한국전산구조공학회
1998년도 가을학술발표회 논문집

p-Version 유한요소 프로그램을 위한 자동절점생성 알고리즘 및 전처리 기법 개발

Development of Automatic Node Generation Algorithm and Preprocessing Technique for *p*-Version Finite Element Program

조 준 형* 홍 종 현** 우 광 성***
Jo, Jun-Hyung Hong, Jong-Hyun Woo, Kwang-Sung

ABSTRACT

Due to the drastic improvement of computer hardware and operating system, it is easy to break through the main defects of limited computer memory and processing time, *etc.* To keep up with this situation, this paper is focused on developing the preprocessor program with the input method based on vector graphic editor and the preprocessing technique including automatic node generation algorithm for the *p*-version finite element program. To develop this preprocessor program, the special data structure and the OOP(Object Oriented Programming) have been used by the Visual Basic 4.0. The Special data structure is proposed to describe the geometric data of node numberings and coordinates suitable for the p-version finite element program, which are quite different from the comvential h-version finite element program.

1. 서론

최근 PC(Personal Computer)의 성능향상 및 GUI 운영체제의 발전으로 인하여 그 응용범위가 크게 확대되었다. 이러한 시대적 상황에 맞추어 구조해석분야에서도 GUI 환경에서 신뢰성있고 원활하게 작동될 수 있는 프로그램의 필요성이 대두되었다. 이미 다수의 유한요소해석 프로그램들은 사용성을 높이기 위해 Preprocessor의 데이터 입력방식에 그래픽을 많이 가미하는 추세로 개발되고 앞으로도 이런 방식은 더욱더 가속화 될 거라고 예상된다.

따라서, 본 연구에서는 *p*-Version 유한요소법의 이점을 최대로 살릴 수 있는 Vector Graphic Editor의 데이터 입력 방식과 자동요소생성을위한 자동절점생성(Automatic Node Generation)알고리즘이 더해진 Preprocessing 기법에 의한 프로그램을 VISUAL BASIC을 이용해 개발하여 이를 통해 *p*-Version 유한요소 프로그램 실행을 위한 입력 데이터 작성시 요소생성부분의 정확도향상 및 소요시간단축에 그 목적이 있다.

앞에서 언급한바 있듯이 본 논문에서 개발될 프로그램의 큰 특징은 사용자가 마우스로 직접 그린 그래픽 객체로부터 유한요소해석 입력데이타를 추출하는데 있다. 본 방식은 사용자에게 친숙하게 느껴질 뿐만 아니라 해석데이타의 오류를 줄이는데에도 크게 기여할 것이다.

* 영남대학교 토목공학과 대학원 석사과정 Email : adbrain@hitel.net
** 탐라대학교 토목환경공학과 전임강사
*** 영남대학교 토목공학과 부교수

2. *p*-Version 유한요소법을 위한 객체 분류

2.1 Node Class

정규 영역에서 르장드르 형상함수의 적분일반형은 다음의 식(1)과 같이 정의 된다.

$$F_{j+1}(\xi)=\sqrt{\frac{2j-1}{2}}\int_{-1}^{\xi}P_j(t)\,dt \tag{1}$$

여기서 $P_j(t)$는 식(2)처럼 정의되며 $i=0,1,2,....$이다.

$$P_j(t)=\frac{1}{2^i\,i!}\frac{d^i}{dt^i}(t^2-1)^i \tag{2}$$

ξ,η가 각각 $-1\le\xi\le 1$ 그리고 $-1\le\eta\le1$인 정규사변영역에서 형상함수는 일차원형상함수 $F_p(\xi)$으로부터 다음과 같이 유도된다.

(1) 4개의 기본 모우드는 $F_i(\xi)\cdot F_j(\eta)$ 이며 이때 $i,j=1,2$ 이다.

(2) 주변 모우드는 $\eta=\pm1$인 변에 대해서 $F_p(\xi)$를 $(\eta+1)$과 $(\eta-1)$에 곱함으로 얻어지며 $\xi=\pm1$인 변에 대해서 $F_p(\eta)$를 $(\xi+1)$과 $(\xi-1)$에 곱함으로 얻어진다. 여기서 $p\ge2$는 형상함수의 차수를 나타낸다. 즉 $p=2$는 2차의 형상함수를 나타낸다.

(3) 내부모우드는 $p\ge4$인 경우 유효하다. 그리고 $F_i(\xi)\cdot F_j(\eta)$에 의해서 구할 수 있다. 따라서 $i+j=p$가 성립하고, 두 i 그리고 j는 2와 같거나 크다.

이처럼 Node를 기본모우드, 주변모우드, 내부모우드 3가지로 나눌수 있는데 이를 각각 Corner Node, Side Node, Bubble Node로 부른다. 그림1에서 형상함수가 8차일 경우 생성되는 Node의 종류를 보여주고 있다. 본 프로그램에서는 이들 Node를 모두 한가지의 Node Class를 이용해서 표현한다. 즉, Node는 외부적으로 독립된 데이터형으로 인식될 수 있고 스스로 동작을 해야할 필요도 있으므로 Class로 정의될 수 있는 조건에 잘 부합한다. 본 전처리 프로그램은 특성인 그래픽 처리 및 자동절점생성(Automatic Node Generation)에 주안점을 두어 Node Class를 실제로 설계해 보겠다.

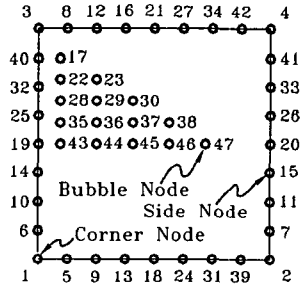


그림1 8차의 형상함수일 때 Node의 생성과 Node의 종류[1]

Node Class는 Preprocessor 프로그램의 가장 기본이 되는 단위로 이미 말한 바와 같이 자신의 성질을 나타내는 Member Variable과 그 성질을 이용해 어떤 표현이나 동작을 행하는 Method(Member Function)으로 구성되어있다. 표1과 표2는 각각 Node Class의 구성요소와 중요한 처리를 수행하는 Method를 설명하고 있다. 특히 각 Class의 Method는 Preprocessor의 특성을 강하게 반영하고 있다.

표 1 Node Class 구성도

	Member Variable(Property)	Member Function(Method)
clsNode	m_CenterX, m_CernterY, m_Boundary Conditon m_intChkXt, m_intChkYt m_index_of_Myself, m_CurrentColor	WriteNodeNumber EraseNodeNumber Draw

표 2 Node Class의 Method

clsNode	
Member Function	기능
WriteNodeNumber	Node가 생성되고 그려질 당시에 부여된 번호를 Node의 옆에 작게 씀.
EraseNodeNumber	수정시에 이미 있던 자신(Node)의 번호을 지움.
Draw	마우스 이벤트에 따라 Node를 그림.

2.2 Line Class

Line Class는 Node Class와 마찬가지로 유한요소모델링과 Element를 생성을 위한 근본적인 객체라고 할 수 있다. 다시 말해서 사변형 Element는 4개의 변으로 표현될 수 있고, 이 4개의 변은 각각 선객체로 표현될 수 있다. 특히, 이 객체의 경우는 유한요소 측면보다는 Preprocessor의 측면이 강하게 반영된 경우에 해당한다. 표3는 Line Class의 구성도를 나타내고 있다.

Line Class의 속성으로 특이할 만한 것은 Side Node부여를 자동으로 하기 위한 새로운 선정렬 값을 저장할 공간이 필요하다는 것이다. Side Node부여를 자동으로 하기 위해서 값은 자동화의 가장큰 부분으로 또 다른 객체에서 정렬루틴을 수행하여 도출된 값을 받아서 저장한다. Line Class의 Method중 표4에서 중요한 것을 설명하고 있다.

표 3 Line Class 구성도

	Member Variable(Property)	Member Function(Method)
clsLine	m_SrartX, m_StartY, m_EndX, m_EndY m_index_of_Myself, m_Node_on_Line m_EndNode(1 to 2) m_Region_X, m_Region_Y m_Region_Y1, m_Region_Y2 m_INNER_Region_X, m_INNER_Region_Y m_Newindex, m_NewindexBoolean	SmallRegion( ), GetINNER_Region_X_Count( ) GetDotX_in_INNER_LineArea( ) SmallRegionY( ), m_INNER_Region_Y_Count( ) GetDotY_in_INNER_LineArea( ) DrawBox_to_Express_LineArea( ) MinusIncremented( )

표 4 Line Class의 Method

clsLine	
Member Function	기능
SmallRegionX ( )	선의 전체 영역중 양끝에서 10pixel만큼 작아진 X좌표값의 영역을 계산 후 저장.
INNER_Region_X_Count ( )	SmallRegionX ( )에서 m_INNER_Region_X Collection에 저장되어진 좌표의 개수(Collectin.Count)를 반환.
GetDotX_in_INNER_LineArea ( )	m_INNER_Region_X Collection에 저장되어있는 좌표값을 반환.
SmallRegionY ( )	선의 전체 영역중 양끝에서 10pixel만큼 작아진 Y좌표값의 영역을 계산 후 저장.
m_INNER_Region_Y_Count ( )	SmallRegionY ( )에서 m_INNER_Region_Y Collection에 저장되어진 좌표의 개수(Collectin.Count)를 반환.
GetDotY_in_INNER_LineArea ( )	m_INNER_Region_Y Collection에 저장되어있는 좌표값을 반환.
Delete_Y_Dot_in_INNER_LineArea ( )	선의 수정 시에 이미 저장되어있는 m_INNER_Region_Y의 item을 지움.
MinusIncremented ( )	SideNode가 그려질 증분치를 구하고, 선이 그려진 방향을 판단해서 증분치의 값이 양 또는 음인지를 결정하는 기능.

2.3 Element Class

Element객체는 앞서 언급한 두가지 객체로 이루어지게 된다. 따라서 앞의 두가지 객체를 내부 데이터로 Element Class내에 선언된 Member Variable내에 저장을 한다.

Element Class는 앞의 두가지 Node객체와 Line객체를 포함하는 것 이외에 다른 여러 가지 속성을 나타내는 데이타를 포함한다. 주목할 사항은 Collection Class라는 또다른 형태를 찾을 수 있는데, 다른 객체를 관리하는 기능을 갖는 객체로 Linked List와 흡사한 역할을 한다.

배열이라는 정적(Static)데이터 형태를 쓰지 않고, Linked List의 동적(Dinamic)개념이 섞인 데이터형을 쓴이유는 *p*-Version 유한요소 프로그램의 특징과 현재 만들고 있는 Preprocessor의 기능을 조합해서 살펴보면 알 수 있다. 즉 Bubble Node의 개수를 정하는 Degree of Polynominal이 Preprocessor 프로그램 실행도중에 설정되며 이에따라 Node의 개수가 설정되므로 동적바인딩(Dinamic Binding)으로 처리를 해야하며 또한 Node 객체라는 일반적인 데이터 형태가 아닌 객체를 저장해야하므로 Collection Class사용이 적절하다.

표 5 Class 구성도

	Member Variable(Property)	Member Function(Method)
clsArea (Element)	m_index_of_Myself m_Node_on_Element as Collection m_Selected_Line_Array(1 to 4) m_CornerNode(1 to 4) as CornerNode m_CornerNode4123(1 to 4) as CornerNode m_CornerNodeNumber(1 to 4) m_CornerNodeNumber4123(1 to 4) m_Material_Number m_LIne_to_draw_Node as XYPoint	BresenHam_Line_to_Draw_Node( ) Contract_EightPoints_into_FourPoints( ) CornerNode_into_m_EndNode_in_clsLine( ) CornerNodeNumber_into_m_CornernodeNumber_in_clsArea( ) CornerNodeNumber4123_into_m_CornernodeNumber_in_clsArea( ) DeleteNode_On_Element( ), Node_on_Element_Auto( ) [Propert Let] Sort_CornerNode_With_CounterClockwise( ) Sort_CornerNode4123_With_CounterClockwise( )

표 6 Element Class의 Method

clsArea(clsElement)	
Member Function	기능
BresenHam_LIne_to_draw_Node ( )	Degree of Polynominal에 맞도록 노드를 잘 정렬해 그리기위해 m_Line_to_draw_Node()배열에 선의 좌표를 저장.
*Contract_EightPoints_into_* FourPoints ( )	*Element*를 형성하는 4개의 선의 양끝 8점을 조사하여 중복되는 점을 추려내고 4개의 점으로 만듬.
CornerNodeNumber4123_into_m_ CornernodeNumber_in_clsArea ( )	Element객체 안의 m_CornerNode4123변수에 저장되어있는 코너노드 좌표를 생성되어진 모든 노드의 좌표와 비교하여 같은 노드의 생성시에 부여된 번호를 m_CornerNodeNumber4123 배열에 저장.
DeleteNode_on_Element ( )	이미 생성된 노드를 다른 Line객체 또는 Element객체에 속하도록 수정할 때 먼저 속해있던 객체에서 노드 객체의 번호를 지우는 역할을 수행
Node_on_Element_Auto ( ) [Property Let]	Degree of Polynominal에 맞도록 자동으로 생성된 노드에 좌표값을 부여하고, 그 좌표값에따라 노드를 그리고 노드의 번호를 씀.
Sort_CornerNode4123_With_Counter ClockWise ( )	우측하단점을 시작점으로 하여 코너노드를 시계반대방향으로 정렬.
WriteAreaNumber ( )	Element의 생성시에 부여된 번호를 씀.

Element Class 구성도는 표5에 나타나 있고 Method중 독특한 연산을 수행하는 함수에 대해 정리해보면 표6과 같다.

3. 자료구조의 형성

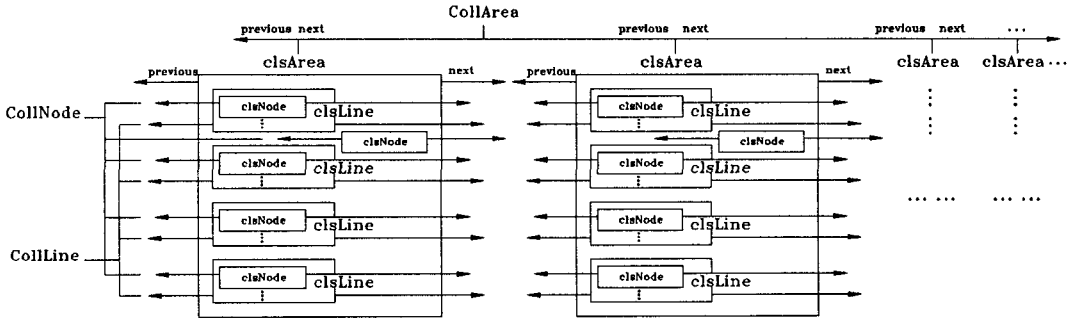


그림 2 객체의 관계

앞절에서 언급한 자료구조는 각각의 분리된 객체를 clsArea(Element)에 넣은 후 그 clsArea를 Linked List로 묶어서 관리하는 것이 아니라, 이미 각각의 생성되어진 객체는 모두 독립된 Linked List(Collection Class)를 이용해서 관리되고 있다. (그림 2의 각 객체의 양쪽으로 나온 화살표가 바로 옆의 clsNode와 clsLine으로 이어지지 않고 있다.) 다시 말하면 clsNode 객체는 CollNode, clsLine 객체는 CollLine Class, clsArea 객체는 CollArea Class를 이용해 관리된다. 그리고 그 각각의 clsNode, clsLine 객체는 필요한 부분으로 정리되어 다시 clsArea에 저장된다. 따라서 하나의 객체는 clsArea를 거치지 않고 바로 참조가 이루어 질 수 있고 clsArea객체의 내부 데이터를 통해서 참조가 이루어 질 수도 있는데, 이러한 참조방식은 후에, 이렇게 연결되어진 자료구조의 기하학적 정보로부터 *p*-Version 유한요소 프로그램의 데이터를 추출해는데 모두 사용된다. 자료객체(Data Object)들의 상호 관련을 상세히 나타내보면 그림2와 같다.

4 PREPROCESSOR에 의한 요소 생성

4.1 일반

요소의 생성을 위해서는 우선 Line 객체를 새롭게 정렬하는데 이는 다시 말하면 Line 객체에 새로운 Member Variable을 선언하고 그 변수에 새로운 번호를 저장하는 것을 의미한다. Line 객체가 형성되어있는 형태에 따라 정렬을 해야하므로 이 정렬을 보통의 숫자로 정렬하는 것과는 좀 다르다. 해석대상의 정보를 그림으로 입력받기 때문에 생길 수 있는 문제 즉, 각각의 객체가 사용자에 따라 항상 같은 순서로 그려지지 않을 수 있기 때문에 발생하는 문제를 방지하고 *p*-Version에서 Side Node와 Bubble Node를 생성시킬 때 따르는 방식을 최대한 수용하고 또 가장 익숙한 방식으로 Node객체를 생성하기 위해서는 자료객체에대한 인식과 정렬을 행해야한다. 즉 그려진 도형의 현재 상태를 인식하는 방법이 필요하다.

4.2 데이터 인식 및 정렬과 입력데이타 생성

AutoNumbering() 프로시저 안에는 표7에 나타나 있는 알고리즘이 포함되어있고, Find_FirstLine_on_Area() 함수 안에는 표7, 표8, 표9의 형태가 모두 포함되어 있다. AutoNumbering() sub procedures는 내부에서 Find_FirstLine_on_Area()를 호출하는 형태를 취하고 있고, 호출된 Find_FirstLine_on_Area()는 수행도중 다시 함수자신을 호출하는 형태를 가진다.

시작과 끝을 정할 수 없는 그림객체를 컴퓨터가 인식하도록하기위해서 재귀호출의 형태를 이용하였다. 처리가 종료되면 각 선객체는 위치에 맞는 번호를 부여받게 된다.

표 7

```
If UBound(OppositePoints) > 3 Then
   .........
ElseIf UBound(OppositePoints) = 3 Then
   .........
ElseIf UBound(OppositePoints) < 3  Then
   .........
End If
```

표 9

```
Select Case UnNumberLineNumber

Case Is >=1
   .........
Case -1
   .........
End Select
```

표 8

```
If There_is_SameCood_ExceptOneCood() Then
 If UBound() = 2 Then
  If Not Numbered_exceptMe() Then
    If Check_1YLargerThan2y_and_1ySmallerThan3Y() Then
      ...
    End If
    If 0 = Check_Wheather_Twovalue_are_Same() Then
      ...
    End If
  End If
  End If
End If
 If -1 =UnNumber_LineNumber_exceptOneLine
       _(TempMatchedLine(),UnNumberLineNumber) Then
 End If
```

다음은 Polynomial Degree에 맞추어서 Node를 자동으로 생성해야하는데, 이런 생성은 같은 종류의 객체를 만들어내는 것을 의미한다. 이렇게 Node를 자동으로 생성 시키는 과정에 그래픽과정도 모두 포함되어있다. 즉, 각 Line객체 안에 정의되어있는 Method를 통해서 Graphic처리를 수행하도록 설계되었다. 위의 알고리즘 및 루틴이 형상을 판단하여 처리하는 가장 일반적이고 간단한 형태의 선객체를 그림3에 나타내었다.

3차의 형상함수까지는 Element의 변을 따라 Side Node만 생성이 되고 4차의 형상함수를 나타내는 Node는 Bubble Node가 Element의 면내부에 생성이 되어야한다. Alpha, Beta, DeltaAlpta, DeltaBeta는 각각 Polynimial Degree에 맞추어 루프문의 상한값과 하한값을 지정하는 역할을 하고 있다. 4차의 Bubble Node가 1개 , 5차의 Bubble Node가 2개, 6차의 Bubble Node가 3개, 7차의 Bubble Node가 4개, 8차의 Bubble Node가 5개로 증가하는 경향을 그대로 반복수행문의 상한값과 하한값에 반영하기위해 표10에 있는 알고리즘을 구성하였다.

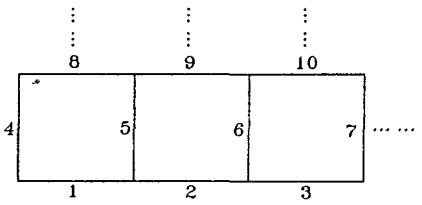


그림 3 일반적인 선의 형상에 따른 번호의 부여

```
Alpha = Alpha + DeltaAlpha
DeltaAlpha = DeltaAlpha + 1
Beta = Beta + DeltaBeta
DeltaBeta = DeltaBeta + 1
```

표 10 형상함수의 차수 증가 알고리즘

절점이 자동으로 생성되고, 하중 및 경계 조건이 입력되면, p-Version 유한요소 계산 프로그램과 자료전달의 수단인 Text File생성 과정을 수행한다.

5. 필렛용접에 의해 접합된 T-joint부

필렛용접에 의해 접합된 T-joint부를 대칭성에 의해 우측 상단 $\frac{1}{4}$모델링을 통하여 해석을 수행하였다. 그림 7는 Polynomial Degree에 따라 본 프로그램으로 모델링하여 계산된 8번 절점의 σ_{yy}를 그래프로 나타내었다.

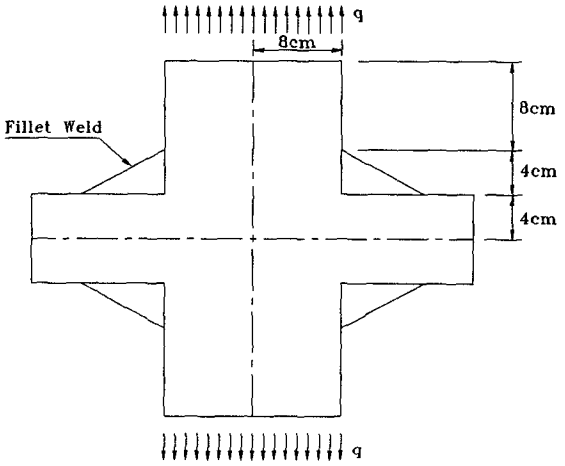
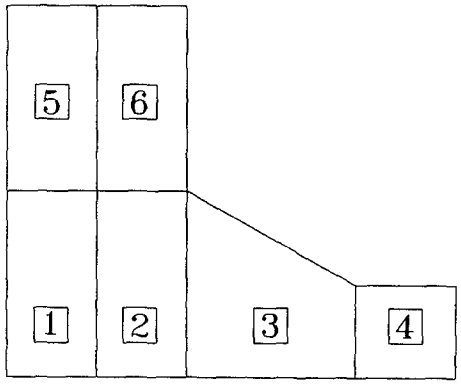


그림 4 필렛용접에의해 접합된 T-joint



$E=2.1\times10^{6}kg/cm^{2}$, $\nu=0.3$

$q=200kg/cm$, $Thickness=1cm$

그림 5 6개의 Element를 사용하여 모델링

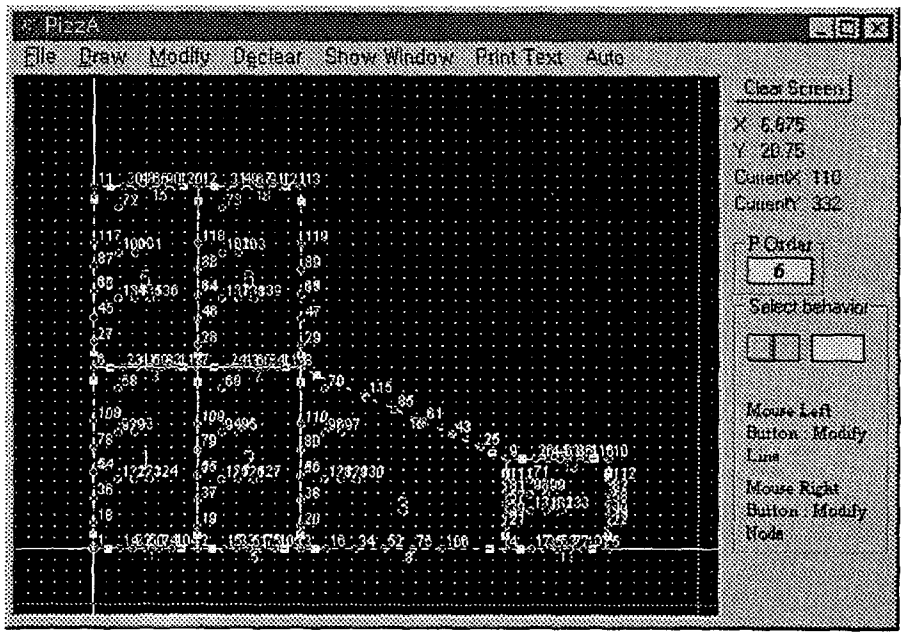


그림 6 6 Element, P-Degree 6

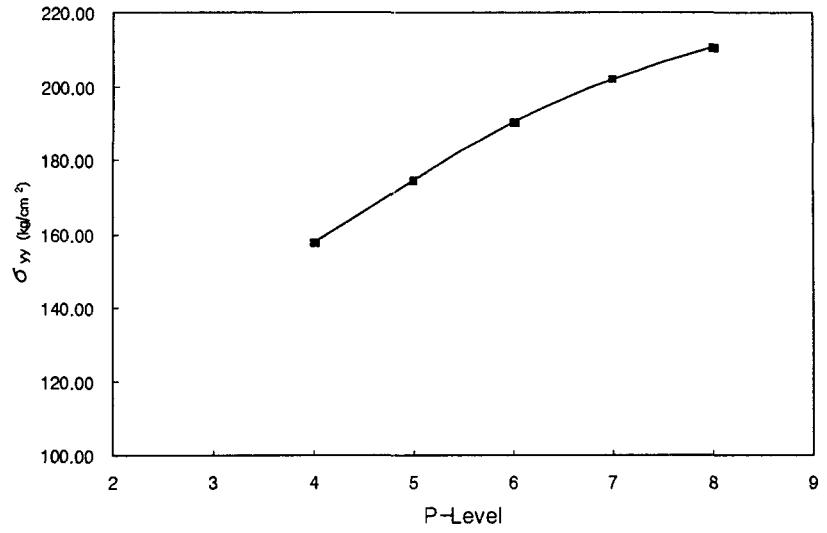


그림 7 8번 절점의 σ_w

6. 결론 및 고찰

본 논문을 통해서 TUI(Test User Interface)방식을 최대한 배제하고 기하학적 정보를 사용자가 직접 그린 도형을 통해 추출하는 방법 및 자체적으로 개발한 절점자동생성(Automatic Node Generation) 알고리즘을 이용해 프로그램을 만들어 보았다. 물론 짧은 개발 시간과 제한된 정보, 정성적데이타를 통해 인간이 판단해 내는 값과 같은 결과를 정량적 데이터를 기반으로 컴퓨터를 이용해서 계산할 때의 해결해야할 문제점등으로인해 도형의 인식 및 알고리즘, 사용자편의성고려 그리고 인터페이스의 효율성 측면에서 완전하지 않을 수 있다. 그러나, 이런 대부분의 것들은 시간적 투자와 좀더 깊은 연구를 통해 해결될 수 있다고 믿는다. 프로그램의 또 다른 기능의 향상을 논하자면, 아마도 3차원이 가장 중요한 문제가 될 것이다. 역시 3차원에서도 가장 중요한 것은 그래픽을 사용해 모델링된 대상체의 자료구조이므로 3D Solid모델을 표현하기위해 사용되고 있는 자료구조를 이용한다면 자동화 및 전처리기 구현이 가능할 것으로 생각된다.

7. 참고문헌

1. K. S. Woo, 『High Precision Analysis of Plates and Cylindrical Shells in the Presence of Singularities by the p-Version of F.E.M.』, *Ph.D. Dissertation*, Vandervilt University, (1988)
2. 장승조, 이상갑 『Visual Basic을 이용한 구조해석 프로그램에 관한 연구』 *한국전산구조공학회* 학술발표회 논문집, 제8권, 제2집, 통권 제15호, p215-p222, 1995
3. 조명철 『A Study on the two-dimensional Automatic Mash Generation Programming』 *인하대학교 대학원 기계공학과(자동화전공) 석사학위논문*
4. Charles Petzold, *Programming Windows95* Microsoft Press
5. Stephen Prata, *비주얼 베이직 4 자율학습* WAITE GROUP PRESS, 도서출판 대림
6. Zane Thomas, Mitchell Waite, *VISUAL BASIC 4 HOW-TO* WAITE GROUP PRESS ™ 200 Tamal Plaza Corte Madera, CA 94925
7. Herbert Schildt, 류성렬, *C : Power, C 프로그램예제* 도서출판세웅