Multi-user Internet environment for gear design optimization

Daizhong Su*, Shuyan Ji, Nariman Amin and J B Hull
(* Corresponding author, E-mail: Daizhong.su@ntu.ac.uk)
Department of Mechanical and Manufacturing Engineering,
The Nottingham Trent University, Burton Street, Nottingham NG1 4BU, U.K

Abstract A Web based multi-user system has been developed to remotely execute a large size software package via the Internet. The software implements genetic algorithm to optimize the design of spur and helical gears. To accomplish this, a combination of HTML, JavaServlets, JavaApplets, JavaScript and HTTP protocol has been employed.

Keywords: Design optimization, Java applet, Java servlet, JavaScript, Internet

1 Introduction

The rapid development in technology and globalization of the world wide web has willingly or unwillingly pushed companies into the world of Internet. Global competition has dramatically increased the need for companies to produce high-quality, competitively priced products both quickly and efficiently. Within such an environment, success reflects the ability of a company to quickly respond to customers' requirements and design, prototype, manufacture, test, and deliver a high-quality product to the market in the least possible time (Peng and Hall 2000). Customers' awareness of the latest trend in the technology is growing, due to the better communication and the availability of the information through the Internet. This has resulted in higher demands from the suppliers and developers. It is essential for companies to use the media of Internet in order to keep up with the rapidly changing technology, either in terms of exchanging information, or designing and delivering the products.

In the web-application server domain, several methods have been devised to let a Web server process the user input, to run custom programs and to serve dynamic contents in response to

client requests, such as Common Gateway Interface (CGI) and Java Servlet. A CGI program can be written with different languages such as C/C++, Perl, while the Java Servlet is written in the Java language only.

The CGI technology has been used to implement the remote execution of a gear optimization design software package by the School of Engineering at The Nottingham Trent University (Su and Wakelam 1999). In the project, a CGI program, written in C++, enables users to run the software from any part of world and from any machine via the Internet, whether it is UNIX, Windows, etc (Su and Amin 2001).

The speed problem caused by the CGI itself, however, has been explored and has affected the package's utilization for multi-users. In order to resolve the problem, another technology for Web server extension, Java Servlets/Applets has been chosen and has been implemented in this project.

In this paper, the process for implementation of the Internet-based design system using Java Servlets/Applets is described. The utilization of other techniques related to this project such as HTML, JavaScript, Web server and protocol are also presented. In addition, an example is provided demonstrating how these techniques are applied.

2 The gear design optimization package

The original gear design optimization software is a stand-along package. It consists of three parts:

- a friendly graphical user interface(GUI)
- a genetic algorithm(GA) program
- a numerical analysis program for gear strength calculation to the British Standard 436 (1986)

The GUI was developed using Visual BASIC, and the other two were written in C++. The GUI is used for design-data input, setting-up optimization specifications (goals, weight

factors, population size and number of tests), and display of results. The GA program conducts the optimization. The numerical analysis program is invoked by the GA program in the optimization process to calculate the tooth strength. All three parts are fully integrated into a single software environment. The package is used to optimize the design of external spur and helical gears with involute tooth profile.

Up to nine gear design parameters can be optimized, including tooth face width, module, pressure angle, helix angle, rack tip radius, addendum coefficients, addendum modification (tooth profile shift) coefficients, and number of pinion teeth.

There are five optimization objectives including minimizing face width, minimizing center distance (variable center 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 have been considered for the optimization: --1) tooth root bending stress cannot exceed the allowable stress, 2) tooth contact stress cannot exceed the allowable stress, and 3) sliding/rolling speed ratio cannot exceed 3.

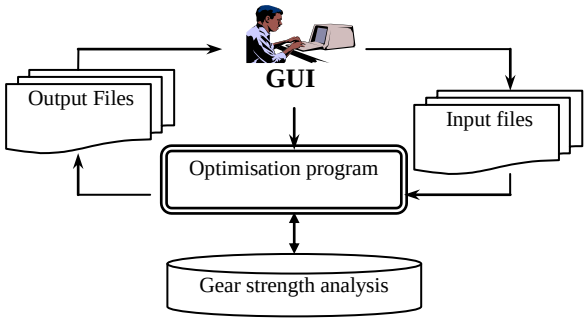


Fig 1

The execution structure of the software is shown in Figure 1. The data inputted by the designer via the GUI are written into the input files. The Visual Basic program then calls the optimization program to perform the genetic algorithm. It will first read the data from the input files and starts the genetic algorithm, whilst invoking the strength analysis program.

After the execution is completed, the results are written into the output files, which are then displayed to the designer.

The optimization process may be time consuming because of a generic algorithm that is known to be computationally expensive. How much time is required depends on the given size of the population, the given number of tests and the selected number of parameters to be optimized.

3 Internet solution

The gear design optimization can not be conducted on the Internet unless the following problems have been resolved:

- How to pass the user's inputs data to the executing package and to send the results to the user.
- How to allow multiple users to run the package at the same time.
- How to allow users to run the existing platform-dependent package from any machine over the Internet.

Besides, the copyright and security problems for the package have to be considered.

To meet the above requirements, a JavaServlet/Applet based system has been developed by the authors. The structure of the system with Servlets is shown in Figure 2.

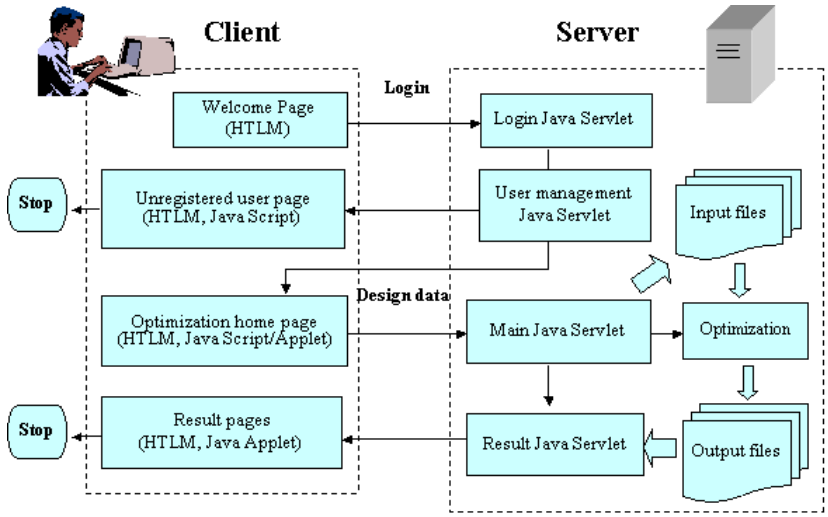


Fig 2

The gear optimization package is located on the server side. A user on the client side could give all the optimization parameters through the interface in the HTML page. Then the parameters are sent to the server which calls the server-side extension such as Servlets. When the user clicks on the submission button, a Servlet program is activated by the HTML code. It parses the data and writes them into the input files and invokes a C++ executable program located on the server. When the execution is completed and output files are created, Servlets would pass the results back to the client.

The gear optimization package resides and runs on the server machine. This is considered to be a secure approach since the program is not downloaded to the user's machine and the client has no access to the source code.

The existing gear optimization package is of large size, is time-consuming and is platform-dependant. The execution of the package on the server depends only on the power and platform of the server-side, and is not relative to the client-side machine. It is possible for users to run the package from any machine whether it is of UNIX or Windows system over the Internet.

A HTML file, instead of Visual Basic program of original optimization package, is used to pass the data from the client to the server which invokes the servlet programs to process them, to call the package and to send back the results to the user. The utilization of the servlet program improves the speed in response to the client request.

The Multi-thread feature of Java makes it possible to support multi-user requests at the same time. In order to support this, the gear design optimization program has been modified. For example, a flag has been used as a symbol when the program is completed and outputs the status information during the running of the package in order for the servlets to control it, and the user information, such as name and address, has been added as the running parameters in

order for the servlets to manage the multiple users. Further details will be described in Section 7.

4 Implementation of The User Interface

The user interface is written in HTML, which is platform-independent and the codes are interpreted locally, i.e., on the client's machine. While the interface of the original gear optimization software was created using Visual Basic and C++, which is platform-dependent and hence is not suitable for the Internet application.

Another important function of the HTML is its features called *form* that make it possible to send information such as design parameters from the user to the server. After the user has inserted the data and pressed the submission button, all the data within the form will be transferred to Servlets located in the server. The Servlet program then performs the processing and sends the results back in response to the user. The page is shown in Figure 3.

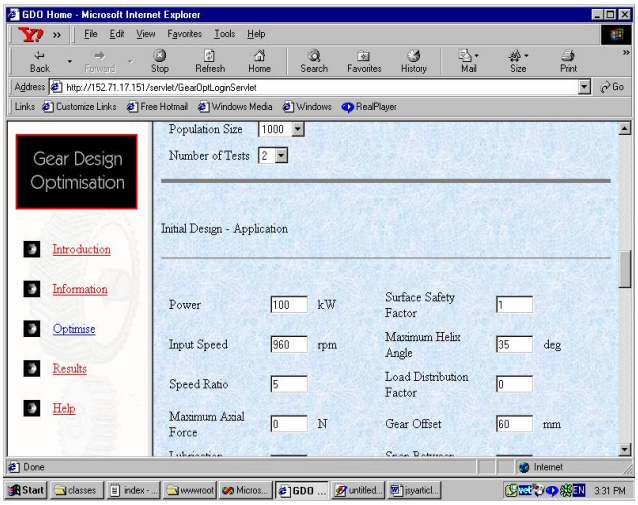


Fig 3

JavaScript, included in the HTML, has been used to offer the HTML dynamical property. Because HTML documents are static, and plain-text files, they do not have the ability to perform any calculation or validity to check the data submitted to the server. To make the web page more dynamic and user-interactive and to perform the data validation, Java Script is

used. Java Script is a scripting language which can be embedded into the HTML code in a text format and 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 main task of the Java Script used in this project is to check the data before passing them to the server. If there is an invalid data, i.e., if any of the data are wrong, are missing or are out of the permitted range, the JavaScript would prevent transfer of the data to the server and display a message box informing the user what the error is. The JavaScript code is embedded into the HTML document using the SCRIPT tag. The user name and password have to be entered in the welcome page of HTML file to be sent to the *login servlet* before data input. To enhance the function of the user interface, Java applets are also embedded in the HTLM. The detail information of the Java applets is given in setion 6.

5 Servlets for this Project

Servlets are some Java classes located on the server and can accept the request in a HTML file from the user. They retrieve the data in the format of a long string from the form and subsequently process the data and send the results back to the user. The following types of servlets are utilized in this project:

5.1 Login servlet

Login servlet is designed to check the user identity, including username and password, to ensure that only a legal user is allowed to access the optimization package form the web site.

5.2 User management servlets

User management servlets are designed to maintain user information, and to add or delete the user from the list of the registered users. If a user's registration is expired, the servlets will delete the user name automatically from the list.

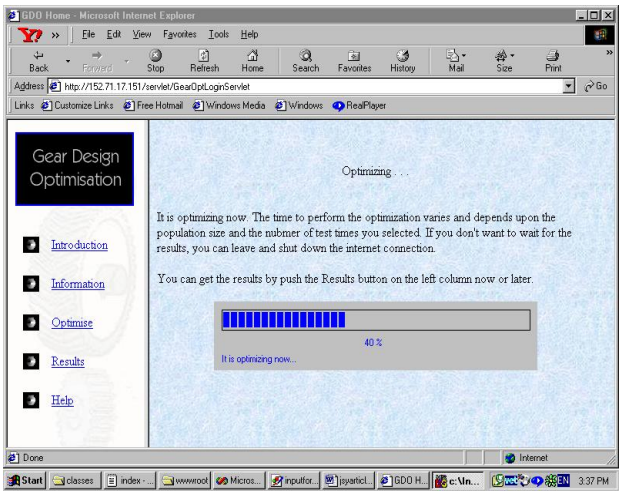


Fig 4

5.3 Main servlet

The main tasks of the servlet are: (1) to accept the request information by the specific protocol and to retrieve the data from the user, for which the data is sent via the HTML *form* using POST method and is extracted using the *getParameter()* method; (2) to execute some necessary calculations for the data using Java API; (3) to create input data files for the gear design optimization using *FileWriter* and *PrintWriter* classes of Java; (4) to run the optimization executing program, using *Process* and *Runtime.getRuntime().exec()* method; (5) to return to the page with a progress bar showing the optimization progress, as shown in Figure 4. The progress bar is the execution of the Java Applet program, which is inserted into the HTML page returned by the servlet. It reads the status data output by the optimization program dynamically and displays the graphical process stripe which shows the progress of the execution.

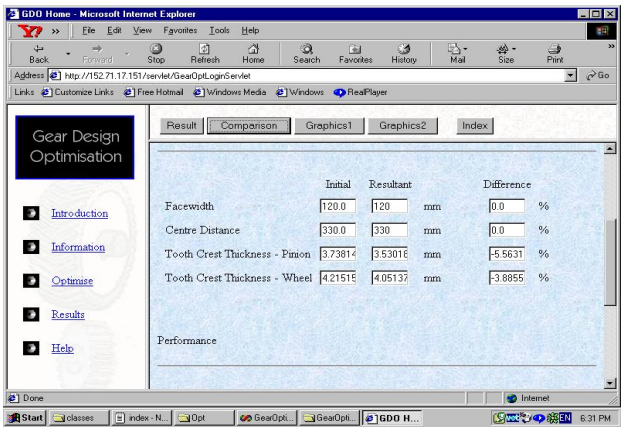


Fig 5

5.4 Result Servlets

When the user presses the button *Result* on the left hand side of the screen, and if the optimization program is not completed, then the progress bar is displayed, otherwise the result page, shown in Figure 5, will appear on the user's screen. As shown in Figure 5, the result servlets, linked to *Result* and *Comparision* on the top, first read the results from the output files created by the execution program, using *FileReader* and *BufferedReader*, and then send the results directly back to the user, using *response.getWrite*, or create result file in the HTML file using *FileWriter* and *PrintWriter*. The Applet program, linked to the button *Graphics1* and *Graphics2* on the top, is in charge of processing the data files and displaying the result curve graphically to the client side.

6 Java Applets

Java applets are located on the server and can be viewed using a World Wide Web browser. They 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 can cause system damage. The restrictions on applets include the following (Lemay et al 1996):

- Applets cannot read or write to the client's file system, which means they cannot delete files or test to see what programs have been 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 Dynamic Link Libraries.

However, these restrictions do not limit the 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.

6.1 The Progress Bar

The genetic algorithm is known to be computationally expensive. The time taken to perform the optimization 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 output when the optimization process performing the 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 optimization 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 optimization 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.

A data file is created by the optimization package on the server. Data within this file indicates the progress of the optimization in converging on a solution. While the optimization program is being executed, the data within this file is continuously being updated. The data is utilized 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 optimization program is running on the remote server, and the value of the data file that determines the progress of the optimization 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.

6.2 Data Retrieval

All the data files that are produced by the optimization 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 6 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.

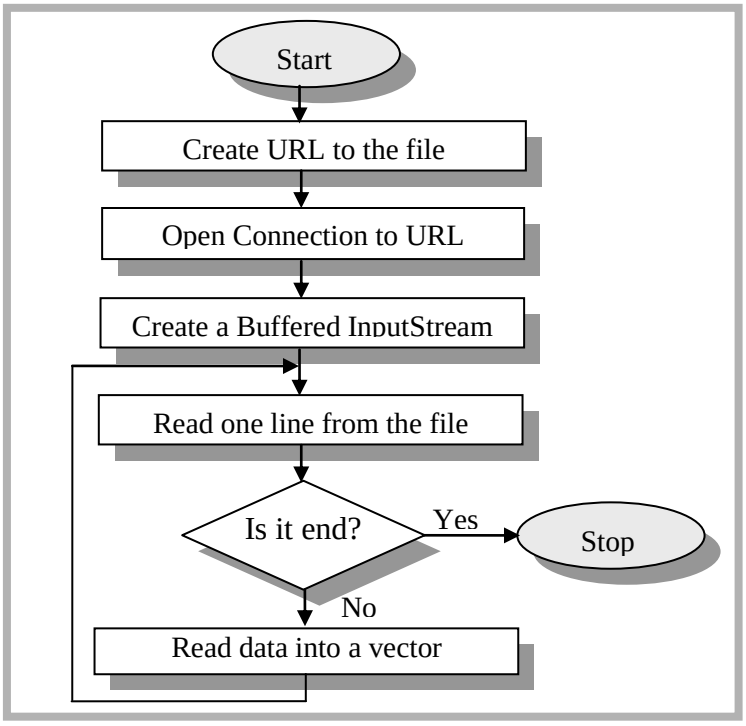


Fig 6

6.3 Optimization Graph

When the execution of the optimization program is completed, in addition to accessing the resultant data, the user has the option of displaying the graph of the optimization. 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 7.

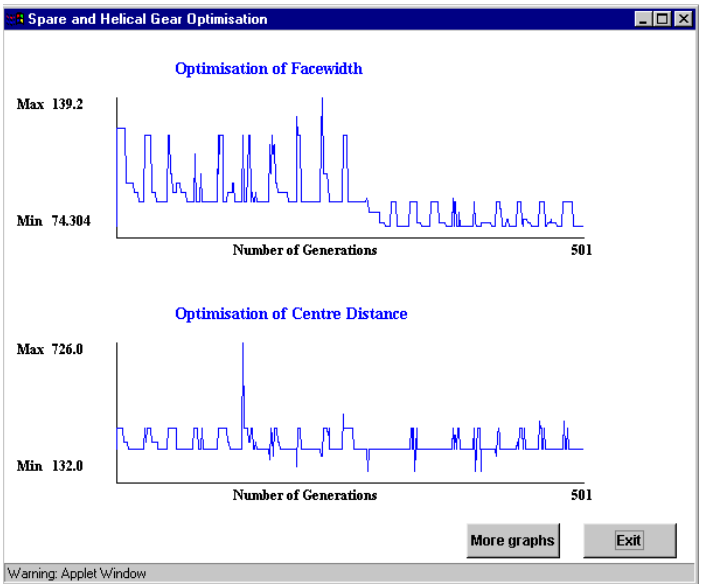


Fig 7

While the optimization 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 [4]. The first step is to read the data from the relevant file and save them into a vector. Because vector is an array of object, 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 generations 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 8 for the flow diagram.

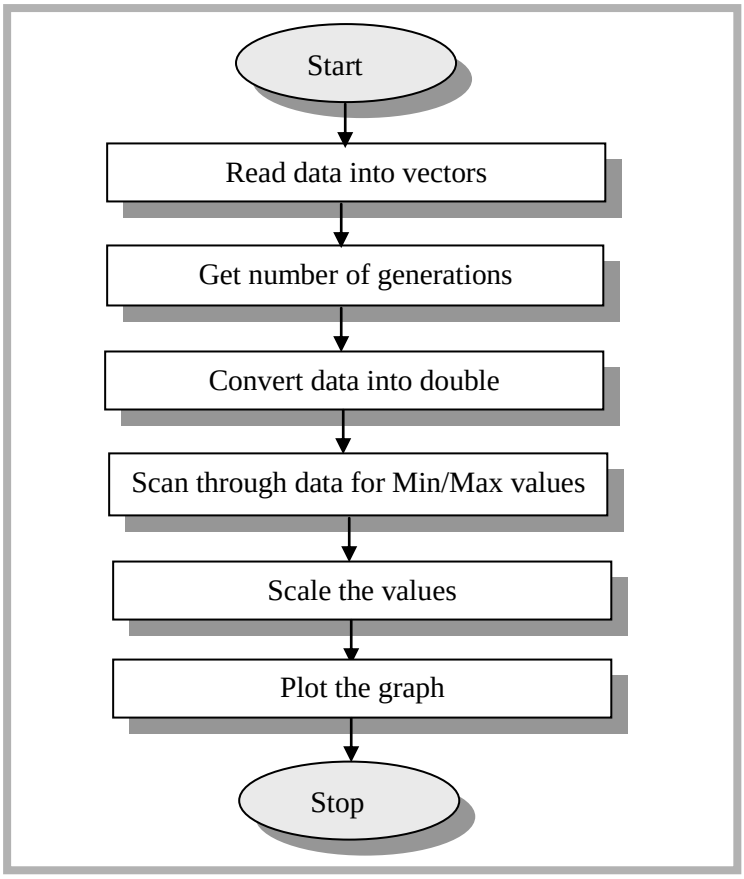


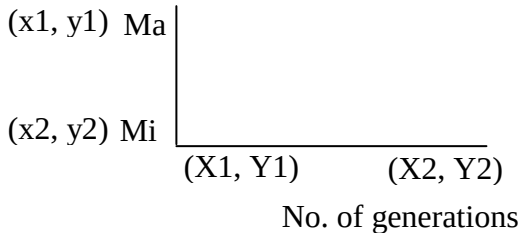
Fig 8

6.4 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).

6.5 Results Page

Figure 9 shows the *Optimization Results* page, containing the progress bar applet while the optimization 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 optimization program is still running. 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 optimization program is completed, i.e. when the progress bar reaches the far right side.

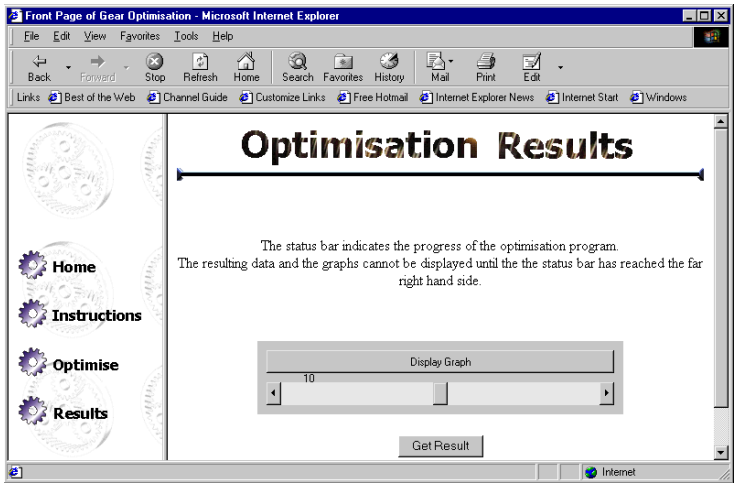


Fig 9

7 Multi-user environment

To run the system, the servlet is requested first and located into the Web server's memory space. Subsequent client requests for the servlet result in calls to the servlet instance in memory. For every request servlet uses the thread to process multiple requests simultaneously. There is no interference between different users.

But the optimization program needs to read input files and to write results to a series of output files and works on a default directory according to the original package. This will request the user to write the data input files to and to read the output files from the same directory even though the different users use different servlet instances. A possible situation may create a problem as shown in Figure 10. It cannot guarantee that the optimization program read from the correct input files and send the correct results back to the user. This is a crucial point which has to be dealt with.

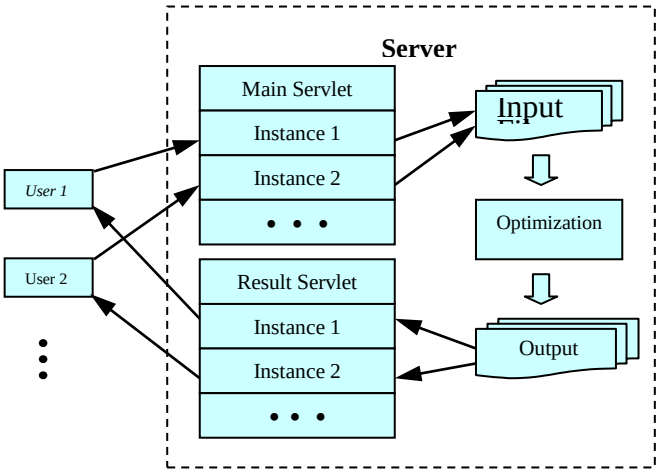


Fig 10

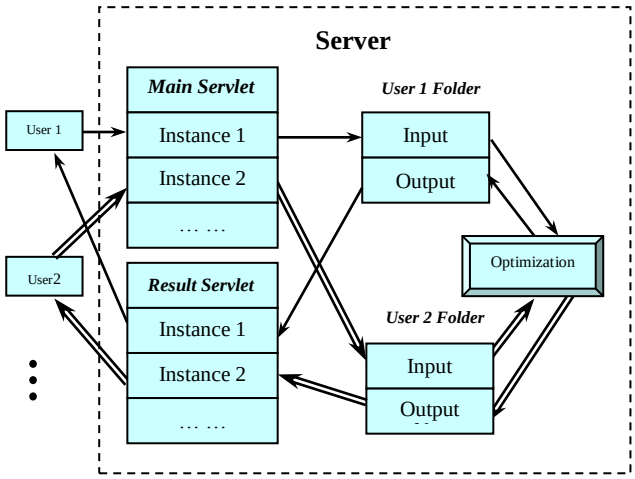


Fig 11

A Multi-user Environment is developed to avoid the above conflict, as shown in Figure 11 where each user is given an identity and a space on the server, so that all the interactions could be within the allocated area. In this way, there would be no conflicts between different users.

When the first time the users log in, they are required to input their usernames and passwords. If permission is given, then the servlet creates a directory on the server under the JRun directory. This is where all the interactions occur for the user. The input files for and output files from the optimization program, HTML files and other information relative to the user are

used in the directory while servlets are respectively in the specific directory of JRun. There is no need for the copies of servlets and optimization program to be located under each user's directory because the multi-threads of servlets support multi-users.

After the optimization program is completed, the results are written into output files in that directory. So when the user presses the result button system will call another servlet (output servlet) which will read the correct files in the directory and will display the results page and accept the directory information. If the user presses a button for displaying a curve of the process of a particular parameter's optimization, a relevant applets program is invoked, and then the curve appears on the user's screen. The curve is of the results of the user identified by the specific username information passed by the servlets to applets. When the user logs off, the servlet deletes all the files in the user's directory.

8 Conclusion

A structure based on a combination of HTML, JavaScript, CGI and Java programming has been developed, which implements gear design optimization over the Internet. The optimization software utilizes genetic algorithm to search for the best possible solution.

The utilization of Java Servlets improved the performance in response to the multi-users environment. The successful application of Java Servlets instead of CGI provides a wider application region. As demonstrated in this project, various tasks can be implemented by Java Servlets, such as sending different kinds of the existing technique resources onto the Internet and implementing the remote design of cross-region, communicating commercially about engineering technique resources and enlarging the Internet service of a technique site, and so on.

It was described that the optimization process is very time consuming, due to the genetic algorithm. An applet is developed that monitors the progress of the optimization program and

displays a progress bar on the client's machine. This would help the users to estimate the time required for the completion of the optimization. The paper also describes the development of Java applets that display the graph of the optimization. The performance of the designs is given in the graphs, indicating the trend of the search and the levels of performance.

The combination of Java servlets and applets and the necessary modification of the gear design program make it possible to master the execution of such a large-size and time-consuming program, to provide the powerful processing of the results and to display the results geographically to the user.

The method is very important for an agile manufacturing environment. It enables the geographically dispersed engineers to optimize a gear design and obtain the results successfully from any part of the world. Any changes or improvement to the optimization program needs to be done only on the main copy of it, which is located on the server and the users would have access to the latest version at any given time. This is cost effective and also enables the company to stay on the competitive level.

References

British Standard 436, Spur and helical gears: Part 3. Method for calculation of contact and root bending stress limitations for metallic involute gears, British Standard Institution, 1986.

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

Peng, Q. and Hall, F. R., 2000, Enhancement of the performance of SMEs through Web-centred virtual manufacturing. In D. Su (ed), Internet-Based Engineering: Applications and Case Studies, ISBN1-84233-021-7, (Nottingham: The Nottingham Trent University and Professional Publishing Ltd), pp131-148.

Su, D. and Amin, N. 2001, A CGI-based approach for remotely excusing a large program for integration of design and manufacturing over the Internet, International Journal of Computer Integrated Manufacturing, Vol 14, No.1, pp 55-65.

Su, D. and Wakelam, M., 1999, 'Evolutionary optimisation within an intelligent hybrid system for design integration', Artficial Intelligence for Engineering Design, Analysis and Manufacturing Journal (ISSN 0890-0604), No.5, November, 1999, Cambridge University Press, pp 351-363.

Figure list