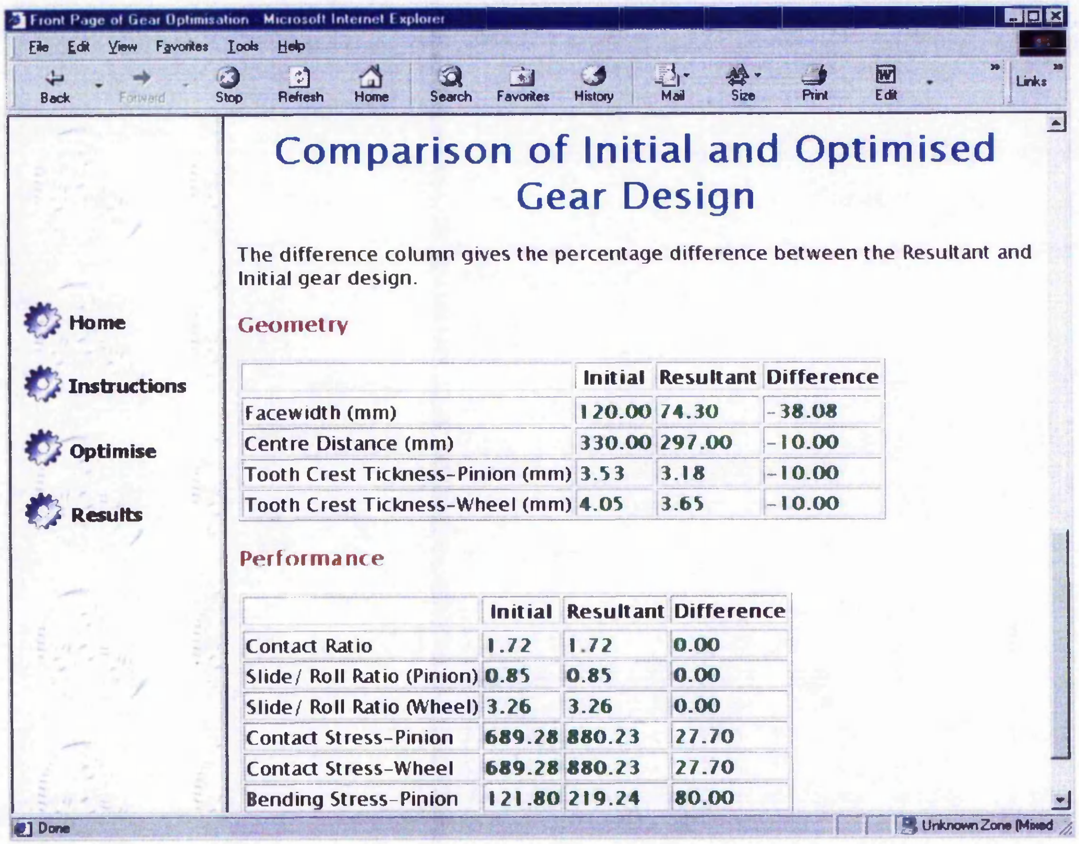
3.5.4.5 Results page

Figure 3-6 showed the *Optimisation Results* page, containing the progress bar applet while the optimisation program is being executed on the server. At this point clicking on *Display Graph* (the applet) button, would not display any graph and the *Get Result* (CGI program) button would inform the user that the optimisation program is still running, as shown in figure 3-10. This is because both applet and the CGI check the value of the flag before displaying any result to the user. The results can only be viewed when the optimisation program is completed, i.e. when the progress bar reaches the far right side.

Once the execution is completed, the results can then be displayed showing the optimised parameters. It also shows a comparison between the original value and the optimised output. Figure 3-11 shows the screen shot of an example of the result page.

3.6 Multi-user environment

The proposed system described in section 3.5 works perfectly as long as we guarantee that only one client will be using the system at a particular time. There is no way this could be the case in an Internet environment. One of the merits of implementing the web-based optimisation software is the ability of multi-users to utilise this software simultaneously. What problem could then arise with the current method?


Front Page of Gear Optimisation - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Size Print Edit
Links
Home
Instructions
Optimise
Results
Done
Unknown Zone (Mixed)

Comparison of Initial and Optimised Gear Design

The difference column gives the percentage difference between the Resultant and Initial gear design.

Geometry

	Initial	Resultant	Difference
Facewidth (mm)	120.00	74.30	-38.08
Centre Distance (mm)	330.00	297.00	-10.00
Tooth Crest Tickness-Pinion (mm)	3.53	3.18	-10.00
Tooth Crest Tickness-Wheel (mm)	4.05	3.65	-10.00

Performance

	Initial	Resultant	Difference
Contact Ratio	1.72	1.72	0.00
Slide/ Roll Ratio (Pinion)	0.85	0.85	0.00
Slide/ Roll Ratio (Wheel)	3.26	3.26	0.00
Contact Stress-Pinion	689.28	880.23	27.70
Contact Stress-Wheel	689.28	880.23	27.70
Bending Stress-Pinion	121.80	219.24	80.00

Figure 3-11 Optimisation results page

The optimisation program interacts with a number of input and output files. The input files contain the initial design of the gear inserted by the client. The output files contain the results of the optimisation. If two clients are using the system simultaneously, the input/output files could be over written and wrong output could be sent to the users.

3.6.1 An example of a problem using the current method

A situation that would create a problem has been shown graphically in figure 3-12. We assume that User 1 makes a connection with the server. The data is sent to the CGI program and the optimisation program starts running. At this moment, the connection with the server might be lost, or since the user knows that it will take some time before the execution is completed, he might decide to come back later to check the results, as indicated in figure 3-12a. The optimisation program is running on the server and it will be completed after a period of time and the results will be written into output files. Meanwhile User 2 makes a connection with the server and starts running the software. The program is completed and the results are written into output files. In another words, the results of the User 1 will be overwritten by the results of User 2, as shown in figure 3-12b. When User 1 returns to check the results, wrong information will be displayed on his screen. This is a crucial point that needs to be dealt with.

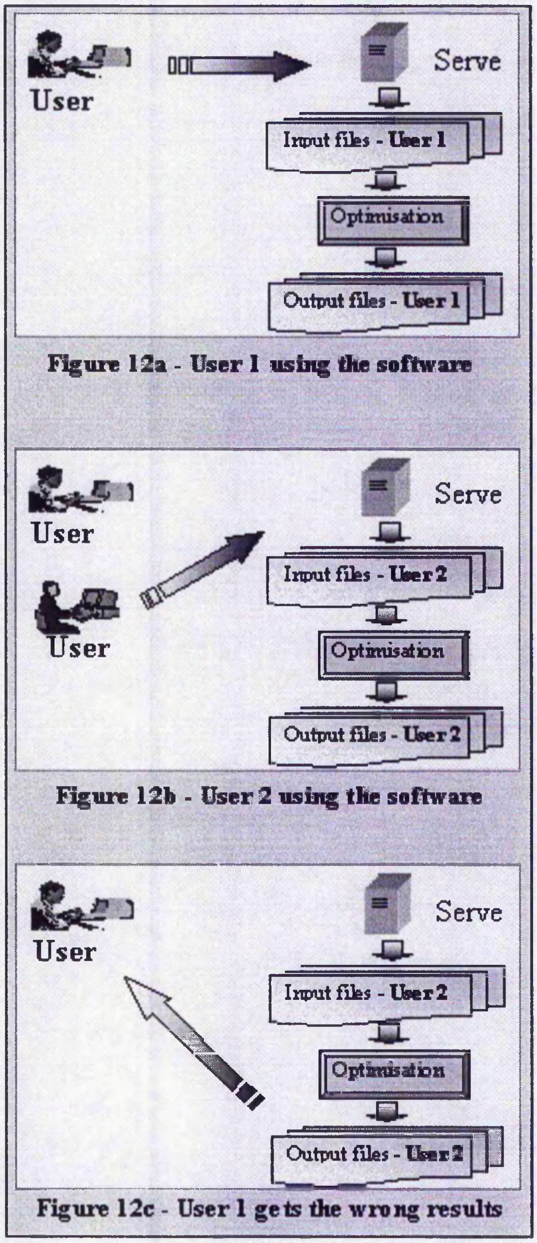


Figure 3-12 An example of a problem in a multi-user situation

3.6.2 The solution for a multi-user environment

It was decided to give each user an identity and a space on the server, so that all the interactions could occur within that allocated area. In this way, there would be no conflicts between different users. The structure of the system is shown in figure 3-13.

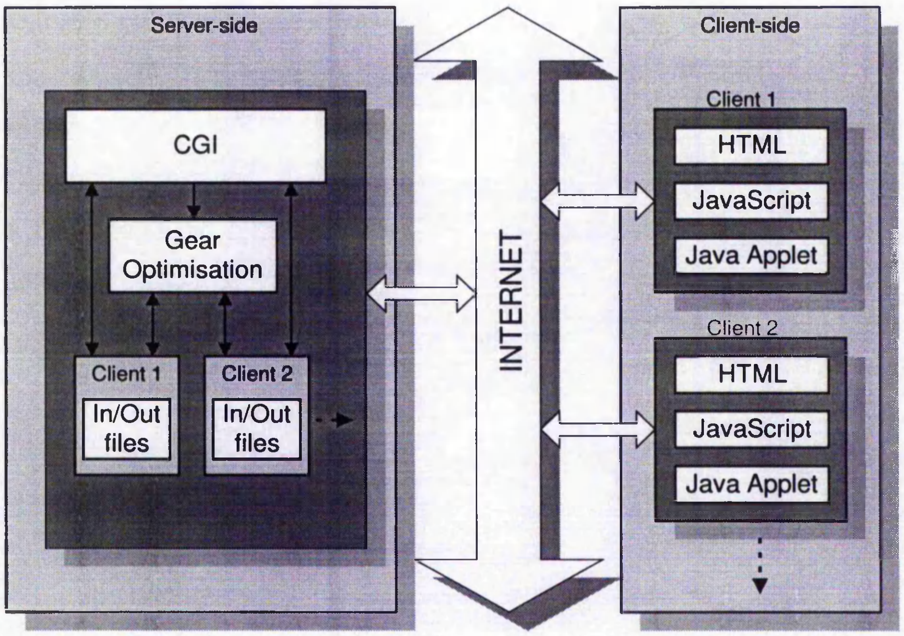


Figure 3-13 Structure of the web-based gear design optimisation system for multi-user environment

The first time the users log in, they are required to input their usernames. If permission is given, then the CGI program will create a directory on the server under the *cgi-bin* directory. The name of this directory would be the same as the username. This is where all the interactions occur for this user.

The next step is to copy all the CGI and HTML files into this directory. The structure was designed in such a way that only one copy of optimisation would be needed. This avoids copying the optimisation program, which is considerably large, into the directory every time a user logs on. However, the path of the directory needs to be passed on to it and also to all other CGI files. This is a vital step, otherwise the optimisation program will not have any information about where it should be reading the data from and where to write the output files.

HTML *form* has a feature called *hidden element* that makes it possible to pass arguments to the CGI program. This characteristic is used to pass the directory name, i.e. username, to the CGI program. The <*BASE*> tag of HTML also defines the current directory where the documents are located. Both these features need to be added to the HTML form dynamically. This is done by the CGI program. It will update the HTML files and adds the hidden element as well as the <BASE> tag with the appropriate path to the HTML. The final step would be to open the file index.html within the current directory. From this point onwards, all the interactions and reading/writing from/to files occur within this location.

When the user submits the data, the CGI program will write the data into input files within this directory and then calls the optimisation program and passes the directory name to it. The optimisation program will read the data from this directory and after the execution is completed, the results are written into output files in that directory. So, when the user presses the *Result* button, the correct data is sent to the user.

In order to prevent taking up the space on the server unnecessarily, the outdated user directories are removed. After the results are displayed on user's machine, the program keeps this directory for the period of 48 hours. Every time the results are displayed, a file called *resultDisplayed.txt* is created within that directory. This file does not contain any data, however the system checks the date of creation of this file, if it is more than 48 hours, this indicates that the directory is outdated and all the files within that directory along with the directory itself will then be deleted. This is done to reduce the space on the server to minimum. The pseudo code is shown below:

```
- Get current date/time
- Get the name of all the user directories
- Ignore the current user directory
- For the rest
    - For every directory
        - Get the date/time of the creation of the file resultDisplayed.txt
        - If more than 48 hours
            - Delete all the files
            - Delete the directory
        - End If
      - End For
  - End
```

Figure 3-14 shows the screen shot of the first page of the Internet-based gear design system. It consists of two main parts: Gear Catalogue and Gear Optimisation. As can be seen, for the gear optimisation, the user is required to enter the username before they can access the software. Gear catalogue will be discussed in detail in chapters 6 and 7.

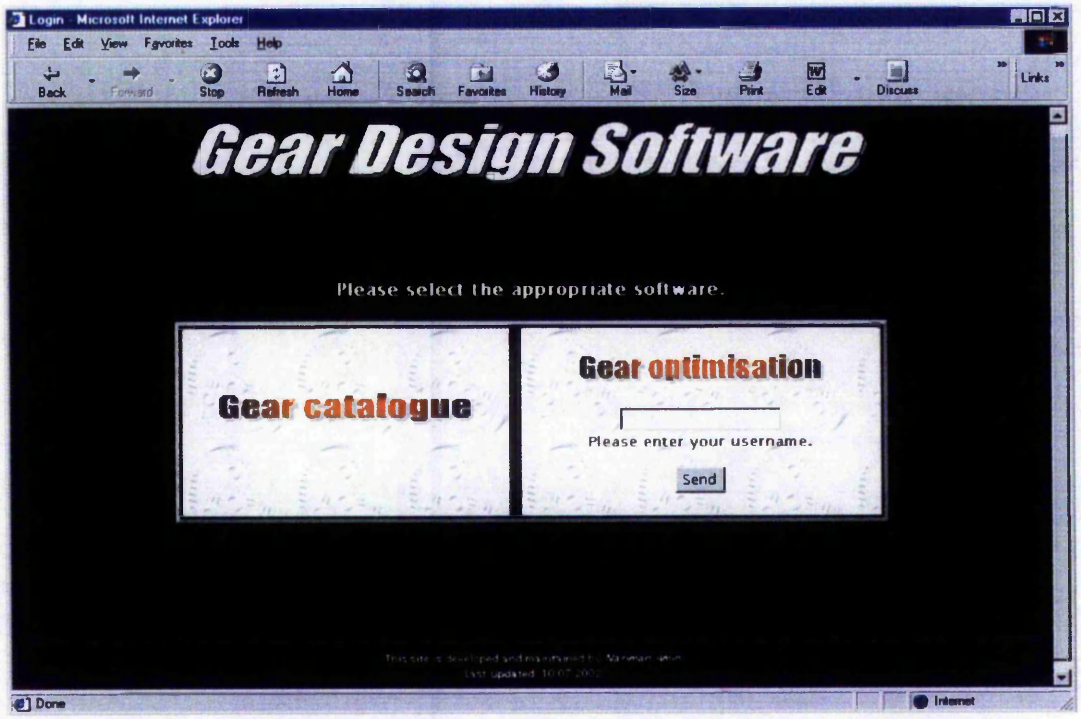

Login - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Size Print Edit Discuss
Links
Gear Design Software
Please select the appropriate software.
Gear catalogue
Gear optimisation
Please enter your username.
Send
Done
Internet

Figure 3-14 Screen shot of the first page of the Internet-based gear design system

The program has been tested using various input data and the results have been compared with the original software running on a single workstation. The results proved to be successful. The program has also been tested using remote stations and from different platforms, all of which have produced successful results.

3.7 Concluding remarks

In this chapter, the approach for implementing the online gear design optimisation process is thoroughly discussed. One of the main issues was how to execute a computationally expensive software package over the Internet. The first requirement of the system is that it must be platform independent, since the remote user might be working on a different operating system. The best solution is to locate the software on the server and the users access it via a server side programming language.

CGI script is used to perform the communication between the client and server. In designing the system, special consideration was taken to determine which tasks belong to the server and which ones belong to the client. JavaScript was introduced into the system to reduce the load on the server. JavaScript is executed locally on users' machine. Its task in this system is to validate the data provided by the user, before they are sent to the server. Java applets are implemented as well, for displaying the results in a graphical format. They are also used for monitoring the execution time of the program using a scrollbar.

The problem of multi-user environment is also addressed in this chapter. This is due to the presence of the input/output files used by the optimisation software. An approach is developed that assigns a space for each user, so that all the interactions occur within this space. In this way, no conflict arises between different users. Once the user retrieves the result, the allocated space is removed to clear the space on the server.

Chapter 4

Integration of Artificial Neural Networks into GA Optimisation

4.1 Introduction

The gear design optimisation described in the previous chapter assists in designing gears without the necessity of being an expert. According to the tests carried out by Wakelam (1998) the optimisation process gives satisfactory results but suffers from a drawback of being slow in finding the solution. This is due to the iterative nature of the genetic algorithm computations. To get a satisfactory result, a large number of population for the GA algorithm is often chosen to ensure that the search area is comprehensively covered, which in terms increases the number of iterations. This leads to a slow moving process of finding the optimised parameters for the gear design.

According to the tests performed by Wakelam (1998), a combination of 2000 for the population size and 10 for the number of tests would give the optimum result. The user is recommended to use this set-up, however, if a quicker result is required, the user has the option of reducing these values. This, however, might degrade the performance. Therefore, there is a trade off between the performance and the processing time. Even though the final output produces the satisfactory result, the time consumed for the program to reach the final solution could be a drawback for the user. This is particularly important in an Internet environment where speed is an essential factor.

The execution time depends on the size of the population defined, the number of tests and the number of parameters selected. A new approach is put forward to address this problem by integrating Artificial Neural Networks (ANN) into the system. To further explain this proposal, a brief description of the optimisation program is first presented.

4.2 Characteristics of the GA optimisation program

In section 2.3.1 and 3.2, a brief description of the structure of the gear design optimisation software package was given. In this section, the structure of the main program of this software package, i.e. GA optimisation will be discussed.

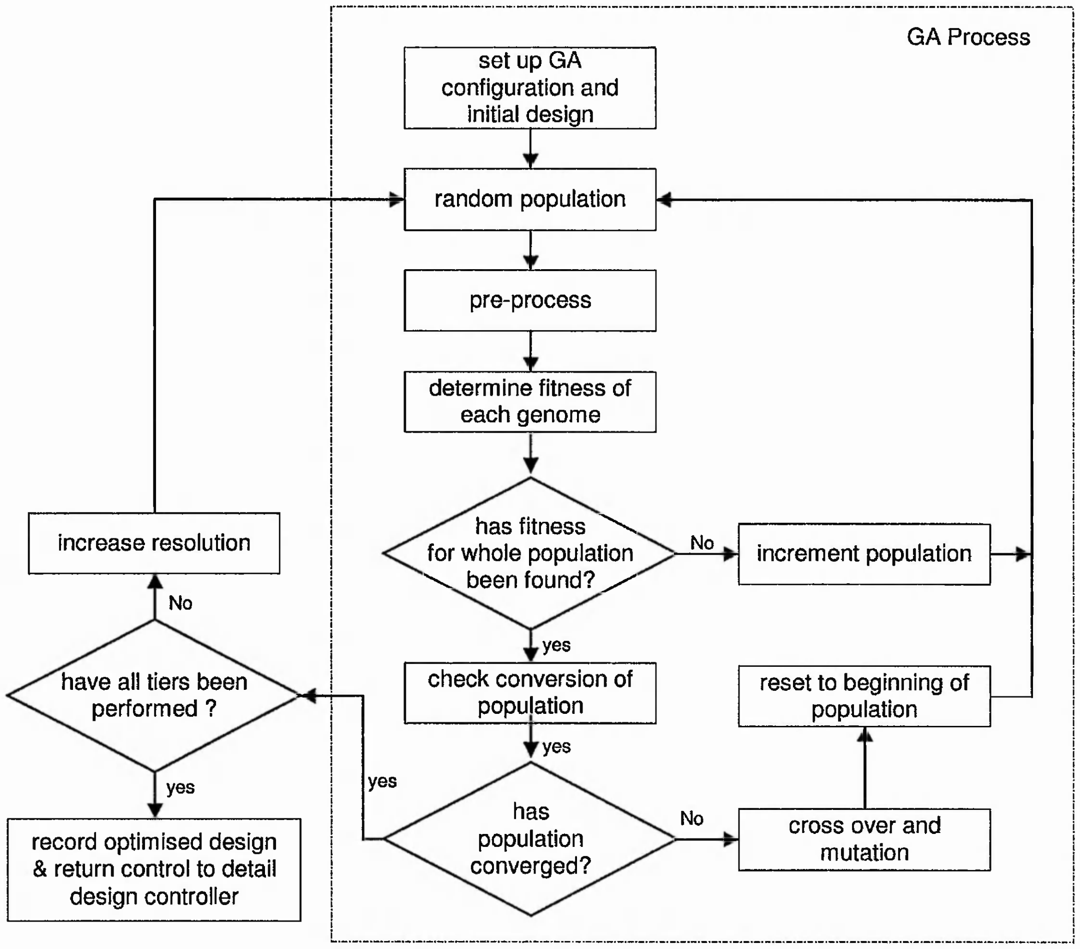

GA Process
set up GA configuration and initial design
random population
pre-process
determine fitness of each genome
has fitness for whole population been found?
No
increment population
yes
check conversion of population
yes
has population converged?
No
cross over and mutation
reset to beginning of population
yes
have all tiers been performed ?
No
increase resolution
yes
record optimised design & return control to detail design controller

Figure 4-1 Schematic of the gear optimisation process, (Wakelam, 1998)

Figure 4-1 illustrates the schematic of the gear optimisation process (Wakelam, 1998). The process commences with the set-up of the population size, convergence parameters, fitness function goals, critical condition limits and the initial design. The population is then filled with random genomes, forming the initial population. The genes within the genome are decoded and the gear geometry adjusted to comply with the new parameters.

Once the performance of all the genomes in the population has been determined, the fitness rating for each genome relative to the population is calculated. The population is checked to determine if it has converged upon a solution. If convergence has not occurred, cross over and mutation of the population is performed to create the new generation. The process is repeated until the population converges on the same solution.

At this point the basic GA process has finished, however as mentioned in section 2.3.1, the GA optimisation program implements a cascade procedure comprising of two tiers, refer to figure 2-5. This means that once the first tier is completed, the process needs to go through the second tier. The resolution of the decoding process during pre-process is increased and the optimisation process repeated. In this case, the initial gear design that formed the base for the search is replaced by the solution obtained from the first tier. Upon completion of the second tier of the optimisation, the resultant solution is taken as the optimised gear design.

4.3 The proposed solution to improve the speed of the execution

It was proposed to build a neural network that would replace the first tier of the cascade procedure of genetic algorithm optimisation. The function of the neural networks would be to estimate the solution of genetic algorithm after the first tier. In other words, the ANN will estimate the output of the first tier, which would then be fed into the second tier as the intermediate warm values, see figure 4-2.

Implementing this proposal is not as straightforward as it seems in figure 4-2. The code of the original optimisation program, written in C++, needed to be thoroughly investigated and every single parameter needed to be scrutinised to filter out those data that contribute significant effects on the output. The purpose of this proposed approach is to find a value as close as possible to the output of tier1. It does not need to be exactly the same, since this would act as a warm input value to tier 2 and not the final output.

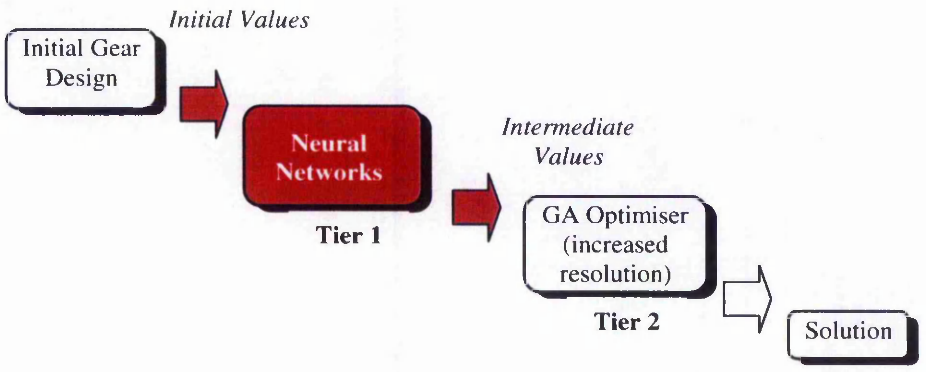


Figure 4-2 Implementing Neural Networks in the cascade procedure of GA optimisation

There are 80 parameters that are fed into the first tier (see appendix A). This is relatively high to be used as input data for ANN. Some of these only have a slight effect on the final solution. Consequently, these parameters need to be singled out and excluded from the input of ANN. This would be very beneficial, since the presence of a large number of input data would increase the number of input patterns drastically, which might not be feasible in terms of time required to train the ANN. The idea is to reduce the data collection process by limiting the number of combinations for the parameters used.

4.4 Implementing artificial neural networks

Artificial neural networks are typically organised in layers. Layers are made up of a number of interconnected nodes which contain an activation function. Patterns are presented to the network via the input layer, which communicates to one or more hidden layers where the actual processing is done via a system of weighted connections. The hidden layers then link to an output layer which contain the final result. Figure 4-3 shows the structure of an ANN system, having two inputs, 1 hidden layer containing three nodes and two outputs.

Most ANNs contain some form of learning rule which modifies the weights of the connections according to the input patterns that it is presented with. In a sense, ANNs learn by example as do their biological counterparts, e.g. a child learns to recognise trees from examples of trees.

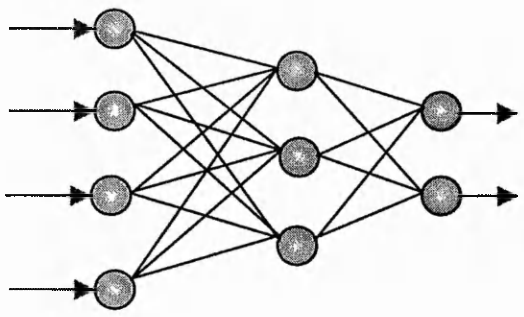


Figure 4-3 A three layer structure of ANN

There are two main characteristics among the ANN algorithms:

- Supervised learning: where the network is presented with a pair of data vectors during the learning phase. The network reduces an error margin between calculated and desired output vectors by adjusting weight values.
- Unsupervised learning: where input space is classified into distinct regions based on regularities and correlations.

4.5 Training by backpropagation of errors

Backpropagation is a multi-layer feed forward network and is probably the most well-known and widely used among the current types of ANN systems available. Backpropagation is an abbreviation for the backwards propagation of errors. It is a supervised learning rule used to adopt the network parameters (weights) based on the error between the target and the actual network output.

To begin with, the network learns a predefined set of input-output example pairs by using a two-phase propagate-adapt cycle. After an input pattern has been applied to the first layer of network unit, it is propagated through each upper layer until an output is generated. This output pattern is then compared to the desired output, and an error signal is computed for each output unit.

The error signals are then transmitted backward from the output layer to each node in the intermediate layer that contributes directly to the output. However, each unit in the intermediate layer receives only a portion of the total error signal, based roughly on the relative contribution the unit made to the original output. This process repeats, layer by layer, until each node in the network has received an error signal that describes its relative contribution to the total error. Based on the error signal received, connection weights are then updated by each unit to cause the network to converge toward a state that allows all the training patterns to be encoded.

The significance of this process is that, as the network trains, the nodes in the intermediate layers organise themselves such that different nodes learn to recognise different features of the total input space. After training, when presented with an arbitrary input pattern that is noisy or incomplete, the units in the hidden layers of the network will respond with an active output if the new input contains a pattern that resembles the feature the individual units learned to recognise during training.

“Backpropagation performs a gradient decent within the solution’s vector space towards a global minimum along the steepest vector of the error surface” (Gurney, 1997). The global minimum is the theoretical solution with the lowest possible error. The error surface itself is a hyperparaboloid but is seldom smooth. In most problems, the solution space is quite irregular with numerous pits and hills which may cause the network to settle down in a local minimum which is not the best overall solution. Figure 4-4 shows an example of local and global minima.

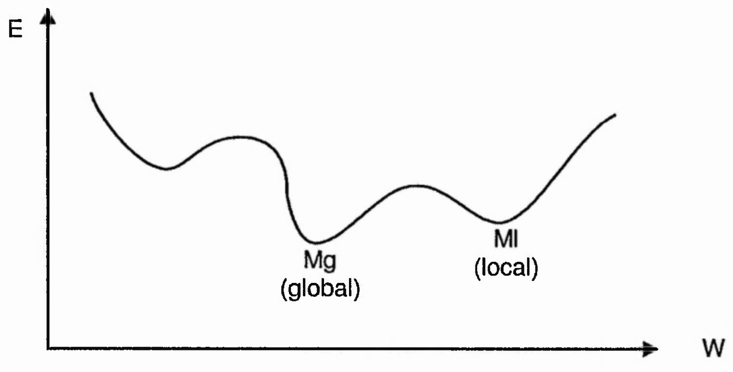


Figure 4-4 Local and global minima

Using a differentiable nonlinear activation function (i.e. sigmoid function) allows the network to bypass most of the local minima and helps it to decent towards a global minimum. However, the danger of getting stuck in local minima always exists. Generally the performance of the network is heavily dependent on certain parameters such as:

- The learning rate and momentum term
- The size of the network (i.e. number of hidden units)
- The initial weights

Once an ANN is trained to a satisfactory level, it may be used as an analytical tool on other data. To do this, the user no longer specifies any training runs and instead allows the network to work in forward propagation only. New inputs are presented to the input pattern where they filter into and are processed by the middle layers as though training were taking place, however, at this point the output is retained and no backpropagation occurs. The output of a forward propagation run is the predicted data which can then be used for further analysis and interpretation.

The pseudo code for the training phase of network development is shown below:

```
- Get input:target patterns
- Initialise network
    - WHILE stopping condition is false
        - FOR each training input:target pattern
        - Feed forward input
        - Calculate and backpropagate errors
        - Adjust weights
        - ENDFOR
    - ENDWHILE
- Save network weights in a file
```

The structure diagram for training the neural networks is shown in figure 4-5. As shown in figure 4-5, each sample will be trained to a RMS error. RMS is the Root Mean Square of all the actual outputs diviation from the target outputs. This has to be specified to give the software an obtainable value to reach. So the training is complete when the value of RMS goes below the specified limit or when the number of iteration exceeds a given limit.

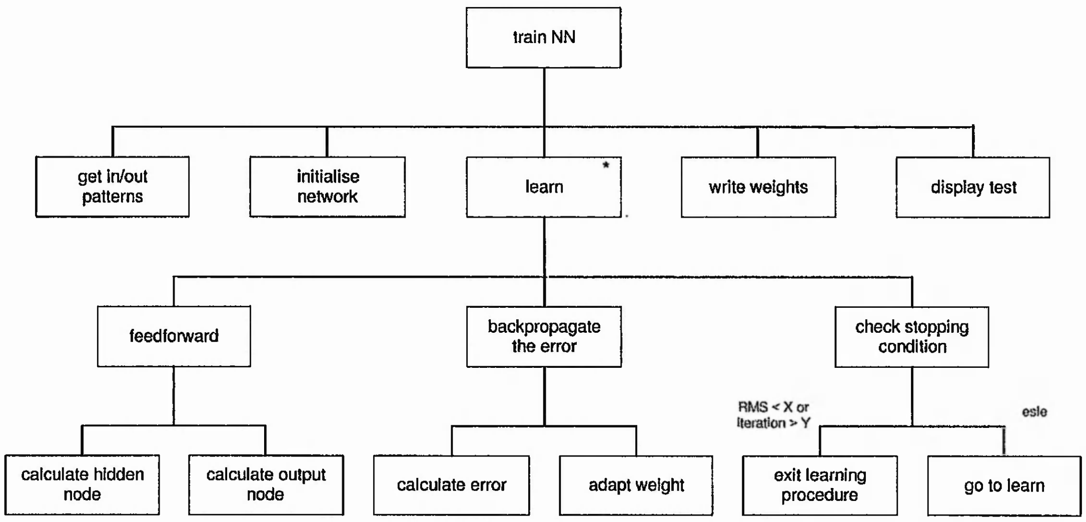


Figure 4-5 Structure diagram for the ANN training program

4.6 Specification of the application area

Gears have an extensive scope of application, ranging from small gears used in watches to large gears used in shipping industry. This research focuses on gears for standard applications, i.e. excluding the light applications and the heavy duty gears. Standard applications include gears used in cars, machine tools, laser machine, etc., where the outside diameter is in the range of 8-400 mm.

To obtain a more efficient result in a limited period of time, the application area is restricted according to the following criteria:

- The test will be carried out for one of the gear optimisation parameters, i.e. *facewidth*.
- Fixed centre distance
- Spur gears only
- Pitting and crowning allowed
- Number of teeth less than 16 is not considered, since they require undercutting.

These criteria were applied to make the experimental test feasible considering the limited time available. The objective is to develop an approach which could then be implemented for a wider range of gears and which could include all the combinations of the optimisation parameters.

4.7 Identification of input/output parameters

In order to be able to implement ANN into the system to replace the first tier, it is necessary to have a clear understanding of the input/output parameters for the first tier and the input parameters to the second tier. As explained in chapter 3, the gear optimisation program creates two input files, *genome.dat* and *optgear.dat*. These files contain the data provided by the user. The former contains the GA set-up parameters and the latter contains the initial data for gear design. The result of the optimisation program are saved in an output file, *result.dat*.

Genome.dat and *optgear.dat* contain 16 and 64 data respectively, and *result.dat* contains 36 data. Refer to appendix A for examples of the files *genome.dat, optgear.dat* and *result.dat*. At first glance, it seems that 80 input data and 36 output data are needed for the training patterns of ANN. Constructing an ANN with these numbers of inputs and outputs for this application is not feasible. Creating different combination produces an enormous number of patterns. If we assume that 5 different values are selected for each input data, the total number of patterns would be 5^{80}. This is an impossible task, bearing in mind that the target for each pattern is obtained by running the original gear optimisation for that pattern. This means running the optimisation program 5^{80} times to get the output and to be able to create the input patterns for the ANN. This will be unattainable due to the time required to execute all the patterns.

Fortunately, not all of these data need to be included in the ANN training patterns. Many of these parameters could simply be omitted, since they are not required for the specific application area that this prototype package is working on. These are those data related to helical gear, percentage of fitness function for centre distance, pitting, crowning and the first 9 parameters in the *genome.dat* file, since the program will focus on facewidth only. In addition, the population size and number of tests for

GA settings need not be included, since the optimum value of 2000 for population size and 10 for number of tests will be used while running the original optimisation file for the patterns to get the target.

The purpose of the proposed approach is to find a value as close to the output of tier 1 as possible, in order to replace tier 1 with the ANN. It is not essential for the estimated value to be exactly the same as the output of tier 1, since this acts as a warm input value to tier 2 and not the final output. This means that a number of initial parameters, that have slight effect on the final solution, could be excluded from the ANN training set. The idea is to reduce the data collection process by limiting the number of contributions for the used parameter.

The original optimisation program was thoroughly investigated and every single parameter was scrutinised to filter out those data that were contributing significant effects on the output. Even though many data parameters were affecting the final output, most of these only had a slight effect. So, these data could be omitted from the input of the ANN, e.g. lubrication viscosity, etc. To have a clear understanding , the following questions were dealt with:

- What are the inputs to the first tier ?
- What are the outputs of the first tier ?
- What are the inputs to the second tier ?
- What data are being changed ?
- Which input data have crucial effect on the output ?
- Which output data are being calculated and which are being derived from ?

The result of the initial investigation of the code showed that the network could be reduced down to 10 input data. This is a promising result, but can this still be reduced even further?

4.8 Determining the final output data

A code was added to the optimisation program that creates a text file providing information about the initial data entering the first tier and the corresponding data exiting that tier, along with their difference. This way it is possible to study which data are being changed. Optimisation program was run using different sets of data and the data file was examined each time. It was noticed that the only real change was in the value of width. The other outputs are either not changing or the change is insignificant. Moreover, their value is obtained through a straightforward calculation rather than through GA process. This makes sense since the optimisation program is trying to optimise the value of facewidth only. Consequently, ANN will only need to predict the value of width. Therefore, the ANN will have one output neuron only.

The target of the ANN will not however be the actual value of width. The target is the difference between the actual output and the initial value. Using this method it limits the range of data in the training set which leads to a better normalisation. The target will be spread out more, hence it will make it easier for ANN to estimate a more efficient value.

4.9 Determining the final input data

The size of the ANN has now been reduced to 10 inputs and 1 outputs. The input data are: *%facewidth, %equal slide, %contact ratio, %equal stress, power, input speed, speed ratio, module, pressure angle* and number of pinion *teeth*. The output is *facewidth*. Even though the number of input/output data to the ANN has been cut drastically compared to the original estimation, since the numbers are real numbers and some of them have a relatively high range (for example, the value of input speed could be between 750-1500), creating different combinations will still result in a large number of patterns. This consequently increases the time consumed to run them using the original gear optimisation to obtain the target values. Therefore, attempt is made to further reduce the number of inputs.

Table 4-1 shows the results of tests that were carried out to determine the extent of the influence of each input data on the output. Each row corresponds to one set of test. For each set, the output of tier 1 is noted (not shown in the table). Then each data is checked by running the optimisation program using the exact values for all the data apart from the one under investigation.

To give a better understanding of the table, one of the tests, for example test #4 is explained in detail. The first step is to run the optimisation program using the values given for test #4, shown in the table. The output *y*, is then noted down. All other outputs will then be compared with *y*. For example, in a case that *%facewidth* is to be checked, the optimisation is run using four different values, 0, 50, 75, 100, keeping the values for the rest of the data fixed (i.e. as shown in the table). The output of each of these is compared with *y*. If any different, this means that *%facewidth* has shown some influence on the output. Whereas if all the outputs are the same as *y*, this indicates that *%facewidth* has had no effect on the output. The same procedure is repeated on the rest of the data within this set.

In table 4-1, the ✓symbol under a data illustrates if that data is affecting the output and × sign shows that the data has no influence over the output.

Studying the table disclosed the following points:

- When *module* is relatively high, e.g. 6, most of the data are not affecting the output.
- *%CR* (%Contact Ratio) has no effect on the output.
- In most cases, *%slide, power, speed, ratio* and *pressure angle* have no effect on the output, unless the module is very low, i.e. 0.5.

%CR can definitely be omitted from the input data. However, the rest of the data affect the output in different ways.

A catalogue from davall gear manufacturing company (Davall, 2000) is used to assist in collecting the training data. Hence the prototype software tries to find the optimum value for the facewidth based on the conditions given in the catalogue. There are 4 fixed choices for pressure angle, i.e. 17.5, 20.0, 22.5, 24.0; and 12 fixed choices for module, i.e. 0.5, 0.7, 0.8, 1.0, 1.25, 1.50, 2.0, 2.5, 3.0, 4.0, 5.0, 6.0. It is proposed to create 48 different ANNs, i.e. for the combination of module and pressure angle. This way module and pressure angle need not be included in the net.

At first three input data were used (*%facewidth, %stress, teeth*) to construct the training patterns. The training errors and generalisation errors were good enough for the purpose of this project. The average training errors were 0.04 and generalisation errors 0.06. It however needed about 1700 training patterns for the ANN program to learn the patterns. Adding to the fact that test patterns were also needed to test the generalisation capability of the ANN network, the time taken to train each ANN was very high. This is due to the fact that for every single pattern, the GA optimisation needs to be executed to get the output of tier 1, which is used as the target value for the ANN pattern. It was causing difficulty due to the time limitation of this project.

Further investigation proved that instead of using *%equal stress* and *%facewidth* separately, their ratio could be used. This reduces the number of input data from three to two. This provides a possibility to reduce the number of training patterns required. The training patterns that were required for the ANN with two inputs, were between 150 and 250 patterns. This is a great reduction in the number of patterns, which consequently reduces the time taken to obtain the target values.

The average errors were 0.08 and 0.095 for the training and generalisation respectively. This shows a degradation in the results as compared to the previous case when the ratio was not used in the training patterns. There is therefore a trade off between speed and accuracy; but for this particular application obtaining the exact target values are not very crucial as the ANN provides the warm values to the tier2. If the estimated values by ANN are close to the desired values, then the GA of tier2 will be able to continue fine tuning the results.

For the cases when module is low, ANN was trained using 2 inputs as well. It was found that even though %slide, power, speed, ratio and pressure angle have some influence on the output, their effect was sufficiently low enough to exclude them from being part of the neural network training patterns without causing a drastic effect on the final result.

The loss in accuracy for both cases, mentioned above, will not be a major problem due to the fact that ANN will only need to provide a value close to the target value to allow GA to search for finer solution

Test #	%Face	%Slide	%CR	%Stress	Power	Speed	Ratio	Module	Press. Ang	Teeth
1	0 ✓	0 ×	0 ×	0 ✓	0.5 ×	750 ✓	2 ×	0.7 ✓	17.5 ✓	16 ✓
2	100 ×	100 ×	100 ×	100 ✓	15 ×	1500 ×	7 ×	6 ✓	24 ×	127 ×
3	50 ✓	50 ×	50 ×	50 ✓	7.5 ×	1050 ×	4 ×	2 ✓	20 ×	50 ✓
4	30 ✓	30 ×	70 ×	70 ✓	4 ×	960 ×	3 ×	1.5 ✓	22.5 ×	40 ✓
5	80 ✓	80 ×	20 ×	20 ✓	12 ×	1200 ×	5 ×	4 ✓	20 ×	80 ✓
6	40 ×	60 ×	80 ×	100 ×	5 ×	1000 ×	6 ×	5 ✓	24 ×	20 ×
7	25 ✓	70 ×	70 ×	25 ✓	2 ×	850 ×	3 ×	1 ✓	20 ×	25 ✓
8	90 ×	10 ×	10 ×	90 ×	13 ×	900 ×	7 ×	2.5 ✓	22.5 ×	75 ✓
9	60 ✓	30 ×	30 ×	60 ×	11 ×	1100 ×	5 ×	3 ✓	22.5 ×	90 ×
10	1 ✓	10 ×	20 ×	30 ×	13 ×	1300 ×	4 ✓	5 ✓	24 ×	67 ✓
11	75 ✓	75 ✓	75 ×	75 ✓	4 ×	850 ✓	3 ×	0.8 ✓	20 ✓	40 ×
12	100 ✓	90 ✓	80 ×	70 ✓	2 ✓	1150 ✓	4 ×	0.5 ✓	20 ✓	16 ✓
13	25 ✓	25 ×	25 ×	25 ×	5 ×	900 ×	2 ×	1.25 ✓	22.5 ×	16 ✓
14	40 ✓	60 ✓	80 ×	100 ✓	5 ✓	1000 ✓	6 ✓	0.5 ✓	24 ×	20 ✓
15	1 ✓	10 ×	20 ×	30 ✓	13 ✓	1300 ✓	4 ×	0.8 ✓	24 ×	67 ✓
16	60 ✓	30 ×	30 ×	60 ✓	11 ×	1100 ×	5 ×	4 ✓	22.5 ×	90 ✓

Table 4-1 Investigating the influence of each input data on the output

4.10 Process of collecting the training pattern

As mentioned earlier, in order to get the target value for every pattern, the optimisation program should run with that input data and the value of *facewidth* after the completion of tier 1 is used for the target of the pattern (the target will be the difference between this value and the initial facewidth). It will be very much time consuming for the developer to create the patterns one by one.

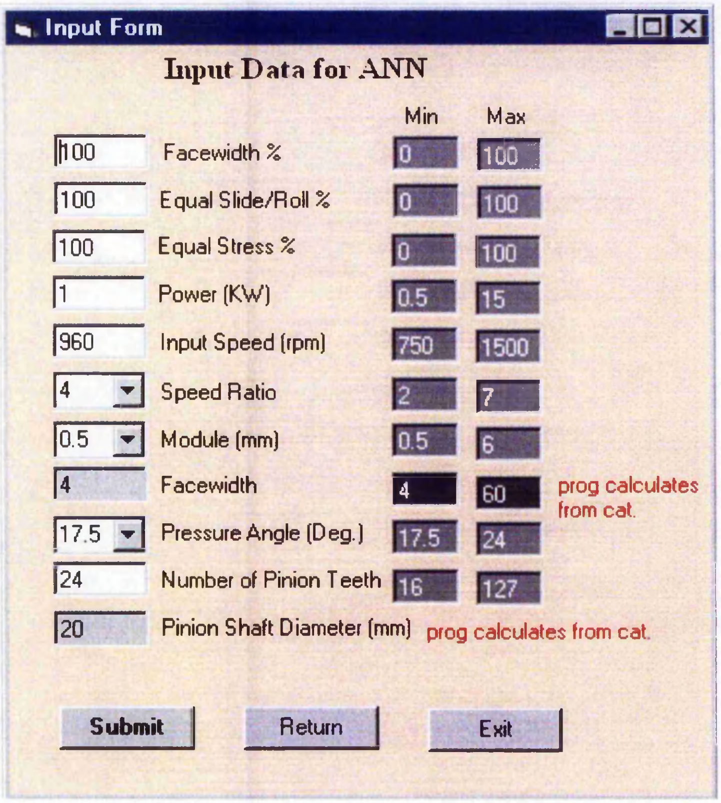
To speed up the process, a software is developed in Visual Basic, *inData*, that automates the process of data collection for ANN training and invokes the optimisation program for each pattern. The final patterns ready to be fed into the ANN, will be saved in the file *patterns.dat*. The software constitutes of two major parts:

- Collecting the input data
- Obtaining the target data

4.10.1 Collecting the input data

Figure 4-6 shows the screen shot of the form that obtains the input data from the developer. Boxes shown in grey colour are locked, i.e. the user cannot enter a value in these boxes. Pinion Shaft Diameter will not be used in the ANN input. Its value depends on the number of teeth and module, which is taken from the catalogue. This is coded in the software and its value is automatically retrieved as soon as the user selects the number of teeth and module. Facewidth is also not included in the ANN, it is shown here as reference. This is the initial value needed for the gear optimisation. Once again the software retrieves the value from the catalogue.

Minimum and maximum values for each data show the range of the data. Inserting a value less than the given minimum and more than the maximum will result in an error message, asking the user to change the value. These maximum and minimum values are used for normalising the input data for ANN process. Figure 4-6 displays the range of the permissible data for all the input parameters. These ranges are corresponded to the type of the gear chosen for this project, as described in section 4.6.


Input Form

Input Data for ANN

		Min	Max	
100	Facewidth %	0	100	
100	Equal Slide/Roll %	0	100	
100	Equal Stress %	0	100	
1	Power (KW)	0.5	15	
960	Input Speed (rpm)	750	1500	
4	Speed Ratio	2	7	
0.5	Module (mm)	0.5	6	
4	Facewidth	4	60	prog calculates from cat.
17.5	Pressure Angle (Deg.)	17.5	24	
24	Number of Pinion Teeth	16	127	
20	Pinion Shaft Diameter (mm)	prog calculates from cat.		

Figure 4-6 Obtaining input data from the developer

For the selection of training sample, the following points are taken into consideration:

- The training samples are organised in a random order rather than structured order. This ensures the maximum excitation of the nodes in a network by reducing the correlation in the training samples.

- Effort is made to present an equal number of training patterns from each class than an unequal number. A disproportionate number of training patterns from one class will pull the decision boundaries in a manner emphasising the most frequent classes, crippling the training of the minority class. Presenting a class in a non-uniform manner generally increases the amount of training needed, in some cases to infinity.

Effort is made to present as many different samples as possible, so that the training patterns represent the pattern variations within a class. Insufficient variation in training samples will produce boundaries that will not separate the feature space.

Sufficient variation will produce boundaries, which better partition the featurespace, minimising the probability of a rare variation in the input data pattern being misclassified.

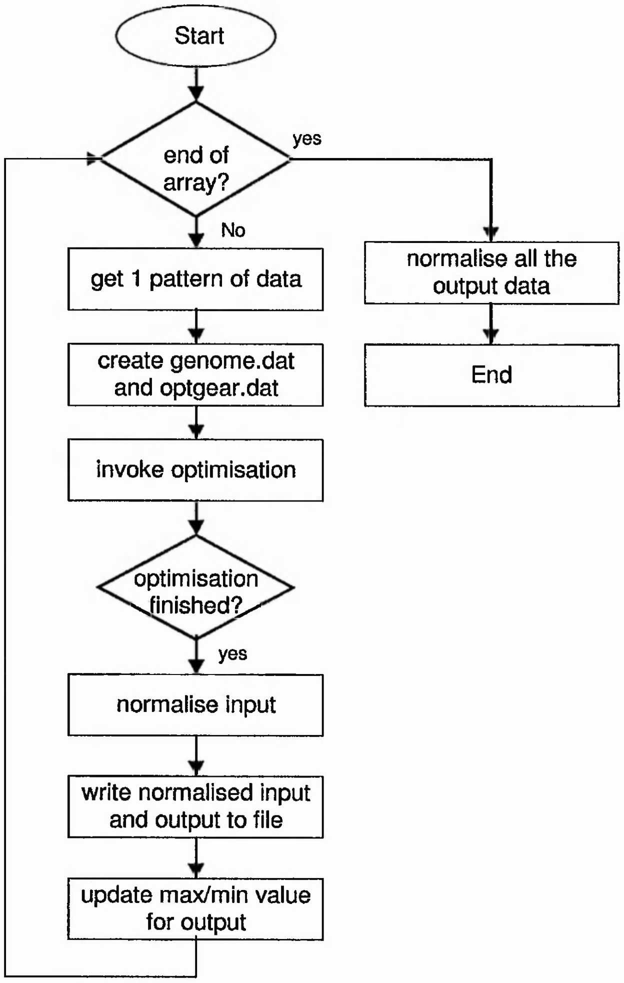

Start
end of array?
yes
No
get 1 pattern of data
create genome.dat and optgear.dat
invoke optimisation
optimisation finished?
yes
normalise input
write normalised input and output to file
update max/min value for output
normalise all the output data
End

Figure 4-7 Flow diagram of the process of creating the input file for ANN

4.10.2 Obtaining the target data

Once all the input data are provided, the software starts invoking the gear optimisation for each pattern. It reads the first pattern and creates the two necessary input files for gear optimisation, i.e. *genome.dat* and *optgear.dat*. The gear optimisation is then called which reads the data from these input files and starts the GA process. The program stops after the completion of the first tier and the value of facewidth is saved which will be used as the target of the first input pattern of ANN. At this stage, the input patterns are normalised and written to a file called *NNpatterns.dat* and the difference between the new facewidth and the initial facewidth is added after the input data. This process is repeated for all the patterns. The flow diagram of the procedure is shown figure 4-7.

4.10.3 Normalising the data

Before feeding the patterns into the ANN, both the input data and the target need to be normalised. Data normalisation is required by the backpropagation algorithm to perform properly. The normalised data for the ANN, when sigmoid activation is used, lies between 0.0 and 1.0; while for the tanh activation function data values between -1.0 and 1.0 are used. One way to create a data that would lie within the limiting values is to use the minimum and maximum values in the dataset, that is:

$$x = (x_r - x_{\min})/(x_{\max} - x_{\min})$$

where x_r is the real value before normalisation and $x_{\min}$ & $x_{\max}$ are the minimum and maximum values found in the dataset. The above equation normalises the dataset between 0.0 and 1.0 which is used when sigmoid function is employed. For the tanh function we have:

$$x = (x_r - x_{cent})/(x_{\max} - x_{\min})$$

Where cent is the centre distance between min and max, that is:

$$x_{cent} = ((x_{\max} - x_{\min})/2) + x_{\min}$$

The input data in the file *NNpatterns.dat* are normalised but not the target. This is because the maximum and minimum values for the input data were already available, therefore these are used to normalise the input data before writing them into the file. However, the minimum and maximum values of the target is not available, therefore while executing the optimisation program, a files is created *(maxMinOutput.dat)* that contains two data, which represent the maximum and minimum values for the target data. For every pattern, the data within this file is updated. When the targets for all the patterns are produced, the *maxMinOutput.dat* will contain the final maximum and minimum data for the target. These two values are then used to normalise the target.

At first *tanh* activation function (y = tanh (x)) was used, which means that the output was normalised between –1 and 1. The program was then tested using the sigmoid activation function ($y = 1/(1+e^{-x})$), that is the output was normalised between

0 and 1. This showed an improvement in the result. The results of five test examples are shown in table 4-2.

Weight file No.	E(t) Sigmoid	E(g) Sigmoid	E(t) Tanh	E(g) Tanh
12	0.039	0.059	0.044	0.174
27	0.08	0.10	0.16	0.22
37	0.087	0.154	0.163	0.30
45	0.044	0.082	0.089	0.16
48	0.055	0.094	0.10	0.20

Table 4-2 Test examples showing the improvement of results using Sigmoid activation function

Weight file No. refers to the different cases; detail explanation is given is section 4.12. As can be seen from the given examples, there is a clear improvement using sigmoid activation function.

4.11 Examining the output data

The result of the primary testing was not very promising. The estimation was far from acceptable. To understand the cause of the problem, tests were conducted to examine the effect of the inputs. Table 4-3 shows the test carried out on the effect of the ratio of %stress over %facewidth. The teeth was kept constant at 50. As can be seen in the table, it is noted that the output has minimum value when the ratio is less than 30%; and maximum value when the ratio is greater than 86%. The test repeated once again keeping the teeth number constant at 127. Similar result was produced, with a slight change. This time the output had a maximum value when ratio is greater than 88% rather than 86%.

This showed that the ANN does not need to be applied for the whole range. As an example, if a ratio of less than 30% is selected, the program will automatically produce the output, which is minimum value and there is no need to apply ANN to estimate the output. Therefore, these minimum and maximum parts will not included in the training patterns for ANN.

In order to implement this process, thorough testing were carried out on the input data using similar methods and for all the 48 ANN cases. Since not all of them have the same boundary points. The tests were applied to all the cases to find their relevant boundary points.

%Facewidth	%Stress	Target
100	1	-1.12
100	2	-1.12
100	3	-1.12
100	4	-1.12
100	5	-1.12
100	6	-1.12
100	7	-1.12
100	8	-1.12
100	9	-1.12
100	10	-1.12
100	11	-1.12
100	12	-1.12
100	13	-1.12
100	14	-1.12
100	15	-1.12
100	16	-1.12
100	17	-1.12
100	18	-1.12
100	19	-1.12
100	20	-1.12
100	21	-1.12
100	22	-1.12
100	23	-1.12
100	24	-1.12
100	25	-1.12
100	26	-1.12
100	27	-1.12
100	28	-1.12
100	29	-1.12
100	30	-0.96
100	31	-0.96
100	32	-0.96
100	33	-0.8
100	34	-0.8
100	35	-0.8
100	36	-0.8
100	37	-0.64
100	38	-0.64
100	39	-0.64
100	40	-0.48
100	41	-0.48
100	42	-0.48
100	43	-0.48
100	44	-0.32
100	45	-0.48
100	46	-0.32
100	47	-0.32
100	48	-0.48
100	49	0
100	50	-0.16
100	51	0
100	52	0
100	53	0

100	54	0
100	55	0
100	56	0
100	57	0.16
100	58	0.16
100	59	0.16
100	60	0.16
100	61	0.16
100	62	0.32
100	63	0.32
100	64	0.32
100	65	0.32
100	66	0.32
100	67	0.48
100	68	0.48
100	69	0.48
100	70	0.48
100	71	0.48
100	72	0.64
100	73	0.64
100	74	0.64
100	75	0.64
100	76	0.96
100	77	0.8
100	78	0.8
100	79	0.8
100	80	0.8
100	81	0.8
100	82	0.96
100	83	0.96
100	84	0.96
100	85	0.96
100	86	0.96
100	87	1.12
100	88	1.12
100	89	1.12
100	90	1.12
100	91	1.12
100	92	1.12
100	93	1.12
100	94	1.12
100	95	1.12
100	96	1.12
100	97	1.12
100	98	1.12
100	99	1.12
100	100	1.12

Table 4-3 Effect of (%stress / %facewidth) on the output

4.12 Replacing the first GA tier with ANN

After all the ANNs have been trained to a satisfactory level, the original gear design optimisation program needs to be altered to implement the ANN. The steps that need to be accomplished in the optimisation program are listed below:

```
– Retrieve %facewidth, %equal stress, module, pressure angle, power, input
  speed, speed ratio and teeth
– Check the condition
    – IF ANN is required
        – Select the relevant ANN
        – Open the appropriate weight file and run ANN
        – Obtain the output
        – De-normalise the output (i.e. facewidth)
        – Add the de-normalised output to the initial facewidth
    – END IF
    – ELSE
        – Select the appropriate maximum or minimum value
    – END ELSE
– Update Initgear
– Start tier 2
```

Initgear is a *struct* within the optimisation program that contains all the data required for GA process. This is fed into tier 1 and after the completion of tier 1, *Initgear* struct is updated and fed into tier 2.

Table 4-4 illustrates the conditions for all the ANNs that the program uses to run the appropriate ANN. Column *weights file No.* refers to the name of the weight files. For example, when module is 4 and pressure angle is 20, the weight has a value of 10. This means that the program should retrieve the file *weight10.dat*. *Input* refers to the number of input nodes and *hidden* refers to the number of hidden nodes. As an example, for the situation when module is 0.5 and pressure angle is 24, the ANN has a 2-3-1 architecture, i.e. 2 input nodes, 3 hidden nodes and 1 output node. This information is necessary for running the ANN.

Weights file No.	Module	Pressure Angle	Max output	Min output	Input	Hidden	% ratio < ➔ = min	% ratio > ➔ = max	Conditions for max & min
1	6	17.5	16.80	-16.80	3	4			
2	6	20	16.80	-16.80	3	4			
3	6	22.5	16.80	-16.80	3	4			
4	6	24	16.80	-16.80	3	4			
5	5	17.5	14	-14	3	4			
6	5	20	14	-14	3	4			
7	5	22.5	14	-14	3	4			
8	5	24	14	-14	3	4			
9	4	17.5	11.20	-11.20	3	4			
10	4	20	11.20	-11.20	3	4			
11	4	22.5	11.20	-11.20	3	4			
12	4	24	11.20	-11.20	3	4			
13	3	17.5	8.40	-8.40	3	4			
14	3	20	8.40	-8.40	3	4			
15	3	22.5	8.40	-8.40	3	4			
16	3	24	8.40	-8.40	3	4			
17	2.5	17.5	7	-7	2	3	0.31	0.85	
18	2.5	20	7	-7	2	3	0.31	0.81	
19	2.5	22.5	7	-7	2	3	0.31	0.81	
20	2.5	24	7	-7	2	3	0.31	0.82	
21	2	17.5	5.60	-5.60	2	3	0.32	0.81	
22	2	20	5.60	-5.60	2	3	0.32	0.81	
23	2	22.5	5.60	-5.60	2	3	0.31	0.82	
24	2	24	5.60	-5.60	2	3	0.30	0.82	
25	1.5	17.5	5.04	-5.04	2	9	0.30	0.82	
26	1.5	20	5.04	-5.04	2	3	0.31	0.82	
27	1.5	22.5	5.04	-5.04	2	3	0.30	0.82	
28	1.5	24	5.04	-5.04	2	3	0.25	0.89	
29a	1.25	17.5	4.20	-4.20	2	4	0.31	0.82	Teeth > 25
29b	1.25	17.5	4.20	-4.20	2	3	-----	0.95	" else
30	1.25	20	4.20	-4.20	2	5	0.31	0.82	Teeth > 22
31	1.25	22.5	4.20	-4.20	2	8	0.30	0.82	Teeth > 20
32a	1.25	24	4.20	-4.20	2	4	0.30	0.82	Teeth > 20
32b	1.25	24	4.20	-4.20	5	7	------	0.92	" else
33	1	17.5	4.20	-4.20	2	5	0.26	1.23	
34	1	20	4.20	-4.20	2	5	0.28	0.98	
35	1	22.5	4.20	-4.20	2	4	0.28	0.98	
36	1	24	4.20	-4.20	2	5	0.28	0.97	
37	0.8	17.5	1.68	-1.68	2	3	0.30	0.88	
38	0.8	20	1.68	-1.68	2	9	0.30	0.88	

39	0.8	22.5	1.68	-1.68	2	5	0.30*	0.87	* If 65<teeth<96→ use ANN
40	0.8	24	1.68	-1.68	2	5	------	0.88	
41a	0.7	17.5	1.40	-1.40	2	3	0.30	0.88	Teeth < 121
41b	0.7	17.5	1.40	-1.40	2	3	-----	0.88	" else
42a	0.7	20	1.40	-1.40	2	3	0.30	0.89	Teeth < 108
42b	0.7	20	1.40	-1.40	2	5	-----	0.89	" else
43	0.7	22.5	1.40	-1.40	2	3	0.30	0.88	
44a	0.7	24	1.40	-1.40	2	3	0.30	0.88	Teeth<89 && teeth > 111
44b	0.7	24	1.40	-1.40	5	3	-----	-----	88 < teeth < 112
45	0.5	17.5	1.12	-1.12	2	3	0.30	0.87	
46	0.5	20	1.12	-1.12	2	3	0.30	0.87	
47	0.5	22.5	1.12	-1.12	2	3	0.26	0.88	
48	0.5	24	1.12	-1.12	2	3	0.24	0.93	

Table 4-4 Conditions for all the ANNs

The last three columns, (i.e. *%ratio < ➔ = min, %ratio > ➔ = max* and *Conditions for max & min*) show the situation if the maximum or minimum output values could be used instead of running the ANN. To give a better understanding of this, two of the cases are explained here. For example for the case when module is 2 and pressure angle is 17.5, if the value of %ratio (i.e. ratio of *%facewidth and %stress*) is less than 0.32%, then minimum output value is used, rather than running the ANN. Similarly if %ratio is greater than 81%, maximum value is used. For the case when module is 1.25 and pressure angle is 17.5, the maximum/minimum case applies only when teeth is greater than 25. If teeth is smaller or equal to 25, then ANN is used; and if %ratio is greater than 95%, then maximum is used.

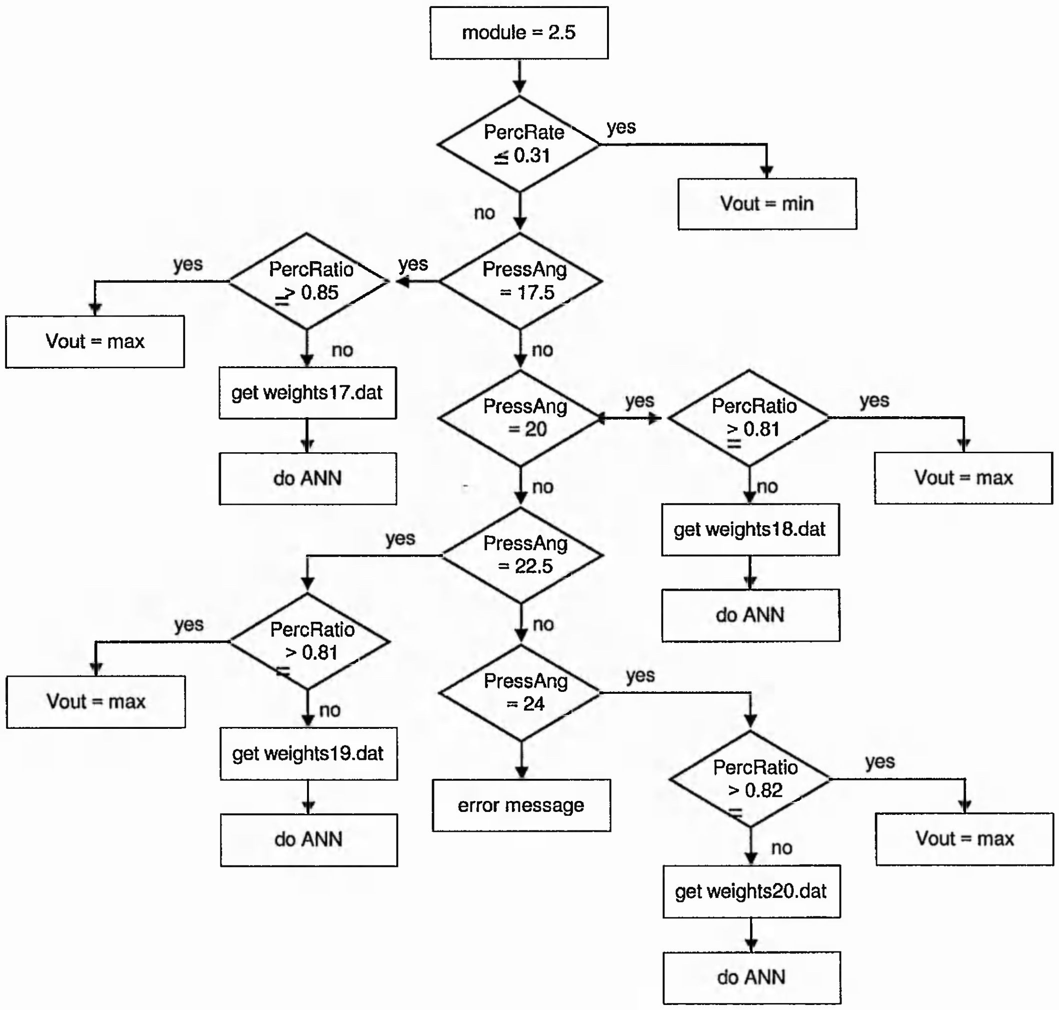


Figure 4-8 Flow diagram for the process of finding the appropriate weight file and ANN

As shown in the table, for the cases with the modules 3, 4, 5 and 6, ANNs have been constructed using three input data. For the rest, two input data have been used. Only two cases needed 5 input data (refer to *weight file No.* 32b and 44b), since using two input data did not give a satisfactory result. For these two cases, *input speed, speed ratio* and *power* were needed as well as *teeth* and ratio of *%facewidth and %stress.*

Figure 4-8 shows the flow diagram of the process of finding the appropriate weight file and ANN for the case when module is 2.5. In table 4-4 this case is shown in red colour. The program first checks the value of percentage ratio of %stress over %facewidth (*PercRate*). If it is equal to or smaller than 0.31, then no ANN needs to be applied and the minimum output value is given out. If it is greater, then it first checks the value for pressure angle (*pressAng*). If it is 17.5, then it checks the value of *PercRate.* If it is equal to or greater than 0.85, then output is assigned the maximum value, else the program called the file *weight17.dat*, which contains the appropriate weights. The program then runs ANN using this weight file to calculate the output. Similar procedure applies for pressure angles 20, 22.5 and 24.

This process shown in figure 4-8 is carried out for all the cases shown in table 4-4.

4.13 Determining ANN architecture

During the training, the ANN was tested with different values of learning rate, hidden nodes, momentum term, initial weight and the number of iteration, ensuring the best combination that would give the optimum result. Some of these tests are discussed here:

4.13.1 How long it needs to be trained?

It is essential not to overtrain the network. The quality of the network often reaches a peak as training progresses and after that it starts deteriorating. To overcome this problem, training is done as follows. For each trial of number of hidden neurons, a random initial weight is to be generated and trained until improvement is negligible. Then more random initial weights are generated and trained. It follows by finding the best possible value for learning rate and momentum term. When a moderate number of these repetitions in a row fail to improve the performance any more, this proves that the network has been trained as best as can be done for the training set. If, after all the training, its performance on the validation set is significantly worse than on the training set, either the training set is defective (i.e. too small or not representative of the population), or there are too many hidden neurons.

Overfitting results when the training set is small relative to the number of hidden neurons. To compensate for the insufficiency of training data, the number of hidden neurons are limited. The training set size and hidden layer are tied together. When they are unbalanced in one direction, the network is unable to learn as well as it should. When unbalanced in the other direction, the network learns too much (memorises) and hence generalises poorly. There should be balance.

As an example, one of the ANNs is evaluated here. This is the case when module is 0.7 and pressure angle is 17.5. Table 4-5 shows the test that was carried out on the number of iterations. A network with 2-3-1 structure was used, having two inputs, one hidden layer with three nodes and one output. The learning rate, momentum term and initial weight were set to be 0.05, 0.05 and 0.1 respectively. The network was then tested with different number of iterations, ranging form 5000 to 60000, refer to table 4-5.

ANN	Learning Rate	Momentum Term	Init Weight	No. of Iteration	εt	εg
2-3-1	0.05	0.05	0.1	5000	0.055	0.076
"	"	"	"	10000	0.045	0.076
"	"	"	"	20000	0.047	0.076
"	"	"	"	30000	0.045	0.076
"	"	"	"	40000	0.044	0.076
"	"	"	"	50000	0.044	0.077
"	"	"	"	60000	0.044	0.077

Table 4-5 Testing for the number of iterations

As shown in the table, there is only a slight change in the output errors. Increasing the number of iteration, causes a reduction in training error, εt, and an increase in generalisation error, εg. This means that the generalisation would be less accurate.

So, the iteration number of 40000 was chosen, which shows an acceptable learning and generalisation capability.

4.13.2 How many hidden nodes should be used?

Choosing an appropriate number of hidden neurons is very important. Using too few will starve the network of the resources it needs to solve the problem. Using too many will increase the training time, perhaps so much that it becomes impossible to train it adequately in a reasonable period of time. Also, as excessive number of hidden neurons may cause overfitting. Overfitting causes all training points to be memorised, which might lead to a poor generalisation performance. If the performance of the network is evaluated with the training set, it will be excellent. However, the network performs poorly on a general pattern. This is because it will consider trivial features, and become confused.

The best approach for finding the optimal number of hidden neurons is through the trial and error. First a number of neurons are chosen which is small (i.e. 1 neuron in this case). The network is trained and tested and its performance is recorded. Then the number of hidden neurons are slightly increased and the network is again trained and tested. This process is repeated until the error is acceptably small, or until no significant improvement is noted.

Table 4-6 shows the test that was carried out to select the best structure for the network. A network with a learning rate of 0.05, momentum term of 0.05, initial weight of 0.1 and iteration number of 40000 was selected. The network was tested with different hidden nodes ranging from 1 to 10. The result shows that as the hidden nodes increase, the training error εt gets increased, but the generalisation error εg gets reduced. It was decided to use a 2-5-1 network with the lowest training error and relatively low generalisation error. So, even though lower number of hidden nodes would improve the training error, this has a negative effect on the generalisation. Therefore, a 2-5-1 network would be a more efficient structure.

ANN	Learning Rate	Momentum Term	Init Weight	No. of Iteration	εt	εg
2-1-1	0.05	0.05	0.1	40000	0.071	0.109
2-2-1	"	"	"	"	0.044	0.078
2-3-1	"	"	"	"	0.044	0.076
2-4-1	"	"	"	"	0.045	0.076
2-5-1	"	"	"	"	0.046	0.074
2-6-1	"	"	"	"	0.046	0.074
2-7-1	"	"	"	"	0.046	0.074
2-10-1	"	"	"	"	0.052	0.074

Table 4-6 Testing for the number of hidden nodes

4.13.3 The learning rate, momentum term and initial weights

The choice of learning rate and momentum term has a significant effect on the performance of a network both during the learning and recall phases. Learning rate is the rate of convergence between the current solution and the global minimum. A fixed value of the learning rate might not be appropriate for all portions of the error surface. In flat regions, where the learning is slow, a relatively large learning rate can help skip the shallow minima at a larger pace. This is because the weights will be changed by a larger amount. On the other hand, in a steep surface, the derivative of the weight is large. This might result in overshooting a minimum since the weights will be updated by a large amount in the vicinity of a minimum. This problem can be solved by having a small learning rate.

The learning rate must always be much less than 1, typically at most 0.4. Up to a point, larger learning rates will make training progress faster. If the learning rate is too large though, convergence may never occur. The weight vectors may oscillate wildly.

Momentum term also helps the network to overcome obstacles (i.e. local minima) in the error surface and settle down at or near the global minimum. It is a constant that is multiplied by the previous weight change. The momentum constant is greater than zero, and to ensure convergence, it is also less than 1.

A proper choice of initial weights often results in shortening training time by several orders of magnitude. The network should be simulated with a different set of small random weights to find out the best set of values for the initial weights.

Similar tests as described in sections 4.13.1 and 4.13.2 were carried out for the learning rate, momentum term and initial weight. Their optimum values were found to be 0.045, 0.05 and 0.05 respectively. Tests were carried out for all the ANNs to find out the best values for their learning rate, hidden nodes, momentum term, initial weight and the number of iterations. The final result of these tests are shown in appendix B.

4.14 Validation test and results

In order to analyse the efficiency of the proposed method, the output of tier 1 was tested for both the original GA program and the new one, i.e. the ANN, refer to figure 4-9. A number of designs were fed into the programs and the output of the first tiers were monitored. The data used for testing is shown in appendix C. As shown in figure 4-9, the result has been satisfactory. In most cases, ANN has estimated a value very close to the output of GA. Only in two cases, i.e. tests number 4 and 6, the output estimated by ANN is slightly different. This does not cause any problem, since the estimated value is not totally off the target and it is still close to the GA output. Moreover, this is not the final value. This value would then be fed into Tier 2, where GA would start doing the search to find the final solution.

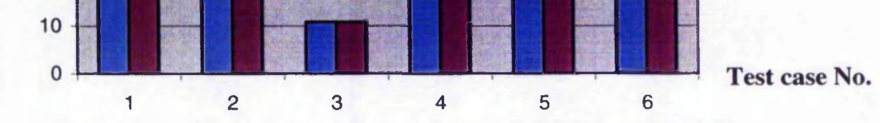


Figure 4-9 Comparison of the output of Tier 1 for GA and ANN

To further analyse this methodology, a number of tests were carried out on the speed and the iteration number, as shown in table 4-7.

	$\tau_1 + \tau_2$	Total	$\tau_{NN} + \tau_2$	Total	%Improvement
Execution Time (s)	25.15 + 25.49	50.64	0.01 + 25.82	25.83	48.99 %
No. of Iteration	82 + 76	158	1 + 80	81	48.73 %
Execution Time (s)	23.84 + 21.84	45.68	0.01 + 25.49	25.50	44.18 %
No. of Iteration	62 + 85	144	1 + 70	71	50.69 %
Execution Time (s)	44.87 + 26.64	66.747	0.01 + 25.18	25.19	62.26 %
No. of Iteration	85 + 102	187	1 + 72	73	60.96 %

Table 4-7 Comparing the speed of the execution of the original optimisation program and the new one that implements ANN

Table 4-7 shows the execution time and the number of iteration for the original optimisation program that uses GA for both tier 1 and 2, and also for the new program which implements ANN for tier 1. Here τ_1 and τ_2 indicate the time taken for the tier1 and tier2 of the original optimisation program respectively while τ_{NN} the execution time for the neural network computation. Tests for three different designs are shown in this table. For a better understanding, the execution time and the iteration number have been recorded for both tiers.

In the first test, both the execution time and the iteration number of the two tiers of the new program, i.e. $\tau_{NN} + \tau_2$ have been reduced almost by half, i.e. execution time has been dropped from 50.64s to 25.83s and the iteration number from 158 to 81. Similar trend appears in the next two tests shown in the table, with the third test having percentage improvement of even higher than 60.

The result of testing proves that ANN has managed to achieve the purpose of the proposed method. In most cases, it can estimate the output of tier 1 and the estimated value is very close to the target. This means that it can be used to replace the GA program in tier 1.

4.15 Discussion and conclusion

Substituting GA used in tier 1 by ANN, reduces the execution time dramatically. Even though ANN requires a lot of time to go through the learning process, once the training is completed, it is a matter of a simple calculation to produce the result (i.e. feed-forward computation). Therefore, it does not need to go through any iteration. Whereas in GA, the program needs to go through a number of iterations to search for the best possible solution. As a result, by replacing one tier that performs GA with ANN, it acts as if that tier has been eliminated, leading to a substantial reduction in the execution time. A quick glance at the simplified structure of both programs would prove this theory, see figure 4-10.

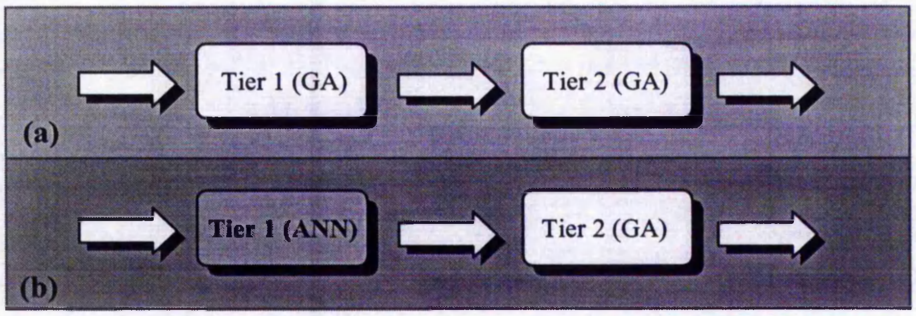


Figure 4-10 Comparing the two programs, (a) Original program, using GA for Tier 1, (b) New approach, using ANN for Tier 1

As discussed earlier, the experimental results show approximately 50% improvement in the execution time. Reducing the execution time improves the efficiency of the program. This is particularly important if applied in an Internet environment. Therefore this new methodology could be applied for online gear optimisation program. This would be very beneficial for the Internet medium where speed is a crucial factor.

ANN could have the capability of replacing both tiers, in such a case the result would be outstanding. Since ANN would be replacing the whole GA in the application, resulting in an immediate output of the result. At the moment, the training and testing errors are not low enough to replace tier 2. However, with further research in this area, this could be accomplishable.

In the next chapter, a methodology is presented to improve the learning technique of the backpropagation learning algorithm. This is done by re-scaling the process of normalisation of the output data.

Chapter 5

Improvement of Performance of Backpropagation Learning Algorithm

5.1 Introduction

Artificial Neural Networks (ANN) have been used in many applications to solve various non-linear problems. The strength of ANN lies in its generalisation to solve unseen patterns. There is however a limitation to how much improvements that can be made to enhance the generalisation. Trying out different hidden nodes and neural parameters is one way of improvement. Another approach is to change the data pattern to allow the neural network to solve the relationship between input and output more easily. This is known as data re-scaling.

In this chapter a method is described that allows an improvement in the performance of the Backpropagation learning algorithm. This is done by normalising the output values between 0.1 and 0.9 instead of 0 and 1 for the sigmoid function and between -0.9 and 0.9 instead of -1 and 1 for the tanh function. This is related to the effect of the output derivative near the saturation levels. At these levels the derivatives tend to zero and hence the weight changes would be near zero. By re-scaling the output data, the minimum value of the derivative will be slightly above zero which allows a slight weight change at the saturation level. This leads to a faster and more efficient learning process for the neural network.

This is very helpful for the problem of gear design optimisation described in the previous chapter. It helps the ANN to estimate a more accurate output value for the facewidth. Moreover, improvement in the speed of learning will help to speed up the training process which is vital for the gear design optimisation problem.

The proposed method is tested on two applications: 1) Function approximation and 2) Gear optimisation. Both produce satisfactory results and show an improvement in the training and generalisation errors.

5.2 Backpropagation algorithm

In this algorithm, the goal is to train a multi-layer perceptron network to approximate an unknown function, based on some training data consisting of pairs (*x*,*d*). The vectors *x* and *d* represent a pattern of input to the network and desired output (target) respectively. The Backpropagation calculation can be divided into three segments, feed-forward, weight-adaptation, and feed-back (or error-backpropagation) (Gurney, 1997).

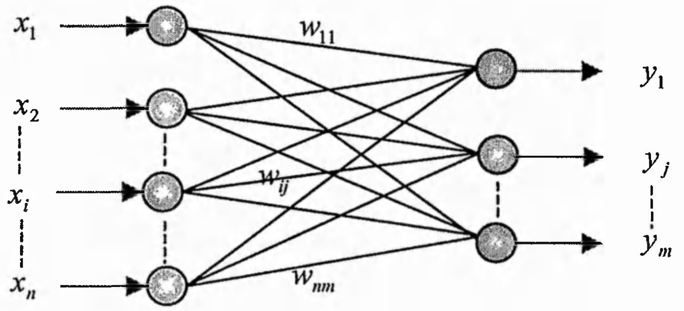


Figure 5-1 Two Layer ANN

5.2.1 Feed-forward

In feed-forward operation, all inputs to each neuron are multiplied by their associated weights, which are then summed up and sent to a limiting function to bound the output values:

$$sum_j = \sum_{i=1}^{n} x_i * w_{ij} \qquad (1a)$$

$$y_j = f(sum_j) \qquad (1b)$$

where x is the input value, y is the output value, as marked on the two layer ANN diagram in figure 5-1. w_{ij} is the weight value between input i and output j and *f(sum$_j$)* is a nonlinear activation function which is commonly represented by Sigmoidal approximation function (Lippmann, 1987) with the advantage of having a simple derivative, that is:

$$f(x) = 1/(1+e^{-x}) \qquad (2a)$$

$$f'(x) = f(x)*(1-f(x)) \qquad (2b)$$

Sigmoid function bounds values between 0 and 1. However for certain applications, where a broader range of input and output values might be required, the tanh function is used instead. This function limits the activation values between –1 and 1.

$$f(x) = \tanh(x) \qquad (3a)$$

$$f'(x) = 1-(f(x)*f(x)) \qquad (3b)$$

Figure 5-2 shows the two different limiting functions used in neural networks. The derivatives are used during the learning process where it behaves somewhat like a filter. It allows a greater change in the weight values when the neuron output is near zero while it permits almost no changes in the weight values when a neuron has reached the saturation (0 or 1 for the *sigmoid* and –1 or 1 for the *tanh* function).

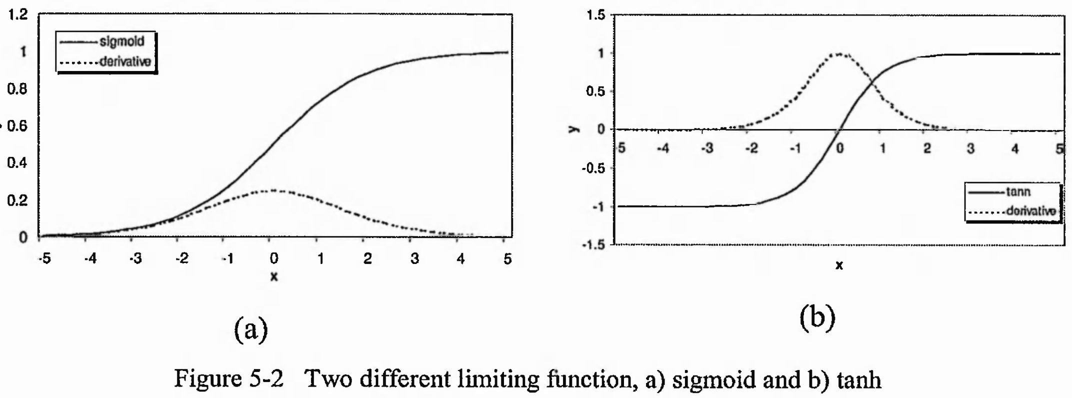


Figure 5-2 Two different limiting function, a) sigmoid and b) tanh

5.2.2 Weight Adaptation

The output weights are updated by an error value that is obtained by,

$$\delta_j^L = f'(y_j^L)*(d_j - y_j^L) \qquad (4)$$

where d_j is a target value and δ_j^L is error value for the jth neuron in the output layer L, and $f'(y_j^L)$ is the derivative of the activation value of the output layer neuron. This error value is then used to update weights of the jth node:

$$\Delta w_{ij}^L = \eta * \delta_j^L y_i^h \qquad (5a)$$

$$w_{ij}^L(t+1) = w_{ij}^L(t) + \Delta w_{ij}^L \qquad (5b)$$

where η is learning rate coefficient, y_i^h is the output value of neuron i in hidden layer h, w_{ij} is weight value between hidden neuron i and output neuron j, and t is a time step (i.e. iteration).

5.2.3 Feed-back

Since there is no explicit target for the internal representation of the network (or hidden nodes), the back-propagated error signal from the output layer is used to obtain the error values for the hidden neurons. This is a reverse calculation to the feed-forward going from output to hidden layer. Here for each hidden node, the error values of all output neurons are multiplied by their associated weight values and the summation of all multiplication will represent the error value:

$$\delta_i^h = f'(y_i^h) * \sum_{j=1}^{n} \delta_j^L * w_{ij} \qquad (6)$$

A similar weight adaptation is performed for the output weight.

5.3 What problems could arise ?

In most applications, the values of the outputs used for training data are not in the ranges of the Neural Networks activation function (i.e. sigmoid or tanh). This means that the output data must be pre-processed by scaling. Scaling data before training the Neural Networks is done by mapping the desired range of the outputs to the full working range of the Neural Networks outputs.

One problem with scaling data to full ranges of activation functions is that the values near the two ends (0 and 1 for sigmoid and –1 and 1 for tanh) reach the saturation level. This causes the weight adaptation of such neurons to seize or change insignificantly. The reason lies in the equation of the backpropagation weight adaptation (see Eq. 4 & 5).

In weight adaptation equation the derivative of the neurons have significant effect on the weight values. As can be seen in figure 5-2a, the value of the derivative reaches a peak value (0.25) at the centre of the sigmoid function (i.e. when y=0.5 for x=0) and progresses towards zero very quickly at the two ends of the activation function (when y=0 or y=1). Therefore the output neurons whose target values are close to 1 or 0, their derivatives become almost zero. Hence any weight adaptation for such output neurons would be zero (or significantly low), since derivative affects the weight change formula (i.e. if derivative=0 $\rightarrow$ weight change=0).

As an example if an output value is 0.999, its derivative value would be 0.000999. Therefore the weight change, as indicated in Equations 4 & 5, will be very small for the neurons that have reached saturation. The same scenario exists for the tanh function (see figure 5-2b). The value of the derivative reaches the peak value at the centre of the activation function, (i.e. when y=0 for x=0) and inclines towards zero at the two ends (i.e. when y = -1 and y = 1). So for tanh activation function, those output neurons that have target values close to 1 and –1, their derivatives becomes zero causing no change in the value of the weights.

5.4 Rescaling the output data

One solution to the problem of the insignificant weight adaptation of the neurons with saturated target values is to allow the derivatives of such neurons to have small effect even though they have reached their maximum or minimum values. This is done by normalising the output values between 0.1 and 0.9 instead of 0 and 1 for the sigmoid function and between –0.9 and 0.9 instead of –1 and 1 for the tanh function. At these levels the derivatives tend to zero and hence the weight changes would be near zero. By re-scaling the output data, the minimum value of the derivative will be slightly above zero which allows a slight weight change at the saturation level. In this case the lowest derivative of the neurons would be 0.09 (i.e. 0.1(1-0.1) or 0.9(1-0.9) refer to Eq.2b) as compared to 0.0 (i.e. 0(1-0) or 1(1-1)) for the 0 to 1 data scaling. Therefore the weight values will still be changing even though the neurons might have reached their saturation levels.

So the output data are normalised between 0.1 & 0.9, and –0.9 & 0.9 for the sigmoid and tanh function respectively. This allows an improvement in the performance of the Backpropagation learning algorithm and leads to a faster and more efficient learning process for the neural network. To analyse this theory, experimental tests were carried out on two applications, namely function approximation and a gear optimisation problem. The training and generalisation errors of these two applications will be investigated.

5.5 Experimental results

As mentioned before, the theory was tested on two applications, i.e. function approximation and a gear optimisation problem.

5.5.1 Function Approximation

The function that was used for this test is defined as (Engelbrecht et al, 1999):

$$f(x) = \sin(2\Pi x) * \exp(-x) + Noise(0,0.1)$$

where x is in the range [-1, 1] and the *Noise* is a random value between 0.0 and 0.1. It is a sinus-line function with noise added to it. It is a non-linear function and is difficult for AI to estimate. The network size of 1:10:1 was used for the test. 10 hidden nodes were found to give the best result. This value was also used by Engelbrecht et al (1999). The output values were scaled in the range [0, 1] for sigmoid activation function and [-1, 1] for tanh activation function.

Act. Func.	NN	εt	εg	New εt	New εg	%Imp εt	%Imp εg
Sigmoid	1-10-1	0.022	0.024	0.011	0.014	50 %	41.67 %
Tanh	1-10-1	0.020	0.028	0.017	0.025	15 %	11 %

Table 5-1 Test result for function approximation

Table 5-1 summarizes the result of the test carried out for the sigmoid and tanh activation function. εt and εg are the training and generalisation errors respectively. New εt and New εg are the errors after the application of the proposed theory and %Imp εt and %Imp εg indicate the percentage improvement to the training and generalisation errors using the new approach.

As shown in the table, in both cases the new network resulted in a better training and generalisation errors. For sigmoid activation function, the errors have been reduced approximately by half, which is an outstanding result. There is a 50% improvement in the training error and 41.67% improvement in the generalisation error for the sigmoid activation function. Tanh activation function shows an improvement as well, even though it is less than sigmoid. The training error has been reduced by 15% and generalisation by 11%.

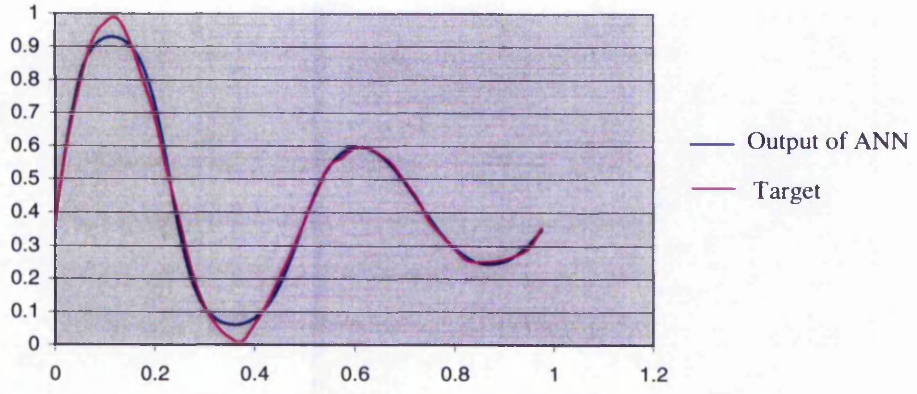


Figure 5-3 ANN training result for the original network for sigmoid activation function

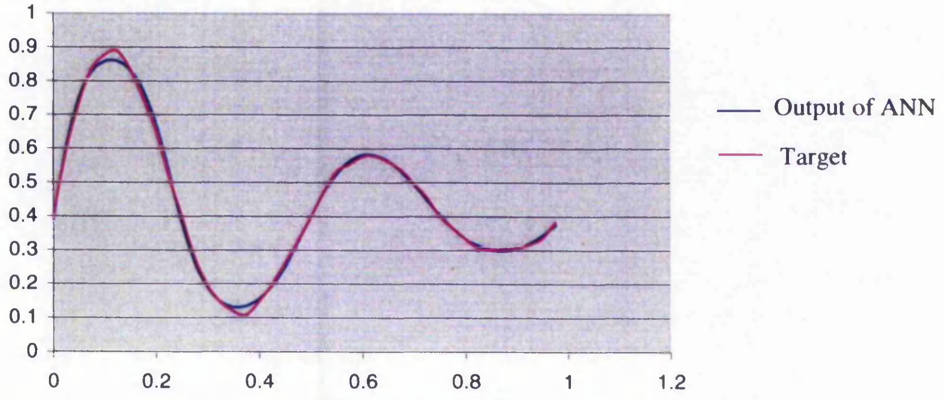


Figure 5-4 ANN training result for the new network for sigmoid activation function

Figures 5-3 and 5-4 show this effect graphically for the sigmoid activation function. The graphs of the ANN training results were plotted for the two dimensional function approximation problem. Figure 5-3 displays the graph for the original network and figure 5-4 displays the graph for the new network with re-scaled output data. There is an obvious improvement, where the outputs become close to maximum. Scaling the output to 0.9, brings the target closer to the output. Similar trend exists where the outputs get closer to minimum.

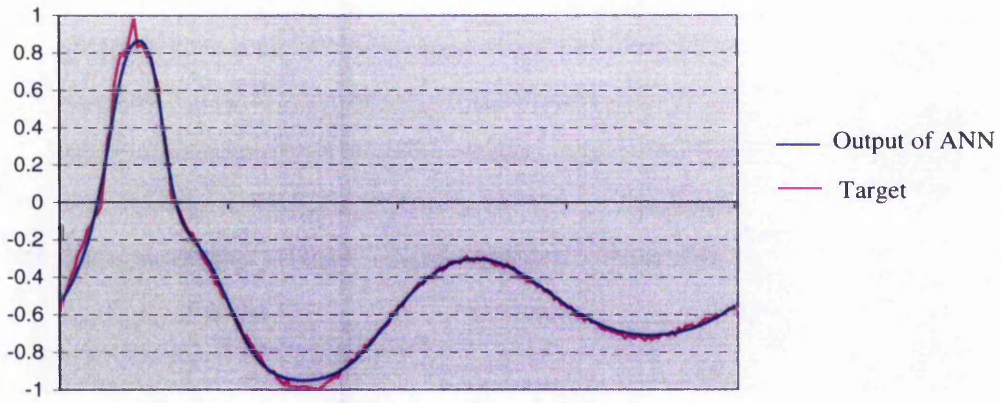


Figure 5-5 ANN training result for the original network for tanh activation function

Figures 5-5 and 5-6 show this effect for the *tanh* activation function. Figure 5-5 represents the original network where the output data were normalised between –1 and 1; and figure 5-6 displays the new network where the output data were normalised between –0.9 and 0.9. The improvement is seen in the latter figure where the target is brought closer to the output.

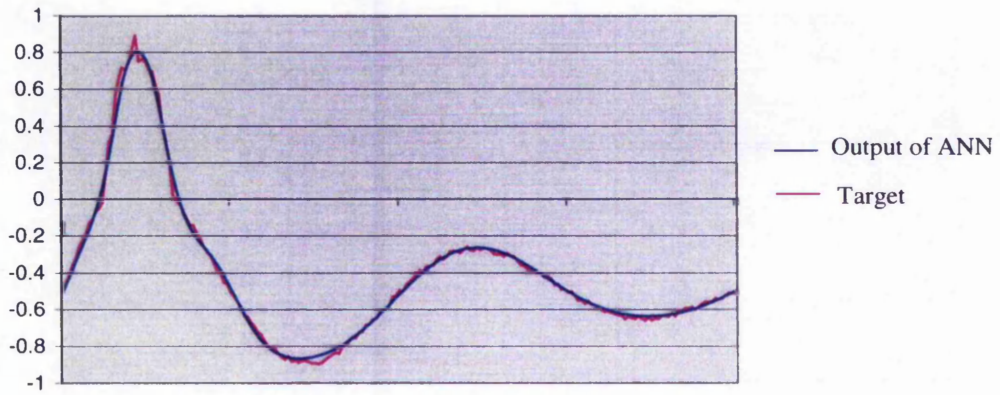


Figure 5-6 ANN training result for the new network for tanh activation function

5.5.2 Gear optimisation

The method was applied to the problem of gear design optimisation described in chapter 4. The result can be seen in table 5-2. The value of *Weight file No.* implies the name of the weight file used, e.g. 3 means that the file weight3.dat is read, i.e. when the module is 6 and pressure angle is 22.5. Refer to table 4-4 and appendix B for the detail about the condition, the pattern and type of ANN used.

Weight file No.	**εt**	**εg**	**New εt**	**New εg**	**%Imp εt**	**%Imp εg**
3	0.038	0.049	0.032	0.040	15.8 %	18.4 %
2	0.038	0.051	0.035	0.035	7.9 %	31.4 %
1	0.041	0.047	0.038	0.029	7.3 %	38.3 %
41a	0.044	0.076	0.039	0.061	11.4 %	19.7 %
Average					10.6 %	29.95 %

Table 5-2 Test result for gear design optimisation

As the table illustrates the result of the test on gear design optimisation is satisfactory. It shows a notable improvement in both training and generalisation errors. The average improvement for the training error for these tests is 10.6 % and for generalisation error is 26.95 %, which is an outstanding result specially for the generalisation error.

Another important point which was observed during test on the gear optimisation problem was that the new network was learning much faster than the original one. As an example, one of the ANNs required 40000 iterations for the original network to reach the solution, whereas for the new network, only 5000 iterations were needed. To further analyse this, a fixed value was chosen for the final error to see how many iterations both the network require in order to reach that fixed value. It was tested for the final value of 0.049 for the gear optimisation problem and the result was outstanding. The original network needed 19,440 number of iterations, whereas the new network reached the final value only in 340 iterations. Similar test was carried out for the function approximation problem and it followed the same pattern.

5.6 What is the optimum solution?

In the previous sections, the result of normalising the output data between 0.1 and 0.9 for the sigmoid activation function and –0.9 and 0.9 for the tanh activation function were presented, which proved to be successful. However the question is whether these are the optimum solutions? What effect would it make if the output data are normalised between 0.05 and 0.95 or 0.15 and 0.85. These were tested on both function approximation and the gear optimisation problem. The results of the tests are shown in tables 5-3 and 5-4.

Act. Func.	**Scale**	**εt**	**εg**	**%Imp εt**	**%Imp εg**
Sigmoid	0.0 → 1.0	0.022	0.024	—	—
"	0.05→0.95	0.016	0.018	27.3 %	25 %
"	0.1 → 0.9	0.011	0.014	50 %	41.67 %
"	0.15→0.85	0.018	0.024	18.2 %	0 %
Tanh	-0.1 →1.0	0.020	0.028	—	—
"	-0.95→0.95	0.017	0.027	15 %	3.6 %
"	-0.9 → 0.9	0.017	0.025	15 %	11 %
"	-0.85→0.85	0.012	0.017	40 %	39.3 %

Table 5-3 Test result for different scaling for function approximation

Figure 5-3 shows the result of the test carried out on function approximation for differenent scales of normalisation. The result shows an improvement of 27.3 % and 25% for the training and generalisation errors respectively for the sigmoid activation function, when the output data are normalised between 0.05 and 0.95. The improvement is higher for 0.1-0.9 and decreases when normalised between 0.15 and 0.85. Therefore, it proves that the optimum solution is to normalise it between 0.1 and 0.9. This might be due to the fact that data is scaled too far (i.e. squashed to a very small range) where useful information within the dataset is degraded which might lead to difficulty for ANN to learn the patterns. This however might depend on different applications as for the gear optimisation problem, ANN still continues learning for lower scaling factors (i.e. see Table 5-4).

The result is however different for tanh activation function. The error keeps reducing as the scale descreases. As the table shows, the optimum solution is achieved when the output data are normalised between –0.85 and 0.85. In this case, the training and generalisation errors are improved by 40% and 39.3% respectively. As can be seen for the tanh function, the data sacling can be reduced further than that for sigmoid function. The reason might lie in the fact that for tanh function the data is spread further (between –1.0 and 1.0) compared to the sigmoid function (between 0.0 and 1.0) and when the scaling is reduced, it still posses useful information for ANN to learn properly.

Table 5-4 displays the result of the tests carried out on one of the cases of the gear design optimisation problem. The result shows a dramatic improvement in both the training and specially the generalisation errors as the scale of the normalisation is reduced. The training and generalisation errors are reduced to 0.035 and 0.045 respectively when the output data are normalised between 0.15 and 0.85. These errors are further reduced when normalised between 0.2 and 0.8.

Scale	εt	εg	%Improvement εt	%Improvement εg
0.0 →1.0	0.044	0.076	—	—
0.05→0.95	0.044	0.062	0 %	
0.1 → 0.9	0.039	0.061	11.4 %	19.7 %
0.15→0.85	0.035	0.045	20.5 %	40.8 %
0.2 →0.8	0.029	0.044	34.1 %	42.1 %

Table 5-4 Test result for different scaling for gear optimisation

Even though normalising between 0.2 and 0.8 shows the best improvement, this might not be feasible. This is because if the range of the scale becomes too small, it might lose some vital information during the de-normalisation. The same situation applies for 0.15-0.85, as well as –0.85 and 0.85 for tanh activation function shown in table 5-3. The point is that the potential is there, however if needs to be applied to the real problem, the result of the denormalised data need to be checked as well to ensure that no vital information is lost. This however does not exist when de-normalised between 0.1 and 0.9, because only a slight reduction has been made to the range of the normalisation scale.

5.7 Concluding remarks

This chapter dealt with the problem of weight adaptation for the situation when the output data reaches the saturation level (0 and 1 for sigmoid and –1 and 1 for tanh). The nature of the weight adaptation equation causes insignificant change in the value of the weights in such conditions. This problem was addressed by reducing the range of the output data to lie between, but not including, the full range of the activation function. The range of 0.1 to 0.9 was tested for the sigmoid function, and –0.9 and 0.9 for the *tanh* function. Function approximation and gear optimisation were used for testing the proposed method. Experimental results illustrates that the proposed method is efficient, resulting in an improvement in the training and generalisation errors. It also speeded up the rate of learning of the neural network.

The second point is particularly important if applied to the method described in chapter 4 in which ANN is used to replace GA tier 1 in gear design optimisation program. Preparing the training data for ANN is time consuming since in order to get the targets, gear optimisation should be run for each pattern. Therefore, speeding up the process of learning will improve the overall time to a great extent.

Tests were also carried out for different range of normalization scales, such as range of 0.05-0.95 and 0.15-0.85 for the sigmoid function; and –0.95&0.95 and –0.85&0.85 for the *tanh* function. Even though reducing the range showed improvement for *tanh* in function approximation and also the gear optimisation problem, it was discussed that important information might be lost if the range is reduced too much. If such small range needs to be used, then the de-normalised data need to be checked thoroughly to ensure that it does not cause any negative effect on the final result.

The next two chapters are concerned with the development of an online electronic catalogue for a gear manufacturing company. The development of the database and the process of remote connection is described in chapter 6. Chapter 7 presents a methodology that improves the feature of the electronic catalogue by creating the dynamic drawing of the gears in the form of DXF files.

Chapter 6

Internet Techniques for Online Gear Catalogue

6.1 Introduction

Before hypermedia was introduced into the engineering catalogues, manufacturing companies, designers and customers depended mainly on the catalogues in the form of a hard copy. There are a number of problems associated with the traditional hard copy catalogues (Crowder et al, 2000):

- Copies of the catalogue need to be distributed among different teams within an organisation. As these teams are often in different locations, it is often necessary to physically follow the catalogue around a factory. This is an added burden to the process and increases the time taken to complete any request. In addition, the process could end with the catalogue being mislaid or lost.
- Moving the catalogue from one location to another is inefficient and slows the process down. Many of the required activities are independent and can thus be performed in parallel.
- There is normally no absolute guarantee that the information in hand is up-to-date, since with a paper-based catalogue, the issuing of updates and the supporting modification notes cannot be fully enforced. There is well-documented evidence that paper based systems will deteriorate in accuracy at a rate of 20-30% per year, and will ultimately be inaccurate enough to be of a value (Crowder et al, 2000).

Many companies in engineering-manufacturing industry find their efficiency severely downgraded by paper-based systems. The manufacturers of standard engineering components still deliver the majority of information to designers in the form of printed catalogues. These are delivered to potential users in response to an enquiry, or automatically via a mailing list. Large sums are spent on the dissemination of this information (Culley, 1998).

Over the last few years, the age of the electronic catalogue has clearly emerged. The advent of the widespread availability of CD-ROM technology means that useful packages of information can be delivered cost effectively to a wide variety of customers. Some of the advantages of such systems are listed below:

- Large amount of information can be stored in CD-ROM.
- The compact size of CD-ROM also makes its transfer much easier.
- Electronic catalogues speed up the searching and selection of a particular component.
- There is a much improved likelihood of identifying the best available component in terms of technical, cost and availability.

Although this kind of offline electronic catalogue addresses the problems associated with the traditional printed catalogue and saves time in design process, it still does not eliminate one problem, i.e. the information given to the user might still be obsolete. Another problem is that the catalogue is platform dependent. This would be a disadvantage, since there is no guarantee that the customer is using the same platform, or even various teams within an organisation could be working on different platforms.

The optimum solution is to make the electronic catalogue available online. This means that the user has the access to the up-to-date information at any time. It will also eliminate the need for designing different application interfaces across different platforms.

As part of this research work, a prototype online-electronic catalogue is developed for davall gear manufacturing company (Davall, 2000). The data is stored in an Access database. The appropriate table of data is retrieved and is displayed on the clients' machine, in a scrollable tabular format. A technique is also proposed that enables the designers to create the CAD drawing files of the selected gear dynamically. The drawing is created in the form of DXF format. Adding this feature makes the electronic catalogue '*assertive*'. As the literature review showed, most of the available design catalogues are '*passive*.' That is they only transfer the information, whereas in the proposed catalogue, new drawings are created dynamically.

The structure of the system is shown in figure 6-1. It is based on a combination of Java applet, JavaScript, Java Database Connectivity (JDBC) and Java Sevlet. The Microsoft access database that contains the gear dimensions is located on the server. JDBC, Java servlet and the C++ executable programs that create the DXF drawings are all located on the server as shown in figure 6-1. JDBC is used to make the connection to the database and to retrieve the appropriate data and Java servlet is used for communication between the client and server for the creation of DXF drawing files. HTML and Java Applet are used for graphical user interface; and are run and interpreted locally.

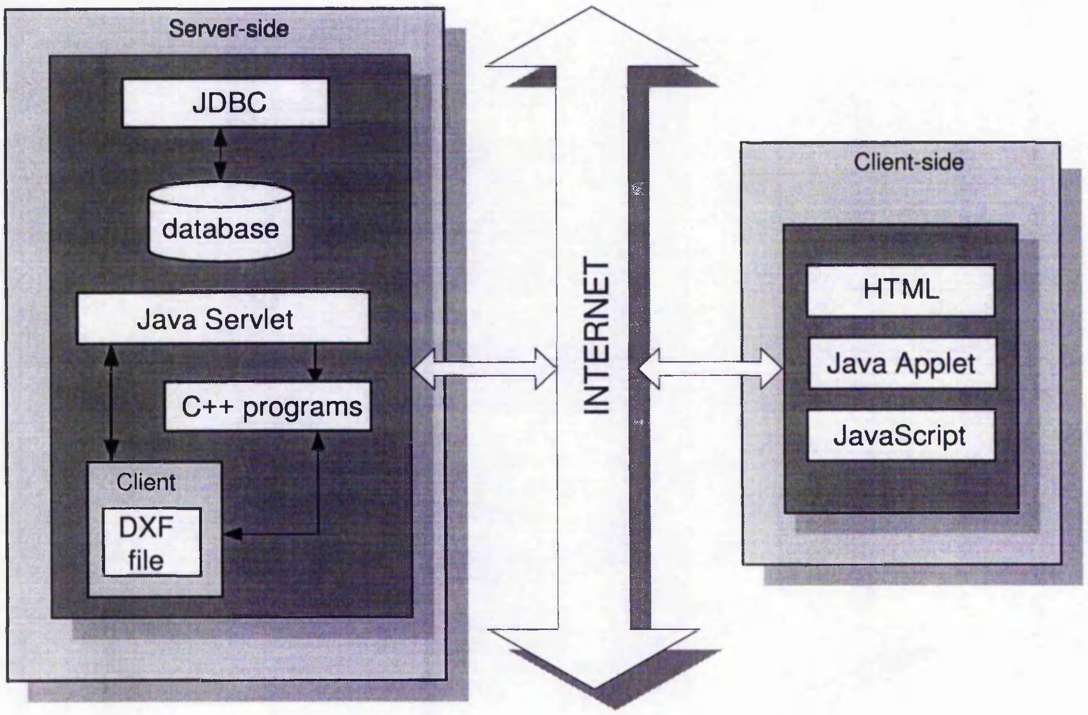
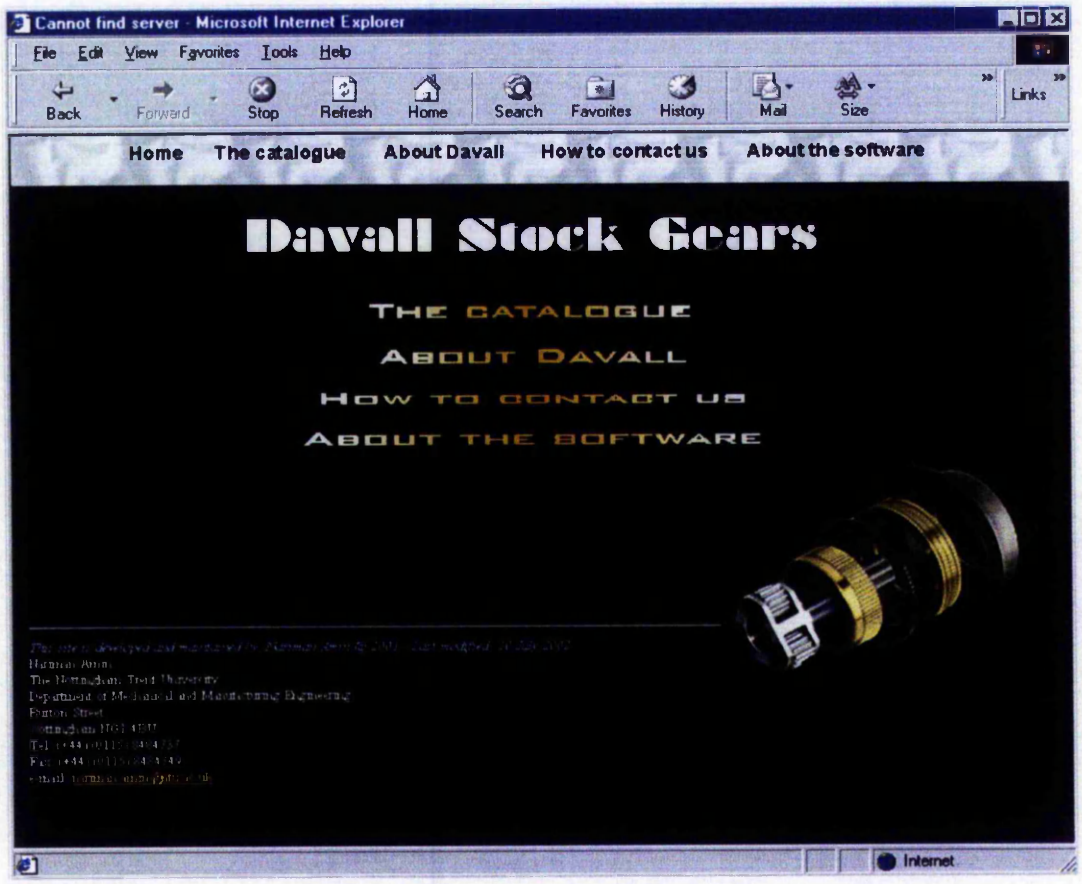


Figure 6-1 Structure of online gear catalogue

Figure 6-2 shows the first page of the online gear catalogue. The texts are hyper linked to their relevant pages. Moving the mouse over them will change their colour, this is achieved using JavaScript. *The Catalogue* opens the main page where the database applet is located. *About Davall* gives a brief description of the company. *How to contact us* contains the address of the company and *About the Software* provides a brief description of the online gear optimisation catalogue. A horizontal frame has also been used as can be seen at the top of the page. This helps the user to navigate through the pages easily.


Cannot find server - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Size
Links
Home The catalogue About Davall How to contact us About the software
Davall Stock Gears
THE CATALOGUE
ABOUT DAVALL
HOW TO CONTACT US
ABOUT THE SOFTWARE
Internet

[illegible]
Hamzah Arum
The Nottingham Trent University
Department of Mechanical and Manufacturing Engineering
Burton Street
Nottingham NG1 4BU
Tel [illegible]
Fax [illegible]
e-mail [illegible]

Figure 6-2 First page of the online gear catalogue

In the following sections the development of the electronic catalogue using relative web techniques will be discussed. The dynamic creation of the DXF files will be presented in chapter 7.

6.2 Java Database Connectivity

Many product design and manufacture applications extensively involve database management. Relational databases are widely used for their well-established and standardised SQL (Structured Query Language). SQL is a language for use with relational databases. Although SQL has been evolving since the early 1980s, it was in 1990 that the SQL Access Group defined the Call Level Interface (CLI) as a standard for accessing databases. To implement it, a driver is needed that can translate a CLI call into the language used to access a particular database (Cornelius, 1998).

There is a wide selection of commercial relational database management systems (RDBMS). For example, Microsoft SQL server, ORACLE and SYBASE are typical RDBMS with a client-server architecture. Microsoft access, FoxPro, etc. are widely used in practice. Most of such popular databases are ODBC compliant (Huang et al, 2001). ODBC is an Application Programming Interface for Microsoft windows that implements an extended version of the CLI. It was first released in 1992. It is an open standard that provides a common set of application interface calls to manipulate databases. Application developers can write applications that make ODBC calls and work with many databases, instead of writing programs specifically for a particular database using its native database APIs. The use of ODBC as a data access API continues to be an effective data access strategy, offering database transparency and a common development API. This can reduce overall development time and allows the use of common applications, either developed in-house or purchased from one of the massive number of vendors offering ODBC-compliant applications, tools and utilities (Gupwell, 1999).

One method of accessing a remote database is the use of Common Gateway Interface (CGI). CGI has been used for this purpose by many developers and performed relatively well, until Java stepped into the market. A major drawback of CGI is that a process needs to be created for every database access request. Creating a process is time-consuming and expensive in terms of the server's main memory and can also exhaust resources available to server applications. A description of the behaviour of CGI was given in section 2.4.2.1.

Java could overcome the problem associated with CGI for database access. However, Java cannot directly make connection to the database. ODBC API is written for the programming language C. Although the Java Native Interface (JNI) allows a Java program to call a C function, calling C code is not usually possible from a Java applet because of security restrictions. So, instead Java has its own API for submitting SQL statements to a database server. This is called Java DataBase Connectivity (JDBC) (Cornelius, 1998).

JDBC inherits all the characteristics of Java. The advantages of Java for web applications were discussed in section 2.4.2.2. JDBC provides an interface between the client and the database server. It is a Java API for accessing virtually any kind of tabular data. It consists of a set of Java classes and interfaces that provide a standard API for database developers and makes it possible to write industrial strength database applications using an all-Java API.

Two major layers make up JDBC: the JDBC API and the JDBC Driver API. JDBC API consists of a set of classes and interfaces written in Java that enable the client to make a connection with the database, send the SQL statement and retrieve the result. It makes it possible for the developers to access database services using standard Java mechanism (Linthicum, 1998). The basic components (classes or interfaces) are (Quan et al, 1999):

1. *DriverManager*, a static class capable of managing multiple JDBC drivers in a Java Virtual Machine execution environment.
2. *Driver*, an interface for all JDBC drivers capable of accessing one or more database systems.
3. *Connection*, an abstraction of a client/server database connection.
4. *Statement*, an abstraction of a database query or operation.
5. *ResultSet*, an abstraction of the database query results
6. *DatabaseMetaData*, an abstraction of the information about database.
7. *ResultSetMetaData*, an interface to the information of the result set itself.
8. A few other classes and interfaces that represent other aspects of a database system.

6.3 Selection of a JDBC driver for the project

The driver manager can support multiple drivers connecting to different databases. JDBC drivers are a group of Java classes, which act as the interfaces between the

JDBC and the database. The JDBC passes the programmer's SQL to the database via the driver. A JDBC driver can either be written entirely in Java so that it can be downloaded as part of an applet or it can be implemented using native methods to bridge to existing database access libraries. There are four types of JDBC drivers, which are explained below.

6.3.1 JDBC-ODBC bridge driver

The JDBC-ODBC bridge driver was developed by JavaSoft to take advantage of the large number of ODBC enabled data sources. It is also known as type 1 driver. The driver translates JDBC calls into ODBC calls, which is then processed by an ODBC driver. The application database resides in the client machine and the client component accesses the database locally. An empty database template is made downloadable from a web page. The user may download it to its client machine for future use. Some ODBC binary code, and in many cases database client code, must be loaded on each client machine that uses this driver. Consequently, this kind of driver is at most appropriate on a corporate network where client installations are not a major problem. Figure 6-3 shows the structure of the JDBC-ODBC bridge driver.

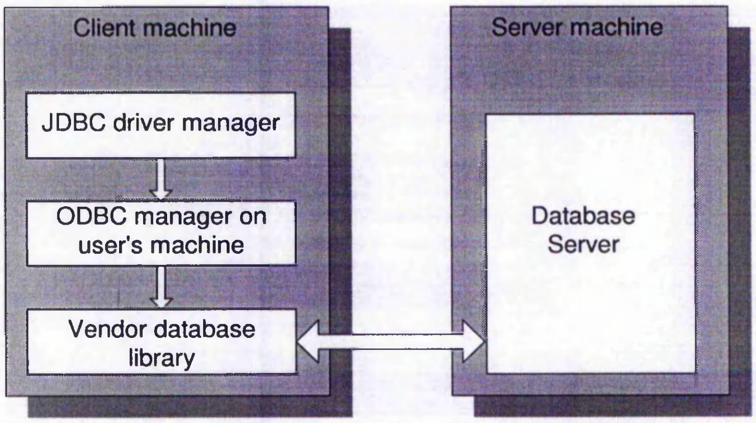


Figure 6-3 Structure of ODBC-JDBC bridge driver

6.3.2 Native-API partly Java driver

This is type 2 driver, based on a two-tier model. In a two-tier model, the connection is built directly between the client and the remote database. The Native-API partly

Java driver connects the client to the database by using vendor-supplied libraries. These drivers are typically written in some combination of Java and C/C++, as the driver must use a layer of C program in order to make calls to the vendor libraries, which are written in C. This kind of drivers translates JDBC calls into the native API calls of the DBMS to be connected. However, same as JDBC-ODBC bridge driver, some binary codes (the vendor library) need to be loaded on each client machine. Figure 6-4 shows the structure of this type of driver.

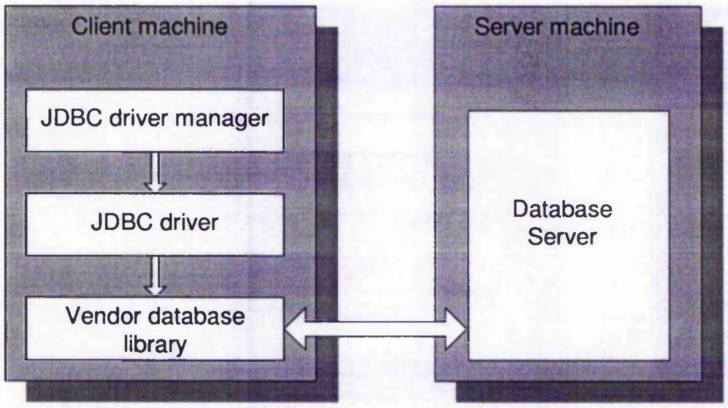


Figure 6-4 Structure of Native-API partly Java driver

6.3.3 Network-protocol, pure Java driver

This driver is known as type 3 driver. It is a three-tier connection between application client, server and DBMS server. In a three-tier model, a connection is built between the application server and the remote database server; and the client obtains data from the application server. It is the middle tier of application server objects that deals with all database access operations. Application clients invoke application server objects that in turn invoke the DBMS servers. Network-protocol, pure Java driver translates JDBC calls into a database-independent network protocol, which is then translated into a database-specific protocol by a middle-tier server. This driver can be written entirely in Java and can provide just-in-time delivery of Java applets. The structure of type 3 is shown in figure 6-5.

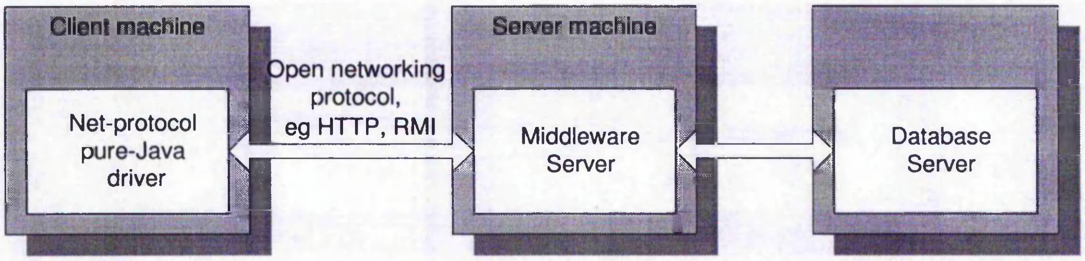

Client machine
Net-protocol pure-Java driver
Open networking protocol, eg HTTP, RMI
Server machine
Middleware Server
Database Server

Figure 6-5 Structure of Network-protocol, pure Java driver

6.3.4 Native-protocol pure Java driver

This is type 4 driver, based on two-tier-plus connection between application client, web server and DBMS server. In two-tier-plus connection, there is a compromise between a two-tier and three-tier connection. In this case, a connection is built between the ODBC/JDBC middleware and the remote database server, and the client obtains the data from the ODBC/JDBC middleware. On one hand, this model is similar to the three-tier model because there is a middle tier- the ODBC/JDBC driver manager that resides on the server-side. On the other hand, this two-tier-plus model is similar to the two-tier model in that it is the client that invokes the normal ODBC/JDBC calls. This is the clear difference from the three-tier model where the server object makes such calls. Invocation calls from the client are then transmitted to the ODBC/JDBC driver manager.

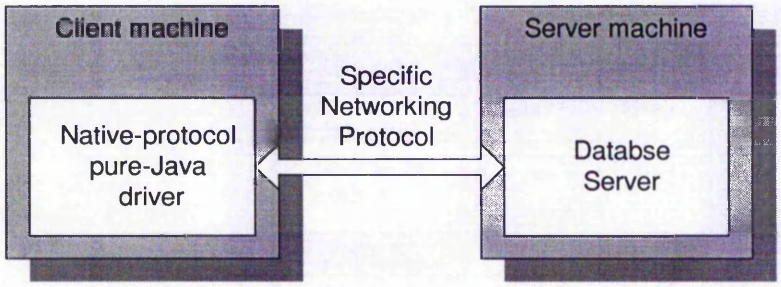

Client machine
Native-protocol pure-Java driver
Specific Networking Protocol
Server machine
Databse Server

Figure 6-6 Structure of Native-protocol pure Java driver

The Native-protocol pure Java driver translates JDBC calls directly into the network protocol used by the specific database vendor. Thus, this driver architecture provides driver calls from the client to the database server. This driver is written entirely in Java. Since these drivers translate JDBC directly into the native protocol, they can provide for very high performance database access. However, it is a 'fat' client approach, as the bulk of the application logic runs on the client. Therefore it is most popular for intranet access. The structure of the Native-protocol pure Java driver is shown in figure 6-6.

6.3.5 What type of driver?

Now the question is which type of driver is most suitable for our application? Table 6-1 illustrates the advantages and disadvantages of each driver (Yang et al, 1998). The first two drivers could be disregarded by the first glance. The reason is the fact that ODBC/JDBC drivers must be installed and properly configured on the client machine. Although this is straightforward, a user may not be aware of this requirement or unwilling to do this before using the application. No client configuration is required for type 3 and 4. Type 3 has proved to be more suitable for an Internet environment. Result of tests on these drivers, carried out by Sood (1998) was discussed in section 2.4.2.2. The large download time of type 4 makes it unsuitable for Internet environment. Thus type 3 was chosen for the purpose of the online gear catalogue.

JDBC Drivers	**Pros**	**Cons**
Type 1: JDBC-ODBC bridge driver	- Integrated into JDK 1.1	- Requires ODBC manager on the client - The ODBC driver on each of the clients needs to be configured
Type 2: Native-API partly Java driver	- Allows the programmer to fully utilise the speed and power that comes from the use of the API specifically developed for the DBMS	- Driver is DBMS-dependent. - Requires a vendor-supplied DLL (Dynamic Link Library) to be installed in the client's JAVA library path.
Type 3: Network-protocol, pure Java driver	- DBMS-independent: Allows the most flexible multi-server configuration – e.g. supports features such as cashing connection and query results - No configuration required on client's machine	- Requires a vendor-supplied intermediate server. - Configuration of the intermediate server
Type 4: Native-protocol pure Java driver	- Allows a direct call from the client to the database, without the need of client pre-configuration.	- Need to load a different driver for each DBMS that it needs to access. - Fat client

Table 6-1 Comparisons of the 4 types of JDBC drivers

6.4 JDBC configuration on the server

The data, i.e. dimension of the gears with different number of teeth, are saved in MS Access database, in different tables depending on the module and the material. The database is located on the server and the users will make a connection to this database and retrieve the appropriate table. To be able to use JDBC, three steps needs to be performed: Install JSDK2.1, Configure the database on the server, and Install a JDBC driver, which are further explained below.

- **Install Java 2 SDK, Standard Edition**

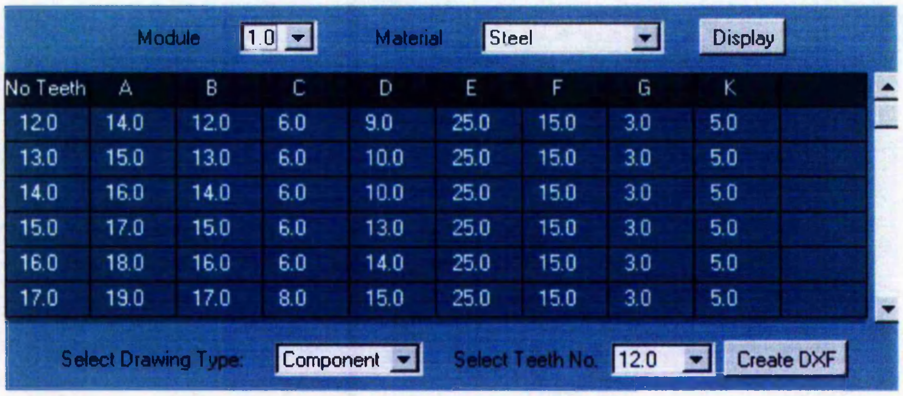
  Java platform needs to be downloaded and installed, which includes the core JDBC API. It includes the JDBC 2.0 core API, the package java.sql [White et al, 1999].

- **Configure the database on the server**

  The second requirement is to create an ODBC data source for the database and register it with the operating system (ODBC set-up, accessed August 2000). ODBC is an API that is able to access data using SQL commands. So an ODBC DSN (Open Database Connectivity/Data Source Name) needs to be set up for the database. Without this, JDBC would not be able to establish a connection with the database. This needs to be done only once on the server's machine, no ODBC configurations required on each client's machine. Appendix D describes the procedure for setting up an ODBC DSN data source for a database.

- **Install a JDBC driver**

  The final step is to install a JDBC driver that supports MS Access. ThinAccess from ThinWeb company was selected for the task (ThinAccess, accessed September 2000). ThinAccess is a JDBC driver of type 3 and can connect to any ODBC data source, such as MS Access. ThinAccess driver has a small footprint, compared to the native Type 4 drivers, so the time taken for downloading the driver is reduced. It also performs caching on the client and server side which results in enhanced database access speed (Sun-servlet, 1998).


Module 1.0 Material Steel

No Teeth	A	B	C	D	E	F	G	K
12.0	14.0	12.0	6.0	9.0	25.0	15.0	3.0	5.0
13.0	15.0	13.0	6.0	10.0	25.0	15.0	3.0	5.0
14.0	16.0	14.0	6.0	10.0	25.0	15.0	3.0	5.0
15.0	17.0	15.0	6.0	13.0	25.0	15.0	3.0	5.0
16.0	18.0	16.0	6.0	14.0	25.0	15.0	3.0	5.0
17.0	19.0	17.0	8.0	15.0	25.0	15.0	3.0	5.0

Select Drawing Type: Component Select Teeth No 12.0

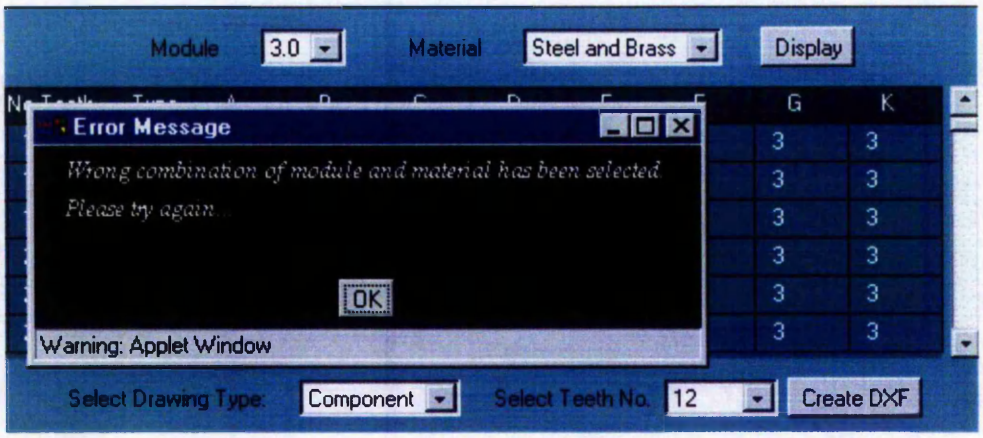
Figure 6-7 Gear catalogue applet

6.5 Establishing a connection with the database

Figure 6-7 shows the applet that has been developed, which provides the graphical user interface for obtaining the required parameters from the user and for displaying the data. In order to retrieve the data, the values of *Module* and *Material* needs to be selected and by pressing the *Display* button, the data will be displayed in the grid. *Module* and *Material* determine the appropriate table of data to be retrieved. In this case, the data is displayed for the situation when module is 1.0 and material is Steel. As can be seen in the figure, the user has the option of creating DXF drawings. This is done by selecting the drawing type and number of teeth. Clicking the *Create DXF* button will then generate the drawing in DXF format which the user can download. This will be explained in detail in the next chapter.

By clicking on the Display button, before making a connection to the database, the applet would check if the selected values for module and material are valid. This is because data does not exist for all the combination of the module and material.

Figure 6-8 shows the situation when the user has selected a non-available table. The applet detects this and prevents making connection to the database. A message is displayed as shown in the figure, informing the user of the error. It is very important that this error is caught at local level. It will speed up the process enormously. This reduces the task of the server and also the number of client/server interaction in case of a possible error. If correct values for module and material have been selected, a connection is then made with the MS Access database located on the server, via the JDBC driver.



Wrong combination of module and material has been selected
Please try again...

Figure 6-8 Applet detects the error and informs the user

In order to connect to the database, some initialisation needs to be performed. A JDBC driver manager is provided as part of the JDBC API. It is necessary to register a driver with the driver manager. JDBC driver needs to be loaded by the Java Virtual Machine and the application needs to check that the driver was successfully loaded. The driver can be registered programmatically by requesting the class of the driver to be loaded using the static method *forName()* in the class '*class*'. This will cause the driver to be loaded into the running application. The code that loads the thinAccess driver called '*ThinDriver*' is shown below:

```
Class.forName("com.thinweb.sql20.jdbc20.ThinDriver").newInstance();
```

It is possible to install more than one driver in the JDBC program. When a request is made to make a connection to a database, each one will be tried in turn until one accepts that request. However, using more than one driver will slow down both system startup (as each must be loaded) as well as the runtime (since each may need to be tried in turn). For this reason, it is recommended to select the most appropriate driver only. For the purpose of the task at hand, one driver is sufficient since a single database is used.

Once the JDBC driver is registered, a connection to the SQL server can be built by sending the message *getConnection(url)* to the object called DriverManager :

Connection con = DriverManager.getConnection (URL);

The string specifying the database to connect to is informed from a JDBC URL. The URL is comprised of three parts:

- The JDBC protocol indicator (*jdbc:*)
- The appropriate subprotocol such as (*odbc:*)
- The driver-specific components (name of the connection within *thinAccess*)

URLs are used because the Java program accessing the database is an applet that needs to connect to the database via the Web. Once a connection has successfully been made to the database, the program closes that connection.

The next step involves creating a statement object to execute queries. A statement object is responsible for sending the SQL statement, and returning a set of results from the query. The following code creates the statement object using *con,* the instance of the connection object:

Statement stmt = con.createStatement();

In the development of online gear catalogue, the connection was made to be pre-configured. This is very important and results in an increase in performance, since the connections are pre-created or configured out of band, as opposed to being created in real time. It avoids having to perform a number of performance intensive operations in real time, such as loading a driver and creating the connection. The same connection can be referenced by many clients simultaneously. In addition to the connection, the statements, such as the queries, are pre-configured. This allows users to share statements across multiple clients and avoid redundant queries (Sun-servlet, 1998).

Pre-configuring the connection and statements is done via thinAccess driver administrator. For pre-configuring the connection, a name and the URL are defined for the connection. For pre-configuring the statement, a name is defined for that particular statement, along with the name of the pre-configured connection and the SQL query. Therefore, the applet would call the *Statement* instead of the *Connectio*n. As an example, for the first table in the gear database, the name *sheet1statement* has been assigned to it. The following SQL command is also defined under that statement:

*SELECT * FROM SHEET1*

Where sheet1 is the name given to the first table within the gear database. Whenever the sheet1statement is invoked, it automatically executes the above query, resulting in retrieving all the data within that particular table. The full description of the process of pre-configuring the connection and statements is given in appendix E.

Figure 6-9 shows the codes developed for the connection to the gear database remotely.

```
Line No.    Code

   1        Properties props = new Properties();
   2        Class.forName("com.thinweb.sql20.jdbc20.ThinDriver").newInstance();
   3        com.thinweb.sql20.jdbc20.Statement20 stmt = new
                                    com.thinweb.sql20.jdbc20.Statement20
                                    ("jdbc:thinweb:" + host + table, props);

   4        ResultSet rs = stmt.executeQuery();
   5        ResultSetMetaData rsmd = rs.getMetaData();
   6        numberOfColumns = rsmd.getColumnCount(); // get number of columns


            // get the label of the columns
   7        for (int i=0; i<numberOfColumns; i++)
   8                colLabel[i] = rsmd.getColumnLabel(i+1);

                // Retrieve the data
   9            while (rs.next())
                {
  10                    for (int i=1; i<=(numberOfColumns); i++)
  11                            vData.addElement(rs.getString(i));
                }

  12        stmt.close();
  13        rs.close();
```

Figure 6-9 The code for database connection

To make JDBC calls, the package java.sql.* needs to be imported into the program which is not shown in the figure 6-9. By sending a messagbe *forName()* to the class called 'C*lass'*, an instance of the specified JDBC driver (i.e. ThinDriver), is created and registered, as shown in line 2 in figure 6-9. Once the JDBC driver is loaded, a connection to the SQL server can be built by creating an instance of the *Statement* class to execute a query operation. The JAVA applet builds such a connection by supplying the URL for the target SQL server. This is shown in line 3 where *host* refers to the path to the server, *table* contains the name for the pre-configured statement and *props* is an instance of *Properties* class.

Statement class is used to retrieve the information from the data source. It contains the *executeQuery()* mehtod which implements the SQL access operation and returns a *ResultSet* class. The *ResultSet* class is used to get the results of the pre-configured statement via the connection. It is linked to the data source via the *Statement* class, refer to line 4.

The result sent back from the SQL server are then processed by the applet using appropriate looping commands to process the data one row at a time. The *while(rs.next())* expression obtains all results for the query operation. The *next* method of the instance *rs* of *ResultSet* class is used to fetch each row one by one, see line 9. The retrieved data are then saved in the *inData* vector, as shown in line 11. After the query operation is finished, the connections of the *ResultSet* and the *Statement* classes are closed, lines 12 and 13.

In this program the *ResultSetMetaData* object has also been utilised. This object provides information about a particular *ResultSet* instance. As shown in line 5 of figure 6-9, an instance of *ResultSetMetaData* is created that makes a call to *getMetaData()* method of the *ResultSet*. The *ResultSetMetaData* object *rsmd* contains metadata information about the columns of the *ResultSet rs*, so *rsmd* is used to invoke *ResultSetMetaData* method to access this information. Then the method *getColumnCount()* is called which retrieves the number of columns in that particular table (line 6). Not all the tables have the same number of columns. Therefore this information is vital for the following reasons:

- For displaying the table of data properly.
- Not all the columns retrieved from the table are displayed. The user does not need these data. They are only needed for the creation of the DXF drawings.

The function *getColumnLabel()* is also used to retrieved the label of the columns, refer to lines 7 and 8. These will be displayed as the headings of the columns on the applet.

In addition, the mechanism for catching any exception is considered throughout the whole program. The driver is loaded and the connection is made within a *try{} catch{}* block. This is because both operations can raise exceptions and these must be caught and handled (since they are not runtime exceptions). The *forName()* method raises the *ClassNotFoundException* if it cannot find the class that represents the specified driver. If the program cannot find the specified database, then the *SQLException* is raised. The *try{} catch{}* block works by trapping any exceptions raised in the *try* part within the *catch* part (assuming that the exception raised is an instance of the specified class of exception or one of its subclasses) (Guan, 1998).

6.6 The applet

The applet is produced using two JAVA files, *scrollerGrid.java* and *database.java.* The former handles the graphical user interface, laying out the retrieved data and obtaining and evaluating user's commands. The latter is a class that performs the database connection only. The code listing shown in figure 6-9 is definded within this class. An instance of *database* class is created, whenever the user clicks on the *Display* button, after the values of Module and Material have been validated as explained earlier.

The structure of the *database* class was explained earlier, which is shown in the form of flow diagram in figure 6-10. When *scrollerGrid* invokes the *database* class, it passes two arguments to it, path of the host server and the connection name for pre-configured statement. It will then use these to make the connection as formerly discussed.

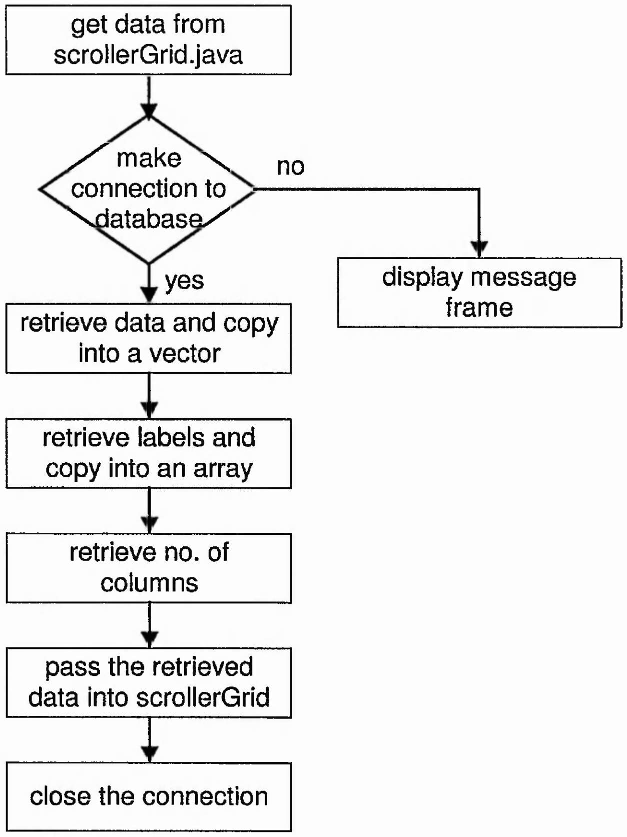

get data from scrollerGrid.java
make connection to database
no
yes
display message frame
retrieve data and copy into a vector
retrieve labels and copy into an array
retrieve no. of columns
pass the retrieved data into scrollerGrid
close the connection

Figure 6-10 Flow diagram for database class

ScrollerGrid applet is made up of these parts:

1. Initialisation phase
2. Layout of the graphical user interface
3. Event handler
4. Retrieving and displaying the data on the applet
5. Invoking Java Servlet to create DXF drawing
6. Terminate applet

The first part and the last part are normal designs for a Java applet. The fifth part will be explained in detail in the next chapter.

In the second part, *BorderLayout()*, a Java layout manager is used to position the components of the user interface in the applet window on the browser. Two *panels* are created, *buttonPanel* and *dxfPanel*. Panels are Java containers and are used for a better organisation of the components within the applet. *buttonPanel* is positioned at the top of the applet, containing the *Display* button, the list box and labels for the *Module* and *Material*. *dxfPanel* is positioned at the bottom of the applet, containing the Create DXF button, the relevant list boxed and labels. A *Scrollerbar* is also created which will be used to navigate through the table. The two panels along with the *Scrollerbar* are added to the *BorderLayout*. For example :

add ("North", buttonPanel);

will add the buttonPanel to the layout manager and position it at the top of the applet.

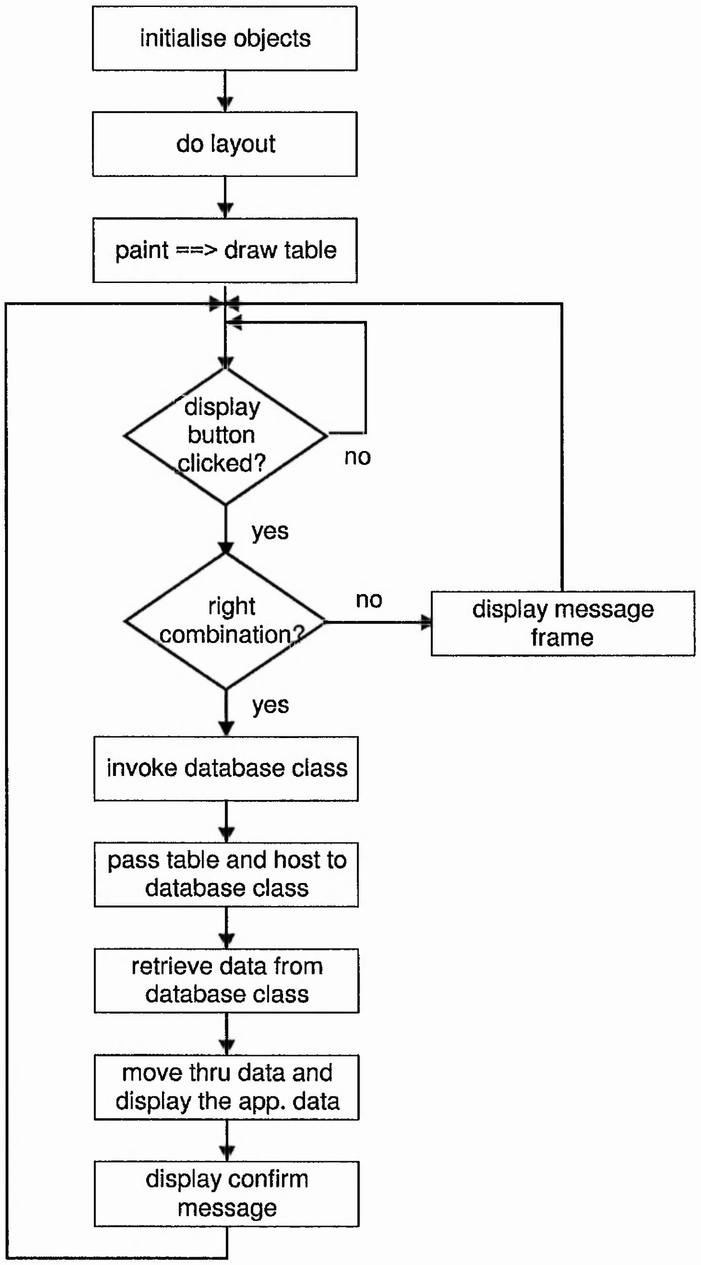


Figure 6-11 Flow diagram for the call to the database within the *scrollerGrid* applet

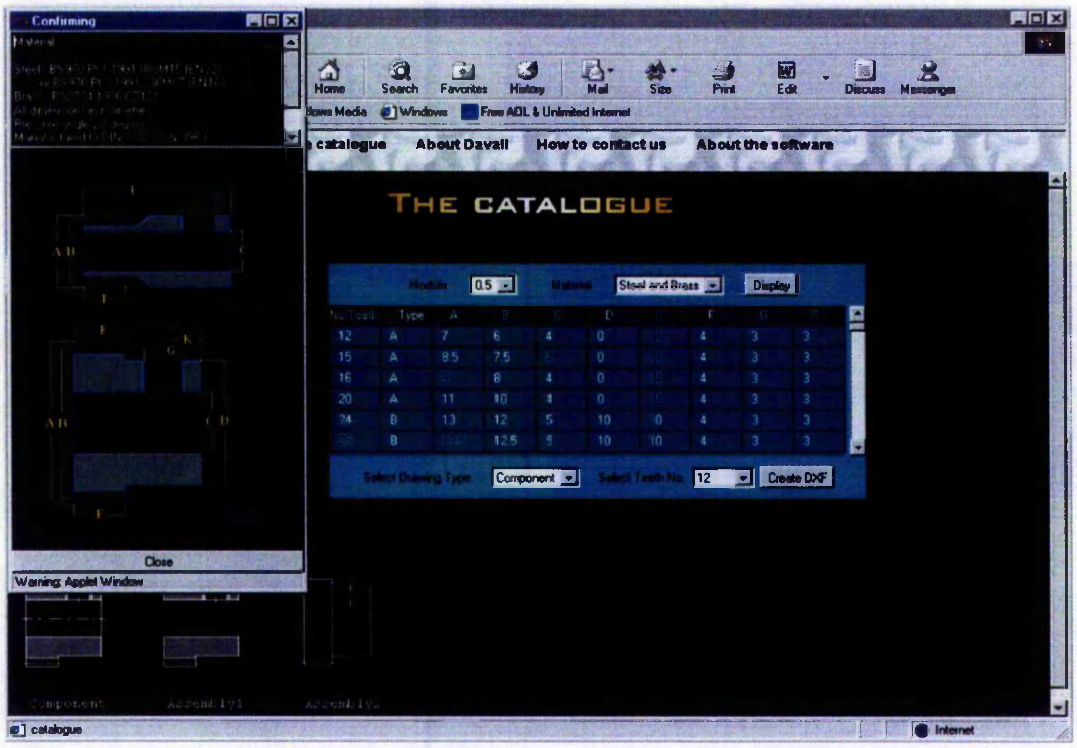
The event handler phase is designed and implemented to scan three kinds of events:

1. When the user presses the *Display* button
2. When the user presses the *Create DXF* button
3. When scrollerbar is activated (i.e. user trying to scroll up or down the table)

For the first case, if the validation procedure does not produce any error, then a connection is made with the database and the appropriate table of data is retrieved. For the third case, a method is invoked to check by what value the scrollbar needs to be incremented or decremented, in order to display the data accordingly.

Displaying the data is done by using vectors. The retrieved data is saved into a vector and its content is then manipulated to be displayed efficiently on the applet. *ScrollerGrid* applet requires all the data from the desired table, number of columns and also the label of the columns. These are passed to it from the *database* class.

When the resulting data is displayed on the applet, the *Select Teeth No* list box is populated simultaneously with the number of teeth from that table of data. This will be needed for creating the DXF drawing. This will be explained in the next chapter.



No Teeth	Type	A	B	C	D	E	F	G	H
12	A	7	6	4	0	14	4	3	3
15	A	8.5	7.5	6	0	14	4	3	3
16	A	9	8	4	0	14	4	3	3
20	A	11	10	4	0	16	4	3	3
24	B	13	12	5	10	0	4	3	3
25	B	13.5	12.5	5	10	10	4	3	3

Figure 6-12 Database page showing the confirmation frame

Figure 6-11 shows the flow diagram for the call to the database, i.e. when the *Display* button is clicked. Figure 6-12 shows the confirmation frame after the connection is created successfully. Some information about the type of gear is given in the frame along with the drawing of the gear, so that the user can visualise the type of gear.

6.7 Concluding remarks

In this chapter, the importance of the electronic catalogues for today's engineering field was outlined. It was discussed that with the available technology, it is possible to make an electronic catalogue online. This is rather essential, since the technology is changing rapidly and the progress of a company relies heavily on how swiftly they can adapt to these changes. Therefore having access to an up-to-date data at any time from any location proves to be very beneficial.

This chapter describes the development of an online catalogue for a gear manufacturing company. The data is saved in Microsoft Access database located on the server. The connection between the user and the database is achieved using Java Database Connectivity. This is an efficient method for connection to a remote database. It is based on Java and inherits all the merits of Java, such as platform independent. Type 3 driver, i.e. Network-protocol pure Java driver is used, which is based on a three tier model. It communicates between the application server and the database server and the client obtains data from the application server.

The process of connection to the database and retrieving the data was described in detail. In addition, the development of the applet that displays the table of data and performs the graphical user interface was also presented.

In the next chapter, a methodology will be presented that expands the functionality of the electronic catalogue by creating the two dimensional drawing in the form of DXF files.

Chapter 7

Dynamic Creation of DXF Files

7.1 Introduction

A lot of present-day mechanical design is compositional design. Many off-the-shelf and vendor supplied components (like bearing, shafts, gears, etc.) are inserted by designers into their design drawings. Some vendors also provide the solid model of their products in a standard format (e.g. AutoCAD DXF) for insertion into designs. Most designers are comfortable with operating in this manner (Rajagopalan et al, 1998).

The efforts towards CAD integration have resulted in the development of various data exchange standards. The five most important files and formats for engineers are DXF, IGES, STEP, SET and VDAFS (Dvorak, 1996). SET is a national drawing translation standard from France while VDAFS, from Germany, has a similar function. DXF is the Drawing Exchange format and supported by Autodesk. It has become a de facto standard because most 2D CAD packages support it. The format has frequently served as a 2D image translator. It enables the interchange of drawings between AutoCAD and other programs. The DXF file format is supported by many applications, and many CAD systems can import drawing files of this format. The DXF file could be in ASCII or binary format. The ASCII format could easily be viewed and analysed in DOS Editor.

As part of this research project, it is proposed to improve the functionality of the gear catalogue by developing a facility for the user to download the drawing of gears of any size in DXF format. How can we provide DXF files for each gear size in the catalogue?

- To physically draw each gear size using AutoCAD and export it as DXF. All the files are located on the server. Every time the user selects a particular gear, the corresponding DXF file will be sent to the users.
- To have only one copy of the DXF file on the server. By users' command, a C++ program located on the server, will be initiated that calculates the dimensions of the selected gear and overwrites the data within that DXF file.

The first method is not feasible, since it will be very costly in terms of the time and effort spent on the creation of each of these gears, which could reach up to hundreds or even thousands, depending on that particular catalogue. Moreover, by placing all of these drawings on the server, a large amount of server space will be wasted. The second method is very efficient and overcomes the problems mentioned above.

In this part of the research, a method is presented that creates the drawing in DXF format dynamically. The user makes connection to the database via JDBC. After the data are retrieved from the database and displayed on the user's monitor, they can select a specific gear design from the table and by his/her command, a C++ program on the server is initiated that creates the drawing. The user can then download the drawing file into his/her machine. Similar to gear design optimisation, the problem of multi-user environment exists. Once again, a directory is created for each user, which will later be removed after it is no longer required.

The communication between client and server is achieved using Java Servlet. The function of the Servlet for this project is to obtain the gear dimensions from the client and pass this to the C++ executable program that creates the drawing in DXF format. After the DXF file is created, the Servlet then automatically invokes *the File Download Dialogue Box* on the client's machine. This would give the clients the option of downloading the DXF file onto their machine.

7.2 Java Servlet

In gear design optimisation presented in chapter 3, the communication between the client and server is performed using CGI. For the DXF creation, however, Java Servelet is used instead of CGI. Java Servlet is a Java class that extends the functionality of a server, similar to applet that extends the functionality of a browser. It is located on the web server and can deal with client request, similar to CGI. Being written in Java, it inherits all the merits of Java, such as cross-platform portability. Servlets can run on any platform without the need for recompilation or rewriting. The server sets up the connection, and once it is successfully connected, it can then be employed to service the client request. A major problem with CGI is its speed because a separate process is invoked for each CGI request. Unlike CGI scripts, a servlet runs in a separate thread of execution that remains alive for the duration of the client connection, with many thread instances of the same servlet servicing multiple clients at any one point. Once a servlet has been loaded into the server, it remains in memory for subsequent client requests, until explicitly unloaded. This eliminates the overhead of reloading and initialising the servlet again, which significantly improves the efficiency of the server (Stein, 1998). When the servlet is loaded, a single method is called that can be used to initialise any startup data. Once the servlet class object has been initialised, it is ready to serve client requests. Servicing a client request is performed by the server, calling a predefined method for servicing requests. Servlets are generally unloaded and removed from memory only when the server shuts down. Figure 7-1 shows the cycle that a servlet goes through (Williamson, 1999).

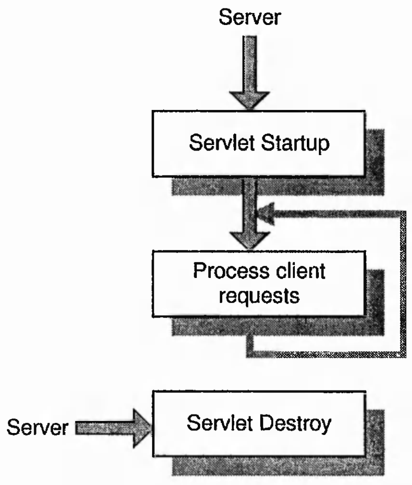


Figure 7-1 Servlet cycle

7.3 Software requirements for servlet installation

These softwares are essential for running the servlet:

– JSDK 2.1

– JRUN 3.1

Javasoft supplies a development kit for creating servlets, the Java Servlet Development Kit (JSDK). It is downloaded from http://jserv.javasoft.com and installed on the server. In order to be able to compile and run the servlets, the directory containing the JSDK classes need to be added to the CLASS-PATH environment variable.

The server used for this research work is Microsoft IIS 4.0 web server. Running a servlet on this web server, requires a servlet engine to host them. A servlet engine does the following tasks:

– It loads the .class file in the Java virtual machine running on the server.

– It runs the servlet and provides it with a runtime environment.

JRun 3.1 from Macromedia has been used to serve as a servlet engine. JRun is a set of Java classes that implement all of the functionality needed for an application to run Java servlets. The servlet classes developed for the creation of the DXF drawings will be located under the JRun directory rather than in the web directory of the server.

7.4 Servlet architecture

Servlets interact with the servlet engine running on the web server through requests and responses. The request-response paradigm is modelled in the behaviour of the HyperText Transfer Protocol (HTTP). A client program which could be a web browser makes a connection across the Internet, accesses the web server and makes a request. This request is processed by the servlet engine that runs with the web server, which returns a response to the servlet. The servlet in turn sends a response in HTTP form to the client (Sun-servlet, 1998). A servlet has three main stages in its life: start-up, servicing requests and Shut down.

- **Start-up**

  When a server loads a servlet, it executes the servlet's *init()* method. This method initialises the servlet. Once initialised, the servlet remains active until the web server is stopped, or until it is explicitly removed by an administrative action. Any resources created under *init()* method will be accessible to every single servlet invocation (Webreview-servlet, accessed June 2000).

- **Servicing requests**

  Service(ServletRequest, ServletResponse) is called each time a servlet is invoked. It is from this method that the servlet processes each client request. The first parameter, *ServletRequest*, encapsulates the information about the HTTP request (e.g. the client's IP address, host name and request parameters). The second parameter, *ServletResponse*, contains methods for setting the type of data that are going to be returned (e.g. text or HTML), finding the output stream that needs to be written to, and sending HTTP-specific header (Clip, 1998). In another words, *ServletRequest* class captures the communication from the client to the server, while the *ServletResponse* class contains the communication from the servlet back to the client, as shown in figure 7-2.

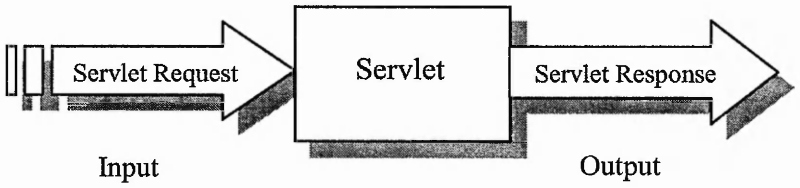


Figure 7-2 Servlet input/output

- **Shut down**

  destroy() method is used to close down the servlet. It stops threads and releases servlet object resources. This method is executed once. The server will not run it again until after it reloads and reinitialises the servlet.

7.5 Structure of the program

When the *Create DXF* button on the applet (refer to figure 6-7) is clicked, the applet passes the selected teeth number and drawing type to the Java Servlet located on the server, *dxfServlet*. The servlet will then invoke a relevant C++ program that reads the gear dimensions selected by the user, performs some calculations and creates the DXF file. Three different types of drawings could be created by the program, shown in figure 7-3.

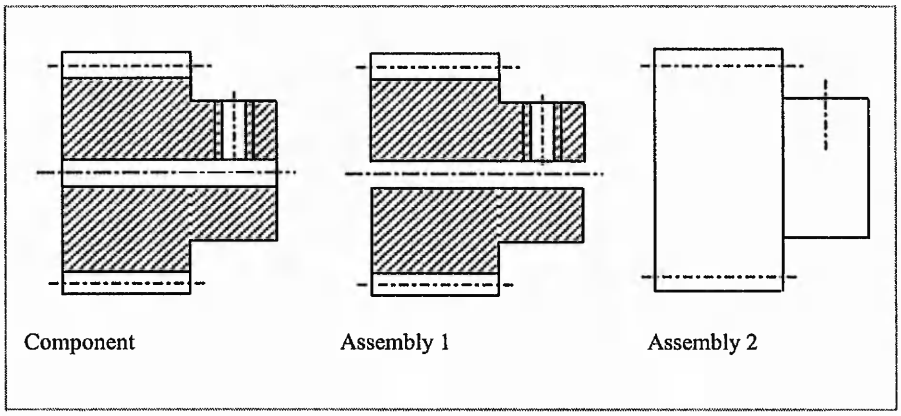


Figure 7-3 Three different types of drawings

The gears with bigger size, i.e. higher module do no have a screw hole. Therefore, the program produces six different types of drawings in DXF format. Six different C++ programs are developed, each responsible for a type of the drawing. These C++ executable programs are shown in table 7-1. The sign ✓ indicates that the *exe* file comprises of that particular feature. For example, *gearb.exe* is responsible for creating component drawings with a screw hole and hatching.

C++ executable program	Component	Assembly	Screw Hole	Hatching
gearb.exe	✓		✓	✓
gearbx.exe	✓		×	✓
asm1.exe		✓	✓	×
asm1x.exe		✓	×	✓
asm2.exe		✓	✓	×
asm2x.exe		✓	×	×

Table 7-1 Six different C++ executable files

It needs to be mentioned that this prototype program will not create DXF drawing for all the gear dimensions in the database. When material is *Steel & Brass* and it is of type *A*, the drawing has a curve feature. The DXF program has not been implemented for this type. If the user has selected this type, the program will not create the DXF file and will inform the user about this. Once again this validation is done by the *scrollerGrid* applet, i.e. it is done locally. When the applet notices this case, it does not invoke the servlet and instead displays a message on user's screen. This process helps in speeding up the system. An example of the table of data from the davall gear catalogue (davall, 2000) is given in appendix F. The diagram of the gear with curve feature is shown in the appendix.

It is in the *scrollerGrid* applet that the program finds the appropriate C++ executable file and passes this along with the type of drawing, teeth number and the relevant data within the selected record to the servlet. The code that invokes the servlet in *scrollerGrid* applet is shown in figure 7-4.

```
Line No.    Code

   1        String path = "http://localhost/servlet/dxfdxfServlet?"+strdxf;
   2        url = new URL(path);
   3        ois = new ObjectInputStream(url.openStream());
   4        Object obj = ois.readObject();
   5        ois.close();
```

Figure 7-4 The code for invoking servlet

The applet opens a connection to the specified servlet URL, as shown in figure 7-4 lines 1 and 2, where *dxfServlet* is the name of the servlet and *strdxf* contains all the data that need to be passed to the servlet. The URL object is initiated in line 2 and then a new instance of *ObjectInputStream* is created that opens a stream to the specified URL object, refer to line 3. Once the connection is made, then the input stream from the servlet is accessed. Finally the connection is closed down.

The code shown in figure 7-4 produces the following exceptions in case of an error: *MalformedURLException, IOException, ClassNotFoundException*. The code is positioned within try{} and catch{} block, where these exceptions will be caught in case they occur and will produce the appropriate message.

For a better understanding, the structure of the process of DXF creation is shown as a flow diagram which is displayed in figure 7-5. The figure shows the data that are passed to the servlet. It is assumed that the user has selected an assembly drawing, with a hole and without hatching. Therefore, the name of the appropriate program is selected in the applet program, *scrollerGrid*, and is then passed on to the servlet. The servlet will then invoke that *exe*. The *exe* program will then create the DXF file using the values that have been passed to it.

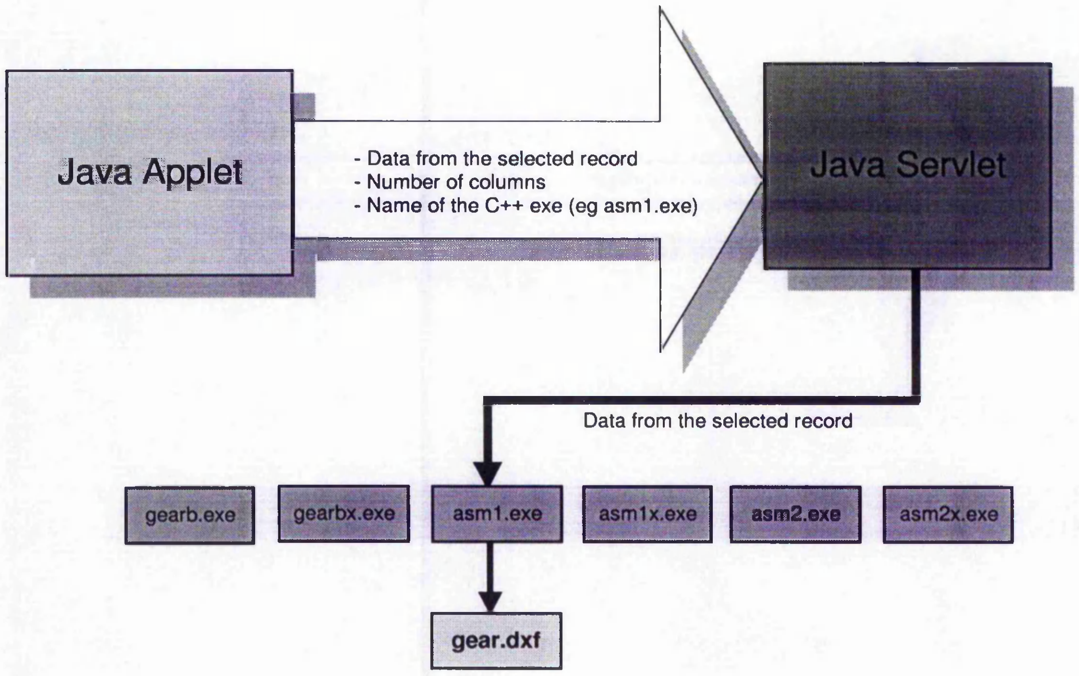


Figure 7-5 Structure of the process of DXF creation

7.6 Servlet program

The package *javax.servlet.** and *javax.servlet.http.** need to be imported into the program. The *doGet()* method will then need to be overwritten to be able to handle the HTTP interactions. It takes two arguments, *HttpServletRequest* object and *HttpServletResponse* object.

The next step is to retrieve the data that have been passed to the servlet from the *scrollerGrid* applet. *getParameterValues()* method is used which returns the value of the named parameter. As an example, the following code returns the name of the appropriate C++ executable file:

```
String dxfProg = request.getParameter("dxfProgExe");
```

where *request* is an instance of *HttpServletRequest* and *dxfProgExe* is the name of the executable file passed to the servlet from the applet.

The C++ executable files take a number of arguments. When invoking them, the appropriate data are passed to them as a command-line argument. The following code invokes the relevant executable program :

```
Process p = Runtime.getRuntime().exec(dxfProgPath);
```

Where dxfProgPath, contains the path of the program, the name of the program as well as the arguments data.

An important method is to set the content type and other response header fields. This specifies how the servlet should response to the client's request. When the DXF file is created, the Servlet then automatically invokes *the File Download Dialogue Box* on the client's machine. This gives the clients the option of downloading the DXF file onto their machine. To download a file to a browser, the content type and header needs to be set. The pseudo-code is shown below:

```
– Read the file into a buffer
– Set the content type to "applicationi/octet-stream"
– Set the header to "Content-Disposition", "attachment;filename=Gear.dxf"
– Get the output stream
– Send the stream to the client
```

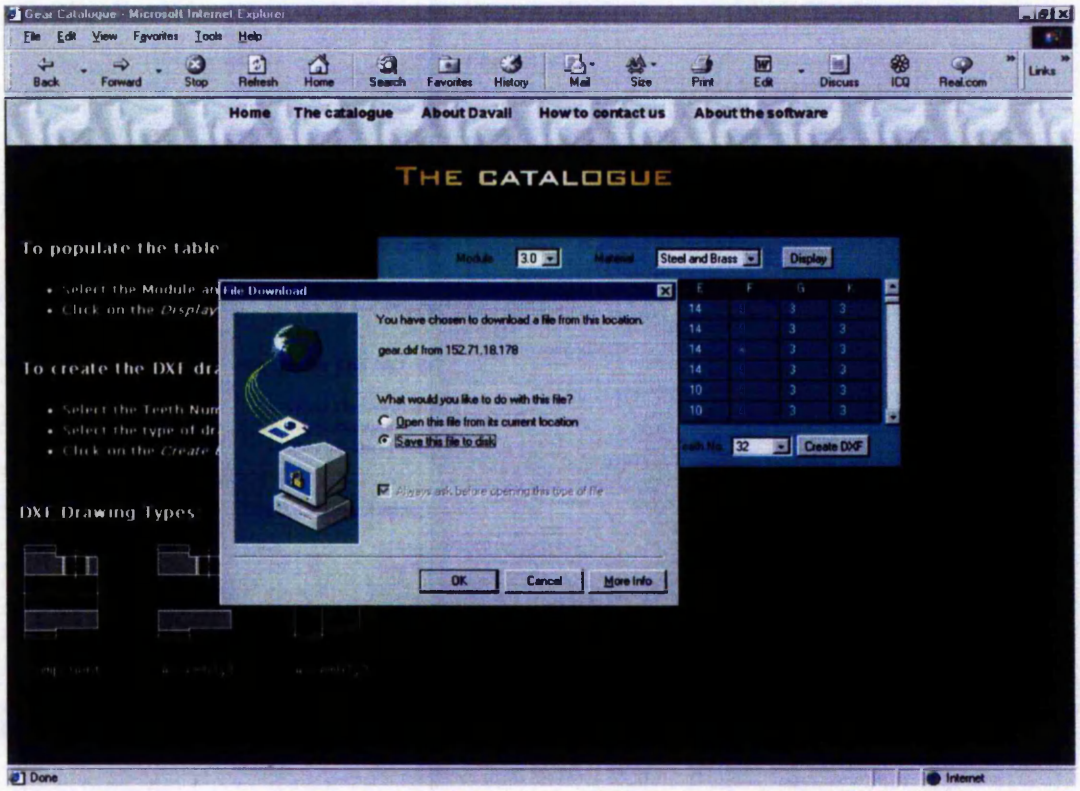
The code for setting the content and header is shown below:

```
response.setContentType("application/dxf");
response.setHeader("Content-Disposition", "attachment;filename="+resultFile);
```

where *resultFile* is the name of the output file, i.e. *gear.dxf* and *response* is an instance of *HttpServletResponse*. The DXF drawing is created on the server. An output stream is opened to send the response to the user. *BufferReader* is then used to read the data from the file and downloaded it to the user, as shown below:

```
ServletOutputStream out = response.getOutputStream();
BufferedReader filein = new BufferedReader(new FileReader(dxfFilePath));
```

where *dxfFilePath* is the path of the new DXF file on the server. The file is then read line by line using filein.readLine() method. Figure 7-6 shows the screen shot of the situation when the user is trying to download the DXF file to his/her machine.


Gear Catalogue - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Size Print Edit Discuss ICQ Real.com Links
Home The catalogue About Davall How to contact us About the software
THE CATALOGUE
To populate the table
DXF Drawing Types
Module 3.0 Material Steel and Brass Display
Create DXF
Done Internet

File Download

You have chosen to download a file from this location.

gear.dxf from 152.71.18.178

What would you like to do with this file?

Open this file from its current location [✓ Save this file to disk]

[✓] Always ask before opening this type of file

OK Cancel More Info

Figure 7-6 User downloading the DXF file

7.7 Multi-user environment

The structure described here cannot simply be applied to a multi-user environment. Similar to the online gear optimisation program discussed in chapter 3, the program produces an output file, i.e. the DXF file on the server. If two or more users are using the program simultaneously, there is a high possibility that they overwrite on each other's DXF file.

Similar procedure as the online gear optimisation program is taken to overcome the problem. A directory is created for each user and the DXF file is written to and downloaded from that folder. The directory name is the name of the remote host name. This is retrieved using the code: *request.getRemoteHost()* , where request is an instance of *HttpServletRequest*.

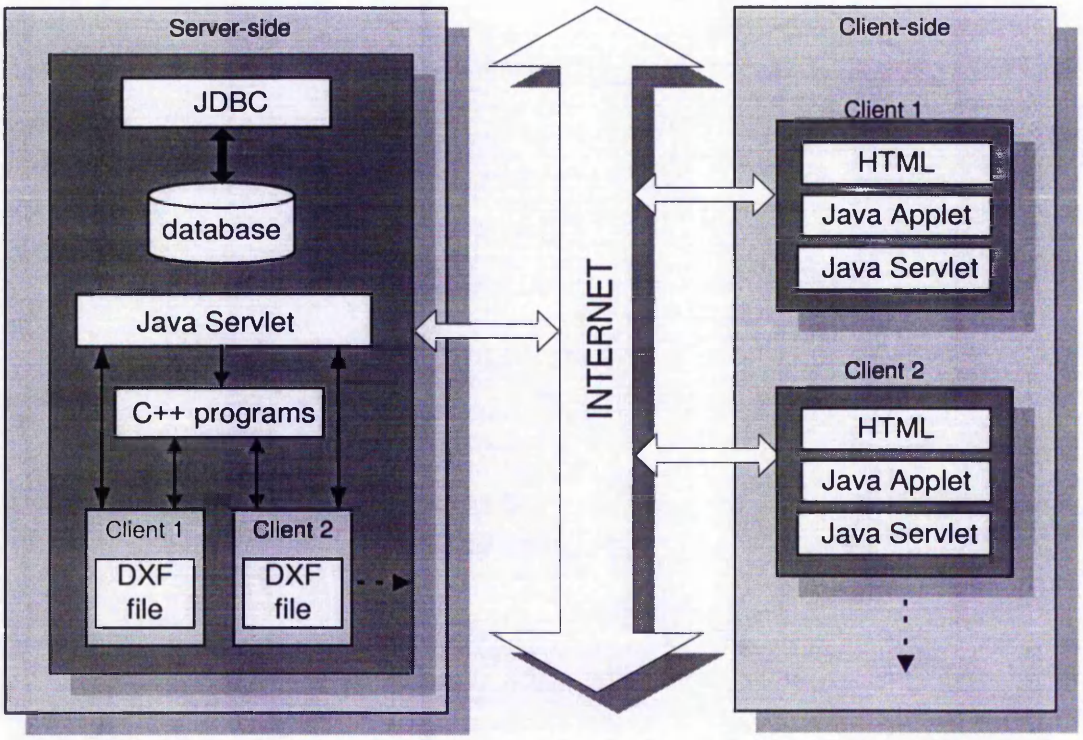


Figure 7-7 Structure of the online gear catalogue system under multi-user environment

The program also checks for the outdated DXF files and their corresponding directories and deletes them to clear up the space on the server. The process of deleting the outdated files and directories is done by the *deleteFiles* class. This is called from the *dxfServlet*. *deleteFiles* runs in its own thread. It removes all the DXF files and their relevant folders that have been used before a certain period of time, which in this case it is two hours. If a directory exists which is the name of the current remote host, it is ignored. The program goes through the directories and if the

DXF file under that directory has been created prior to two hours, both file and the directory is deleted.

Finally the structure of the online gear catalogue can be summarised as shown in figure 7-7.

7.8 Creating DXF drawings

The Drawing Exchange Format was developed and is maintained by AutoDesk. It enables the interchange of drawings between AutoCAD and other programs. Some of the merits of using DXF for the purpose of this work are:

- The DXF file format is supported by many applications, and many CAD systems can import drawing files of this format.
- The DXF file could be in ASCII or binary format. The ASCII format could easily be viewed and analysed in DOS Editor.
- Only a single DXF file is needed for each type of gear. Every time the user selects a dimension, the data is overwritten over the original value. So, there is no need to keep all of the different drawings on file.

A DXF file consists of 5 sections (AutoCAD, 1996):

- **Header section**

  It contains general information about the drawing. Each parameter has a variable name and an associated value.

- **Tables section**

  It contains definitions of data such as line types, layers, text styles, views, etc.

- **Blocks section**

  It contains entities for block definitions. These entities define the blocks used in drawing. The format of the entities in the block section is identical to entities in the entity section.

- **Entities section**

  It contains the drawing entities, including any block references. Items in the entity section exist in the block section as well and the appearance of entities in the two sections is identical.

- **End of file.**

  It indicates that the end of file has reached.

```
0
LINE
8                        these are fixed
1
6
CONTINUOUS               indicates the line is solid
62
0                        these are fixed
5
21                       handler which is unique for each line (it is incremented
                         by 1 for the next line)
 10                         tag for x co-ordinate of the first point
100                      x co-ordinate of the first point
 20                         tag for y co-ordinate of the first point
108                      y co-ordinate of the first point
 30                         tag for z co-ordinate of the first point
0.0                      z co-ordinate of the first point
 11                         tag for x co-ordinate of the second point
104                      x co-ordinate of the second point
 21                         tag for y co-ordinate of the second point
108                      y co-ordinate of the second point
 31                         tag for z co-ordinate of the second point
0.0                      z co-ordinate of the second point
```

Figure 7-8 DXF code within the Entities section, that defines a line.

DXF is organised in these sections and not all of these sections need to be edited. Depending on the task at hand, some of these sections could be completely omitted. In relation to this, an investigation was carried through which it was found that the data for the actual lines of the gear are situated in the *Entities* section and the hatching are located in the *Blocks* section. Therefore, only the data in the *Blocks* and *Entities* section needed to be changed. Each co-ordinate component of line has an identification tag, shown below:

1a *represents the x co-ordinate*

2a	*represents the y co-ordinate*
3a	*represents the z co-ordinate*

where *a* is the vertex number. For example, the line that follows the tag 10 indicates the 'x' component of the first co-ordinate. Figure 7-8 shows the codes, taken from the DXF file which define the co-ordinate of the two points that create a line. This is in the Entities section. The explanation of each line is given in the figure.

Therefore, in order to create the drawings with new dimension, these points need to be overwritten. This is not that straightforward. This is because the data, given in the catalogue, specify the length of the line and not the x and y co-ordinates, so the data could not simply be copied onto the DXF file. To overcome this problem, it was decided to define each point parametrically. In this way, the co-ordinates could be calculated from the data given in the table.

7.9 Calculating co-ordinates of the lines

The first step is to create a drawing in AutoCAD with one of the dimensions given in the catalogue. However, it was noticed that two data were unavailable, i.e. the dimensions that are labelled 'L' and 'M' in figure 7-9. Therefore, these values needed to be calculated for all the individual gears manually. Figure 7-9 shows the relation between B, the pitch diameter and L, the root diameter, graphically.

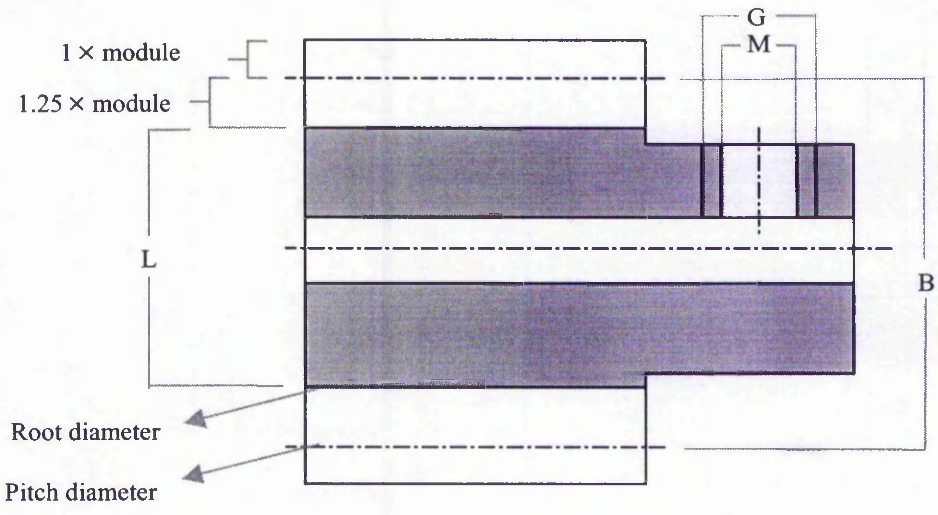


Figure 7-9 Relation between pitch diameter and root diameter

Therefore, the value of *L* could be calculated by the equation shown below:

$$L = B - (2 \times 1.25 \times m)$$

where *m* is the module. The value of *M*, the inner diameter of the hole, could be calculated from the following equation:

$$M = G \times 0.8$$

Now that all the data is available, it is possible to determine the parametric values of each point. It is first required to determine an origin on the diagram. The left co-ordinate of the centre line of the gear is assigned to be the origin. This is shown in figure 7-10, as point 1, the letters correspond to the dimensions given in the table of data.

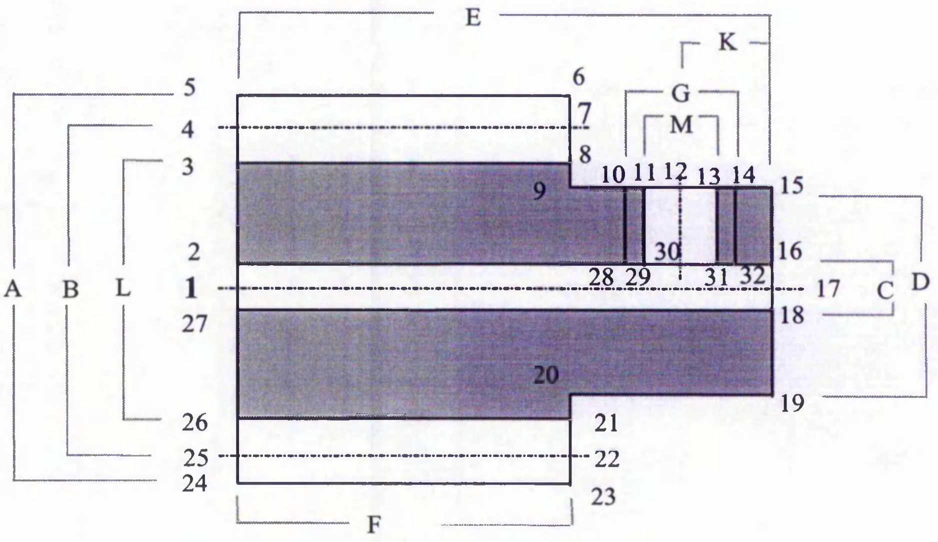


Figure 7-10 Spur Gear with the co-ordinates being marked

The drawing is made up of 32 points. Each of these needs to be assigned an equation (a parametric value). Since point 1 is the origin, its value is (0, 0). Now, the value of point 2 needs to be calculated. Its x co-ordinate is the same as point 1, i.e. 0. Its y co-ordinate is +C/2. It is positive because it is located above the origin (point 1). So, its co-ordinate is (0,C/2). The co-ordinate of the point 27 would be negative, i.e. (0, -C/2). This is due to the fact that it is located below the origin. In this way, the parametric equation for all the co-ordinates are evaluated.

7.10 Overwriting the original DXF file

Using AutoCAD, a drawing of the gear was created using one of the dimensions given in the catalogue and then exported as DXF file. By investigating this file, it was noticed that it is made up of 16 solid lines and 4 dotted lines. The lines were not drawn in order, refer to figure 7-11. The numbers indicates the order of the appearance of the lines in the DXF file. It was also found out that the order is not important, as long as the co-ordinates are correct.

When the user clicks on the *Create DXF* button, the servlet invokes the appropriate C++ executable file and passes the required data to it as a command-line argument. The C++ program then calculates the co-ordinates of each point using the appropriate equation and then overwrites the co-ordinate values of the lines on the DXF file that has been created. Only a part of the DXF file, i.e. the *Blocks* and *Entities* sections of the DXF file need to be edited, refer to figure 7-12.

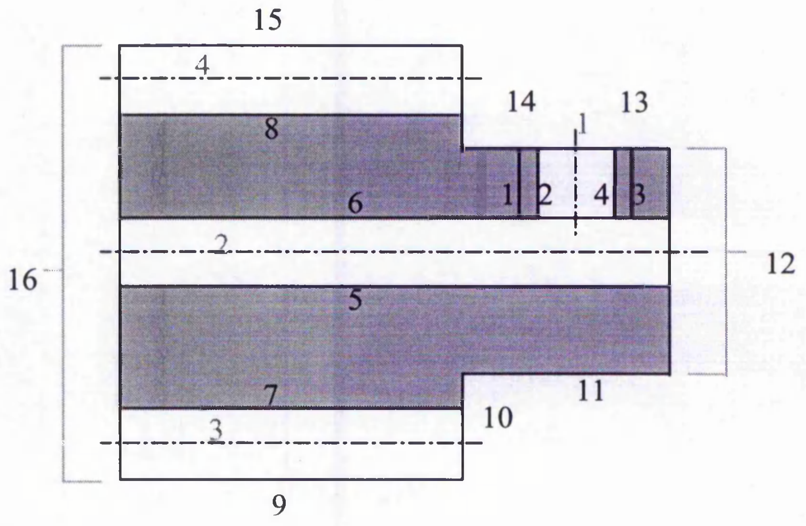


Figure 7-11 Spur Gear with the lines being numbered in order of their appearance in the DXF file

The C++ program first reads the default DXF file and saves the fixed parts into two arrays, *arrayA* containing section A and *arrayC* containing section C (refer to figure 7-12). The C++ program will then write the content of arrayA into the file *gear.dxf.* The program then reads the dimension of the gear from the file dimens.txt, performs the calculation and appends the appropriate data to the DXF file. Finally it adds the data from *arrayC* to the end of the output file.

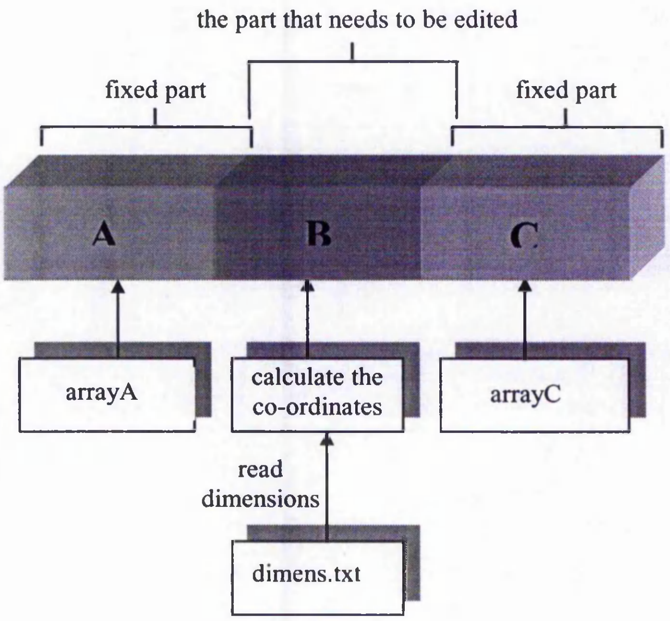


Figure 7-12 DXF file showing the fixed parts

7.11 Hatching

The information about the hatching is located in the *Blocks* section of the DXF file. The lines that construct the hatching are defined exactly as the lines that construct the outline of the drawing. However, a lot of calculations and conditions are needed to be taken into consideration in order to determine the co-ordinates of the hatching lines. When the dimension of the gear changes, the co-ordinates of the lines that construct the hatches, should be changed relatively. The program holds the information on all the points that construct the gear. This information is used to calculate the co-ordinates of the lines that define the hatches.

7.11.1 Hatching the top right section

This section is shown in figure 7-13. The marked points on the diagram are those that are needed in the calculation for hatching this section. The program starts hatching from the top left corner of this section, i.e. point 13. The lines are drawn from the top edge to the left edge. Initially, the co-ordinates of the two points of the lines are set to be the same as point 13. The spaces between the hatches are set to be 1.0.

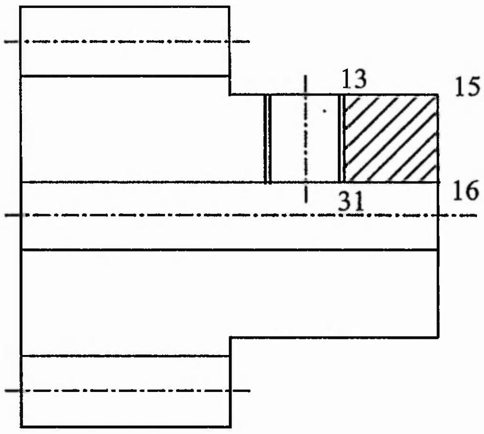


Figure 7-13 The top right section that needs to be hatched

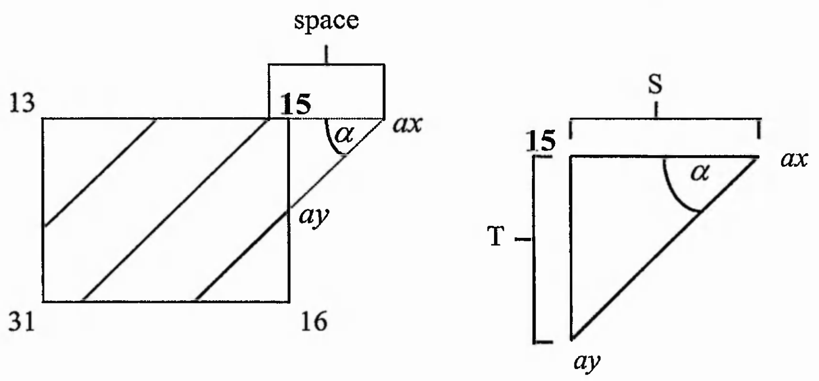
Let us assume that (*ax, ay*) is the co-ordinate of the first point of the line and (*bx, by*) is the co-ordinate of the second point of the line. The program first starts by incrementing *ax* and decrementing *by*. *ay* and *bx* stay unchanged. The program then checks if *ax* has passed the top right corner of the gear, i.e. point 15, and also if *by* has passed the point 31. For the first condition, the program then starts decrementing *ay* and sets the value of *ax* the same as the x co-ordinate of the point 15. In this way, the co-ordinate (*ax, ay*) will lie on the right edge. For the second condition, the program starts incrementing *bx* and sets the value of *by* the same as the y co-ordinate of the point 16. The co-ordinate (*bx, by*) will then lie on the bottom edge of this section.

A problem that arises here is that when *ax* reaches point 15, if *ay* is decremented by 1.0, the hatching will not be consistent and the space between this line and its previous one does not seem to be 1.0. The same problem applies when *by* passes point 31. Therefore the distance of *ay* to the bottom of the point 15 must be calculated. The equation that calculates this, is shown below:

$$space = 1.0 \text{ (the distance between the hatches)}, \quad \alpha = 45^\circ$$

$$\tan 45^\circ = \frac{T}{S} \quad \Rightarrow \quad 1 = \frac{T}{S} \quad \Rightarrow \quad T = S \quad \therefore \quad T = ax - 15x$$

$$ay = 15y - T \quad \Rightarrow \quad ay = 15y - ax + 15x$$



This value of *ay* is used only once. From the second time onwards, the value of *ay* is decremented by the value of space, i.e. 1.0. A similar calculation needs to be done for the value of *bx*, i.e. the distance to the right of point 31. The program then stops drawing the lines when either the value of *ay* becomes less than the y co-ordinate of the point 16, or the value of *bx* becomes more than the x co-ordinate of the point 16. It should be noted that depending on the dimension of the gear, these conditions could occur simultaneously or at different times. All these situations needed to be taken into account in the C++ program.

7.11.2 Hatching the top left section

Hatching the top right section is the most straightforward one compared to the remaining sections. For the top left section, more conditions are needed to be taken into account. The order of the appearance of these conditions is also important. To prevent repeating the calculations, only the conditions with their final results are going to be listed here.

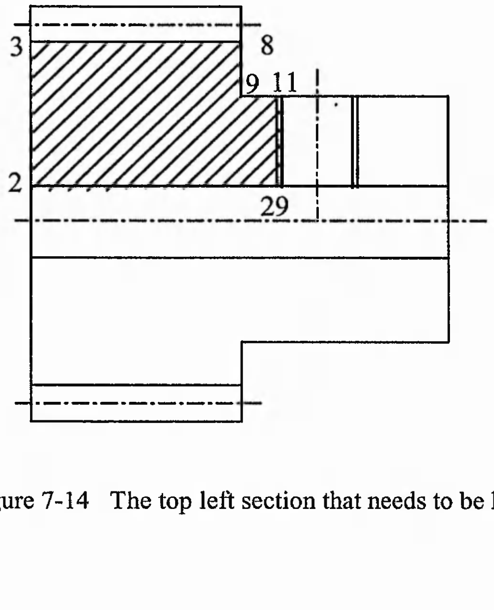


Figure 7-14 The top left section that needs to be hatched

Figure 7-14 indicates which section is going to be hatched with the relevant points being marked. The program starts hatching from the top left corner, i.e. point 3. Therefore, (*ax, ay*) and (*bx, by*) are initialised to the value of point 3. The program then starts incrementing *ax* and decrementing *by*. The conditions exist for the points 2, 8, 9 and 11. The hatching of this section is completed when either the value of *ay* becomes less than the *y* co-ordinate of the point 29, or the value of *bx* becomes more than the *x* co-ordinate of the point 29. The conditions are listed below:

For point 2

If $by < 2y \Rightarrow by = 2y$

$bx = 2x + (2y - by)$ *(for the first time only)*

For point 8

If $ax > 8x \Rightarrow ax = 8x$

$ay = 8y - (ax - 8x)$ *(for the first time only)*

For point 9

If $ay < 9y \Rightarrow ay = 9y$

$ax = 9x + (9y - ay)$

For point 11

If $ax > 11x \Rightarrow ax = 11x$

$ay = 11y - (ax - 11x)$

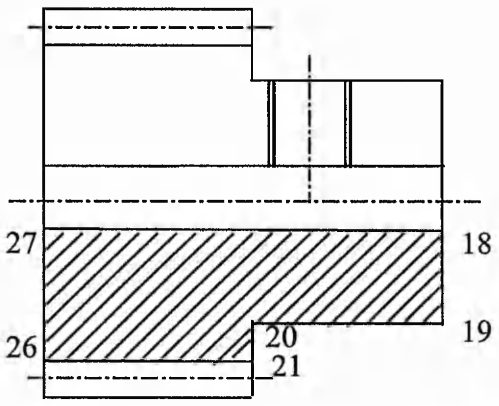


Figure 7-15 The bottom section that needs to be hatched

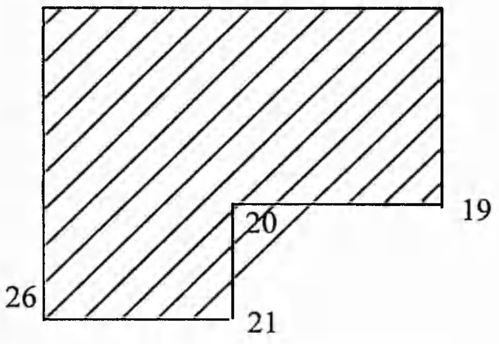


Figure 7-16 Problem caused in the bottom section

7.11.3 Hatching the bottom section

This section is the most complicated part, since the number of conditions that needs to be taken into account are far more than the previous two sections. This section is shown in figure 7-15. While trying to hatch this section, a different problem was encountered. This problem is shown graphically in figure 7-16. As it is shown in the diagram, the problem arises when the line crosses over the point 20. The difficult point is to figure out when the line has reached that point. This would be difficult if the hatching is started from the left side and is progressed to this point, since it is the line itself that is passing point 20 and not its co-ordinate.

To break the problem down, the section was divided into 3 parts. A line was drawn that is tangent to the point 20 and cuts the top and bottom edges at 45°. Using this method the co-ordinate of the points, where the line cuts the top (*ax, ay*) and bottom (*bx, by*) edges could be calculated and each of these parts could be treated separately. The calculations are shown below:

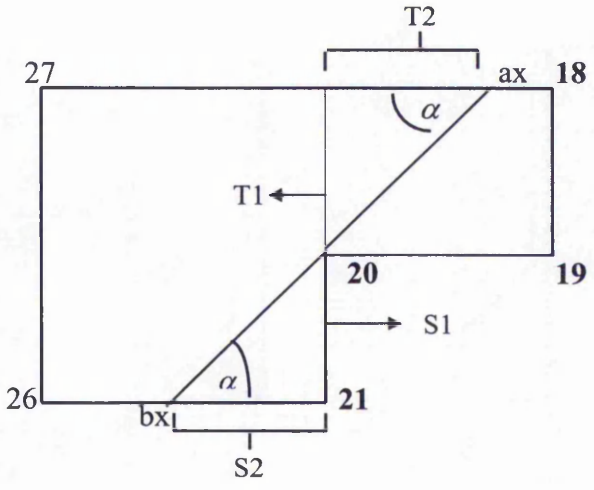
$$\alpha = 45^\circ$$
$$\therefore\ T1 = T2\ \ \&\ \ S1 = S2$$

$$S1 = 20y - 21y$$
$$bx = 21x - S2$$
$$bx = 21x - (20y - 21y)$$

$$\boxed{\begin{aligned} bx &= 21x - 20y + 21y \\ by &= 21y \end{aligned}}$$

$$T1 = 18y - 20y$$
$$ax = 20x + T2$$

$$\boxed{\begin{aligned} ax &= 20x + 18y - 20y \\ ay &= 18y \end{aligned}}$$



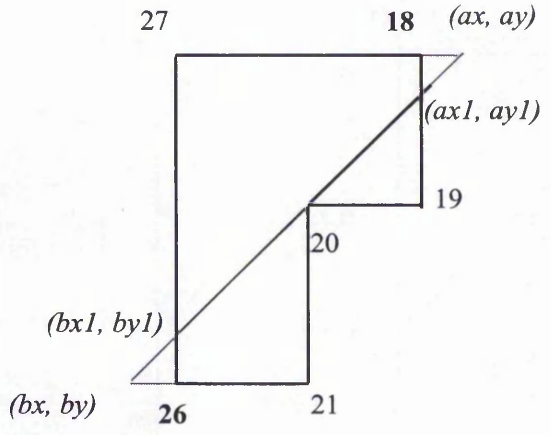
Now that the equations of the co-ordinates of the line, ie (*ax, ay*) and (*bx, by*) have been determined, the next step would be to start hatching the individual parts. However, before moving into the next task, another problem needs to be dealt with. It was found out that the top co-ordinate of the line (*ax, ay*) does not necessarily lie on the top edge and similarly the bottom co-ordinate (*bx, by*) does not necessarily lie on the bottom edge. Depending on the dimension of the gear, the line could cut the right edge, or it could cut the left edge, or both. Therefore, it was essential to take these conditions into consideration. When these conditions occur, the co-ordinates of (*ax*, *ay*) and (*bx*, *by*) would obviously be different. The calculations are shown below:

If $ax > \mathbf{18x}$
$ay1 = \mathbf{18y} - (ax - 18x)$

$$\boxed{\begin{array}{l} ay1 = 18y - ax + 18x \\ ax1 = 18x \end{array}}$$

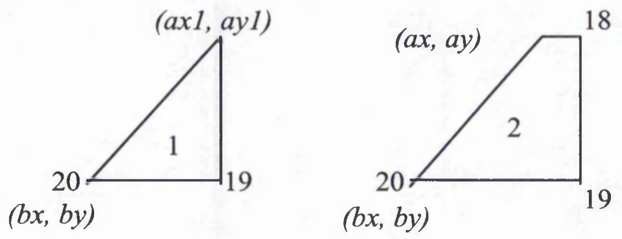
If $bx < \mathbf{26x}$

$$\boxed{\begin{array}{l} by1 = 26y + 26x - bx \\ bx1 = 26x \end{array}}$$



7.11.3.1 Hatching the top right part

Depending on the dimension of the gear, and therefore depending on whether the line cuts the top edge or the right edge, this part could have either of the following shapes:



The initial value of (*bx, by*) would be the same as point 20. The program would start by incrementing *bx;* and *by* would be fixed. For shape 1, the initial value of (*ax, ay*) would be (ax1, ay1) and the program starts decrementing *ay*; and *ax* would be fixed. For shape 2, the program would start incrementing *ax* and *ay* would be fixed. When the value of *ax* exceeds the point 18, the value of *ax* and *ay* are going to be changed, which are shown below:

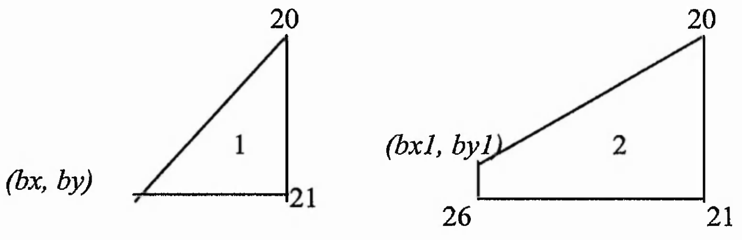
$ax = 18x$

$ay = 18y - (ax - 18x)$ *(for the first time only)*

The program terminates hatching this section, if *ay* becomes less than the *y* co-ordinate of the point 19 and *bx* exceeds the *x* co-ordinate of the point 19.

7.11.3.2 Hatching the bottom right part

This part could have any of the following shapes depending on the dimension of the gear and whether the line cuts the bottom edge or the left edge:



The initial value of (*ax, ay*) would be the same as the point 20. The program starts by decrementing *ay*, but *ax* would be fixed. The initial value of (*bx, by*) for shape 2 would be *(bx1, by1)*. For shape 1, the program starts by incrementing *bx*, but *by* would be fixed, whereas for shape 2, the program starts by decrementing *by*; and *bx* would be fixed. For shape 2, when the value of *by* becomes less than the *y* co-ordinate of the point 26, the values of *(bx, by)* are going to change. These are shown below:

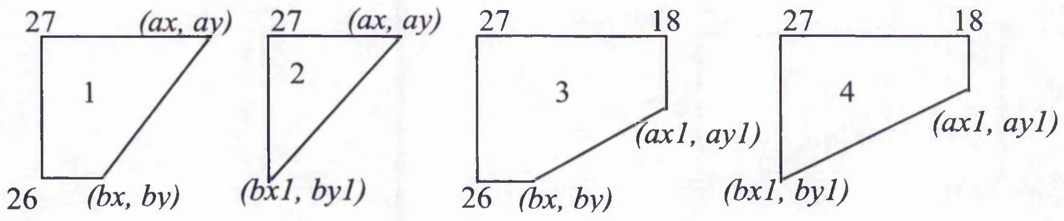
$bx = 26x + 26y - by$

$by = 26y$

The program terminates hatching this part when the value of *ay* becomes less than the *y* co-ordinate of the point 21 or *bx* exceeds the *x* co-ordinate of the point 21.

7.11.3.3 Hatching the left part

This part is more complicated than the other two and it could get any of the following shapes depending on the dimension of the gear and whether the line cuts the right or top edge, and/or the left or bottom edge.



For the shapes 3 and 4, the initial value of *(ax, ay)* would be *(ax1, ay1)* and the program starts by incrementing *ay, ax* is fixed. For the shapes 1 and 2, the program starts by decrementing *ax, ay* is fixed. For the shapes 2 and 4, the initial value of *(bx, by)* would be *(bx1, by1)* and the program starts by incrementing *by, bx* is fixed. For the shapes 1 and 3, the program starts by decrementing *bx, by* is fixed.

For the shapes 1 and 3, when the value of *bx* becomes less than the *x* co-ordinate of the point 26, the new values of *(bx, by)* would be:

$$bx = \mathbf{26}x$$
$$by = \mathbf{26}y + \mathbf{26}x - bx$$

For the shapes 3 and 4, when the value of ay exceeds the *y* co-ordinate of the point 18, the new values of *(ax, ay)* would be:

$$ax = \mathbf{18}x - (ay - \mathbf{18}y)$$
$$ay = \mathbf{18}y$$

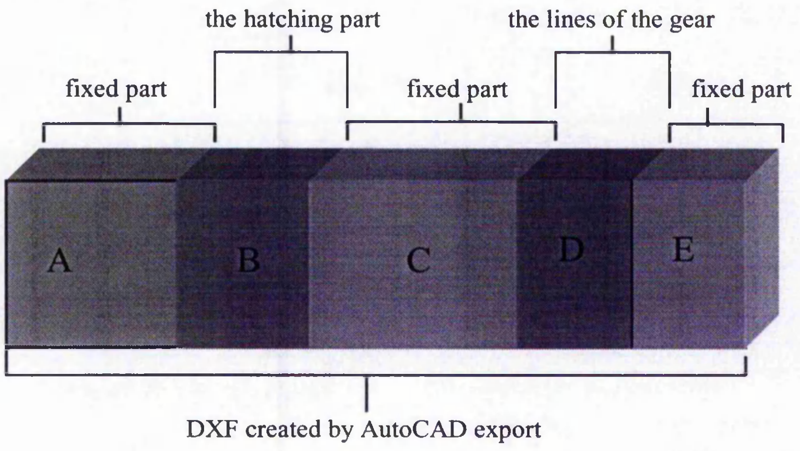


Figure 7-17 DXF file, showing the fixed areas

7.12 C++ implementation

With introducing the hatching, the structure of the process needs to be altered as well. Without the hatching as was explained earlier, only two arrays were needed that contributed to the fixed part of the DXF file. When the hatching process is added, there will be three fixed sections rather than two. This is shown graphically in figure 7-17.

7.13 Concluding remarks

The online gear catalogue has an outstanding feature that makes it dynamic and assertive. The user has the option of creating a two dimensional drawing of the selected gear dynamically. The drawing is in the form of DXF file and it is downloaded to the users' machine, which can then be imported into a CAD software. Such feature was not found in the review of the available online catalogues. This chapter presented this methodology in full detail, outlining the problem areas, specially in the hatching section, and providing the solution to them.

The program is written in C++ and is located on the server. Java servlet is used to perform communication between the client and the server. Servlets are fully written in Java which makes them platform independent.

Since the program creates DXF files on the server, a problem arises when more than one client are using it simultaneously. A solution was presented that overcomes the problem associated with the multi-user environment. A separate directory is created for each user where the DXF will be created for that particular client. The outdated directories will be removed to clear the space on the server.

Chapter 8

Discussion and Conclusion

8.1 Concluding Remarks and discussion

The aim of this research was to study the efficient implementation of Internet into the engineering and manufacturing design. It is to address the problems associated with heterogeneous computing environments. The system developed should therefore be accessible by users as well as engineering teams who are geographically dispersed throughout the world and provide a design environment to achieve the agility of engineering design and manufacturing.

The investigation was undertaken with particular reference to the development of web-based distributed support system for design of gears. With regard to this, two main areas of investigation were considered: online gear design optimisation and online electronic gear catalogue, with dynamic creation of two-dimensional drawings. The development of online gear optimisation gave rise to the problem of speed, which is vital for an Internet environment. To tackle this problem, Artificial Neural Network (ANN) was introduced into the system to estimate the result of the genetic algorithm within the gear optimisation software. An approach was also presented which improved the learning rate of the back propagation learning algorithm.

8.1.1 Online gear design optimisation

The literature review presented the integration of Internet in different areas of engineering and manufacturing. Even though the application of Internet into design and manufacturing has been explosive, it is far from optimum. There are still many areas that need to be explored and pushed to adopt to the new technologies. No work was found that provides the facility to utilise gear optimisation software remotely.

A gear design optimisation software for a stand-alone machine had been produced by the department of Mechanical and Manufacturing engineering at The Nottingham Trent University (Wakelam 1998). Result of the research carried out by Wakelam (1998) showed that high performance gear designs can be achieved without the necessity for a gear design expert. Implementing this over the Internet is of high importance, since designers could have access to it at their own convenience.

Chapter 3 presented in full detail the techniques used to develop the online gear design optimisation. The allocation of the tasks to the client and the server were carefully taken into consideration. CGI, located on the server, were used to communicate between the client and the server and to invoke the gear design optimisation software. JavaScript, located on client's machine, were used to validate the data before they are sent. This reduces the load on the server, specially in the situation where there is a wrong entry. Java applet was also used for graphical user interface and for monitoring the execution time.

The problem of multi-user environment were also discussed and a solution was given. The problem arises due to the presence of the input and output data files. A space was created on the server for each user so that all the interactions would occur within that space; this is to ensure that different users would not interfere with each other's data. These spaces were later removed to clear space on the server.

It might be argued that CGI is slow. This statement might be true depending on the nature of the application. CGI would be inefficient for a situation where a very large number of access requests are to be established simultaneously and where there is a high interaction between the client and the server. This is due to the fact that CGI creates a new process for every client, therefore having to create a new process for a large number of users simultaneously would slow down the system. This is however not the case with the gear design optimisation. The software is a specialised one, so only a limited number of users (gear designers) will be using them. Moreover, there are a limited number of interactions between the client and the server and all the data are simple textual format.

In spite of this, it needs to be mentioned that the requirement to change software tools always exists as the Internet is constantly evolving. A method that is considered efficient today, might be inefficient in a few years time due to the introduction of new technology. Therefore within this dynamic environment, it is difficult to conclude that certain method is best for a given application. Fortunately the gear software is implemented in C++ and runs as a stand-alone application on the server. Therefore changing the environment tools from current configuration to another configuration will not require major changes to the system.

8.1.2 Online electronic catalogue

Chapter 6 presented an approach for the development of the online electronic catalogue. Most companies have a wide range of information, either in the form of printed catalogue or CD-ROM. A CD-ROM can deliver vast amount of information for very little cost, therefore making significant savings over the traditional paper catalogues. Making these data available over the Internet will be a major step in delivering the up-to-date information as quickly as possible. This results in a dramatic reduction in delivery cost. Since only the database on the server needs to be updated, whereas in the off-line catalogues, the updated CD-ROM or the printed catalogue needs to be sent to the user.

The database is in the form of a Microsoft Access located on the server. Java Database Connectivity is utilised to make the connection between the client and the database server. JDBC is written entirely in Java, therefore it inherits all the merits of Java, most importantly being platform independent. Type 3 driver, i.e. Network protocol, pure Java driver is used to make the connection between the application server and the database. No configuration is needed to be set up on user's machine, which makes it efficient for Internet environment. Many online catalogues have been created using CGI. CGI is however inefficient for database connection as described in chapter 2.

Many engineering companies have adapted to the online electronic catalogue. In chapter 2, some of these available catalogues have been discussed. Most of these catalogues are passive, i.e. they solely deliver information. Supplying additional features such as providing design drawings can be very helpful for the customers. As discussed in chapter 7, some of these available online catalogues do provide drawings in the form of DXF, that the user can download. However, these drawings have been created before hand. As a result, only a limited number of drawings are available to download, since creating the drawing for every single gear is not efficient in terms of time taken to create them and the space they would take on the server.

The approach developed by this research, which is described in chapter 7, however can create the drawings in the form of DXF files dynamically. DXF is a result of efforts on the exchange of graphics databases between CAD systems, which deal with geometric information such as lines, circles, etc. Most CAD softwares (e.g. AutoCAD and ProEngineer) can import DXF files. Only one DXF file is available on the server, and this file is overwritten and then sent to the user. The programs that create the DXF file are written in C++ and located on the server. Java Servlet is used to make the communication between the client and server and to invoke the C++ programs. The problem of multi-user environment exists for this approach as well, due to the presence of the DXF file. Similar solution as the gear design optimisation was adapted to implement this approach in a multi-user environment.

8.1.3 Artificial Neural Networks

The development of online gear design optimisation was successful and achieved its objective. However it takes some time, depending on the input data, for the execution to be completed. As was described in chapter 4, this is due to the fact that the optimisation software performs genetic algorithm. GA is known to be computationally expensive. Investigation was carried out to increase the speed of the execution, and Artificial Neural Networks have been integrated in the system. The original optimisation program consisted of two tiers, both utilising GA. ANN was used to replace one of the tiers in order to estimate the result of GA in the first tier. This is a novel approach and no such technique has been found in the literature review. The methodology was presented in detail in chapter 4. Using this method, the speed of execution can be reduced to half. With ANN, time is required to train the network, but once it is trained, the result is produced instantly.

The development of ANN initiated a method that improves the learning rate of the back propagation learning algorithm. This was done by normalising the output value between 0.1 and 0.9 for the Sigmoid activation function and between –0.9 and 0.9 for the tanh activation function; whereas the neuron outputs are traditionally normalised between 0.0 and 1.0 for sigmoid and –1.0 to 1.0 for tanh. This method was tested on a function approximation, for both sigmoid and tanh activation functions; and also on the gear design optimisation problem and both proved to give a satisfactory result. The result of the tests showed that normalising the output to an even smaller scale proved to produce a better performance when using for tanh activation function and for the gear optimisation function, but not for the case when the sigmoid function was employed. Therefore it depends on the nature of the application to see which is more suitable. It needs to be taken into consideration that reducing the scale too much might also remove some vital information. Hence care should be taken when data scaling of smaller range than 0.1&0.9 for sigmoid and –0.9&0.9 for tanh are used.

This approach has been investigated thoroughly for the current project and it has been successfully applied to gear design optimization, which is a novel contribution in this area. Although in a paper (Ozel et al, 2002) it is mentioned that the 0.1&0.9 scaling is used, no detailed work was found in the literature to give thorough examination of the approach. The current research work has proved that data re-scaling for neural network is very useful to improve results in the gear design optimisation.

8.1.4 Graphical User Interface

Throughout the development of this project, special consideration was taken for the efficient design of the graphical user interface. This issue was discussed in chapter 3. It was described that engineers are not interested in a flashy web site, they need the fast and efficient delivery of information. Attempt was made to keep the size of the documents as low as possible. Limited number of graphics were used and HTML code was employed as much as possible, rather than Java applet. HTML is much faster to display than Java applet.

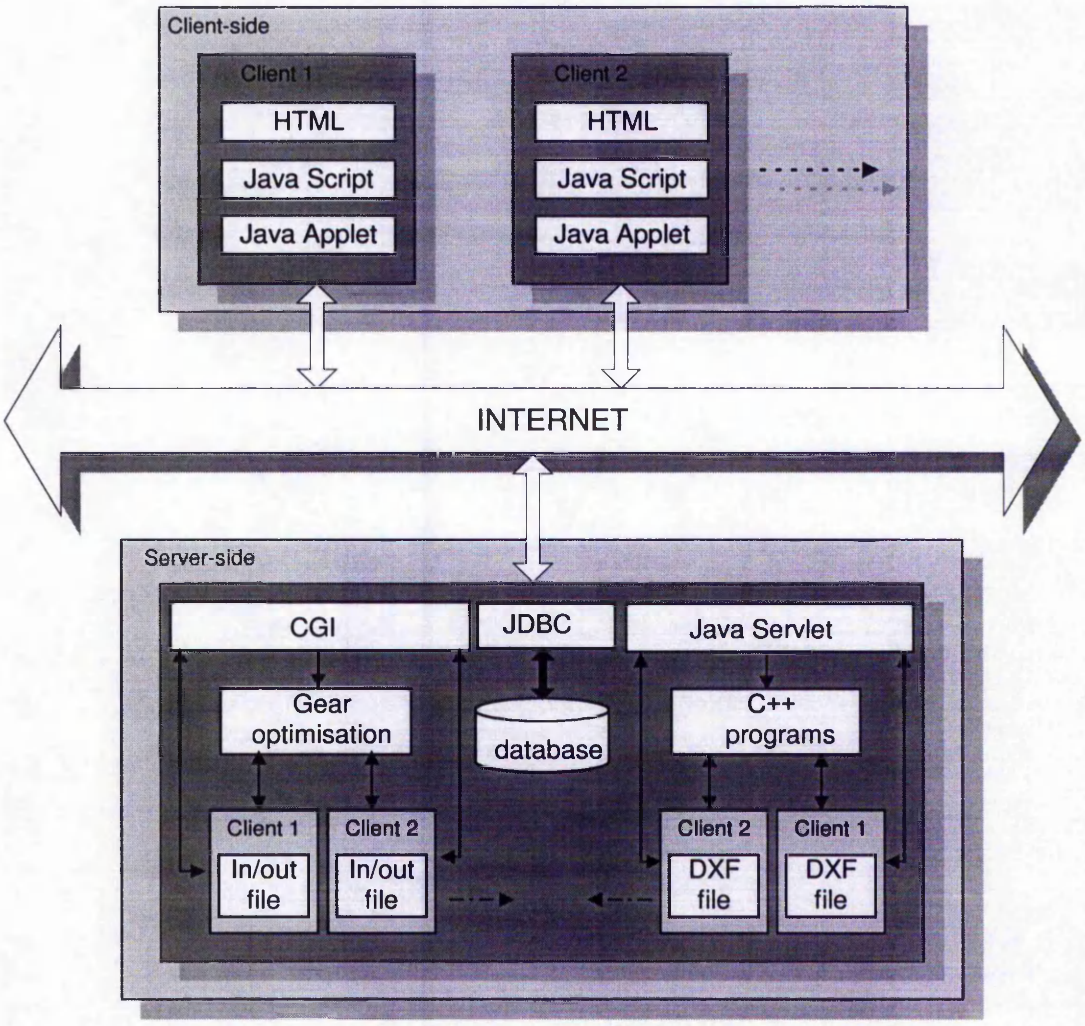


Figure 8-1 Structure of the web-based system

8.1.5 Overall Structure of the system

Figure 8-1 presents the overall structure of the developed web-based system. The figure displays the tasks allocated to server and the client vividly. CGI, JDBC and Java servlet that perform communication between the client and the server are located on the server. The database and the C++ programs, i.e. the gear design optimisation program and the programs that creates DXF drawing files are also located on the server. On the other hand, HTML, JavaScript and Java applet are located and interpreted on the client's machine.

As the figure illustrates, a separate space is allocated for each user for the gear design optimisation package, as well as for the creation of DXF files. These allocated spaces will later be removed when they are no longer required.

8.2 Conclusion

This research project fulfilled its aim, which was to investigate the feasibility of an efficient implementation of Internet for engineering design with gear design as an application area. The approach was developed and implemented successfully by integrating artificial Intelligence (AI), Internet technologies and electronic database with gear design technique. The major objectives achieved within this thesis can be summarised as below:

– The literature review indicated the urge towards the implementation of Internet in engineering and manufacturing companies. No work was found in the application of Internet in the gear design optimisation.

– An approach has been proposed that integrates web-based applications with native codes, such as C or C++. Through this approach, different applications implemented in different languages can be logically and efficiently integrated together. Based on this approach, the development time for a web-based system can be saved dramatically.

– Artificial Neural Networks were integrated into the gear design optimisation to overcome the problem of speed associated with genetic algorithm. ANN is used to estimate the result of the GA. No such work was found in the literature review. Experimental tests produced a successful result and the speed of the execution was cut down dramatically.

– An investigation into ANN triggered a theory to improve the performance of back propagation algorithm. It was proposed to change the scale of normalisation for the output data between 0.1 and 0.9 for sigmoid activation function; and between –0.9 and 0.9 for tanh activation function. Tests carried out on function approximation and gear design optimisation proved the proposed theory.

- The literature review revealed that many engineering companies are implementing online design catalogue. However, many of these catalogues are passive and solely transfer the data. A few online catalogues were found that provide the user with a limited number of design drawings in the form of DXF file. However, these drawings are static. They have previously been created and saved on the server and the user downloads these specific drawings.

- An approach has been developed for the creation of online catalogue for a gear manufacturing company. The database is located on the server and is accessible from any part of the world. JDBC has been used to connect to the remote database. JDBC is written entirely in Java which makes it platform independent. The online catalogue give the user the up-to-date data at all time.

- A methodology has been developed that adds an outstanding feature to the online electronic catalogue. It enables the user to create the two dimensional drawing for the selected gear dynamically. The drawing is in DXF format, which can be imported in most CAD packages, such as AutoCAD and ProEngineer. Only one copy of the DXF file exists on the server, so no space is taken on the server.

- A framework has been designed that addresses the problem associated with the multi-user environment. Both gear design optimisation and online gear catalogue utilise data files that are created on the server. The proposed approach prevents the clients who are using the system simultaneously to overwrite each other's data files. A space is allocated for each user and all the interactions occur within that space. This will later be removed to clear the space on the server.

- The proposed framework described within this thesis does not limit to gear design only. It could easily be modified and adapted for other engineering design fields.

- Special considerations were taken for the creation of graphical user interface. The main objective was to reduce the download time by limiting the number of graphics. The information was also arranged in an efficient way to enable the clients to find the required information easily. The user interface provides a user friendly and convenient interactive environment.

8.3 Further work

This research resulted in an efficient web-based design environment for gear designers and manufacturers. It is however a prototype and due to the time limit, detail testing and validation could not be done. There are also some areas that could be modified and new features could be added to improve its performance, including:

- A systematic evaluation of the developed web-based design system. Detail tests should be carried out to outline any shortcomings or errors.

- The validation and modification of the web-based design system based on the feedback and criticism from the real users.

- A queuing system needs to be implemented for the online gear design optimisation. This is due to the fact that if many users were running the optimisation software simultaneously, this would bring a huge load on the server which results in the server to slow down or in a worse situation, even to crash. Therefore there is a need to introduce a queuing system to limit the number of users running the optimisation at the same time.

- The area of integration of ANN into the gear design optimisation is open to further investigation. At the moment, the approach is implemented only for one condition, i.e. to optimise facewith only. Further research needs to be done to include the rest of the parameters as well. Moreover, further investigation could be carried out to evaluate if the approach could replace both tiers. The potential exists, however at this stage, the result of the tests was not good enough to be able to replace both tiers. Achievement of this objective would be a major step in the design automation and agility.

- Once the ANN is implemented for all the parameters, it should then be launched in the web-based system.

- The current system performs dynamic creation of two-dimensional drawings in the form of DXF files. DXF file has the capability of displaying the drawings in three-dimensional. Therefore further research could be undertaken to create the drawings in three dimensional rather than two-dimensional format. This would improve its functionality to a great extent and facilitates the design immensely.

- The graphical user interface for both online gear design optimisation and online gear catalogue could be improved. Further information, such as background information about gear structure, or further details about the gear manufacturing company could be added to the system.

The result of this research project achieved so far has laid down a solid foundation for carrying out further research into the proposed recommended further work. It is obvious that the Internet technology, both software and hardware, will continue to progress. New technologies or tools could emerge in the near future, which could be integrated into the current system.

References

Adida, B., 1997, "It all starts at the server", IEEE Internet Computing, pp. 75-77.

Agranat, I. D., 1998, "Engineering web technologies for embedded applications", IEEE Internet Computing, pp. 40-45

AutoCAD, 2000, "AutoCAD 2000, User's Guide", Autodesk, Inc., USA

Automec, Automec and Oakes Tools, http://www.automec.co.uk/ [Accessed: March 2002]

Beck, H. and Xin, J., 1998, "Using Java, CORBA, and ODBMS to develop agricultural databases", Computers in Agriculture Conference, pp. 537-544.

Beier, K. P., 2000, "Web-based virtual reality in design and manufacturing applications", 1st International EuroConference on Computer Applications and Information Technology in the Maritime Industries, Germany.

Bernold, T., 1987, "Artificial Intelligence in manufacturing, key to integration?", Gottlieb Duttweiler Institute, Netherlands

Braham, J., 1997, "Designing winning Web sites for engineers", Machine Design, pp. 30-40.

Cheng, B. and Titterington, D.M., 1994, "Neural Networks: a review from a statistical perspective", Statistical Science, Vol. 9, pp. 2-54.

Cheng, K., Pan, P.Y. and Harrison, D.K., 2000, "The Internet as a tool with application to agile manufacturing: a web-based engineering approach and its implementation issues", International Journal of Production Research, Vol. 38, No. 12, pp. 2743-2759.

Cheng, K., Pan, P.Y. and Harrison, D.K., 2001, "Web-based design and manufacturing support systems: implementation perspectives", International Journal of Computer Integrated Manufacturing, Vol. 14, No. 1, pp. 4-27.

Clip, P., 1998, "Servlets: CGI the Java way", BYTE, pp. 55-56.

CORBA, 1997, CORBA: Catching the next wave,
http://developer.netscape.com/library/wpapers/corba/ [Accessed: March 2002]
Cornelius, B., 1998, "Using CORBA and JDBC to produce three tier systems", ACM SIGPLAN Notices, Vol. 33, pp. 44-52.

Coutinho, M, Kim, J., Neches, R., Kumar, V., Eleigh, R., Ling, S. R. and Will, P., 1998, "Active catalogs: integrated support for component Engineering", Proceedings of DETC'98 - 1998 ASME Design Engineering Technical Conference, Atlanta, GA, on CD-ROM, Paper No. DETC/CIE-5521.

Crowder, R. M., Hall, W., Heath, I. And Wills, G., 2000, "Industrial strength hypermedia: design, implementation and application", International Journal of Computer Integrated Manufacturing, Vol. 13, pp. 173-186.

Culley, S., 1998, "Designing with standard parts", Engineering Designer, Vol. 24, pp. 8-11.

Cybercut, Wright, P.K., Integrated Manufacturing Lab,
http://kingkong.me.berkeley.edu/cybercut/, [Accessed: February 2002].

Davall, 2000, Davall Stock Gears Catalogue, UK.

Duke, A. K. and Anumba, C. J., 1999, "A telepresence-based environment for concurrent engineering construction projects", Advances in Concurrent Engineering - CE99, Bath, UK, pp.360-367.

Dvorak, P., 1996, "Stepping over translation troubles", Machine Design, pp. 117-120.

Engelbrecht, A. P. Fletcher, L., Cloete, I., 1999, "Variance analysis of sensitivity information for pruning multilayer feedforward neural networks", IEEE IJCNN, Washington DC, paper 379.

Fesca, Fesca Pacific Gears, http://www.fesca.com.au/cat_gears__frame.htm [Accessed: March 2002]

Frame-world, Frame World Structural Aluminium Framing Components, http://frame-world.com/ [Accessed: March 2002]

Friesenhahn, B., 1998, "Writing JavaScript applications", Bypte, pp. 59-60.

Guan, H., Ip, H.H.S. and Zhang, Y., 1998, "Fava-based approaches for accessing databases on the Internet and a JDBC-ODBC implementaion", Computing and Control Engineering Journal, pp. 71-78.

Gulesian, M., 1998, "JDBC and firewalls", DBMS, Vol. 11, pp. 57-62.

Gupwell, K., 1999, "Corporate data access in a heterogeneous infrastructure", Enterprise MiddleWare, pp. 28-35.

Gurney, K., 1997, “An introduction to neural networks”, UCL Press Limited, London.

Hadjiefthymiades, S. P. and Martakos, D. I., 1997, “Improving the performance of CGI compliant database gateways”, Computers Networks and ISDN Systems, Vol. 29, pp. 1291-1304.

Harvey, J and Mo, J., 2001, “Internet based design system for globally distributed concurrent engineering”, Sybernetics and Systems: An International Journal, Vol. 32, pp. 737-754.

Houser, D.R., Harianto, J., Chandrasekaran, B., Josephson, J. and Iyer, N., 2000, “A multi-variable approach to determine the best gear design”, Proceedings of DETC’2000 - 2000 ASME Power Transmission and Gearing Conference, Baltimore, Maryland, USA, on CD-ROM, Paper No. DETC/PTG-14362.

Huang, G.Q. and Mak, K.L., 2001, “Issues in the development and implementation of web applications for product design and manufacture”, International Journal of Computer Integrated Manufacturing, Vol. 14, pp. 125-135.

Hunt, J., 1998, “Jumping into JDBC”, Application Development Advisor, pp. 42-46.

Itschner, R. Pommerell, C. and Rutishauser, M., 1998, “GLASS: Remote monitoring of embedded systems in power engineering”, IEEE Internet Computing, pp. 46-52

ITTA, 2000, State of the Internet 2000, http://www.itta.com/internet2000.htm, [Accessed: March 2002]

Java 3D, Java 3D API, http://java.sun.com/products/java-media/3D/ [Accessed: March 2002]

JavaSun, The Source for Java Technology, http://java.sun.com/ [Accessed March 2002]

Jiang, H. and Mo, J., 2001, “Internet based design system for globally distributed concurrent engineering”, Cybernetics and Systems: An International Journal, Vol 32, Part 7, pp. 737-754.

Jiang, P. Y. and Fukuda, S., 1999, “Internet service and maintenance for RP-oriented Tele-manufacturing”, Concurrent Engineering: Research and Applications, pp. 179-188.

Karn, R. K., Dandekar, S. V., Poluri, S., Chen, G., Baras, J. S., Nau, D. S., Ball, M.O., Lin, E., Trichur, V. S., Williams, J. T., 1998, “Web-It-Man: A web-based integrated tool for manufacturing environment”, Proceedings of DETC’98 - 1998 ASME Design Engineering Technical Conference, Atlanta, GA, on CD-ROM, Paper No. DETC/CIE-5524.

Kim, D.Y., Kim, N., Kim, Y. and Kang, S.H., 1998, "Distributed concurrent engineering: Internet-based interactive 3-D dynamic browsing and markup of STEP data", Concurrent Engineering: Research and Applications, Vol. 6, pp. 53-70.

Lee Spring, 1997, Lee Spring Electronic Catalogue, Lee Spring Ltd, U.K.

Lemay, L., Perkins, C. L. and Morrison, M., 1996, "Teach yourself Java in 21 days, Professional Reference Edition", Sams.net Publishing.

Levy, M., 1998, "Web programming guide", Software – Practice and Experience, Vol. 28, pp. 1581-1603.

Lippmann, R.P., 1987, "Introduction to computing with neural nets", IEEE Acoustics, Speech and Signal Processing Magazine, Vol. 4, No. 2, pp. 4-22.

Linthicum, D. S., 1998, "Database APIs and Java Meeting connectivity requirements", Component Strategies, Vol. 1, pp. 47-51.

LiveSteamModels, Live Steam Models, http://www.livesteammodels.com/index.html [Accessed March 2002]

Maher, M. L. and Rutherford, J. H., 1997, "A model for syschronous collaborative design using CAD and database management", Research in Engineering Design, Vol. 9, pp. 85-98.

Mentornet, Micro R/C and Indoor Electric Catalogue Index, http://www.mentornet.org/WES.htm [Accessed: March 2002]

Mok, S. M., 1998, "Development of a web-based controller for remote sensing and control of manufacturing processes", Proceedings of DETC'98 - 1998 ASME Design Engineering Technical Conference, Atlanta, GA, on CD-ROM, Paper No. DETC/CIE-5526.

Mullins, Mullins Steering Gears Catalog Index, http://mullinssteeringgears.com/Catalog.htm [Accessed: March 2002]

Nidamarthis, S., Allen, R. H. and Sriram, R. D., 2001, "Observations from supplementing the traditional design process via Internet-based collaboration tools", International Journal of Computer Integrated Manufacturing, Vol. 14, pp. 95-107

NIST (National Institute of Standards and Technology), 1994, "Putting the information infrastructure to work: Report of the information infrastructure task force committee on applications and technology", NIST Special Publication, US. Department of Commerce, pp. 857

ODBC set-up, "Set up ODBC data sources", http://msdn.microsoft.com/library/officedev/off2000/achowStartODBCManager.htm [Accessed: August 2000]

Ozel, T. and Nadgir, A., 2002, "Prediction of Flank Wear by Using Back Propagation Neural Network Modeling When Cutting Hardened AISI H-13 Steel with Chamfered and Honed CBN Tools", International Journal of Machine Tools and Manufacture, Vol.42, pp. 287-297.
Paidly, S. and Sciortino, J., 1999, "Web based applications for manufacturing control", Industrial Engineering Solutions'99 Conference Proceedings, USA, pp. 117-126

Qiang, L., Zhang, Y.F. and Nee, A.Y.C., 2001, "A distutive and collaborative concurrent product design system through the WWW/Internet", The International Journal of Advanced Manufacturing Technology, Vol. 17, pp. 315-322.

QTC, Quality Transmission Components, http://www.qtcgears.com/KHK/index_spur.htm [Accessed: March 2002]

Rajagopalan, S., Pinilla, J.M., Losleben, P., Tian, Q. and Gupta, S.K., 1998, "Integrated design and rapid manufacturing over the internet", Proceedings of DETC'98 – ASME Design Engineering Technical Conferences, Atlanta, GA, on CD-ROM, Paper No. DETC/CIE-5519.

Rou, H., 1997, "Visual J++ Unleashed", Sams.net Pub.

Roy, R., Cooper, S.C., Fan, I.S., Sehdev, K. and Ward, N.J., 1998, "Management of supplier capability information in the extended enterprise using internet technology", CE98 – Advances in Concurrent Engineering, pp. 137-144.

RS, 1998, RS Catalogue on CD-Rom, RS Component Ltd, U.K.

Rzevski, G., Adey, R.A. and Russell, D.W., 1994, "Applications of artificial intelligence in engineering IX", Computational Mechanics Publications, Southampton, Boston.

SAIA-Burgess, SAIA-Burgess Electronics, http://www.saia-burgess.com/us/products/motors/gear_boxes/ [Accessed: March 2002]

Savage, M., Coy, J.J. and Townsend, D.P., 1982, "Optimal tooth numbers for compact standard spur gear sets", Journal of Mechanical Design, Vol. 104, pp. 749-757.

Sherline, Sherline Products, http://www.sherline.com/ [Accessed: March 2002]

SHM, S.H.M On-line Catalogue, http://www.muffett.co.uk/cat-online/main.htm [Accessed: March 2002]

SKF, "SKF", http://www.skf.co.uk/ , [Accessed: March 2002]

Sood, M., 1998, "Examining JDBC Drivers", Dr. Dob's Journal, Vol. 23, pp. 82-87.

Smith, C.S. and Wright, P. K., 1996, "CyberCut: a world wide web based design to fabrication tool", Journal of Manufacturing Systems, Vol. 15, pp. 432-442

Stein, L., 1998, "Java Servlets: Back to the future", Web Techniques, pp. 8-12.

Stevens, A., 1997, "Kicking and Scripting- JavaScript and CGI", Dr. Dobb's Journal, pp. 92-114.

Su, D., Wakelam, M. and Jambunathan, K., 2000, "Integration of a knowledge-based system, artificial neural networks and multimedia for gear design", Journal of Materials Processing Technology, Vol. 107, pp. 53-59.

Sun-servlet, 1998, "About Java Servlets", Http://java.sun.com/products/servlet/2.1/html/introduction.fm.html [Accessed: June 2002]

ThinAccess, "ThinAccess 2.0 User Manual", http://www.thinweb.com/support/thinaccess/ta20/manual.html [Accessed: September 2000]

Tumkor, S., 2000, "Internet-based design catalogue for the shaft and bearing", Research in Engineering Design, Vol. 12, pp. 163-171.

Veeramani, R., Viswanathan, N. and Joshi, S.M. , 1998, "Similarity-based decision support for internet enables supply-web interactions", Proceedings of DETC98 – ASME Design Engineering Technical Conference, Atlanta, GA, on CD-ROM, Paper No. DETC/CIE-5523.

Wakelam, M., 1998, "Intelligent Hybrid Approach for Integrated Design.", Ph.D. thesis, The Nottingham Trent University, UK.

Wang, J., 1993, "Multiple-objective optimisation of machining operations based on neural networks", International Journal of Advanced Manufacturing Technology, Vol. 8, pp. 235-243.

Wang, F.F. and Wright, P.K., 1998, "Web-based CAD tools for a networked manufacturing service", Proceedings of DETC'98 – ASME Design Engineering Technical Conferences, Atlanta, GA, on CD-ROM, Paper No. DETC/CIE-5517.

Wang, H. and Wang, H.P., 1994, "Optimal engineering design of spur gear sets", International Journal of Mechanism and Machine Theory, Vol. 29, pp. 1071-1080.

WebCAD, WebCAD User Page, http://cybercut.berkeley.edu/html/design/webcad_user.htm, [Accessed: February 2002]

Webreview-servlet, "Developing Java Servlets", http://webreview.com/wr/pub/97/10/10/feature/main.html [Accessed: June 2000]

White, S., Fisher, M., Cattell, R., Hamilton, G. and Hapner, M., 1999, "JDBC API Tutorial and Reference, Second Edition – Universal Data Access for the Java 2 Platform", Addison-Wesley, California.

Widrow, B., Rumelhart, D.E. and Lehr, M.A., 1994, "Neural Networks: Applications in Industry, Business and Science", Communications of the ATM, pp. 93-105.

Williams, M. E., 1999, "Highlights of the online database industry and the Internet: 1999", 20th Annual Online Meeting, pp. 1-6.

Williamson, A. R., 1999, "Java servlets by example", Manning Publications Co., Greenwich.

Xia, Q., Feng, L. and Lu, H., 1999, "Supporting web-based database application development", 6th International Conference on Database Systems for Advanced Applications, pp. 17-24.

Yang, A., Linn, J. and Quadrato, D., 1998, "Developing integrated web and database applications using Java applets and JDBC drivers", ACM SIGCE Bulletin, Vol. 30, pp. 302-306.

Yao, Y.L. and Fang, X.D., 1993, "Assessment of chip formation patterns with tool wear progression in machining via neural networks", International Journal of Machine Tools and Manufacture, Vol. 33, pp. 89-102.

Appendix A

Examples of Data Files

In this appendix, an example of input files *genome.dat* and optgear.dat; and the result file *result.dat* is presented.

Genome.dat

1	Facewidth
0	Module
0	Addendum modification coefficient
0	Pressure angle
0	Helix angle
0	Rack tip radius
0	Pinion addendum modification
0	Wheel addendum modification coefficient
0	Number of pinion teeth
2000	Population size
10	Number of tests
100	Fitness function for facewidth
100	Fitness function for centre distance
100	Fitness function for equal stress
100	Fitness function for contact ratio
100	Fitness function for equal slide/roll

Optgear.dat

1	Power
960	Speed
4	Speed ratio
100000	Life
1	Variable centre distance
60	Centre distance
1	Module
20	Pressure angle
0	Helix Angle
24	Number of pinion teeth
96	Number of wheel teeth
15	Pinion facewidth
15	Wheel facewidth
26	Pinion tip diameter
98	Wheel tip diameter
2.25	Pinion tooth depth
2.25	Wheel tooth depth
6	Manufacturing accuracy (Pinion)
6	Manufacturing accuracy (Wheel)
1	Flank surface roughness (Pinion)
1	Flank surface roughness (Wheel)
3	Root surface roughness (Pinion)
3	Root surface roughness (Wheel)
1	Basic addendum coefficient (Pinion)
1	Basic addendum coefficient (Wheel)
.3	Hob tip radius (Pinion)
.3	Hob tip radius (Wheel)
0	Pinion addendum modification coefficient
0	Wheel addendum modification coefficient
1	Crowning
0	Load distribution factor
0	Gear offset
0	Span between bearings
10	Pinion shaft diameter
2	Pinion material type
2	Wheel material type
2	Pinion material quality
2	Wheel material quality
1	Pinion Hardness process
1	Wheel Hardness process
700	Pinion surface hardness
700	Wheel surface hardness
.32	Pinion effective case depth
.45	Wheel effective case depth
2250	Pinion ultimate tensile strength
2250	Wheel ultimate tensile strength
950	Pinion core tensile strength

950	Wheel core tensile strength
-400	Pinion surface residual stress
-400	Wheel surface residual stress
240	Pinion core residual stress
240	Wheel core residual stress
1397	Pinion yield strength for bending stress
1727	Wheel yield strength for bending stress
300	Pinion 0.2% proof stress
300	Wheel 0.2% proof stress
168	Lubrication Viscosity
1	Application factor
1.4	Bending safety factor
1	Surface safety factor
1	Pitting
0	End relief
0	Maximum axial force
35	Maximum helix angle

Result.dat

1	Indicates if program is finished
161	Number of generations
1	Module
20	Pressure angle
0	Helix angle
24	Number pinion teeth
96	Number of wheel teeth
21.888	Pinion facewidth
21.888	Wheel facewidth
26	Pinion tip diameter
98	Wheel tip diameter
2.25	Pinion tooth depth
2.25	Wheel tooth depth
1	Pinion Basic addendum coefficient
1	Wheel Basic addendum coefficient
0.3	Pinion hop tip radius
0.3	Wheel hop tip radius
0	Pinion addendum modification coefficient
0	Wheel addendum modification coefficient
1.72491	Contact ratio
60	Centre distance (resultant)
4	Speed ratio
1367.74	Pinion permissible contact ratio
1455.04	Wheel permissible contact ratio
883.034	Pinion actual contact ratio
883.034	Wheel actual contact ratio
645.963	Pinion permissible bending stress
651.287	Wheel permissible bending stress
237.211	Pinion actual bending stress
238.765	Wheel actual bending stress
0.3	Pinion minimum crest thickness
0.3	Wheel minimum crest thickness
0.71555	Pinion actual crest thickness
0.805807	Wheel actual crest thickness
0.841317	Pinion slide/roll ratio at tip
2.47684	Wheel slide/roll ratio at tip

Appendix B

Determining Architecture for all the ANNs

The table shown here displays the architecture for all the ANNs. Detail tests were applied to find the best possible architecture for them. Due to the limited space, it is not possible to provide the detail of these tests here.

Weights file No.	ANN	Learning Rate	Momentum Term	Initial Weight	Iteration	E(t)	E(g)	%ratio < ➔ = min	%ratio > ➔ = max	Conditions for max & min
1	3-4-1	0.045	0.05	0.05	1100	0.041	0.047			
2	3-4-1	0.045	0.05	0.05	1100	0.038	0.051			
3	3-4-1	0.045	0.05	0.05	1100	0.038	0.049			
4	3-4-1	0.045	0.05	0.05	1100	0.039	0.046			
5	3-4-1	0.045	0.05	0.05	1100	0.039	0.052			
6	3-4-1	0.045	0.05	0.05	1100	0.038	0.054			
7	3-4-1	0.045	0.05	0.05	1100	0.037	0.058			
8	3-4-1	0.045	0.05	0.05	1100	0.042	0.058			
9	3-4-1	0.045	0.05	0.05	1100	0.047	0.052			
10	3-4-1	0.045	0.05	0.05	1100	0.037	0.061			
11	3-4-1	0.05	0.05	0.05	1100	0.039	0.061			
12	3-4-1	0.045	0.05	0.05	1100	0.039	0.059			
13	3-4-1	0.045	0.05	0.05	1100	0.041	0.077			
14	3-4-1	0.045	0.05	0.05	1100	0.039	0.076			
15	3-4-1	0.050	0.05	0.05	1100	0.039	0.070			
16	3-4-1	0.050	0.05	0.05	1100	0.039	0.077			
17	2-3-1	0.045	0.05	0.05	20000	0.037	0.098	0.31	0.85	
18	2-3-1	0.045	0.05	0.05	30000	0.030	0.090	0.31	0.81	
19	2-3-1	0.045	0.05	0.05	20000	0.054	0.095	0.31	0.81	
20	2-3-1	0.045	0.05	0.05	40000	0.030	0.095	0.31	0.82	
21	2-3-1	0.045	0.05	0.05	20000	0.040	0.096	0.32	0.81	
22	2-3-1	0.045	0.05	0.05	20000	0.051	0.080	0.32	0.81	
23	2-3-1	0.050	0.05	0.05	20000	0.059	0.085	0.31	0.82	
24	2-3-1	0.045	0.05	0.05	20000	0.044	0.102	0.30	0.82	
25	2-9-1	0.045	0.05	0.05	5000	0.070	0.103	0.30	0.82	
26	2-3-1	0.045	0.05	0.05	10000	0.074	0.099	0.31	0.82	
27	2-3-1	0.045	0.05	0.05	20000	0.070	0.085	0.30	0.82	
28	2-3-1	0.045	0.05	0.05	15000	0.076	0.074	0.25	0.89	
29a	2-4-1	0.040	0.05	0.05	10000	0.058	0.086	0.31	0.82	Teeth > 25
29b	2-3-1	0.045	0.05	0.05	100000	0.090	0.108	-----	0.95	" else
30	2-5-1	0.045	0.05	0.05	3000	0.075	0.107	0.31	0.82	Teeth > 22
31	2-8-1	0.045	0.05	0.05	5000	0.071	0.113	0.30	0.82	Teeth > 20

32a	2-4-1	0.040	0.05	0.05	10000	0.06	0.107	0.30	0.82	Teeth > 20
32b	5-7-1	0.045	0.05	0.05	20000	0.058	0.108	------	0.92	“ else
33	2-5-1	0.045	0.05	0.05	30000	0.095	0.112	0.26	1.23	
34	2-5-1	0.045	0.05	0.05	30000	0.090	0.104	0.28	0.98	
35	2-4-1	0.050	0.05	0.05	30000	0.086	0.108	0.28	0.98	
36	2-5-1	0.045	0.05	0.05	90000	0.070	0.106	0.28	0.97	
37	2-3-1	0.045	0.05	0.05	40000	0.087	0.104	0.30	0.88	
38	2-9-1	0.045	0.05	0.05	50000	0.083	0.103	0.30	0.88	
39	2-3-1	0.045	0.05	0.05	50000	0.049	0.081	0.30*	0.87	* If 65<teeth<96→ use ANN
40	2-5-1	0.045	0.05	0.05	30000	0.059	0.091	------	0.88	
41a	2-3-1	0.045	0.05	0.05	40000	0.044	0.076	0.30	0.88	Teeth < 121
41b	2-3-1	0.045	0.05	0.05	150000	0.014	0.084	-----	0.88	“ else
42a	2-3-1	0.040	0.05	0.05	50000	0.051	0.071	0.30	0.89	Teeth < 108
42b	2-5-1	0.045	0.05	0.05	100000	0.062	0.070	-----	0.89	“ else
43	2-4-1	0.045	0.05	0.05	70000	0.08	0.098	0.30	0.88	
44a	2-5-1	0.050	0.05	0.05	100000	0.059	0.106	0.30	0.88	Teeth<89 && teeth > 111
44b	5-3-1	0.045	0.05	0.05	20000	0.091	0.107	-----	-----	88 < teeth < 112
45	2-3-1	0.045	0.05	0.05	50000	0.044	0.082	0.30	0.87	
46	2-3-1	0.045	0.05	0.05	50000	0.045	0.100	0.30	0.87	
47	2-3-1	0.045	0.05	0.05	30000	0.054	0.094	0.26	0.88	
48	2-3-1	0.045	0.05	0.05	20000	0.055	0.094	0.24	0.93	

Appendix C

Data used for Comparing Output of Tier1 using ANN and GA

	Case 1	**Case 2**	**Case 3**
Facewidth	1	1	1
Module	0	0	0
Addendum modification coefficient	0	0	0
Pressure angle	0	0	0
Helix angle	0	0	0
Rack tip radius	0	0	0
Pinion addendum modification	0	0	0
Wheel addendum modification coefficient	0	0	0
Number of pinion teeth	0	0	0
Population size	2000	2000	2000
Number of tests	10	10	10
Fitness function for facewidth	100	100	100
Fitness function for centre distance	100	100	100
Fitness function for equal stress	55	10	1
Fitness function for contact ratio	100	100	100
Fitness function for equal slide/roll	100	100	100
Power	2	.5	15
Speed	1000	800	1500
Speed ratio	3	2	7
Life	100000	100000	100000
Variable centre distance	1	1	1
Centre distance	150	54	635
Module	2.5	2	1.25
Pressure angle	20	22.5	20
Helix Angle	0	0	0
Number of pinion teeth	30	18	127
Number of wheel teeth	90	36	889
Pinion facewidth	25	20	15
Wheel facewidth	25	20	15
Pinion tip diameter	80	40	161.25
Wheel tip diameter	230	76	1113.75
Pinion tooth depth	5.625	4.5	2.8125
Wheel tooth depth	5.625	4.5	2.8125

Manufacturing accuracy (Pinion)	7	9	6
Manufacturing accuracy (Wheel)	7	9	6
Flank surface roughness (Pinion)	1	1	1
Flank surface roughness (Wheel)	1	1	1
Root surface roughness (Pinion)	3	3	3
Root surface roughness (Wheel)	3	3	3
Basic addendum coefficient (Pinion)	1	1	1
Basic addendum coefficient (Wheel)	1	1	1
Hob tip radius (Pinion)	.75	.6	.375
Hob tip radius (Wheel)	.75	.6	.375
Pinion addendum modification coefficient	0	0	0
Wheel addendum modification coefficient	0	0	0
Crowning	1	1	1
Load distribution factor	0	0	0
Gear offset	0	0	0
Span between bearings	0	0	0
Pinion shaft diameter	19	20	12
Pinion material type	2	2	2
Wheel material type	2	2	2
Pinion material quality	2	2	2
Wheel material quality	2	2	2
Pinion Hardness process	1	1	1
Wheel Hardness process	1	1	1
Pinion surface hardness	700	700	700
Wheel surface hardness	700	700	700
Pinion effective case depth	.32	.32	.32
Wheel effective case depth	.45	.45	.45
Pinion ultimate tensile strength	2250	2250	2250
Wheel ultimate tensile strength	2250	2250	2250
Pinion core tensile strength	950	950	950
Wheel core tensile strength	950	950	950
Pinion surface residual stress	-400	-400	-400
Wheel surface residual stress	-400	-400	-400
Pinion core residual stress	240	240	240
Wheel core residual stress	240	240	240
Pinion yield strength for bending stress	1397	1397	1397
Wheel yield strength for bending stress	1727	1727	1727
Pinion 0.2% proof stress	300	300	300
Wheel 0.2% proof stress	300	300	300
Lubrication Viscosity	168	168	168
Application factor	1	1	1
Bending safety factor	1.4	1.4	1.4
Surface safety factor	1	1	1
Pitting	1	1	1
End relief	0	0	0
Maximum axial force	0	0	0
Maximum helix angle	35	35	35

	Case 4	**Case 5**	**Case 6**
Facewidth	1	1	1
Module	0	0	0
Addendum modification coefficient	0	0	0
Pressure angle	0	0	0
Helix angle	0	0	0
Rack tip radius	0	0	0
Pinion addendum modification	0	0	0
Wheel addendum modification coefficient	0	0	0
Number of pinion teeth	0	0	0
Population size	2000	2000	2000
Number of tests	10	10	10
Fitness function for facewidth	100	100	100
Fitness function for centre distance	100	100	100
Fitness function for equal stress	100	80	45
Fitness function for contact ratio	100	100	100
Fitness function for equal slide/roll	100	100	100
Power	3	12	10
Speed	1500	1200	1300
Speed ratio	5	5	7
Life	90000	100000	100000
Variable centre distance	1	1	1
Centre distance	840	120	2400
Module	4	1	6
Pressure angle	22.5	24	24
Helix Angle	0	0	0
Number of pinion teeth	70	40	100
Number of wheel teeth	350	200	700
Pinion facewidth	40	15	60
Wheel facewidth	40	15	60
Pinion tip diameter	288	42	612
Wheel tip diameter	1408	202	4212
Pinion tooth depth	9	2.25	13.5
Wheel tooth depth	9	2.25	13.5
Manufacturing accuracy (Pinion)	8	7	6
Manufacturing accuracy (Wheel)	8	7	6
Flank surface roughness (Pinion)	1	1	1
Flank surface roughness (Wheel)	1	1	1
Root surface roughness (Pinion)	3	3	3
Root surface roughness (Wheel)	3	3	3
Basic addendum coefficient (Pinion)	1	1	1
Basic addendum coefficient (Wheel)	1	1	1
Hob tip radius (Pinion)	1.2	.3	1.8
Hob tip radius (Wheel)	1.2	.3	1.8
Pinion addendum modification coefficient	0	0	0
Wheel addendum modification coefficient	0	0	0
Crowning	1	1	1

Load distribution factor	0	0	0
Gear offset	0	0	0
Span between bearings	0	0	0
Pinion shaft diameter	35	10	40
Pinion material type	2	2	2
Wheel material type	2	2	2
Pinion material quality	2	2	2
Wheel material quality	2	2	2
Pinion Hardness process	1	1	1
Wheel Hardness process	1	1	1
Pinion surface hardness	700	700	700
Wheel surface hardness	700	700	700
Pinion effective case depth	.32	.32	.32
Wheel effective case depth	.45	.45	.45
Pinion ultimate tensile strength	2250	2250	2250
Wheel ultimate tensile strength	2250	2250	2250
Pinion core tensile strength	950	950	950
Wheel core tensile strength	950	950	950
Pinion surface residual stress	-400	-400	-400
Wheel surface residual stress	-400	-400	-400
Pinion core residual stress	240	240	240
Wheel core residual stress	240	240	240
Pinion yield strength for bending stress	1397	1397	1397
Wheel yield strength for bending stress	1727	1727	1727
Pinion 0.2% proof stress	300	300	300
Wheel 0.2% proof stress	300	300	300
Lubrication Viscosity	168	168	168
Application factor	1	1	1
Bending safety factor	1.4	1.4	1.4
Surface safety factor	1	1	1
Pitting	1	1	1
End relief	0	0	0
Maximum axial force	0	0	0
Maximum helix angle	35	35	35

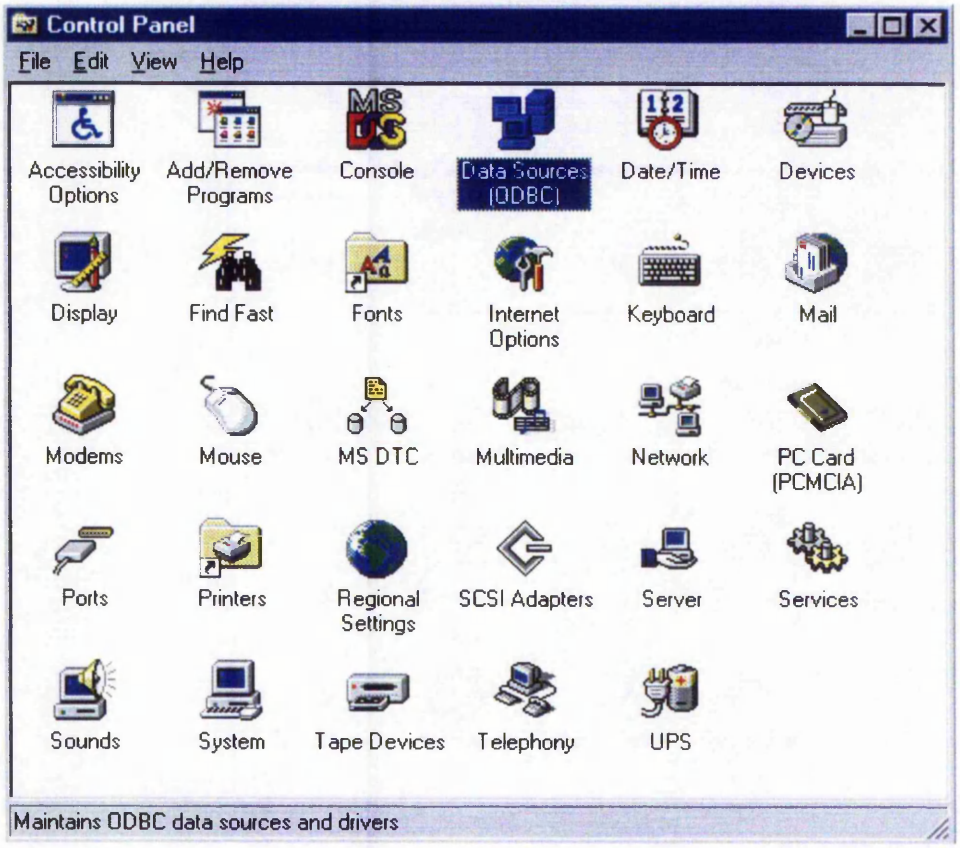
Appendix D

Setting up an ODBC System Data Source

Instructions given in this appendix is for setting up an ODBC DSN for gear.mdb database (Microsoft Access) under Windows NT environment.

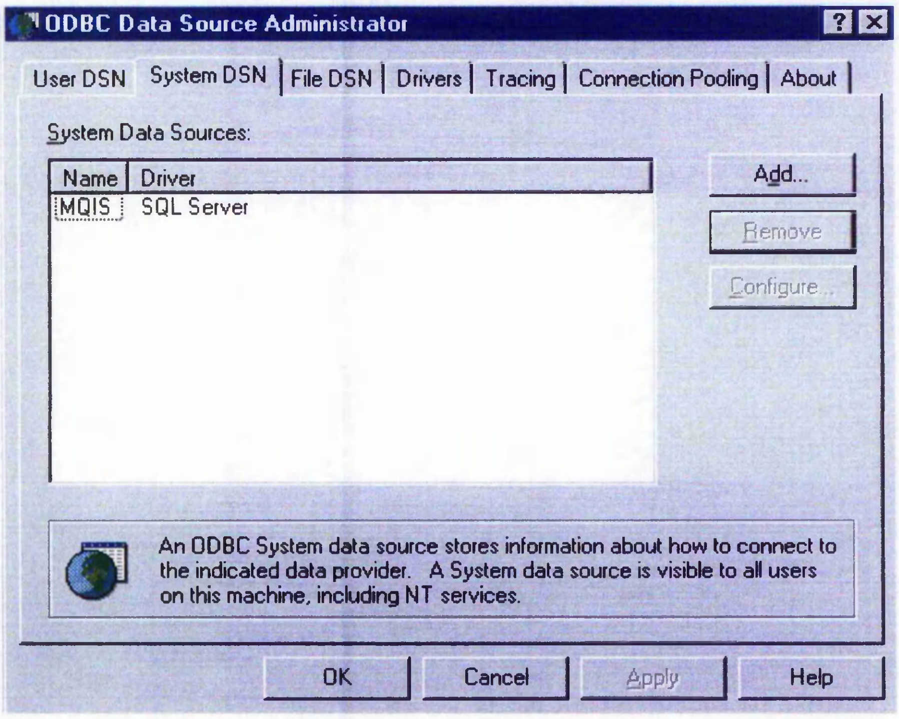
1. Click on : *Start* ➔ *Settings* ➔ *Control panel*

2. In *Control Panel*, double click on *Data Source (ODBC)*, as shown below:



Maintains ODBC data sources and drivers

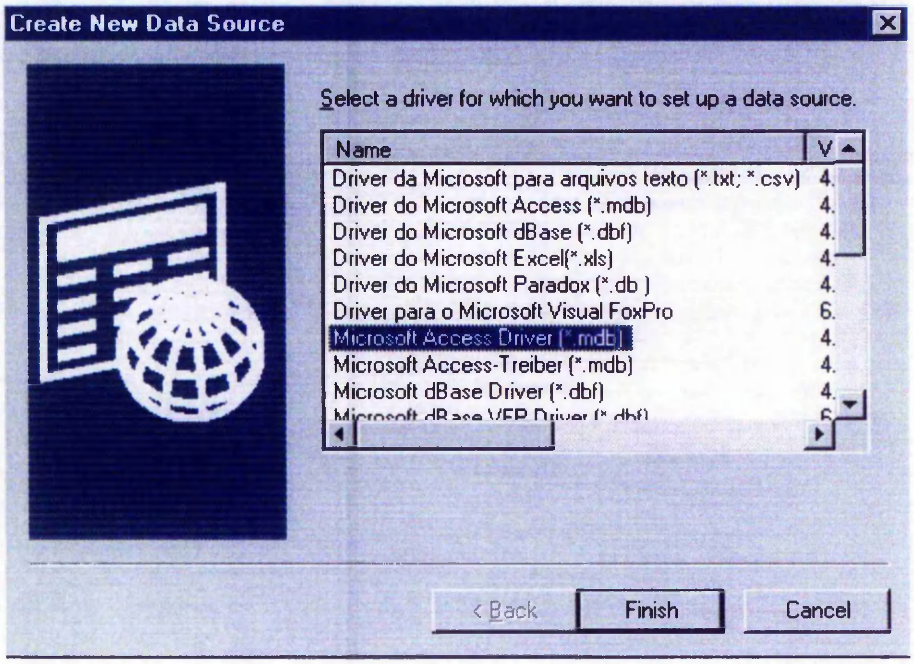
3. In *ODBC Data Source Administrator*, select the *System DSN*, as shown below:



Name	Driver
MQIS	SQL Server

An ODBC System data source stores information about how to connect to the indicated data provider. A System data source is visible to all users on this machine, including NT services.

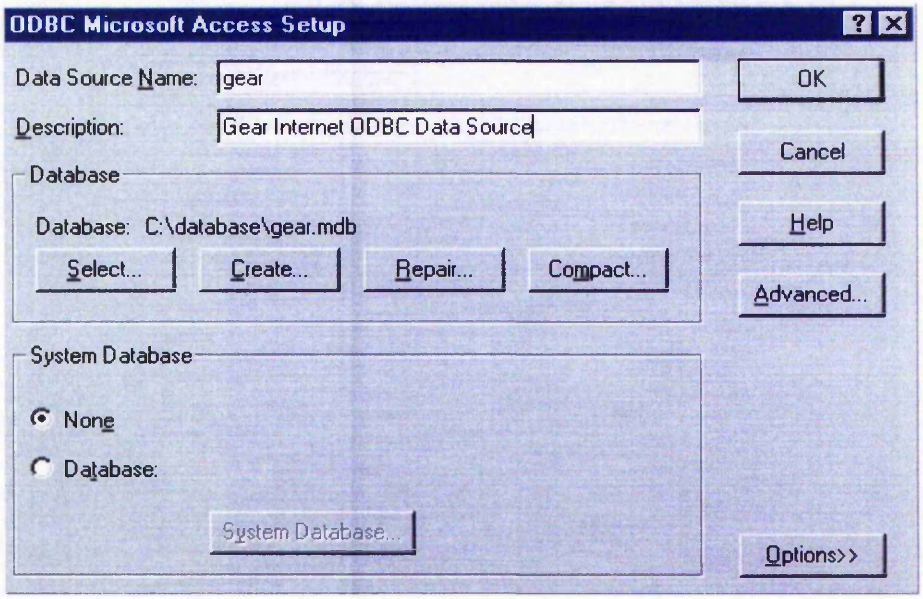
4. Click on *Add* button and select *Microsoft Access driver(*.mdb)* and click on the *Finish* button:



Select a driver for which you want to set up a data source.

Name	V
Driver da Microsoft para arquivos texto (*.txt; *.csv)	4.
Driver do Microsoft Access (*.mdb)	4.
Driver do Microsoft dBase (*.dbf)	4.
Driver do Microsoft Excel(*.xls)	4.
Driver do Microsoft Paradox (*.db)	4.
Driver para o Microsoft Visual FoxPro	6.
Microsoft Access Driver (*.mdb)	4.
Microsoft Access-Treiber (*.mdb)	4.
Microsoft dBase Driver (*.dbf)	4.
Microsoft dBase VFP Driver (*.dbf)	6.

5. In *ODBC Microsoft Access Setup* dialog box, under *Database*, click the *Select* button. Locate the *gear.mdb* file on the hard disk, select it and click the *OK* button. In the *Data Source Name box*, type *gear* and in the Description box write the appropriate description as shown below.


ODBC Microsoft Access Setup

Data Source Name: gear

Description: Gear Internet ODBC Data Source

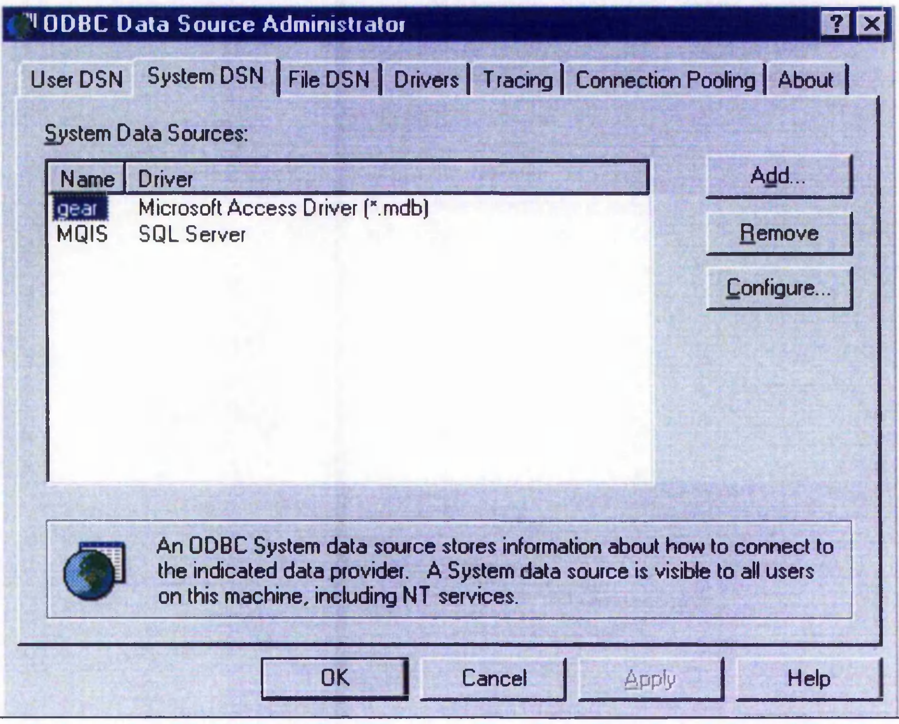
Database

Database: C:\database\gear.mdb

System Database

(● None) ○ Database:

6. Clicking on the *OK* button, will take you back to *ODBC Data Source Administrator*. The *gear* database has now appeared in the *System Data Source*, as shown below. Clicking on the *OK* button completes the process.


ODBC Data Source Administrator

System Data Sources:

Name	Driver
gear	Microsoft Access Driver (*.mdb)
MQIS	SQL Server

An ODBC System data source stores information about how to connect to the indicated data provider. A System data source is visible to all users on this machine, including NT services.

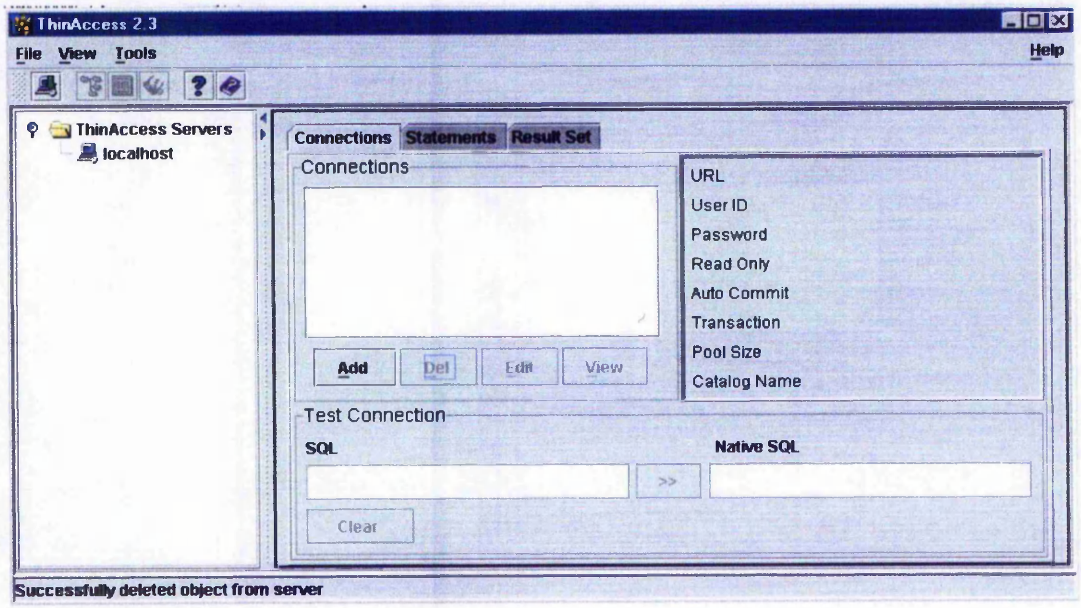
Appendix E

Pre-configuring Connections and Statements

In this section, the process of pre-configuring connections and statements are described using *ThinAccess* driver administrator for the *gear* ODBC database.

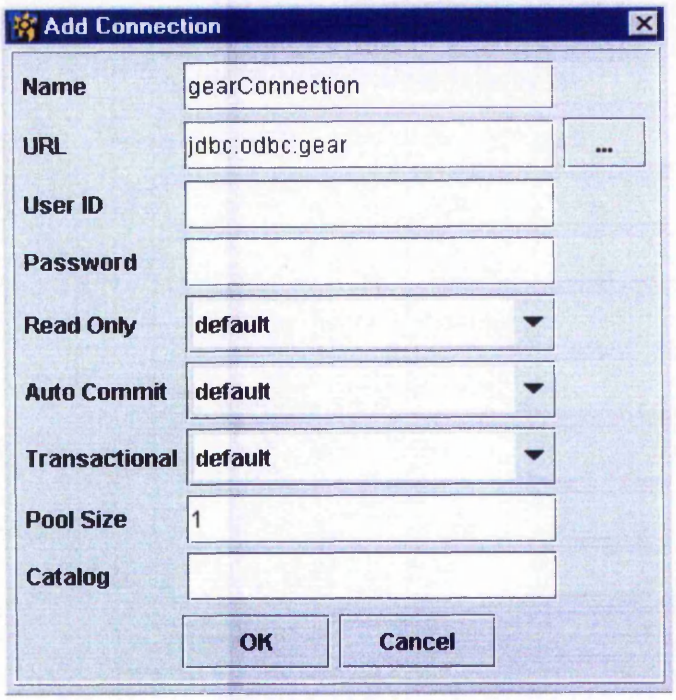
Pre-configuring the connection:

1. Open *ThinAccess administer* and click on the *Add* button in the *Connections* tab.



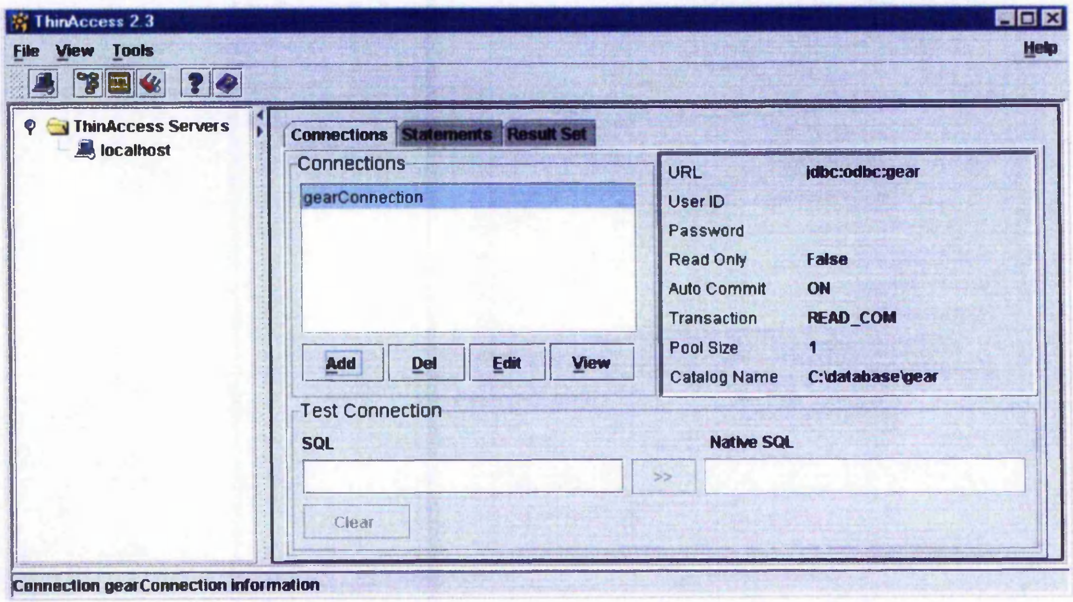
Successfully deleted object from server

2. In the *Add Connection* dialog box, define the name and URL of the connection as shown below:


Add Connection

Name	gearConnection
URL	jdbc:odbc:gear
User ID	
Password	
Read Only	default
Auto Commit	default
Transactional	default
Pool Size	1
Catalog	

3. Click on the *OK* button. The name of the connection appears in the *Connections* box and this shows that the connection has been configured successfully, see below:

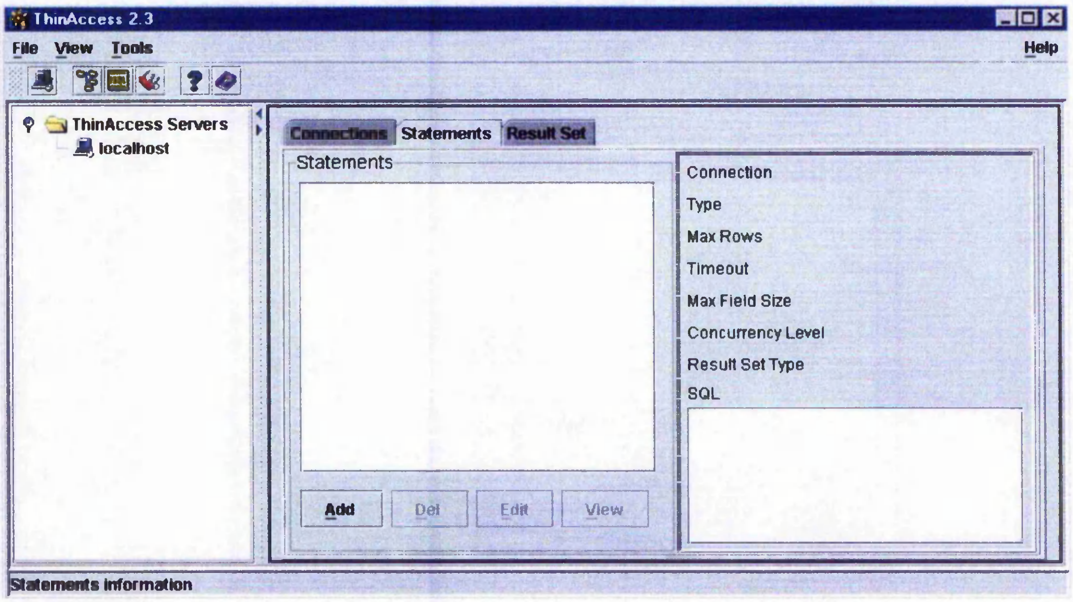

ThinAccess 2.3

URL	jdbc:odbc:gear
User ID	
Password	
Read Only	False
Auto Commit	ON
Transaction	READ_COM
Pool Size	1
Catalog Name	C:\database\gear

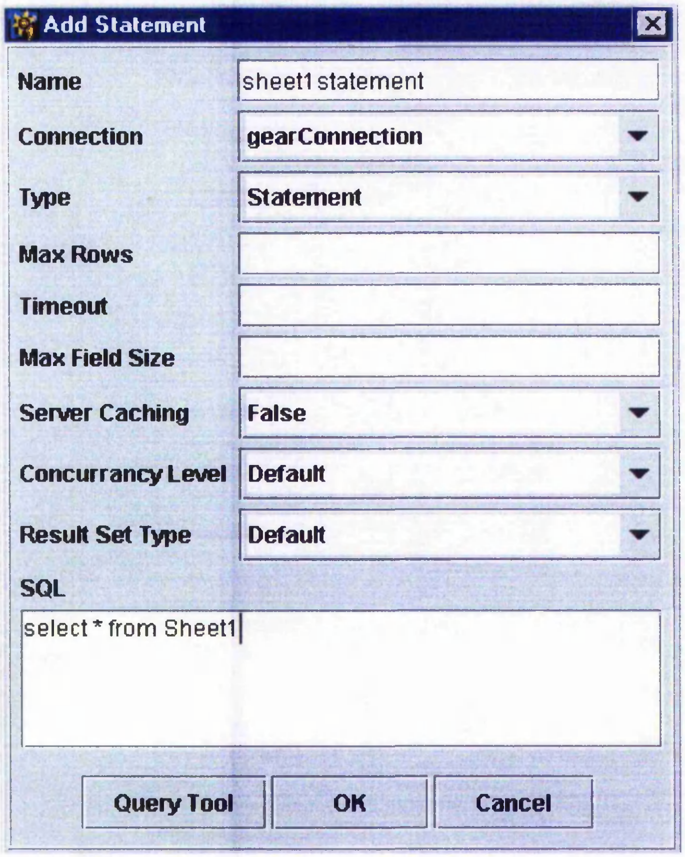
Connection gearConnection information

Pre-configuring the statements:

1. In the *Statements* tab, click on the *Add* button.

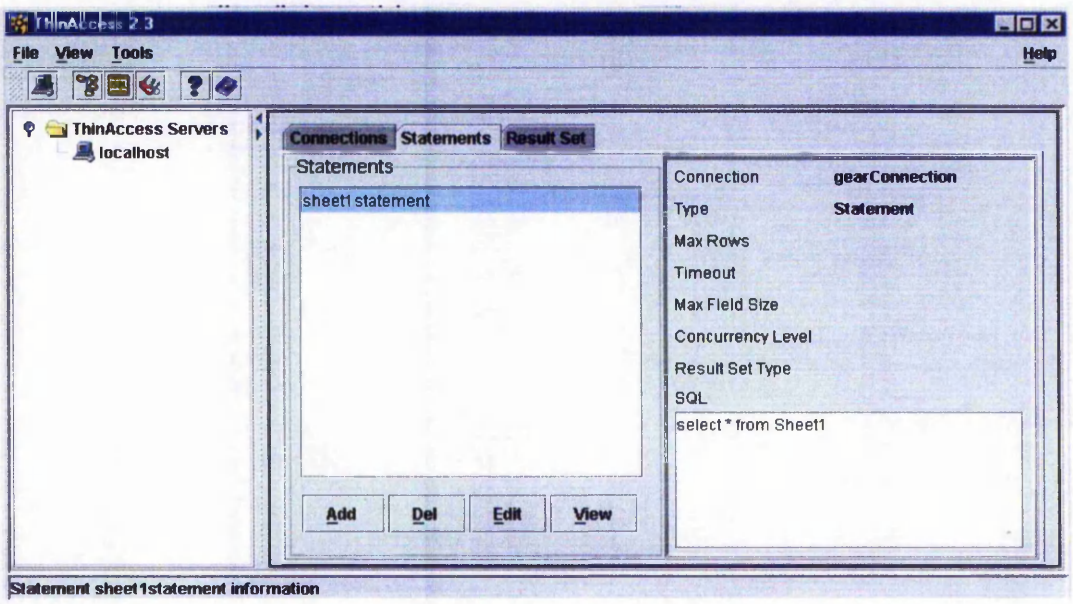


2. In the *Add Statement* dialog box, define the name for the statement. Select the connection name from the list and in the *SQL* text area, type the appropriate SQL command, see below:



Name	sheet1 statement
Connection	gearConnection
Type	Statement
Max Rows	
Timeout	
Max Field Size	
Server Caching	False
Concurrancy Level	Default
Result Set Type	Default
SQL	select * from Sheet1

3. Click on the *OK* button. The name of the statement appears in the *Statements* box and this shows that the statement has been configured successfully, see below:



sheet1 statement

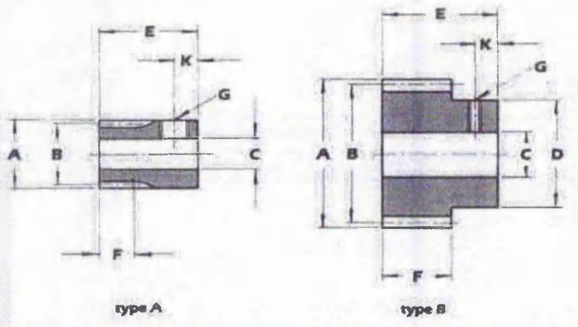
Connection	gearConnection
Type	Statement
Max Rows	
Timeout	
Max Field Size	
Concurrency Level	
Result Set Type	
SQL	select * from Sheet1

Statement sheet1statement information

Appendix F

An Example Sheet of Davell Catalogue

An example of the Davall gear catalogue (Davall, 2000) is given here. This is for the case when module is 0.5 and material is steel and brass. In this figure, a drawing of type A is given, that has a curve feature. This has not been implemented in the development of dynamic creation of DXF files.



DAVALL GEARS

Spur Gears

0·5 module

Steel and Brass

GEARS

Preferred sizes are shown in bold type

No. Teeth	Type	Outside dia. A	Pitch dia. B	Bore dia. C	Boss dia. D	Overall length E	Facewidth F	Setscrew G	K	Catalogue No. Steel	Catalogue No. Brass
12	**A**	**7·0**	**6·0**	**4**	**—**	**14**	**4**	**M3**	**3**	**MA05-12-S**	**MA05-12-B**
15	**A**	**8·5**	**7·5**	**4**	**—**	**14**	**4**	**M3**	**3**	**MA05-15-S**	**MA05-15-B**
16	**A**	**9·0**	**8·0**	**4**	**—**	**14**	**4**	**M3**	**3**	**MA05-16-S**	**MA05-16-B**
20	**A**	**11·0**	**10·0**	**4**	**—**	**14**	**4**	**M3**	**3**	**MA05-20-S**	**MA05-20-B**
24	**B**	**13·0**	**12·0**	**5**	**10**	**10**	**4**	**M3**	**3**	**MA05-24-S**	**MA05-24-B**
25	B	13·5	12·5	5	10	10	4	M3	3	MA05-25-S	MA05-25-B
30	**B**	**16·0**	**15·0**	**5**	**10**	**10**	**4**	**M3**	**3**	**MA05-30-S**	**MA05-30-B**
32	**B**	**17·0**	**16·0**	**5**	**10**	**10**	**4**	**M3**	**3**	**MA05-32-S**	**MA05-32-B**
35	B	18·5	17·5	5	10	10	4	M3	3	MA05-35-S	MA05-35-B
36	**B**	**19·0**	**18·0**	**5**	**10**	**10**	**4**	**M3**	**3**	**MA05-36-S**	**MA05-36-B**
40	**B**	**21·0**	**20·0**	**5**	**10**	**10**	**4**	**M3**	**3**	**MA05-40-S**	**MA05-40-B**
45	**B**	**23·5**	**22·5**	**5**	**10**	**10**	**4**	**M3**	**3**	**MA05-45-S**	**MA05-45-B**
48	**B**	**25·0**	**24·0**	**5**	**10**	**10**	**4**	**M3**	**3**	**MA05-48-S**	**MA05-48-B**
50·	**B**	**26·0**	**25·0**	**6**	**12**	**12**	**4**	**M3**	**3**	**MA05-50-S**	**MA05-50-B**
55	B	28·5	27·5	6	12	12	4	M3	3	MA05-55-S	MA05-55-B
60	**B**	**31·0**	**30·0**	**6**	**12**	**12**	**4**	**M3**	**3**	**MA05-60-S**	**MA05-60-B**
64	**B**	**33·0**	**32·0**	**6**	**12**	**12**	**4**	**M3**	**3**	**MA05-64-S**	**MA05-64-B**
65	B	33·5	32·5	6	12	12	4	M3	3	MA05-65-S	MA05-65-B
70	**B**	**36·0**	**35·0**	**6**	**12**	**12**	**4**	**M3**	**3**	**MA05-70-S**	**MA05-70-B**
72	**B**	**37·0**	**36·0**	**6**	**12**	**12**	**4**	**M3**	**3**	**MA05-72-S**	**MA05-72-B**
75	**B**	**38·5**	**37·5**	**6**	**12**	**12**	**4**	**M3**	**3**	**MA05-75-S**	**MA05-75-B**
80	**B**	**41·0**	**40·0**	**6**	**12**	**12**	**4**	**M3**	**3**	**MA05-80-S**	**MA05-80-B**
84	**B**	**43·0**	**42.0**	**6**	**12**	**12**	**4**	**M3**	**3**	**MA05-84-S**	**MA05-84-B**
85	B	43·5	42·5	6	12	12	4	M3	3	MA05-85-S	MA05-85-B
90	**B**	**46·0**	**45·0**	**6**	**12**	**12**	**4**	**M3**	**3**	**MA05-90-S**	**MA05-90-B**
95	B	48·5	47·5	6	12	12	4	M3	3	MA05-95-S	MA05-95-B
96	**B**	**49·0**	**48·0**	**6**	**12**	**12**	**4**	**M3**	**3**	**MA05-96-S**	**MA05-96-B**
100	**B**	**51·0**	**50·0**	**6**	**12**	**12**	**4**	**M3**	**3**	**MA05-100-S**	**MA05-100-B**
108	**B**	**55·0**	**54·0**	**6**	**12**	**12**	**4**	**M3**	**3**	**MA05-108-S**	**MA05-108-B**
120	**B**	**61·0**	**60·0**	**6**	**12**	**12**	**4**	**M3**	**3**	**MA05-120-S**	**MA05-120-B**
127	**B**	**64·5**	**63·5**	**6**	**12**	**12**	**4**	**M3**	**3**	**MA05-127-S**	**MA05-127-B**

Material	
Steel	BS970 Pt.1 1991 080M15 (EN32)
	or BS970 Pt.3 1991 230M07 (EN1A)
Brass	BS2874:1986 CZ121

All dimensions in millimetres.
Pressure angle 20°.
Manufactured to DIN 3962, DIN 3963, DIN 867, quality 8gS".
Bore tolerances to BS. 4500: 1969, H8

General tolerance unless otherwise stated ± 0·25.
These gears are not hardened.

For guidance on the gear tolerances refer to the Technical Section of this catalogue.

24-hour boring and keywaying service.

25

Appendix G

Publications Arising from this Research

1. Su, D and Amin,N., 2001, "A CGI-based approach for remotely executing a large program for integration of design and manufacture over the Internet", International Journal of Integrated Manufacturing, Vol. 14, pp. 55-65.

2. Amin, N. and Su, D., "Enhancement of speed and efficiency of an Internet-based gear design optimisation", Submitted to the International Journal of Automotive Technology and Management (IJATM),Special Issue on Integration and Co-operation Tools for World-Wide Manufacturing.

3. Su, D., Ji, S., Amin, N. and Hull, J.B., "Multi-user Internet environment for gear design optimisation", Submitted to International Journal of Computer Integrated Manufacturing.

4. Amin, N. and Su, D., 2000, "Gear Design Optimisation over the Internet", in: D. Su(ed) 'Internet-Based Engineering – Applications and Case Studies', ISBN: 1-84233-021-7, Professional Engineering Publishing Limited, Nottingham, pp 61-80.

5. Luo, J., Liu, Y. and Amin, N., 2000, "Comprehensive Techniques for Network Security Defence and Transmission", In: D. Su (ed), Internet-Based Engineering – Applications and Case Studies, ISBN: 1-84233-021-7, Professional Engineering Publishing Limited, Nottingham, pp 149-169.

6. Su, D., Ji., S., Amin, N. and Chen, X., 2001, "A framework of web support for collaborative design", Proceedings of the 5th International Conference on Frontiers of Design and Manufacturing, Dalian, P.R.China, pp 492-498.

7. Su, D., Ji, S., Amin, N. and Hull, B., 2001, "An Internet-based system of gear design optimisation using Java servlets", Proceedings of 4th International Conference on Computer-Aided Industrial Design and Conceptual Design, Jinan, P.R.China, pp 30-36.

8. Amin, N. and Su, D., 2001, "Development of an Online Gear Catalogue using Java Database Connectivity", Proceedings of the International Conference on Mechanical Transmissions, China Machine Press, Chongqing, pp. 137-141.

9. Amin, N. and Su, D., 2000, "Utilization of Java Applets for Gear Optimisation", Proceedings of the International Conference on Gearing, Transmissions, and

Mechanical Systems, Professional Engineering Publishing Limited, London, pp 425-434.

10. Luo, J., Liu, Y. and Amin, N., 2000, "A network security system model – design and implementation", Proceedings of the International Conference on Gearing, Transmissions, and Mechanical Systems, Professional Engineering Publishing Limited, London, pp 425-434

11. Amin, N. and Su, D., 1999, "Development of Internet-based Approach for Optimising Gear Design in a Multi-user Environment", in: Advances in Concurrent Engineering – CE99, P.K. Chawdhry et al (eds.), Technomic Publishing Company, Pennsylvania, 368-374.

12. Amin, N. and Su, D., 1998, "A method for Dynamically Producing CAD Drawing Files within a Multimedia-Based Electronic Catalogue", in: Advances in Manufacturing Technology – XII, Baines, et al (eds.), Professional Engineering Publishing Limited, London, pp 329-334.