APLICAÇÕES DA MANIPULAÇÃO ALGÉBRICA COMPUTACIONAL EM MECÂNICA DOS SÓLIDOS

Maria Cristina Valente de Almeida
Ricardo Mendes Junior
Centro de Estudos de Engenharia Civil - CESEC
Universidade Federal do Paraná - UFPr
Curitiba - Brasil

RESUMO

Neste trabalho apresentamos programas para geração de matrizes de rigidez de elementos finitos e solução de problemas de flexão pelo método de Ritz, baseados em uma linguagem simbólica de desenvolvimento recente, o PASCAL-MP (INPE BRASIL). A linguagem PASCAL-MP é um compilador PASCAL com facilidades para manipulação algébrica de polinômios. Elementos da implementação computacional são mostrados para ilustrar os procedimentos adotados e as facilidades da linguagem utilizada. Espera-se com isto ilustrar o potencial de uso do PASCAL-MP e extrapolar futuras aplicações de linguagens de manipulação simbólica em mecânica dos sólidos.

ABSTRACT

Programs for the generation of finite element stiffness matrices and for the solution of flexural problems by the Ritz Method, based on a recently developed symbolic manipulation system, PASCAL-MP, are presented. The PASCAL-MP is a PASCAL compiler with special commands for symbolic manipulations of polynomials. Elements of the computational implementation are presented in order to illustrate the procedures adopted and the processimg facilities utilized. In this way, we expect to show the potencial use of the PASCAL-MP and to extrapolate future applications of symbolic manipulation systems in solid mechanics.

INTRODUÇÃO

As primeiras aplicações da manipulação da manipulação algébrica computacional em mecânica dos sólidos, foram feitas em meados da década de 70 com os trabalhos de NOOR, JENSEN, ANDERSEN [1], entre outros .

O desenvolvimento de uma série de sistemas gerais para manipulação algébrica, tais como ALTRAN, MACSYMA, REDUCE, [1] , premitiu grandes simplificações nos tediosos desenvolvimentos algébricos necessários no método dos elementos finitos e outras técnicas de solução em mecânica dos sólidos. Ainda, a manipulação simbólica permitiu estabelecer uma importante ligação entre a análise e os cálculos numéricos, facilitando a parametrização dos problemas.

Entretanto, os sistemas disponíveis não se mostram indicadõs para uso generalizado em mecânica dos sólidos em virtude da falta de padronização nas aplicações, do alto custo de máquina e da própria falta de portabilidade dos sistemas .

No Brasil, alguns sistemas para manipulação simbólica computacional já se encontram em uso , e se iniciam desenvolvimentos nacionais . Entre estes, destaca-se a linguagem PASCAL-MP para manipulação de polinômios de múltiplas variáveis, que acreditamos poder, futuramente, apresentar as características desejáveis para linguagens desse tipo.

Os objetivos principais deste trabalho são apresentar a linguagem PASCAL-MP como ferramenta de desenvolvimento de soluções em mecânica dos sólidos e mostrar vantagens de utilização do sistema e sua potencialidade para outras aplicações.

As aplicações consideradas neste trabalho são: Aplicações do método de Ritz na solução do problema de flexão de uma viga engastada de ínércia variável; e geração de matriz de rigidez do elemento triangular de deformação constante (CST) pela formulação de elementos finitos .

Para cada um dos exemplos, a formulação matemática é fornecida , juntamente com elementos da implementação computacional e análise das principais vantagens no uso do sistema proposto.

A LINGUAGEM PASCAL-MP

A linguagem PASCAL-MP [2] é composta de um subconjunto da linguagem PASCAL, acrescido de comandos especiais para manipulação algébrica de polinômios de multiplas variáveis, e apresenta dois modos de operação : modo algébrico e modo numérico. Os comandos padrão da linguagem são executados no modo numérico, enquanto que os comandos especiais são executados no modo algébrico.

Além dos tipos de variáveis da linguagem, o PASCAL-MP possui o tipo POLI para o tratamento algébrico de polinômios.

Os polinômios podem ser definidos na forma de expressões polinomiais em função de suas variáveis, usando os operadores aritméticos, sendo que os coeficientes literais são tratados como variáveis. Por exemplo, para definir o polinômio $6+3x+2y+2x^2+5xy$, usaremos a declaração : P:= 6+3*x+2*y+2*x**2+5*x*y , sendo que P é do tipo POLI e x e y do tipo REAL.

As operações aritméticas no modo algébrico tem a mesma notação que no modo numérico (* , + , - , **) , com exceção da operação de divisão, que é uma função, no modo algébrico. Expressões com variáveis são sempre executadas em modo numérico, ao passo que expressões com polinômios são executadas em modo algébrico, não se permitindo a mistura de tipos .

As operações do modo algébrico são realizadas com o polinômio definido na forma de fração racional e portanto , operações de divisão são sempre mantidas implícitas. Por exemplo, sejam A, B, C, D polinômios na forma usual, e definindo P1: = A/B e P2: = C/D , então :

$$P3 := \frac{P1}{P2} = \frac{A/C}{C/D} = \frac{A}{B} * \frac{D}{C} = \frac{AD}{BC}$$

As funções para manipulação de polinômios são MMC, DERI, INTI , INTD, DIVI, DEN, NUM, EVAL, COEF, IVAR.

A seguir, descreveremos as funções utilizadas nesse trabalho:

DERI(P,x) - retorna um polinômio que é o resultado da derivação do polinômio P em relação à variável x.

INTI(P,x,c) - retorna um polinômio que é o resultado da integração indefinida do polinômio P em relação à variável x, com constante de integração c.

INTD(P,x,L1,L2) - retorna um polinômio que é o resultado da integração definida do polinômio P em relação à variável x, com limite inferior de integração L1 e limite superior de integração L2. Neste, caso, as variáveis L1 e L2 são substituídas por seus valores quando no cálculo da integral .

NUM(P) - retorna o numerador da fração polinomial P.

DEN(P) - retorna o denominador da fração polinomial P.

DIVI(P) - retorna uma fração polinomial cujo numerador é o quociente da divisão de NUM(P) por DEN(P) e cujo denominador é o resto dessa divisão .

IVAR(P,x) - retorna um polinômio que é o polinômio P em função apenas da variável x, tendo sido substituídas as outras variáveis de P pelos respectivos valores numéricos atuais .

COEF(P,x,e,) - retorna o valor do coeficiente do termo do polinômio P cujo expoente da variável x é "e". P deve estar definido apenas em função de x .

EVAL(P) - retorna o valor numérico do polinômio P após substituir todas as variáveis pelos seus valores numéricos atuais .

Para a impressão de polinômios usa-se o procedimento PWRITE (P), que imprime o polinômio P.

O PASCAL-MP, entretanto, não permite a estruturação por ARRAY e RECORD com polinômios .

APLICAÇÃO AO MÉTODO DE RITZ

Formulação Matemática

O problema de uma viga engastada de inércia variável submetida a um carregamento distribuído qualquer (figura 1) , é regido pelo seguinte funcional :

$$I = \frac{E}{2} \int [I(\frac{d^2v}{dx^2})^2 - qv] \, dx \qquad (1)$$

que representa a energia potencial total considerando apenas deformações por flexão .

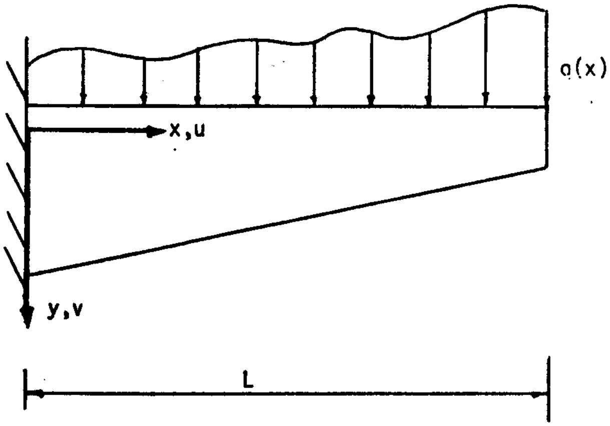

Fig.1 Viga Engastada

Este problema pode ser resolvido pelo Método de Ritz [3], onde se supõe uma solução aproximada da forma :

$$v_i = \sum_{j=1}^{i} A_j x^{j+1} \qquad (2)$$

para i fixo e ≥ 1

Após a substituição de v_i no funcional I da expressão (1), e executadas as operações de derivação e integração, obtém-se o funcional em função dos coeficientes A_j. A solução do problema é encontrada tornando-se estacionário o valor do funcional, resultando no seguinte sistema de equações algébricas lineares :

$$\frac{\partial I}{\partial A_i} = 0 \qquad , j = 1, 2, .., i \qquad (3)$$

Resolvido este sistema de equações por um método numérico conveniente (eliminação de Gauss), obtêm-se os coeficientes A_i, que seriam substituídos na expressão (2) para dar a solução aproximada.

Neste exemplo consideramos os parâmetros da viga como abaixo:

Carga $q(x) = q = 1{,}2$ tf/m
Altura $h(x) = ho + h_\ell . x/L$, sendo $ho = 70$ cm e $h_\ell = 40$ cm
Comprimento $L = 3$ m
Base $b = 20$ cm

Implementação Computacional

Na sequência apresentamos a programação em PASCAL-MP (programa VIGA) para a solução do problema especificado com i = 1, 2, 3, 4. Como resultados calcula-se o valor de v_i(deslocamento) e sua derivada v'_i (rotação) na abscissa x = L , bem como o valor do momento fletor e força cortante em ambos os extremos ,(ver figura 2) .

```
PROGRAM VIGA(INPUT,OUTPUT);
TYPE  MATRIZ= ARRAY [1..4,1..5] OF REAL;
      VETOR = ARRAY [1..4] OF REAL;
VAR   V,PI,INER,Q,PA,PX,TX,T        : POLI ;
      E,L,X,HL,HO,VN,IX             : REAL ;
      LIM,I,J,K                     : INTEGER ;
      A                             : VETOR ;
      B,H,DI,VF,DFV,D2FV,D3FV       : POLI ;
      C                             : MATRIZ ;

PROCEDURE GAUSS ( VAR X:VETOR; A: MATRIZ; N: INTEGER);
VAR I,J,K:INTEGER;
    S,T  :REAL;

PROCEDURE TROCALINHA(K:INTEGER);
VAR L:INTEGER;
    T:REAL;
BEGIN
  FOR L:=K+1 TO N-1 DO
    IF A[K,K] < A[L,K]
      THEN
        FOR J:=K TO N+1 DO
          BEGIN
            T:=A[L,J];
            A[L,J] :=A[K,J];
            A[K,J] :=T;
          END;
 END; (*TROCALINHA*)

 BEGIN (*GAUSS*)
   FOR K:=1 TO N DO
     BEGIN
       IF A[K,K] =0
        THEN
         TROCALINHA(K);
       T:=A[K,K] ;
       FOR J:=K+1 TO N+1 DO
         A[K,J] :=A[K,J] /T;
```

Fig.2 Programa VIGA

```
          FOR I:=K+1 TO N DO
            FOR J:=K+1 TO N+1 DO
               A[I,J] :=A[I,J] -A[I,K] *A[K,J] ;
         END; (* LOOP K *)
       X[N]:=A[N,N+1] ;
       I :=N;
       FOR K :=1 TO N-1 DO
         BEGIN
           I:=I-1;
           S:=0;
           FOR J:=I+1 TO N DO
             S:=S+A[I,J] *X[J] ;
           X[I] :=A[I,N+1] -S;
          END;
     END;


   PROCEDURE CALCULAFUNC;
   (* DEDUCAO ALGEBRICA DO FUNCIONAL *)
   VAR DX,FX,IFX,PE,K1    : POLI;
   BEGIN
     PE:= E;
     K1:= 1/2;
     DX:= DERI(V,X) ;
     DX:= DERI(DX,X) ;
     DX:= DX*DX ;
     FX:= INER*DX - Q*V;
     IFX:= INTD(FX,X,0,L);
     PI:= PE*K1*IFX;
     PWRITE(PI);
     WRITELN;
   END;


   BEGIN (* PROGRAMA PRINCIPAL *)
     LIM:=4 ;
     FA:=A[I];
     PX:=X;
     TX:=X ; T:=1/12 ;   (*INICIALIZACOES*)
     L:=3;
     H0:=0.7; HL:=0.4;
     H:= HL/L*X+H0;
     B:=0,2;
     INER:= B*H**3*T;
     E:=1.0;
     V:=0;
```

Fig. 2 Programa VIGA (continuação)

```
FOR I := 1 TO LIM DO
    BEGIN
      TX:=TX*PX;
      V :=V+PA*TX ; (* PA:=A[I] *)
      WRITELN;WRITELN;
      WRITELN(' POLINOMIO APROXIMADO V :');
      WRITELN;
      PWRITE(V);
      WRITELN;WRITELN('FUNCIONAL PI ');
      CALCULAFUNC;
      FOR J:=1 TO I DO
          BEGIN
            DI:=DERI(PI,A[J]);
            DI:=IVAR(DI,A);
            FOR K:=1 TO I DO
                C[J,K]:=COEF(DI,A[K],1);
            C[J,I+1]:=COEF (DI,A,0);
          END; (* J *)
          GAUSS(A,C,I);
          VF:=IVAR(V,X);
          WRITE('FUNCAO DESLOCAMENTO');
          PWRITE(VF);
          WRITELN;
          X:=L;
          VN:=EVAL(VF);
          WRITELN('DESLOCAMENTO NO EXTREMO ',VN);
          DVF:=DERI(VF,X);
          D2VF:=DERI(DVF,X);
          D3VF:=DERI(D2VF,X);
          VN:=EVAL(DVF);
          WRITELN('ROTACAO NO EXTREMO',VN);
          X:=0;
          IX:=EVAL(INER);
          VN:=EVAL(D2VF)*E*IX;
          WRITELN('MOMENTO DE ENGASTAMENTO',VN);
          VN:=-EVAL(D3VF)*E*IX;
          WRITELN('CORTANTE NO ENGASTE',VN);
          X:=L;
          IX:=EVAL(INER);
          VN:=EVAL(D2VF)*E*IX;
          WRITELN('MOMENTO NO EXTREMO',VN);
          VN:=-EVAL(D3VF)*E*IX;
          WRITELN('CORTANTE NO EXTREMO',VN);
      END;
  END;
```

Fig. 2 Programa VIGA (continuação)

No programa VIGA inicialmente definimos os tipos de variáveis. Os blocos seguintes contêm os "procedures" GAUSS e CALCULAFUNC. O procedure GAUSS resolve o sistema de equação de equações resultantes e o procedure CALCULAFUNC obtém a expressão polinomial do Funcional I (eq.1), onde as variáveis polinomiais V, INER, Q, são respectivamente a função deslocamento , inércia e a carga da viga. Note-se que para a definição do polinômio PI do funcional, foi necessária a cópia das variáveis numéricas , neste caso E/2 e Q, para variáveis polinomiais, uma vez que o PASCAL-MP não permite a mistura de variáveis algébricas e numéricas.

De acordo com a função INTD, as variáveis do limite da integral definida, foram substituídas por seus valores numéricos e o polinômio PI é agora uma função dos coeficientes A_j.

O próximo bloco após os "procedures" representa o programa principal. O trecho inicial do programa contém a definição das variáveis necessárias . O polinômio PA representa o coeficiente genérico A_i que será usado no desenvolvimento do polinômio V.

O trecho seguinte gera a solução do problema , para cada nova aproximação da função deslocamento, controlado pelo laço da variável I.

Inicialmente gera-se a função deslocamento em função dos coeficientes incógnitos A_i e da função de menor grau gerada anteriormente, armazenando-a em V. Em seguida dá-se a montagem do sistema de equações , de acordo com a equação (3). A matriz dos coeficientes resultante é numerica e a solução do sistema é obtida pelo procedure GAUSS. A função solução para o deslocamento é o resultado do polinômio V, onde os coeficientes incógnitos são substituidos pelos seus valores obtidos na solução do sistema, fornecendo a função final aproximada .

O trecho final correspondente ao pós-processamento onde calculam se a rotação e deslocamento em x = L , e os esforços nos extremos, a partir do polinômio V.

Procuramos com um exemplo, explorar a característica única do PASCAL-MP de combinar os modos numéricos e algébricos em um mesmo programa, e a aceitação de coeficientes reais para polinômios . Com isso, conseguiu-se reduzir o algebrismo necessário para a implementação do método de Ritz e ao mesmo tempo obter num único programa , a solução para diversas funções de aproximação.

GERAÇÃO DE MATRIZ DE RIGIDEZ

Formulação Matemática

A geração da matriz de rigidez do elemento triangular (fig.3) de deformação constante (CST) é feita seguindo-se a sistemática estabelecida do método dos elementos finitos [4] . Assim, a matriz de rigidez do elemento é dada pelas equações abaixo , considerando a ordenação dos nós conforme a figura 3 .

$$\underset{\sim}{K}^e = \int_{\Omega e} \underset{\sim}{B}^T \underset{\sim}{D} \underset{\sim}{B} \, d\Omega \qquad (4)$$

onde :

$$\underset{\sim}{B} = \begin{bmatrix} N_{1x} & N_{2x} & N_{3x} & 0 & 0 & 0 \\ 0 & 0 & 0 & N_{1y} & N_{2y} & N_{3y} \\ N_{1y} & N_{2y} & N_{3y} & N_{1x} & N_{2x} & N_{3x} \end{bmatrix} \qquad (5)$$

e

$$\underset{\sim}{D} = \begin{bmatrix} d_{11} & d_{12} & 0 \\ d_{21} & d_{22} & 0 \\ 0 & 0 & d_{33} \end{bmatrix} \qquad (6)$$

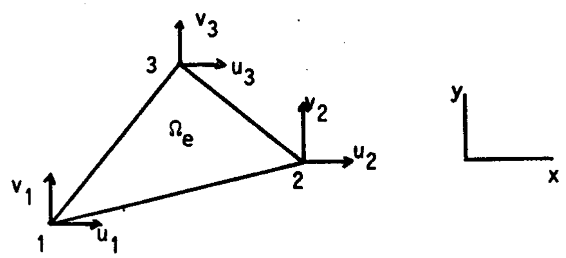

Fig.3 Elemento Finito CST

Adotando-se a formulação em coordenadas naturais , ℓ_1, ℓ_2, ℓ_3 teremos para as funções de forma desse elemento :

$$N_1 = \ell_1 \; ; \; N_2 = \ell_2 \; ; \; N_3 = \ell_3 \qquad (7)$$

A relação entre as coordenadas cartesianas e coordenadas naturais é dada por :

$$\ell_1 = c_{i1} + x\, c_{i2} + y c_{i3} \quad ; \; i = 1,2,3 \qquad (8)$$

esplicitamente temos

$$\ell_1 = (x_2y_3 - x_3y_2) + x (y_2 - y_3) + y (x_3 - x_2)$$
$$\ell_2 = (x_3y1 - y_3x_1) + x (y_3 - y_1) + y (x_1 - x_3) \quad (9)$$
$$\ell_3 = (x_1y_2 - y_1x_1) + x (y_1 - y_2) + y (x_2 - x_1)$$

onde x_i , y_i são as coordenadas dos nós do elemento.

Os elementos da matriz B são expressos por :

$$N_{ix} = \sum_{j=1}^{3} \frac{\partial N_i}{\partial \ell j} \frac{\partial \ell_i}{\ell x}$$
$$N_{iy} = \sum_{j=1}^{3} \frac{\partial N_j}{\partial \ell j} \frac{\partial \ell_j}{\ell y} \quad (10)$$

Para o estado plano de tensões temos os elementos não nulos da matriz D como :

$$d_{11} = E/(1 - \nu^2)$$
$$d_{12} = d_{21} = \nu E/(1 - \nu^2) \quad (11)$$
$$d_{33} = \frac{E}{2(1+\nu)}$$

onde E é o modulo de Young e ν o coeficiente de Poissou .

A integração sobre a área do elemento (A) é feita em coordenadas naturais , utilizando-se a fórmula :

$$\int_A \ell_1^r \ell_2^s \ell_3^t \ dA = \frac{r!\ s!\ t!}{(r+s+t+2!)} \ (2A) \quad (12)$$

Como resultado, ter-se-á a matriz de rigidez em função apenas das coordenadas de seus nós, e das propriedades mecânicas .

Implementação Computacional

O programa CST objetiva deduzir simbólicamente a matriz da rigidez do elemento CST, em função apenas das coordenadas cartesianas de valores numéricos em função de coordenadas fornecidas (ver fig.4).

Inicialmente procedure CALFORMA, define as funções de forma como polinômios, em coordenadas naturais. As coordenadas naturais formam um arranjo de 3 variáveis reais , L.

Procedure CALDERI, realiza a derivação das funções de forma, utilizando as expressões (10). Os coeficientes $\frac{\partial \ell_i}{\partial x}$ e $\frac{\partial \ell_i}{\partial y}$ são constantes

```
PROGRAM CST(INPUT,OUTPUT)
TYPE VETOR= ARRAY [1..3] OF REAL
     MATRIZ= ARRAY [1..6,1..6] OF REAL

VAR  N1,N2,N3,PC1,PC2,PC3,BJI,DJK,BKL: POLI;
     L,X,Y,D1,D2,D3                  : VETOR;
     N1X,N2X,N3X,N1Y,N2Y,N3Y         : POLI;
     C,MK                            : MATRIZ;
     A,E,NI                         : REAL
     NNE,NGL,NGLE,I,LL,J,K,ORD       : INTEGER;
     KIL,KINT,IKIL                   : POLI;

PROCEDURE CALCFORMA;
BEGIN
  N1:=L[1];
  N2:=L[2];
  N3:=L[3];
END;

PROCEDURE CALCDERI;
VAR J:INTEGER;
BEGIN
  N1X:=0.;
  N2X:=0.;
  N3X:=0.;
  N1Y:=0.;
  N2Y:=0.;
  N3Y:=0.;
  FOR J:=1 TO 3 DO
    BEGIN
      N1X:=N1X+DERI(N1,L[J])*PC2;
      N1Y:=N1Y+DERI(N1,L[J])*PC3;
      N2X:=N2X+DERI(N2,L[J])*PC2;
      N2Y:=N2Y+DERI(N2,L[J])*PC3;
      N3X:=N3X+DERI(N3,L[J])*PC2;
      N3Y:=N3Y+DERI(N3,L[J])*PC3;
    END;
END;

PROCEDURE CALCD;
BEGIN
  D1 :=E/(1-NI**2)
  D2 :=NI*E/(1-NI**2)
  D3 :=E/(2*(1+NI))
END;
```

Fig. 4 Programa CST

```
PROCEDURE PEGAD(VAR J:INTEGER,K: INTEGER, DJK:POLI);
BEGIN
  CASE J*3-3+K OF
    1: DJK:=D1;
    2: DJK:=D2;
    3: DJK:=0.;
    4: DJK:=D2;
    5: DJK:=D1;
    6: DJK:=0.;
    7: DJK:=0.;
    8: DJK:=0.;
    9: DJK:=D3;
END; (* FIM PEGAD*)

PROCEDURE PEGAB(VAR I:INTEGER; J:INTEGER; BIJ:POLI);

PROCEDURE PEGANX(J: INTEGER);
BEGIN
  CASE J OF
    1: BIJ:=N1X;
    2: BIJ:=N2X;
    3: BIJ:=N3X;
END;

PROCEDURE PEGANY(VAR J:INTEGER);
BEGIN
  CASE J OF
    1: BIJ:=N1Y;
    2: BIJ:=N2Y;
    3: BIJ:=N3Y;
END;

BEGIN (* PEGAB *)
  CASE I OF
    1: IF J>NNE THEN BIJ:=0. ELSE PEGANX(J);
    2: IF J<NNE THEN BIJ:=0. ELSE PEGANY(J-NNE);
    3: IF J<NNE THEN PEGANY(J) ELSE PEGANX(J-NNE);
END;  (* FIM DA PEGAB *)

PROCEDURE INMAT(VAR PIN:POLI; POUT:POLI);
VAR PA:POLI
BEGIN
  PA:=A;
  POUT:=PIN*PA;
END;
```

Fig. 4 Programa CST (continuação)

```
BEGIN  (* PROGRAMA PRINCIPAL *)
  FOR I:=1 TO 3 DO
    READ (X[I],Y[I] );
  READ (E,NI);
  C[1,1]:=X[2]*Y[3]-X[3]*Y[2];
  C[2,1]:=X[3]*Y[1]-X[1]*Y[3];
  C[3,1]:=X[1]*Y[2]-X[2]*Y[1];
  C[1,2]:=Y[2]-Y[3];
  C[2,2]:=Y[3]-Y[1];
  C[3,2]:=Y[1]-Y[2];
  C[1,3]:=X[3]-X[2];
  C[2,3]:=X[1]-X[3];
  C[3,3]:=X[2]-X[1];
  A:=(C[2,2]*C[3,3]-C[3,2]*C[2,3])/2;
  PC1:=C[J,1];
  PC2:=C[J,2];
  PC3:=C[J,3];

  CALCFORMA;
  CALCDERI;
  CALCD;
  NNE:=3;      (* NUM. NOS POR ELEMENTO *)
  NGL:=2;      (* NUM. GRAUS DE LIBERDADE POR NO *)
  NGLE:=NNE*NGL;
  FOR I:=1 TO NGLE DO
    BEGIN
    FOR LL:=1 TO NGLE DO
        BEGIN
        KIL:=0.;
        FOR J:=1 TO 3 DO
          BEGIN
          KINT:=0.;
          PEGAB(J,I,BJI);
          FOR K:=1 TO 3 DO
            BEGIN
            PEGAD(J,K,DJK);
            PEGAB(K,LL,BKL);
            KINT:=KINT+DJK*BKL;
          END;
          KIL:=KIL+BJI*KINT;
        END;
        INTNAT(KIL,IKIL);
          MK[I,LL]:=EVAL(IKIL);
          PWRITE(IKIL);
        WRITELN;
      END;
    END;
    FOR I:=1 TO NGLE DO
      BEGIN
      FOR J:=1 TO NGLE DO
        WRITE(MK[I,J]);
    END;
    WRITELN;
  END;
END;
```

Fig. 4 Programa CST (continuação)

(eq.8) e estão armazenados nas colunas 2 e 3 da matriz C, inicializada no início do programa principal , em função das coordenadas nodais .

Nota-se a matriz C, sendo composta de números reais , nas expressões de N1X, N2X e N3X é representada pelos polinômios PC2 e PC3 .

A definição dos elementos não nulos da matriz de constantes elásticas D , é feita no procedure CALCD , considerando estado plano de tensões . As variáveis E e NI são respectivamente o módulo de Young e o coeficiente de Poisson .

As rotinas PEGAD e PEGAD apontam os elementos das matrizes D e B necessários no produto matricial para a formação da matriz de rigidez. Estas rotinas são necessárias uma vez que o PASCAL-MP não permite a definição de ARRAY's de polinômios . Os parâmetros de entrada são os índices do elemento da matriz e o resultado é o polinômio correspondente a esse elemento .

A rotina INTNAT faz a integração em coordenadas naturais na área do elemento . Neste caso, os polinômios a serem integrados são constantes , e a rotina é específica para o elemento considerado .

No caso de funções de maior ordem no integrando, a utilização da expressão (11) implicaria em uma simulação da técnica de "pattern matching" [1] . Esta técnica consiste em procurar , termo a termo , em um polinômio dado, os expoentes de suas variáveis , até um expoente máximo definido em função do grau da interpolação utilizada .

O programa principal divide-se em 3 blocos : inicialização,dedução algébrica dos elementos da matriz de rigidez e cálculo numérico da matriz de rigidez , dadas as coordenadas dos nós do elemento . (ver figura 4)

A inicialização lê as coordenadas nodais (X e Y) e as constantes do material , define uma matriz C de transformação de coordenadas cartesianas para naturais (equação 8) e passa a calcular as funções de forma e suas derivadas .

O próximo bloco deduz um elemento $\{K_{i\ell}\}$ da matriz de rigidez utilizando o somatório :

$$K_{i\ell} = \sum_{j=1}^{3} \sum_{K=1}^{3} B_{ji}\, D_{jk}\, B_{k\ell} \;,\; i,\ell=1,2,\ldots,6 \qquad (13)$$

Cada elemento resultante é impresso , e o resultado numérico é armazenado em um arranjo MK para a impressão no bloco seguinte .

O uso do PASCAL-MP neste exemplo serviu para reforçar algumas de suas melhores características e ainda, mostrar certas limitações encontradas .

Entre as características a ressaltar , memciona-se o fato de combinar em um só programa os modos algébricos e numéricos e a facilidades de generalização do programa para a dedução de outros elementos triangulares .

Entretanto, sentiu-se , com mais frequência , neste exemplo a necessidade de outras formas de armazenamento de polinômios , comoARRAYS e RECORDS.Neste caso, a existência dessas estruturas de dados permitiria a eliminação das rotinas apontadoras (PEGAB e PEGAD) reduzindo o código do programa. Além disso, a inclusão de limites algébricos na interação definida, viria a eliminar a necessidade da técnica de " pattern matching" para integração de polinômios de maior ordem, pois as integrais seriam calculadas em ordenadas cartesianas .

CONCLUSÕES

A utilização da linguagem PASCAL-MP para manipulação algébrica em mecânica dos sólidos foi introduzida através de duas aplicações na solução de problemas pelo método de Ritz e na geração de matrizes de elementos finitos .

Em face dos resultados obtidos, concluí-se que o uso do PASCAL - MP em mecânica dos sólidos é viável e de grande potencial para aplicações futuras, em virtude de sua fácil utilização, grande portabilidade e da combinação de modos algébricos e numéricos em um só programa .

Para sua difusão como linguagem de manipulação algébrica, para uso em mecânica dos sólidos, cremos que a linguagem PASCAL-MP deveria possuir algumas capacidades adicionais, que listamos em ordem de prioridade :

i) ARRAYS de polinômios e RECORD comtipo POLI.

ii) Mistura de tipos algébricos (POLI) e numérico em expressões, evitando a definição de novas variáveis para geração de polinômios através de expressões .

iii) Comandos para substituição de uma variável por uma expressão, no modo algébrico .

iv) Comandos para leitura de polinômios e uso de outros arquivos além de INPUT e OUTPUT, para permitir que resulatdos algébricos gerados por um programa possam ser armazenados e posteriormente lidos por outro programa .

v) Comandos para fatoração de polinômios .

É esperado que este sistema, ou outros similares, reduzam significamente o esforço computacional na geração de elementos, bem como permitam a pesquisa em outros problemas, entre os quais , à primeira vista podemos enumerar[5] : extender a formulação para matriz de massa, vetores de carga, matrizes de rigidez geométricas e efeito térmicos ; admitir derivadas de maior ordem na componentes de deformações ; trabalhar com diferentes funções de forma nos diversos graus de liberdade ; trabalhar com diferentes graus de liberdade por nó; estudar matrizes para elementos com descontinuidade geométricas e materiais ; extender para materiais não lineares .

Estudos também podem ser desenvolvidos no sentido de comparar o tempo de execução para matrizes geradas simbolocamente com as geradas com integração numérica utilizada normalmente .

Finalmente de uma forma mais geral, o futuro da computação na engenharia (bem como da computação em si) será fortemente influenciado pela computação simbólica, promovendo modificações inclusive a nível de filosofia de trabalho e educacional em relação ao computador. Contribuições significativas podemos esperar em qualquer área cuja manipulações algébricas atualmente são difíceis ou intratáveis , bem como em área onde os estudos paramétricos são o interesse principal .

Com a necessária evolução dos processadores simbólicos problemas inerentes à manipulação simbólica terão de ser resolvidos para possibilitar o aumento da complexidade das aplicações, bem como a disponibilidade de facilidade de utilização conversacional e o problema da padronização .

REFERÊNCIAS

[1] NOOR,A.K. e ANDERSEN, C.M., "Computerized Symbolic Manipulation in Structural Mechanics - Progress and Potential". Computers & Structures. Vol. 10, 1979, págs. 95-118.

[2] LOMBARDI, J.C., "PASCAL-MP - Manipulador Algébrico e Numérico de Polinômios". Dissertação de mestrado em Computação Aplicada, 1984. Instituto Nacional de Pesquisas Espaciais, São José dos Campos, Brasil.

[3] BREBBIA, C.A. e FERRANTE, A.J., "Computational Methods for the Solution of Engineering Problems". Pentech Press, 1978, 354 págs.

[4] BREBBIA, C.A. e FERRANTE, A.J. (editores), "The Finite Element Technique". Editora URGS, 1975. Porto Alegre, Brasil, 410 págs.

[5] KORNCOFF, A.R. e FENVES, S.J., "Symbolic Generation of Finite Element Stiffness Matrices". Computers & Structures. Vol. 10, 1979, págs. 119-124.