Gilberto Luis Valente Costa

hp^2FEM - Uma Arquitetura de *Software* p Não-Uniforme para o Método de Elementos Finitos de Alta Ordem

119/2012

CAMPINAS
2012

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA

Gilberto Luis Valente da Costa

hp^2FEM - Uma Arquitetura de *Software* p Não-Uniforme para o Método de Elementos Finitos de Alta Ordem

Dissertação de Mestrado (ou Tese de Doutorado) apresentada à Faculdade de Engenharia Mecânica da Universidade Estadual de Campinas como parte dos requisitos exigidos para obtenção do título de Mestre(a) (ou Doutor(a)) em Engenharia Mecânica, na Área de

Orientador: Prof. Dr. Marco Lúcio Bittencourt
Coorientador:

ESTE EXEMPLAR CORRESPONDE À VERSÃO FINAL DA DISSERTAÇÃO DEFENDIDA PELO ALUNO Gilberto Luis Valente da Costa, E ORIENTADO PELO PROF. DR. Marco Lúcio Bittencourt.

.....
ASSINATURA DO ORIENTADOR

CAMPINAS
2012

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA E ARQUITETURA - BAE - UNICAMP

V234a

Valente da Costa, Gilberto Luis, 1983-
hp2FEM - uma arquitetura de software
p não-uniforme para o método de elementos finitos de alta
ordem \ Gilberto Luis Valente
da Costa. –Campinas, SP: [s.n.], 2012.

Orientador: Marco Lúcio Bittencourt.
Dissertação de Mestrado - Universidade Estadual de
Campinas, Faculdade de Engenharia Mecânica.

1. Arquitetura de software. 2. Método de elementos
finitos. 3. Framework (Programa de computador). I.
Bittencourt, Marco Lúcio, 1972-. II. Universidade
Estadual de Campinas. Faculdade de Engenharia
Mecânica. III. hp2FEM - uma arquitetura de software
p não-uniforme para o método de elementos finitos de alta.

Título em Inglês:	hp2FEM - a p non-uniform software architecture to the high order finite element method.
Palavras-chave em Inglês:	Software Architecture, Finite Element Method, Framework.
Área de concentração:	Mecânica dos Sólidos e Projeto Mecânico
Titulação:	Mestre em Engenharia Mecânica
Banca Examinadora:	Carlos Alberto Cimini Júnior, Edson Borin
Data da defesa:	27-08-2012
Programa de Pós Graduação:	Engenharia Mecânica

UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA MECÂNICA

COMISSÃO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

DEPARTAMENTO DE PROJETO MECÂNICO

DISSERTAÇÃO DE MESTRADO ACADÊMICO

hp^2FEM - Uma Arquitetura de *Software* p Não-Uniforme para o Método de Elementos Finitos de Alta Ordem

Autor: Gilberto Luis Valente da Costa
Orientador: Marco Lúcio Bittencourt

A Banca Examinadora composta pelos membros abaixo aprovou esta Dissertação:

Prof. Dr. Marco Lúcio Bittencourt, Presidente
DPM/FEM/UNICAMP

Prof. Dr. Carlos Alberto Cimini Júnior
DPM/FEM/UNICAMP

Prof. Dr. Edson Borin
IC/UNICAMP

Campinas, 27 de Agosto de 2012.

Dedicatória

Aos meus queridos avós, Pedro Catarino e Luiza Marçal, os quais sinto muito orgulho e agradeço por todo carinho e o apoio de sempre.

Agradecimentos

À Deus, minha fonte de força, fé e determinação. E como em todos os dias, agradeço pelo dom da vida.

Ao meu orientador, Prof. Dr. ...

Aos membros das bancas de qualificação e defesa ...

A toda minha família ...

Aos amigos e companheiros de laboratório ...

A CAPES - Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, pelo indispensável apoio financeiro.

O gênio consiste em um por cento de inspiração e noventa e nove por cento de transpiração.

Thomas Alva Edison

Resumo

VALENTE DA COSTA, Gilberto Luis. hp2FEM - uma arquitetura de software p não-uniforme para o método de elementos finitos de alta ordem. 2012. 122p. Dissertação (Mestrado). Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, Campinas.

Este trabalho tem como objetivo principal a implementação de uma arquitetura de *software* para o Método de Elementos Finitos de Alta Ordem (MEF-AO), baseando-se no paradigma de programação orientada a objeto (POO) e no uso de técnicas de otimização de código fonte. O *software* foi escrito em linguagem C++ e desenvolvido sobre um *framework* com ferramentas que auxiliaram no desenvolvimento. A modelagem do sistema foi realizada de forma a facilitar e promover o reuso e manutenção do código. Buscou-se, também, a flexibilidade e generalização do MEF-AO ao permitir a variação nos procedimentos da construção das equações e o uso de malhas p não-uniforme. Neste caso, cada elemento pode ser interpolado com uma ordem polinomial diferente, além de permitir o uso de um algoritmo local de solução. Tal característica pode diminuir o número de operações e de armazenamento, pois o número de funções de forma é aumentado apenas onde é necessário o uso de mais pontos para interpolação da malha de solução. No final, o *software* é avaliado aplicando o problema de projeção para malha de quadrados e hexaedros.

Palavras-chave: Arquitetura de *Software*, Método de Elementos Finitos, *Framework* (Programa de computador).

Abstract

VALENTE DA COSTA, Gilberto Luis. hp2FEM - a p non-uniform software architecture to the high order finite element method. 2012. 122p. Dissertação (Mestrado). Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, Campinas.

The main objective of this work is the implementation of a software architecture for the High-Order Finite Element Method (HO-FEM), based on the Object Oriented Paradigm (OOP) and on source-code optimization techniques. The software was written in C++ programming language and developed over a framework which provided tools that assisted the implementation. The system was modeled so to promote code reuse and maintainability. Furthermore, the system modeling also provided flexibility and generalization for the HO-FEM by allowing modifications on the procedures for equation assembling and the use of p-non-uniform meshes. In this case, each element can be interpolated with different polynomial order, and allows the application of an algorithm for local solution. Such features can reduce the number of operations for memory allocation, since the number of shape functions is increased only where a higher density of points is needed by the solution mesh. Finally, the software is assessed by applying the projection problem for meshes of squares and hexahedros.

Keywords: Software Architecture, Finite Element Method, Framework.

Lista de Ilustrações

Lista de Tabelas

Lista de Abreviaturas e Siglas

Matrizes e Vetores

$\{F\}$,$\{f\}$	-	Vetor de forças aplicadas
	-	Termos independentes
$\{F_e\}$	-	Vetor local de forças aplicadas
$F_{e,i}$	-	Coeficientes do vetor local de forças aplicadas
f_e^{1D}	-	Vetor de carga unidimensional associado à matriz de massa
f_i^{1D}	-	i-ésimo elemento do vetor de carga unidimensional associado à matriz de massa
$\{f_{1D}^m\}$	-	Vetor de carga associado à matriz de massa unidimensional
$[K]$	-	Matriz de rigidez global
$[K_e]$	-	Matriz de rigidez local
$[M]$	-	Matriz de massa global
$[M_{i,j}]$	-	Coeficientes da matriz de massa
$[M_{1D}]$, $\left[M_e^{1D}\right]$	-	Matriz de massa unidimensional
$[M_e]$	-	Matriz de massa local
$[M_e]^{-1}$	-	Inversa da matriz de massa local
$M_{e,ij}$	-	Coeficientes da matriz de massa local
$M_{e,ij}^{1D}$	-	Coeficientes da matriz de massa unidimensional
M_{ij}^{2D}	-	Coeficientes da matriz de massa bidimensional
$[N_{Sol,Pos}]$	-	Matriz de funções de interpolação da malha de solução nos pontos de colocação da malha de pós-processamento
$\{q\}$, $\{u_e\}$, $\{a_e\}$	-	Vetor de deslocamentos locais
$\{R\}$	-	Resíduo
$\{w\}$	-	Função ponderadora
$\{u\}$,$\{a\}$	-	Vetor de deslocamentos

$\left\{u_e^{Pos}\right\}$	-	Vetor de solução local da malha de pós-processamento
$\left\{u_e^{Sol}\right\}$	-	Vetor de solução local da malha de solução
$\left\{u_e^1\right\}, \left\{u_e^2\right\}, \left\{u_e^3\right\}$	-	Vetores de deslocamentos para os elementos 1, 2 e 3

Letras Latinas

b	-	Intensidade do termo independente
b_j	-	Coeficientes de aproximação
$d, '$	-	Derivada de primeira ordem
$''$	-	Derivada de segunda ordem
e	-	Erro na aproximação polinomial
$\|J\|$	-	Determinante do Jacobiano
L^p	-	Norma p
L_j	-	Comprimento de um elemento j
L_1, L_2	-	Comprimento dos elementos 1 e 2
R	-	Resíduo
w	-	Função ponderadora
$u(.)$	-	Função polinomial qualquer
$u_{ap}(.)$	-	Solução aproximada
u_i	-	Deslocamentos nodais para o nó i
u_{ij}	-	Deslocamentos nodais para o nó i no elemento j
u_{12}, u_{22}	-	Deslocamentos nodais do nó 2 nos elementos 1 e 2
$u_{11}, u_{12}, u_{22}, u_{23}, u_{33}, u_{34}$	-	Deslocamentos nodais calculados nos elementos 1, 2 e 3

Letras Gregas

Ω	-	Domínio
ξ_i	-	Coordenada local em [-1,1]
ξ, ξ_1, ξ_2	-	Coordenadas locais
ϕ_i,ϕ_j	-	Funções de interpolação
$\{\phi_i\}$	-	Base formada por um conjunto de funções ϕ_i
$\phi_a, \phi_b, \phi_p, \phi_q$	-	Funções de interpolação unidimensionais
ϕ_1, ϕ_2, ϕ_3	-	Funções de interpolação do polinômio de Lagrange

Siglas

ACDP	-	Ambiente Computacional para Desenvolvimento de Programas
DOFs	-	*Degrees of Freedom* (Graus de Liberdade)
EDP	-	Equações Diferenciais Parciais
GLC	-	Gramática Livre de Contexto
hp^2**FEM**	-	Programa do Método de Elementos Finitos de Alta performance - 2^a versão (*High Performace Finite Element Method Software*)
MEF	-	Método de Elementos Finitos
MEF-AO	-	Método de Elementos Finitos de Alta Ordem
POO	-	Programação Orientada a Objeto
	-	Paradigma de Orientação a Objetos
UML	-	*Unified Modeling Language*
XMI	-	*XML Metadata Interchange*
XML	-	*eXtended Markup Language*

Outras Notações

p-uniforme	-	Distribuição polinomial uniforme na malha de elementos finitos.
p-não-uniforme	-	Distribuição polinomial não uniforme na malha de elementos finitos.
$< , >$	-	Produto interno entre dois vetores
$\{\}^T$	-	Transposto de um vetor
1D	-	Elemento unidimensional
2D	-	Elemento bidimensional
3D	-	Elemento tridimensional

SUMÁRIO

1 Introdução

A simulação computacional é uma poderosa ferramenta utilizada em vários setores da indústria e em importantes centros de pesquisas relacionados às diversas áreas do conhecimento, como química, física, biologia molecular, meteorologia, e principalmente pelas engenharias. Uma simulação faz uso de um modelo numérico do problema em questão, cujos parâmetros podem ser modificados para fornecer informações relevantes a respeito da dinâmica do problema. Desta forma, a simulação permite reduzir amplamente os custos, tempo e recursos através da economia de equipamentos e a eliminação de longas e onerosas etapas de testes no processo de desenvolvimento e pesquisa.

2 Título do Capítulo 2

Nesse capítulo, apresentam-se conceitos sobre o método de elementos finitos ...

2.1 Título da Seção 1

Descrição . . .

Para utilizar as referências em latex acrescente o comando

```
~\cite{referencia}
```

Esses são 2 exemplos de citações: (RYLO, 2007) e (LIMA *e outros*, 2004).

O comando

```
~\citet{referencia}
```

é utilizado quando se coloca o autor no começo da oração. Exemplo:

BITTENCOURT (2010) propôs o método X para o desenvolvimento de . . .

2.1.1 Título da Subseção 1

Descrição . . .

Exemplo para definir imagens. Veja o arquivo capitulo2.tex e verifique como a Figura 2.1 é definida.

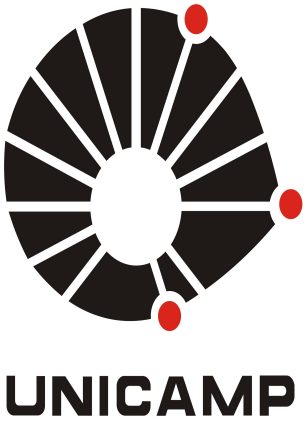

Figura 2.1: Logo da Unicamp.

2.2 Título da Seção 2

Descrição . . .

Agora, um exemplo da para descrever as equações em latex. Veja o arquivo capitulo2.tex verifique como as equações 2.1 e 2.2 estão definidas.

$$M_{e,ij}^{1D} = \int_{-1}^{1} \phi_i \phi_j(\xi_1) d\xi_1, \tag{2.1}$$

$$f_i^{1D} = \int_{-1}^{1} b(\xi_1)\phi_i(\xi_1) d\xi_1, \tag{2.2}$$

3 Título do Capítulo 3

Nesse capítulo, abordam-se conceitos relacionados as técnicas de estruturação . . .

Em seguida, apresenta-se a arquitetura proposta demonstrando com as visões arquiteturais . . .

3.1 Título da Seção 1

A seguir, demonstramos um tipo de enumeração utilizado em latex.

Enumeração:

- Visão de dados . . .
- Visão de módulo . . .
- Visão de implantação . . .
- Visão de execução . . .
- Visão de implementação . . .
- Visão de caso de uso . . .

3.1.1 Título da Subseção 1

Nesta subseção, demonstramos o uso de tabelas em latex como a Tabela 3.1 a seguir:

Verifique a definição da Tabela 3.1 no arquivo capitulo3.tex

Tabela 3.1: Avaliação de Desempenho da Simulação

Grau Polinomial	Tempo de execução (segs)		Consumo de memória (kilobytes)	
	Caso 1	Caso 2	Caso 1	Caso 2
1	0.005406	0.030677	2468	2468
2	0.015912	0.018219	2584	2508
3	0.049234	0.044774	2996	2632
4	0.164143	0.122129	4220	2852
5	0.549411	0.326865	7540	4088
6	1.681481	0.767674	14552	5952
7	5.336972	1.742116	28904	10656
8	14.411422	3.473562	56480	19172
9	35.352515	6.938881	101868	33736
10	79.008939	12.892254	180152	58080
11	166.510947	22.722333	297308	95972
12	336.605528	39.144736	480016	153708

Referências

BITTENCOURT, M.L. **Análise computacional de estruturas com aplicação do Método de Elementos Finitos**. Editora da Unicamp, 2010.

LIMA, L.M.; MELOTTI, B.Z.; CATABRIGA, L. e VALLI, A.M.P. Uma implementação paralela eficiente do método dos elementos finitos para a equação de advecção e difusão. **XXV Iberian Latin American Congress on Computational Methods in Engineering.**, v. 25, 01–09, 2004.

RYLO, E. C. **Adaptatividade hp em paralelo**. 2007. Tese (Doutorado). Universidade Estadual de Campinas.

ANEXO A – Arquivos de Entrada do hp^2FEM

Os dados apresentados a seguir correspondem os arquivos de entrada utilizados para testar o problema de projeção para quatro elementos quadrados.

A.1 Arquivo .fem

O arquivo a seguir apresenta as configurações para uma malha estruturada.

```
*TITLE
Title
*

*ELEMENT_GROUPS
1
1   4   SQUARE  1   4

*INCIDENCES
1 1 2 5 4
2 2 3 6 5
3 4 5 8 7
4 5 6 9 8

*DIMENSION
2

*COORDINATES
9
1   0   0
2   1   0
3   2   0
4   0   1
5   1   1
6   2   1
7   0   2
8   1   2
9   2   2
```

```
*GM_KEYPOINT
0

*GM_LINE
0 0

*GM_SURFACE
0 0

*GM_VOLUME
0 0

*END
#
# Developed by Gilberto Luis Valente da Costa
# e-mail: betogil@gmail.com
```

A.2 Arquivo .def

O arquivo a seguir apresenta as configurações para uma malha uniforme e p-uniforme.

```
*POST_PROCESSOR
4 GID

*SOLUTION_DIRECTIVES
*SOLVER_TYPE
ELEMENT_ELEMENT

*ANALYSIS_TYPE
PROJECTION  CONSIST

*SOLUTION_ALGORITHM
LINEAR_SOLUTION

*LINEAR_SOLUTION_METHOD
GAUSS

*OPERATOR_TYPE
SYMMETRIC

*THEORETICAL_SOLUTION
L2 1 1 P s exp(3.1416*X)*sin((3.1416/4)*Y)*((X-1)^2)*(Y^2)


*GROUP_PARAMETERS_1

*MESH_TYPE_1
UNIFORM pUNIFORM

*SOLUTION_ORDER_1
4

*DOFS_GROUP_1
1 P 0 1 P 2 QX QY 0 0 0

*MATERIAL_1
HOOKE 2100 0.3 1.0E-06 1
```

```
*GEOMETRIC_PROPERTIES_1
1 1 1

*MAPPING_1
ISOPARAMETRIC

*KINEMATICS_1
INFINITE MATERIAL TOTAL_LAGRANGIAN

*PROBLEM_PARAMETERS_1
POISSON D1_MATRICES CONSIST CONSISTENT

*INTERPOLATION_1
STAND NODAL NON_HIERARCHICAL STANDARDLAGRANGE

*COLLOCATION_POINTS_1
EQUALLY_SPACED

*INTEGRATION_POINTS_1
GAUSS_LEGENDRE 2

*DOF
1
P
0
1
P
2
QX QY
0
0
0

*HOMOGENEOUS_DIRICHLET_BC
*HDBC_NODES
0
*HDBC_EDGES
0
*HDBC_FACES
0
*HDBC_KEYPOINTS
```

```
0
*HDBC_LINES
0
*HDBC_SURFACES
0

*NON_HOMOGENEOUS_DIRICHLET_BC
*NHDBC_NODES
0
*NHDBC_EDGES
0
*NHDBC_FACES
0
*NHDBC_KEYPOINTS
0
*NHDBC_LINES
0
*NHDBC_SURFACES
0


*LOAD_SETS
1

*LOAD_SET_1

*NODAL_LOADS_1
0
*KEYPOINT_LOADS_1
0
*EDGE_LOADS_1
0
*LINE_LOADS_1
0
*FACE_LOADS_1
0
*SURFACE_LOADS_1
0
*BODY_LOADS_1
4
1 1 P s exp(3.1416*X)*sin((3.1416/4)*Y)*((X-1)^2)*(Y^2)
2 1 P s exp(3.1416*X)*sin((3.1416/4)*Y)*((X-1)^2)*(Y^2)
```

```
3 1 P s exp(3.1416*X)*sin((3.1416/4)*Y)*((X-1)^2)*(Y^2)
4 1 P s exp(3.1416*X)*sin((3.1416/4)*Y)*((X-1)^2)*(Y^2)
*VOLUME_LOADS_1
0
*EDGE_PRESSURE_1
0
*FACE_PRESSURE_1
0
*LINE_PRESSURE_1
0
*SURFACE_PRESSURE_1
0

*END
#
# Developed by Gilberto Luis Valente da Costa
# e-mail: betogil@gmail.com
```

APÊNDICE A – Arquivos do analisador simbólico-numérico

A.1 Arquivo do analisador léxico

Código A.1: Arquivo para geração de um analisador léxico

```
%{
#include "global.h"
#include "y.tab.h"

#include <stdlib.h>

#undef YY_INPUT
#define YY_INPUT(buf, result, max_size)      \
      {                                      \
          if (global_expr[0]=='\0')          \
              result = YY_NULL;              \
          else {                             \
              strcpy(buf, global_expr);      \
              result = strlen(global_expr)-1;\
      }                                      \
          global_expr[0]='\0';               \
      }
%}


white     [ \t]+
digit     [0-9]
integer   {digit}+
exponant  [eE][+-]?{integer}
real      {integer}("."{integer})?{exponant}?
variable  [a-z]


PLUS              "+"
MINUS             "-"
MULTIPLY          "*"
DIVIDE            "/"
POWER             "^"
SQRT              "sqrt"
```

```
35 SIN                  "sin"
36 COS                  "cos"
37 TAN                  "tan"
38 LOG                  "log"
39 LN                   "ln"
40 EXP                  "exp"
41 EQUAL                "="
42 LEFT_PARENTHESIS     "("
43 RIGHT_PARENTHESIS    ")"
44 NEW_LINE             "\n"
45 END                  ("end"|"END")
46
47
48 %%
49
50
51 {white}      { /* We ignore white characters */ }
52
53 {real}       {
54                     yylval = atof(yytext);
55                     return NUMBER;
56                 }
57
58 {variable}   {
59                     yylval = *yytext - 'a';
60                     return VARIABLE;
61                 }
62
63 {PLUS}                  {return PLUS; }
64 {MINUS}                 {return MINUS; }
65 {MULTIPLY}              {return MULTIPLY; }
66 {DIVIDE}                {return DIVIDE; }
67 {POWER}                 {return POWER; }
68 {SQRT}                  {return SQRT; }
69 {SIN}                   {return SIN; }
70 {COS}                   {return COS; }
71 {TAN}                   {return TAN; }
72 {LOG}                   {return LOG; }
73 {LN}                    {return LN; }
74 {EXP}                   {return EXP; }
75 {EQUAL}                 {return EQUAL; }
76 {LEFT_PARENTHESIS}      {return LEFT_PARENTHESIS; }
```

```
77 {RIGHT_PARENTHESIS}   {return RIGHT_PARENTHESIS; }
78 {NEW_LINE}             {return NEW_LINE; }
79 {END}                  {return END; }
80
81 %%
```

A.2 Arquivo do analisador sintático

Código A.2: Arquivo para geração de um analisador sintático

```
1  %{
2
3  #include "global.h"
4  #include <stdio.h>
5  #include <stdlib.h>
6  #include <math.h>
7  #include <string.h>
8  #include <ctype.h>
9
10 int yyerror(char*);
11
12 %}
13
14
15 %token   NUMBER
16 %token   VARIABLE
17 %token   PLUS MINUS MULTIPLY DIVIDE POWER SQRT
18 %token   SIN COS TAN
19 %token   LOG LN EXP
20 %token   EQUAL
21 %token   LEFT_PARENTHESIS RIGHT_PARENTHESIS
22 %token   NEW_LINE
23 %token   END
24
25 %left    EQUAL
26 %left    PLUS MINUS
27 %left    MULTIPLY DIVIDE
28 %left    SQRT
29 %left    SIN COS TAN
30 %left    LOG LN EXP
```

```
31 %left     NEG
32
33 %right    POWER
34
35 %start    Input
36
37
38 %%
39
40
41 Input:
42     /* Empty */
43     | Input Line
44     ;
45
46 Line:
47     NEW_LINE
48     | Expression NEW_LINE {
49           if (terminal) {
50               printf("Resultado: %lf\n",$1);
51           } else {
52               out_parse = $1;
53               return(0);
54           }
55          }
56     | VARIABLE EQUAL Expression { int ind = $1; sym[ind] = $3; }
57     | END { return(0); }
58     ;
59
60 Expression:
61     NUMBER      { $$ = $1; }
62     | VARIABLE { int ind = $1; $$ = sym[ind]; }
63     | Expression PLUS Expression      { $$ = $1 + $3;     }
64     | Expression MINUS Expression     { $$ = $1 - $3;     }
65     | Expression MULTIPLY Expression { $$ = $1 * $3;     }
66     | Expression DIVIDE Expression    { $$ = $1 / $3;     }
67     | MINUS Expression %prec NEG      { $$ = -$2;         }
68     | Expression POWER Expression     { $$ = pow($1,$3); }
69     | SQRT Expression     { $$ = sqrt($2);  }
70     | SIN Expression      { $$ = sin($2);   }
71     | COS Expression      { $$ = cos($2);   }
72     | TAN Expression      { $$ = tan($2);   }
```

```
73      | LOG Expression      { $$ = log10($2); }
74      | LN  Expression      { $$ = log($2);   }
75      | EXP Expression      { $$ = exp($2);   }
76      | LEFT_PARENTHESIS Expression RIGHT_PARENTHESIS { $$ = $2; }
77      ;
78
79
80  %%
81
82
83  int yyerror(char *s) {
84          printf("%s\n",s);
85  }
```

A.3 Arquivo de interface entre o hp^2FEM e o analisador simbólico-numérico

Código A.3: Arquivo de interface entre o *software* hp^2FEM e o analisador simbólico-numérico

```
1   %{
2
3   #include "global.h"
4   #include <stdio.h>
5   #include <stdlib.h>
6   #include <math.h>
7   #include <string.h>
8   #include <ctype.h>
9
10  int yyerror(char*);
11
12  %}
13
14
15  %token    NUMBER
16  %token    VARIABLE
17  %token    PLUS MINUS MULTIPLY DIVIDE POWER SQRT
18  %token    SIN COS TAN
19  %token    LOG LN EXP
20  %token    EQUAL
21  %token    LEFT_PARENTHESIS RIGHT_PARENTHESIS
22  %token    NEW_LINE
```

```
23 %token    END
24
25 %left      EQUAL
26 %left      PLUS MINUS
27 %left      MULTIPLY DIVIDE
28 %left      SQRT
29 %left      SIN COS TAN
30 %left      LOG LN EXP
31 %left      NEG
32
33 %right     POWER
34
35 %start     Input
36
37
38 %%
39
40
41 Input:
42     /* Empty */
43     | Input Line
44     ;
45
46 Line:
47     NEW_LINE
48     | Expression NEW_LINE {
49           if (terminal) {
50               printf("Resultado: %lf\n",$1);
51           } else {
52               out_parse = $1;
53               return(0);
54           }
55          }
56     | VARIABLE EQUAL Expression { int ind = $1; sym[ind] = $3; }
57     | END { return(0); }
58     ;
59
60 Expression:
61     NUMBER      { $$ = $1; }
62     | VARIABLE { int ind = $1; $$ = sym[ind]; }
63     | Expression PLUS Expression      { $$ = $1 + $3;     }
64     | Expression MINUS Expression     { $$ = $1 - $3;     }
```

```
65      | Expression MULTIPLY Expression { $$ = $1 * $3;     }
66      | Expression DIVIDE Expression   { $$ = $1 / $3;     }
67      | MINUS Expression %prec NEG     { $$ = -$2;         }
68      | Expression POWER Expression    { $$ = pow($1,$3); }
69      | SQRT Expression     { $$ = sqrt($2);  }
70      | SIN Expression      { $$ = sin($2);   }
71      | COS Expression      { $$ = cos($2);   }
72      | TAN Expression      { $$ = tan($2);   }
73      | LOG Expression      { $$ = log10($2); }
74      | LN  Expression      { $$ = log($2);   }
75      | EXP Expression      { $$ = exp($2);   }
76      | LEFT_PARENTHESIS Expression RIGHT_PARENTHESIS { $$ = $2; }
77      ;
78
79
80  %%
81
82
83  int yyerror(char *s) {
84          printf("%s\n",s);
85  }
```