한국해양공학회지 제24권 제4호, pp 66-71, 2010년 8월 (ISSN 1225-0767)

인터넷 기반 선박용 프로펠러 설계 및 해석 시스템 개발

장현길* · 안병권* · 문일성** · 이창섭*
*충남대학교 선박해양공학과
**한국해양연구원 해양시스템안전연구소

Development of Internet-Based Marine Propeller Design and Analysis System

Hyun-Gil Jang*, Byoung-Kwon Ahn*, Il-Sung Moon** and Chang-Sup Lee*
*Dept. of Naval Architecture and Ocean Eng., Chungnam National Univ., Daejon, Korea
**Maritime and Ocean Engineering Research Institute, KORDI, Daejon, Korea

KEY WORDS: Internet 인터넷, Client 클라이언트, Server 서버, Propeller design 프로펠러 설계, Propeller performance prediction 프로펠러 성능추정

ABSTRACT: *Numerical prediction of propeller performance plays an important role in a marine propeller design process. Program developers are consistently trying to improve diminish predicted errors, and program users need to keep up with the latest ones with minimum expenditure of time and money. We have developed an internet based design system in which clients can design propellers with remote access. In this paper, optimized Internet based Propeller Design and Analysis System (iProDAS) for transferences of the massive data is presented, and a sample design using iProDAS is examined.*

1. 서 론

일반적인 공학적 설계 또는 해석 과정에는 수많은 프로그램이 연속적, 반복적으로 활용된다. 마찬가지로 그동안 선박용 프로펠러의 설계과정도 설계자의 경험과 수많은 요소 프로그램을 활용해야 하며 매 단계마다 설계자의 공학적 판단을 필요로 해왔다. 본 연구 그룹은 선박용 프로펠러에 대한 이론해석 및 설계뿐만 아니라 이 과정을 체계화하여 개인용 컴퓨터의 Windows 환경에서 설계를 수행할 수 있도록 한 프로펠러 설계 및 해석 패키지 ProDAS(Propeller design and analysis system)를 지속적으로 개발해 왔다(정인숙 등, 1996; 정인숙 등, 1997; Lee and Cho, 2001). 현재 조선공업 1위 국가인 우리나라 대부분의 조선소에서 이 해석 패키지를 사용하여 선박용 프로펠러를 설계하고 있다. 그러나 이러한 패키지의 한계 중 하나는 모든 계산이 설계자가 직접 설계현장의 컴퓨터에서 개별적으로 수행된다는 점이었다.

설계 또는 해석과정에 사용되는 소프트웨어가 설계자가 휴대하고 있는 컴퓨터에 직접 설치되어 있지 않아도 원격으로 계산을 수행할 수 있는 기능이 필요한 시대가 되었으며, 인터넷의 보급으로 이를 구체화할 수 있는 방법이 개발되었다. 즉 서버-클라이언트 방식에 의해 프로그램의 역할이 분리가 가능해 진 것이다. 이동 중이거나 원거리에 있는 설계자는 인터넷망 접속이 가능한 장소에서 클라이언트 프로그램만을 구동함으로써 설계를 수행할 수 있게 된다. 이 방식의 장점은 핵심 프로그램을 직접 휴대하지 않음으로써 휴대용 컴퓨터의 분실에 따른 프로그램 유출의 위험성을 방지할 수 있음은 물론이고, 중, 소형 조선소와 같이 해석에 필요한 모든 프로그램의 보유와 유지관리 및 수행 인력이 제한되는 경우에 클라이언트만을 소유함으로써 예산을 크게 절감할 수 있는 등 이점이 상당히 많다. 이 경우는 서버에 설치된 프로그램은 서버 관리자에 의해 정기적으로 최신 버전의 프로그램들로 업그레이드됨으로써 프로그램의 성능 향상에 따라 매번 프로그램을 추가 구입해야하는 예산절감 외에, 비전문가 프로그래머의 입장에서는 프로그램 내용에 필요 이상 관여할 필요가 없고 성능이 향상된 최신 프로그램을 활용할 수 있다는 장점이 있다. 프로펠러 설계와 같은 교육과정에서는 특히 교육생들에게 설계 프로그램 제공에 따른 부담을 없애면서도 프로그램을 활용하여 해석 및 설계를 수행할 수 있다는 큰 이점이 있다.

이러한 이점을 고려하여 이왕수 등(2003)은 인터넷 기반 프로펠러 설계 시스템을 개발하여 발표한 바 있다. 그러나 인터넷을 이용한 설계과정 개발의 첫 시도로서 실제로 수많은 복잡한 설계과정을 일일이 확인하며 전체 시스템을 개발하는 데 따른 어려움이 있었고, 통신 프로토콜(Protocol) 및 기본적인 동작을 설계하는데 그쳤다. 특히 용량이 큰 데이터의 송수신에 따른 문제점, 여러 사용자의 동시 사용에 따른 문제점, 하드웨어에 CPU 성능에 따른 프로그램 동작의 불안정성 등의 문제점이 있었다.

교신저자 이창섭: 대전광역시 유성구 궁동 220, 042-821-7762, csleepro@cnu.ac.kr
본 연구는 2007년 제주도에서 개최된 대한조선학회 추계학술대회에 발표된 논문을 근간으로 하고 있음을 밝힙니다.

본 연구를 통해 이러한 초기 버전의 문제를 해결하고 인터넷 기반 프로펠러 설계해석 시스템(이하 iProDAS, Internet-based propeller design and analysis system) 개발을 완성하였다. 본 논문에서는 iProDAS에 이용된 인터넷 통신 구현 부분의 개선점을 설명하고, 선박용 프로펠러 설계과정 계산(예)를 통한 고찰을 수행하고자 한다.

2. TCP 통신의 실제 구현

서버-클라이언트 분리에 의한 프로그래밍은 인터넷을 통해 원격의 두 컴퓨터가 정보를 교환할 수 있는 통신 방법을 먼저 이해할 필요가 있다. 이왕수 등(2003)은 서버-클라이언트로 설계된 프로그램이 소켓함수에 의해 두 컴퓨터가 연결되는 과정을 설명하고 있으며, 두 컴퓨터가 TCP(Transmission control protocol, 전송제어규약)에 의해 연결되며 양방향으로 스트림(Stream) 방식의 통신이 가능함을 보였다. TCP통신의 기본사항은 부록에 정리하였다.

기존 연구에서는 대부분의 핵심기능이 구현되었음에도 불구하고 한정된 네트워크 환경 및 입출력 데이터 특성만을 고려하여 개발되었다. 이로 인해 대용량의 파일을 전송할 경우 클라이언트에서 일부 파일을 완전히 전송받지 못하고 누락되는 치명적인 문제가 발견되기 시작하였다. 본 논문에서는 이왕수 등(2003)이 제안한 스트리밍 프로토콜을 수정하는 과정을 먼저 보이기로 한다.

2.1 서버 측 패킷 전송 구현부분

최대 패킷사이즈는 일정한 크기(경험에 의해 결정한 크기 1024byte)로 정의되어 사용된다. 따라서 최대 패킷사이즈를 넘는 데이터를 전송할 경우 패킷을 분할해서 보내야 한다. 기존에 제시된 패킷의 헤더 구성방식(Fig. 1, before)은 전송되고 있는 패킷이 대용량 파일의 부분 패킷일 경우 수신측에서 부분 패킷들을 결합할 기준이 없으므로 처음 도착한 패킷만을 처리하고 나머지 패킷은 처리되지 않거나 다른 패킷과 결합되어 의도하지 않은 방향으로 처리될 것이다 . 그러나 개선된 방식(Fig. 1, current)에서는 헤더에 전송플래그를 추가하여 파일의 최초 패킷을 1로 표시하고 연속 패킷을 0으로 표시하면, 수신측에서 전송플래그를 기준으로 데이터를 하나의 파일로 결합할 수 있다. 또한 데이터의 안정성을 강화하기 위하여 헤더에 패킷사이즈 외에도 전체 파일사이즈를 기록해 전송량을 체크하도록 하였다.

2.2 서버 측 패킷 전송 스레드 동기화 문제

기존의 방식은 결과파일의 전송 시 Process스레드(계산 및 결과처리를 수행)와 IO스레드(네트워크로의 입출력 수행)의 두 개의 스레드가 중첩되어 수행되는 문제가 있다. 즉, 데이터를 패킷단위로 분할하여 전송 요청하는 Process스레드와 전송완료 결과를 체크하여 전송되지 못한 부분을 재전송하도록 요청하는 IO스레드의 실행순서가 제어 되지 않아 데이터의 순서가 뒤바뀌게 된다. 이러한 문제점을 해결하기 위해 스레드 동기화(Thread synchronization)기법을 사용하였으며, 본 연구에서는 스레드의 실행 순서를 제어하기위해 세마포어(Semaphore) 오브젝트를 사용하였다. 예를 들면 Fig. 2에서 보는 바와 같이 첫 번째 패킷의 전송요청이 완료되면 IO스레드에서는 완료 결과로써 전송되지 못한 ⓑ를 파악하여 추가 전송을 시작한다. 동시에 Process스레드는 두 번째 패킷인 ⓒ를 보내려고 할 것이다. 이 때 ⓒ는 ⓑ가 전부 전송되고 크기가 0이 될 때까지 대기할 수 있도록 하였다.

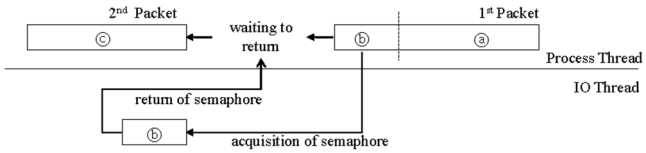


Fig. 2 The transmission sequence of the packet which uses a thread synchronization technique

2.3 클라이언트 측 패킷 수신 구현부분

기존의 방식에서는 수신측에서 하나의 데이터가 여러 개의 패킷으로 분할하여 들어올 경우를 대비 하지 않았다. 본 연구에서는 2.1절과 같은 방법에 의해 분할된 패킷을 분석하여 원래의 데이터로 결합되도록 구현하였다.

한편 다양한 크기의 여러 가지 패킷들이 전송되는 경우 TCP의 Nagle알고리즘에 의해 여러 패킷들이 서로 결합되거나 분리되어 전송된다. 예를 들면, 10KB 크기의 패킷을 서버에서 전송했다고 할 때 클라이언트 측에서는 9KB가 먼저 들어오고 나머지 1KB는 다음에 전송될 패킷의 일부와 결합하여 들어오는 문제가 발생할 수 있다. 따라서 수신측에 적당한 크기의 버퍼를 준비해 패킷 헤더에 기록된 패킷사이즈 만큼의 데이터를 전송받은 후 하나의 패킷으로 처리해야 한다. 이를 위해 링버퍼(Ring buffer)방식을 도입하였다. 링버퍼의 크기는 오버플로우(Overflow)가 발생하지 않도록 최대 패킷사이즈의 두 배로 하였으며 구현된 링버퍼 방식은 다음과 같다. 데이터가 전송되면 일단 링버퍼에 저장 후 패킷 분석을 시도한다. 분석결과로 얻어진 패킷사이즈와 링버퍼에 저장된 데이터 사이즈를 비교 후, 해당 패킷이 완전히 전송되었다고 판단되면 바로 링버퍼에서 데이터를 읽어옴으로써 하나의 패킷이 완성된다. 이후 완성된 패킷의 End point를 다음 패킷의 Start point로 지정하여 연속적으로 데이터를 저장한다. 저장할 데이터(■)가 링버퍼의 끝단에 남은 공간보다 클 경우, 링버퍼의 끝부분을 넘어서는 데이터는 버퍼의 처음으로 돌아가 저장하는 링버퍼의 개념을 Fig. 3에 도시하였다.

before :

header		body
packet size	packet type	data

current :

header				body
file size	packet size	packet type	status flag	data

Fig. 1 Structural alterations to a packet

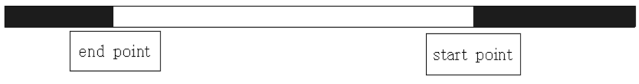


Fig. 3 The processing of the data when reached the buffer

2.4 해석코드 실행시간에 의한 대기자 처리

프로펠러 성능추정 수치 프로그램의 특성에 따라 계산시간이나 수치계산 결과가 방대한 경우가 있다. 기존의 PC에서 설계하는 방식은 최종사용자가 프로그램의 실행과정을 직접 확인하면서 계산을 수행하는 관계로 계산시간이나 개인 PC의 성능을 직접적으로 확인할 수 있다. 그러나 인터넷기반 설계 프로그램의 경우 프로그램의 실행은 서버에서 이루어지는 관계로 계산시간의 증가, 데이터의 전송 시간지연 등의 문제가 발생할 경우 최종사용자는 어느 부분에서 문제가 발생하였는지 알 수 없다. 특히 다수의 사용자가 서버에 접속할 경우 서버는 요청순서에 따라 Process스레드의 대기자 큐(Queue)에 넣어 1명씩 계산이 진행된다. 따라서 대기열이 긴 최종사용자는 자신의 계산순서를 알 수 없는 관계로 인터넷접속 및 서버에 대한 불신을 가지게 된다. 이러한 문제를 해결하기 위하여 최종사용자의 작업이 대기자 큐에 들어가는 순간과 큐에서 나오는 순간 큐에 대기 중인 모든 대기자에게 자신의 대기 순위를 알 수 있도록 하였다.

3. iProDAS를 이용한 프로펠러 설계(예) 및 고찰

서버는 실행초기에 클라이언트와의 접속, 데이터의 송수신 그리고 수치해석 모듈 등을 수행하는 소켓 및 스레드를 초기화하면서 실행된다. 초기화 과정을 마치고 클라이언트를 기다리고 있는 서버와 프로펠러 설계자가 직접 수행할 클라이언트인 iProDAS의 서버와 클라이언트 초기 실행화면을 Fig. 4에 도시하였다.

iProDAS의 실행 후 설계를 수행하기 위해서는 서버에 사용자로 등록되어 있어야 한다. 따라서 사용자는 Fig. 5와 같이 접속 창을 통해 서버에 접속하여 승인 과정을 거친 후 서버의 설계 모듈을 사용하게 된다.

설계과정은 유효반류추정→프로펠러 최적 직경 선정→설계프로펠러 정상 성능해석→설계프로펠러 캐비테이션 추정→선속추정의 흐름에 따른 수치계산을 수행하였으며. 설계결과는 주로 데이터 송수신 및 계산 흐름을 확인할 수 있도록 도시하였다. 본 논문은 프로펠러 설계를 위한 iProDAS의 사용 및 결과에 대한 것을 목표로 하였기 때문에 방대한 양의 수치적 모델링과

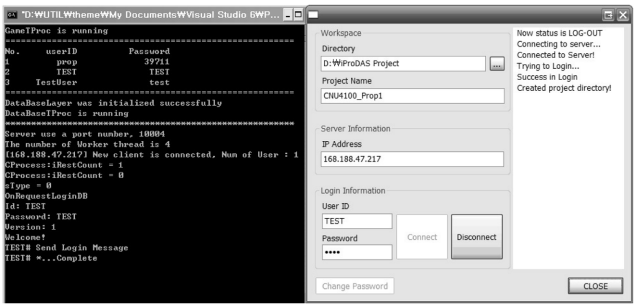


Fig. 5 Log-in process (L: server, R: client)

Table 4 Predicted self-propulsion coeff

MCR (BHP x rpm) : 54460 ps x 102.0 rpm NCR (BHP x rpm) : 49014 ps x 98.8 rpm					
Vs	22.00	23.00	24.00	24.50	25.00
EHP	21494	24571	28759	31433	34496
w	0.2757	0.2747	0.2737	0.2732	0.2727
t	0.2010	0.1999	0.1988	0.1983	0.1977

계산과정 및 데이터 처리절차는 생략하였다.

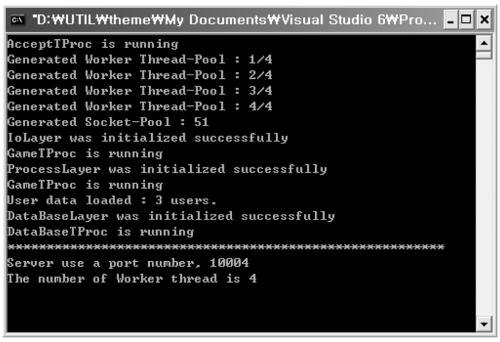
3.1 설계조건

iProDAS의 동작 안정성 및 계산데이터의 송수신 검증을 위해 4,100TEU급 컨테이너 선박용 프로펠러를 설계하였다. 사용된 프로펠러 요구조건 및 자항추정 값을 Table 1에 나타내었다.

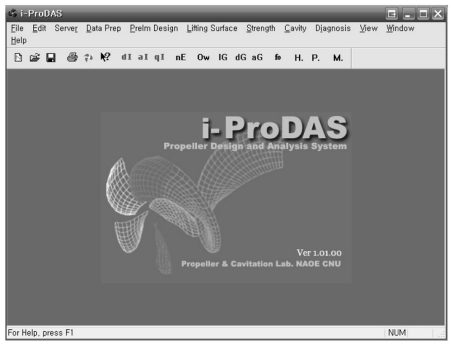
3.2 프로펠러 수치계산(예)

설계를 위한 입력파일 및 수치계산 결과는 설계 프로펠러별로 구성된 작업 디렉토리의 하위에 위치한 INPUT, OUTPUT 디렉토리에서 송수신 된다. 프로펠러 설계를 위하여 클라이언트 PC에 구성된 작업 디렉토리(예, CNU4100_Prop1)와 수치계산을 위한 입력파일이 생성된 형상을 Fig. 6에 도시하였다.

사용자는 포함된 여러가지 모듈을 사용하여 설계 · 해석을 수행한다. 각 모듈의 접근은 Fig. 7에서와 같이 일반적인 메뉴를 통해 수행된다.



(a) Initial message of the server



(b) Client main window

Fig. 4 Initial view of server consol and client window

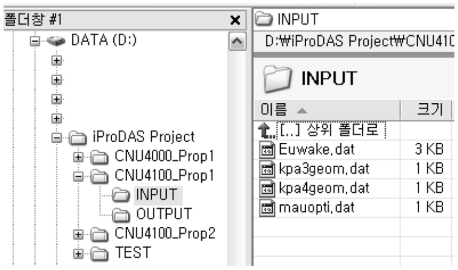


Fig. 6 Working directory structure

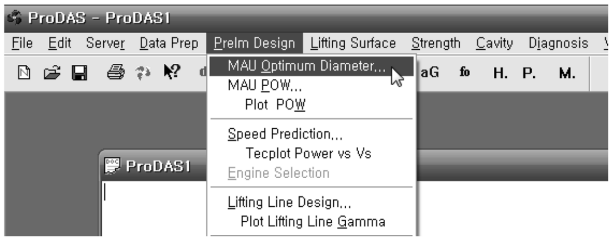


Fig. 7 Access to the menu for the execution

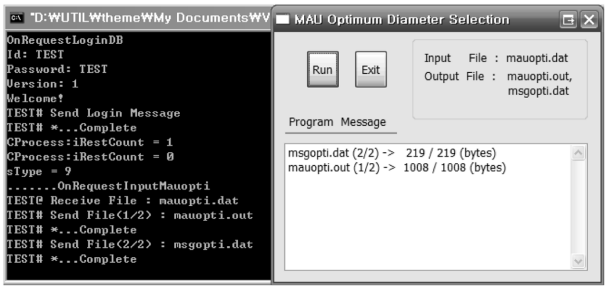


Fig. 8 Execution of MauOpti module

(1) 최적 직경 선정: 먼저 설계조건에서의 최적 직경을 구하기 위하여 MAU series를 이용한 MauOpti모듈을 수행하였다. Fig. 7은 MauOpti모듈의 실행을 위해 해당 메뉴에 접근하는 모습이고, Fig. 8에서 계산수행 및 결과가 전송되는 화면을 보여주고 있다. 사용자는 계산 결과로 생성되어 전송된 파일명, 용량, 파일개수를 화면을 통해 확인할 수 있다. 본 예에서는 2개의 파일(msgopti.dat(219byte), mauopti.out(1,008byte))이 전송되었다.

최적 직경 계산을 수행한 후 프로펠러 직경 및 확장 면적비, 평균 피치비등을 결정하여 상세 설계는 생략하고 설계된 프로

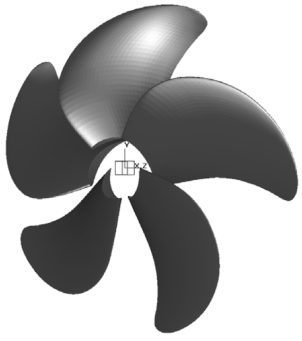


Fig. 9 Designed propeller

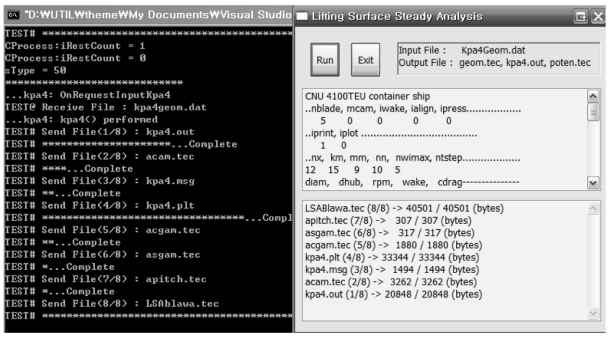


Fig. 10 Calculation of propeller open-water performance and loading distribution

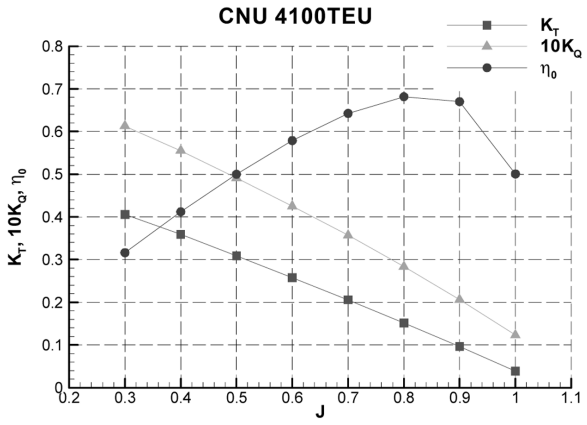


Fig. 11 Propeller performance prediction

펠러 형상을 별도의 후처리 프로그램을 사용하여 Fig. 11에 도시하였다.

(2) 단독 성능 계산: Fig. 9은 초기 설계된 프로펠러 형상에 대한 프로펠러 단독성능을 계산하는 과정으로, 계산 결과를 주고받는 서버와 클라이언트의 화면을 보여주고 있다. 계산 결과 생성된 8개의 파일이 모두 정상적으로 전송되었으며 최대 파일크기는 40,501byte였다. 계산 결과인 단독성능 곡선은 Fig. 10에 도시하였다. 이것을 기준으로 설계자는 요구조건을 충족시키는 효율 특성을 갖는 프로펠러 설계 유무를 판단하게 된다.

(3) 캐비테이션 발생 추정 : 설계자가 프로펠러의 캐비테이션 발생정도를 파악하는 것은 초기 설계에 있어 매우 중요한 요소라 할 수 있다. 본 연구 그룹에서는 수년간의 연구 과정을 거쳐 프로펠러에서 발생하는 캐비테이션을 예측할 수 있는 기술을 개발하였으며 현재 단일 패키지 형식으로 조선소 및 일반 선박용 프로펠러 설계자에게 보급되어 사용되고 있다. iProDAS는 사용자가 설계한 프로펠러의 성능해석뿐만 아니라 캐비테이션 발생정도를 확인할 수 있도록 연동되어있다. 본 논문의 마지막 확인단계로 프로펠러의 캐비테이션 추정 수치계산을 수행하였다. 계산 결과로 12개의 파일이 생성되었으며 최대 크기는 359KB, 총 크기는 887KB로 모두 정상적으로 전송되었다. Fig. 12에 클라이언트에 전송된 파일의 디렉토리를 나타내었으며, 계산 후 전송된 캐비테이션 추정결과를 후처리 프로그램을 사용해 Fig. 13에 도시하였다.

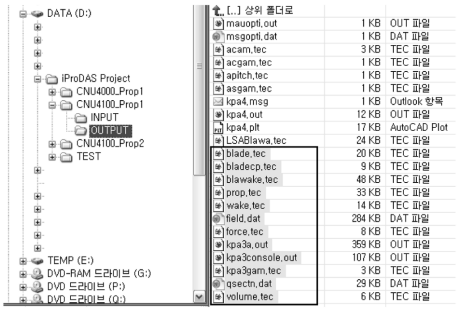


Fig. 12 Client's working directory

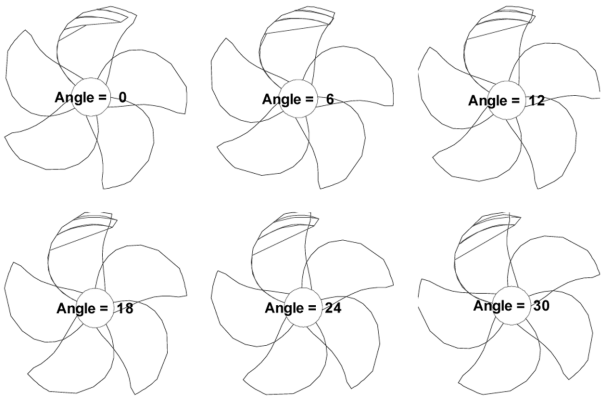


Fig. 13 Cavity prediction

이상의 수치계산(예)를 통하여 프로그램의 정상 동작을 확인하였다. iProDAS를 이용하는 경우 최종 사용자는 설계프로그램의 보관 및 유지에 신경을 쓰지 않아도 되는 장점이 있으므로 PC에 구애받지 않고 설계를 수행할 수 있다. 특히 프로펠러 설계를 위한 입력 자료의 준비 및 입력파일의 생성방법은 기존의 개별 모듈 방법과 동일하게 하여 사용자로 하여금 친밀감을 느끼도록 하였다.

프로펠러 설계를 위한 수치계산 수행에서 비교적 계산 시간이 짧고 계산결과가 적은 프로그램의 경우(MauOpti 등) 계산시간 및 데이터 전송 시간의 차이를 느끼지 못하였다. 반면에 설계자의 인터넷 전송속도 및 계산결과 데이터파일의 크기나 수가 많을 경우 다소 전송시간에서 지체되는 모습을 보였다. 따라서 iProDAS의 활용에 따른 설계자의 설계시간 증가는 서버에 존재하는 수치계산 프로그램의 계산시간에 따른 특성보다는 계산된 결과 파일의 용량 및 수에 의존한다는 것을 알 수 있다. 하지만 프로펠러 설계 전체과정으로 보면 미미한 수준이므로 수치계산을 위하여 각각의 PC에 설계 프로그램을 모두 설치하고 관리하는 노력과 경제적인 손실에 비하면 iProDAS의 효율성은 매우 높다고 판단된다.

4. 결 론

기존의 연구에서는 서버-클라이언트 환경을 구현하여 원격지 설계 작업이 가능하도록 하는 기반을 마련하였다. 본 연구에서는 기존 연구를 활용하여 각종 수치해석 프로그램 모듈을 통합하고 문제점의 해결 및 개선이 이루어 졌다. 클라이언트 프로그램은 통합프로그램으로써의 사용자 환경을 마련하였고, 파일의 크기에 따른 문제점을 패킷설계의 수정 및 스레드 동기화 기법을 통해 해결하여 계산 결과가 사용자측에 완전히 전송되도록 하였다. 개인용 PC에서 수행하던 여러 종류의 프로펠러 설계프로그램을 인터넷기반에서 작동하도록 함으로써 사용자의 입장에서 프로그램의 업그레이드 및 유지관리에 대한 장점을 제시하였다. 향후 프로펠러의 설계 및 해석 절차에 대한 분석을 통하여 사용자 환경을 개선한다면 중·소형 조선소와 같이 전산장비의 유지, 보수가 힘들고 설계인력이 부족한 경우 경제적, 기술적으로 매우 유용할 것으로 판단된다.

후 기

본 연구는 충남대학교 2007년 교원 연구력강화 과제의 일부로 수행되었습니다. 지원에 감사드립니다.

참 고 문 헌

윤성우 (2003). TCP/IP 소켓 프로그래밍, 프리렉.

이왕수, 박범진, 이창섭 (2003). "인터넷 기반 프로펠러 설계 시스템 개발", 대한조선학회 논문집, 제40권, 제6호, pp 69-79.

정인숙, 이창섭, 임효관 (1996). "GUI를 이용한 프로펠러 설계법", 대한조선학회 춘계학술대회 논문집, pp 220-223.

정인숙, 이창섭, 조충호 (1997). "데이터베이스와 응용프로그램 사이의 인터페이스 설계 및 구현", 대한조선학회 논문집, 제34권, 제4호, pp 139-147.

Jones, Anthony and Ohlund, Jim, 김남식(역) (2003). Network Programming for Microsoft Windows, Second Edition, 정보문화사.

Lee, C.-S. and Cho, C.-H. (2001). "Propeller Design and Analysis System using Object-Oriented Database in Windows Environment", Practical Design of Ships and Other Floating Structures, Vol 2, pp 685-691.

부 록

TCP 통신:

TCP란 인터넷기반의 데이터 전송방법을 정의하고 전송을 제어하기 위한 통신규약이다. TCP는 목적지를 찾아가기 위해 IP(Internet protocol)와 함께 사용되는데 IP가 데이터의 배달을 관장하는 동안, TCP는 데이터 패킷을 추적 관리한다. TCP계층은 데이터를 효율적으로 라우팅(Routing)하기 위해 여러 개의 작은 조각으로 나누거나 합치는데, 이것을 패킷(Packet)이라고 부른다.

연결지향 프로토콜인 TCP는 각 지점간의 가상연결이 설정되고 응용프로그램들이 이러한 연결라인을 사용하여 통신한다. 따라서 데이터의 경계가 없이 스트림 방식으로 전송이 이루어지며 데이터의 도착순서가 보장되고 연결 상의 문제가 없다면

데이터의 무결성을 보장 받을 수 있다(윤성우, 2003; Jones and Ohlund, 2003). 예를 들면, iProDAS-Server로부터 Client에게 보내질 때, 서버 내에 있는 TCP 계층은 Nagle알고리즘에 의해 전송 요청받은 데이터들을 네트워크 상황에 맞는 크기의 패킷들로 나누거나 결합하고, 패킷 번호를 붙인 다음, IP 계층으로 보낸다. IP는 데이터를 올바른 주소지에 전달하는 것에만 전념하는 계층으로 각 패킷이 동일한 수신지 주소(IP주소)를 가지고 있더라도, 네트워크의 서로 다른 경로를 통해 뒤바뀐 순서로 전송될 수 있다. 다른 편(클라이언트 프로그램)에 있는 TCP계층에서 각 패킷들을 순서대로 재조립하여 하나의 완전한 데이터로 도착하게 된다(윤성우, 2003).

2010년 7월 12일 원고 접수
2010년 8월 12일 심사 완료
2010년 8월 16일 게재 확정