XL CILAMCE
IBERO-LATIN AMERICAN CONGRESS ON COMPUTATIONAL METHODS IN ENGINEERING
NOVEMBER 11-14, 2019

ALGORITHM TO PERFORM GENERALIZED ASSEMBLY OF FINITE ELEMENT MATRICES

Natalie von Paraski
gatterr@gmail.com
IFPA - Santarém
Av. Mal. Castelo Branco, 621, 68020-570, Pará/Santarém, Brazil
Ítalo Rangel P. Mendes
italopenaforte@gmail.com
UFOPA/IEG - Santarém
Rua Vera Paz s/n, 68040-470, Pará/Santarém, Brazil
Alexandre da Silva Galvão
asgalvao@ufsj.edu.br
DTECH/CAP/UFSJ
Rodovia MG 443, s/n, km 7, 36420-000, Minas Gerais/Ouro Branco, Brazil

Abstract. Each type of computational system, with its dimension and degrees of freedom, makes the implementation of the finite element (FE) matrices assembly developments very specific, sometimes generating unnecessary redundancy in codes, which make them more difficult to understand, implement, read and update. A generalization of the assembly numerical methodologies for any kind of FE matrices can give us a unique algorithm that can be implemented once for all FEM computational problems, without the necessity to create a new code for each different case. This work presents an algorithm based on the general assembly strategy for finite-elements matrices for plane frames, implemented in Paraski [2]. This one algorithm tries to extrapolate the previous idea, being capable to perform the assembly of FE matrices for simple and multi connected systems, not only for plane frames, with one-dimensional FE bar, having three degrees of freedom, but for other kinds of finite element types, like FE triangular and quadrilateral in two dimensions, and FE hexahedron and tetrahedron in three dimensions, all of them having their own variations on their quantity of degrees of freedom. Some validation tests using MatLab codes for static analysis for plane frames with one dimensional FE bar, and Helmholtz equations problems with two-dimensional FE quadrilateral, was successfully made, showing the efficiency of this algorithm being used in different cases, for different elements working in different dimensions, with specific degrees of freedom.

Keywords: General, Assembly, Algorithm, FEM , Matrices

Introdução

A dificuldade na Montagem de matrizes de rigidez de sistemas com elementos fortemente conectados, para a realização da solução pelo método dos elementos finitos, ainda é uma problemática a ser explorada e que merece a atenção dos pesquisadores.

A solução de sistemas lineares é apresentada, para o sistema já montado, na forma.

$$\Delta F = K\Delta U \tag{1}$$

Onde K é a matriz de rigidez do sistema, ΔU é o vetor de deslocamentos ou de incógnitas do sistema e ΔF é o vetor de Forças ou vetor de solicitações do sistema Galvão [3].

Entretanto, no método dos elementos finitos, para que se chegue à equação definida acima, primeiramente o sistema é discretizado em várias partes menores, denominadas elementos finitos, os quais deverão ser definidos quanto ao tipo, quantidade de nós, dimensão e graus de liberdade, e calculados separadamente de acordo com a metodologia adotada, para posteriormente serem montados no sistema quando, finalmente, realiza-se a solução do sistema linear apresentada na eq. (1).

O foco deste trabalho encontra-se na montagem de um sistema FE qualquer, onde o sistema anteriormente discretizado em elementos finitos deve ser corretamente remontado para que a solução apresentada na eq. (1) seja calculada. Neste trabalho é apresentado um algoritmo que realiza de forma generalizada a montagem da matriz K, assim como do vetor de solicitações ΔF, visando o alcance de todas as possíveis combinações das variedades de situações citadas anteriormente.

Para tal, devem ser relembrados, definidos e generalizados alguns conceitos de MEF, como tipos de elementos finitos, restrições e conectividade entre elementos com seus respectivos graus de liberdade.

A seguir, a estratégia descrita por Paraski[5] que aborda a montagem de matrizes de rigidez apenas para pórticos planos, será utilizada como ponto de partida para a descrição de uma estratégia generalizada para a montagem de sistemas de elementos finitos quaisquer, e definição do algoritmo baseado nessa estratégia.

1. Tipos de Elementos Finitos

Parte-se do princípio de que um elemento finito pode ser definido, primordialmente, em função de sua dimensão, sua quantidade de nós e sua quantidade e tipos de graus de liberdade. Eles variam de acordo com o problema apresentado, podendo ser unidimensionais, bidimensionais, tridimensionais ou multidimensionais. Podem ser observados na Fig. 1, alguns exemplos:

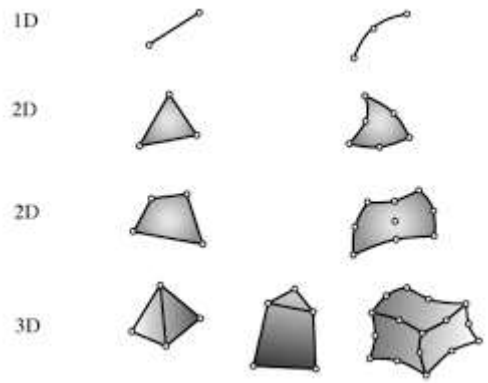


Figura 1: Alguns tipos de elementos finitos (Ferreira[8]).

Pode-se observar na Fig. 1 a variedade de elementos finitos em relação à quantidade de nós e em relação à dimensão, podendo-se observar elementos de linha, de área e de volume. Esses são apenas alguns exemplos, sendo o universo dos tipos de elementos finitos muito variado.

1.1 Multiconexões

Cada elemento finito é uma pequena parte de um sistema previamente definido, com um propósito a ser alcançado. Eles se conectam uns aos outros através de nós em comum, que são compartilhados por mais de um elemento no sistema. Os elementos podem estar simplesmente conectados, definindo sistemas "bem comportados" (de montagem mais simples) ou fortemente conectados, definindo sistemas mais complexos, onde a montagem se torna mais complexa, exigindo técnicas mais elaboradas, como as descritas em Paraski[5].

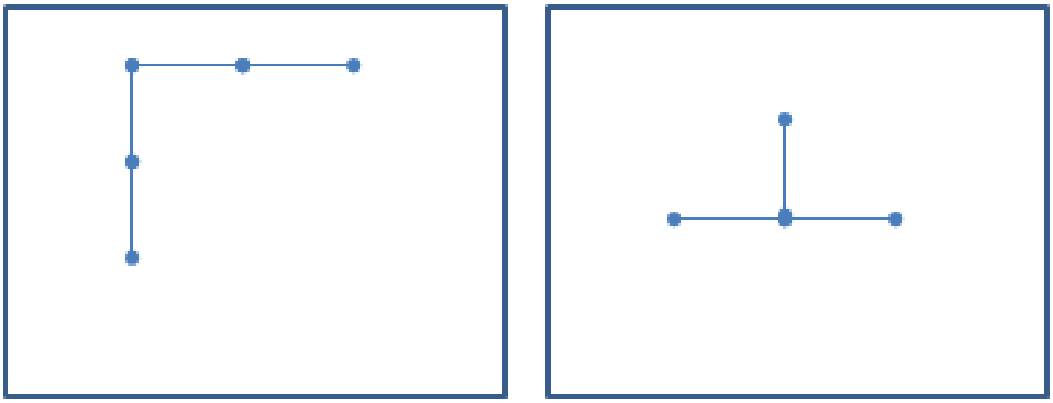


Figura 2: Exemplo simples de um sistema FE simplesmente conectado e um sistema FE fortemente conectado

Pode-se observar, na Fig. 2, que o sistema à esquerda representa um sistema de conexão simples, onde cada nó é compartilhado por no máximo 2 elementos, definindo uma estrutura "bem comportada". Já na figura à direita, pode-se notar a existência de um nó comum a vários elementos, o que já representa uma estrutura mais complexa, onde uma montagem mais acurada se tornará necessária para a garantia da solução do problema. Uma solução eficaz para esses casos reside na criação de uma matriz de conectividade de elementos, onde é possível definir, para qualquer situação, todas as conexões entre os elementos de um sistema através da identificação única de seus respectivos nós F. Teixeira-Dias [6].

Na próxima seção é definida uma matriz de conectividade entre elementos de um sistema FE de forma generalizada para aplicação dos conceitos definidos.

1.2 Matriz de Conectividade de Elementos

Essa matriz é responsável por definir quais nós pertencem a cada elemento. Cada linha dela representa um elemento e as colunas representam os nós daquele elemento. Sua organização pode ser representada pela Tabela 1.

Tabela 1. Representação Visual da Matriz de Conectividade

Elemento	Nó 1	Nó 2	...	Nó N
1				
2				
...	...	...	...	...
N				

A matriz de conectividade será utilizada para definir como ocorrerá a transição dos dados da matriz de rigidez do elemento, K_e para a matriz de rigidez do sistema, $K_{Sistema}$.

A Fig. 3 exemplifica um sistema com 4 elementos finitos e 6 nós. Numeram-se os nós e os elementos, independente uns dos outros.

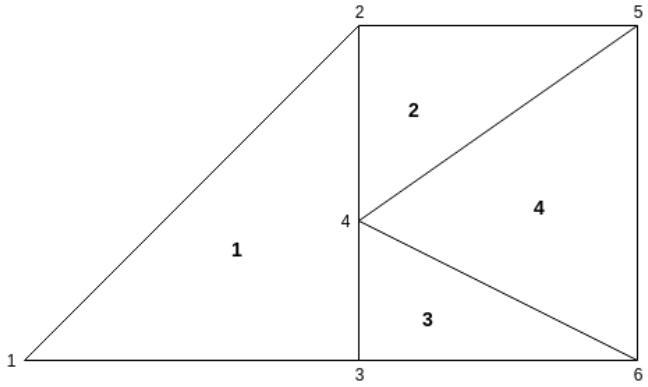


Figura 3. Elementos e seus respectivos nós

Em seguida, define-se a matriz de conectividade, com 4 linhas para os elementos e 6 colunas para os nós, marcando-se, para cada linha, os nós pertencentes a cada elemento.

Tabela 2. Representação visual da Matriz de Conectividade referente ao sistema exemplificado na Fig. 3

Elemento	Nó 1	Nó 2	Nó 3	Nó 4	Nó 5	Nó 6
1	X	X	X	-	-	-
2	-	X	-	X	X	-
3	-	-	X	X	-	X
4	-	-	-	X	X	X

Pode-se observar, na Tabela 2, que os nós pertencentes ao cada elemento foram marcados com um X. Essa matriz é definida no algoritmo de forma genérica, podendo possuir variadas quantidades de nós para cada elemento, possuindo dimensão referente à quantidade de elementos do sistema por quantidade de nós do sistema.

1.3 Definições do Sistema

Neste trabalho, parte-se do princípio que já é conhecido pelo leitor o método dos elementos finitos, estando o foco do mesmo voltado para a montagem da matriz K e a redução do vetor de solicitações ΔF. Para tal, deve-se definir e generalizar as estratégias referentes a alguns elementos, definidos em Paraski [5], que serviram de base para este trabalho.

1.4 Vetor de Forças ou Vetor de Solicitações

O vetor de solicitações ΔF, também definido na literatura tradicional, representa as solicitações impostas ao sistema para que ele altere sua configuração original. Ele é montado com a mesma configuração do vetor de restrições e deve ser reduzido para possuir a mesma dimensão do sistema, para que a solução da Eq. (1) seja alcançada.

Neste trabalho será abordada a mesma estratégia da matriz de conectividade para a organização dos dados, para posterior montagem do vetor de solicitações, descrita em Paraski[5]. Neste caso, as

informações constantes nas linhas serão os nós do sistema e as informações das colunas serão os tipos de solicitações existentes no sistema, conforme a Tabela 3.

Tabela 3. organização das solicitações

Nó\Solicitação	Solicitação 1	Solicitação 2	...	Solicitação N
1				
2				
...	...	...	...	...
N				

O preenchimento desta tabela é realizado através da inserção do valor numérico do tipo de solicitação referente a cada nó do sistema. Nas células onde não houver solicitação a ser realizada, deve ser inserido o valor zero, como pode ser exemplificado na Tabela 4.

Supondo-se que no sistema haja um total de 4 possibilidades de solicitações (não sendo especificado aqui quais são, apenas a sua existência) podendo ou não contemplar todos os nós do sistema, o preenchimento poderia ser feito como representado na Tabela 4.

Tabela 4. representação da organização das solicitações do sistema

Nó	Solicitação 1	Solicitação 2	Solicitação 3	Solicitação 4
1	4	0	0	0
2	0	-12	0	0
3	0	0	0	0
4	0	0	0	-7
5	0	0	0	0
6	0	0	0	0

As informações dessa tabela, combinadas com as informações da matriz de graus de liberdade, servirão para a montagem do vetor de solicitações descrito em Paraski[5].

1.5 Definição do Vetor de Restrições

O vetor de restrições segue, basicamente, as mesmas regras do vetor de solicitações.

Tabela 5. Organização das restrições do sistema

Nós/restrições	R 1	R 2	...	R N
1				
2				
...	...	...	...	...
N				

Pode-se observar, na Tabela 5, que sua representação é análoga à representação das solicitações

do sistema, com diferença de que deverão ser apenas marcadas as células da tabela onde houverem restrições.

Deve-se marcar os graus de liberdade que possuem restrições em cada nó, de maneira análoga ao preenchimento da matriz de conectividade, conforme exemplo na Tabela 6.

Tabela 6. Marcado os graus de liberdade

Nós/Restrição	R1	R 2	R3	R4
1				
2			R	
3	R			
4		R		
5				
6				

Partindo do princípio de que elementos distintos podem ser utilizados em um mesmo sistema, e esses podem possuir quantidades e tipos de graus de liberdade diferentes, torna-se necessária uma matriz que define quais graus de liberdade existem em cada nó do sistema.

1.6 Matriz de Graus de Liberdade

A matriz de graus de liberdade é criada utilizando-se as informações da matriz de restrições. Ela é de grande importância, pois seus dados servirão de base para a inserção das células de K_e em $K_{Sistema}$.

A montagem da matriz de graus de liberdade é feita inserindo-se nas linhas todos os nós do sistema (na mesma ordem em que foram inseridos na matriz de conectividade) e nas colunas todos os possíveis graus de liberdade do sistema.

Tabela 7. Matriz de graus de liberdade

Nós/GL	GL 1	GL 2	...	GL N
1				
2				
...	...	...	...	...
N				

Neste trabalho, seu preenchimento será realizado em duas etapas. Na primeira, para garantia da generalização do sistema, deverá ser inserida a informação da existência ou não de graus de liberdade para os respectivos nós do sistema, como representado na Tabela 8.

Suponha que, hipoteticamente, na Tabela 8 são representados os graus de liberdade do sistema definido na Fig. 3, onde todos os elementos possuem apenas os 3 primeiros graus de liberdade, com exceção do nó 4, que possui 4 graus de liberdade. São marcados, portanto, os nós referentes aos elementos que não possuem o determinado grau de liberdade, enquanto os outros permanecem desmarcados, definindo a existência dos graus de liberdade para os nós do sistema.

Tabela 8. Definição da existência dos graus de liberdade para cada nó

Nós/GL	GL1	GL2	GL3	GL4
1				X
2				X
3				X
4				
5				
6				

A etapa seguinte visa enumerar todos os graus de liberdade existentes no sistema que não possuam restrições, mantendo-os associados aos seus respectivos nós. Essa tabela será utilizada, juntamente com a tabela de conectividade, na realização da transição das informações dos dados de K_e para $K_{Sistema}$.

Tabela 9. Dados das tabelas 6 e 8

Nós/GL	GL1	GL2	GL3	GL4
1				X
2			R	X
3	R			X
4		R		
5				
6				

Cada posição da tabela que não possuir restrição explícita na Tabela 6, de restrições, e não tiver sido marcada como inexistente na Tabela 8, que define os graus de liberdade existentes, deverá ser numerada, sequencialmente, de 1 até a última célula da tabela. As células marcadas com restrições (Tabela 6) ou inexistência de graus de liberdade (Tabela 8) deverão ser preenchidas com o valor 0 (zero), como pode ser exemplificado na Tabela 10.

Tabela 10. Exemplo de matriz de graus de liberdade já preenchida

Nós/GL	**GL1**	**GL2**	**GL3**	**GL4**
1	1	2	3	0
2	4	5	0	0
3	0	6	7	0
4	8	0	9	10
5	11	12	13	14
6	15	16	17	18

Repare na Tabela 10 que a última posição diferente de zero da matriz possui o valor 18, que é

exatamente a quantidade de graus de liberdade existentes nos nós do sistema (Tabela 8) subtraída da quantidade de restrições (Tabela 6). Este valor define a dimensão da matriz global do sistema $K_{Sistema}$, onde serão inseridos os valores da matriz K_e. Ambos serão descritos nas próximas etapas.

2 Definição da Matriz Global do Sistema - $K_{Sistema}$

A Matriz Global do Sistema $K_{Sistema}$ tem sua dimensão definida pelo valor da última célula diferente de zero da matriz de graus de liberdade, que no caso do exemplo hipotético da Tabela 10 obteve o valor 18. Essa simples verificação pode ser utilizada para garantir a eficácia da análise, pois a não confirmação dos valores indicará erro no processo, que deverá ser verificado desde o início.

2.1 Inserção da K_e em $K_{Sistema}$

Antes da realização desse passo, deve-se lembrar que, neste trabalho, parte-se do princípio de que as matrizes K_e já se encontram previamente calculadas. Para maiores detalhes e exemplificação, pode-se consultar o trabalho de Paraski[5].

Uma vez calculada a matriz K_e, o primeiro passo para a inserção dos valores de K_e na matriz $K_{Sistema}$ reside em definir o "local de destino" dos elementos de cada K_e. Para que esta transição ocorra, são utilizados os valores associados aos nós na Matriz de Graus de Liberdade (Tabela 10). Deve-se mudar a referência dimensional de K_e de forma que a nova dimensão mostre em que posição de $K_{Sistema}$ o valor da célula deverá ser inserido.

Deve-se lembrar que cada elemento da estrutura possui uma determinada quantidade de nós, sendo que cada um desses nós possui uma determinada quantidade de graus de liberdade. Esse encadeamento dos graus de liberdade de cada nó de um determinado elemento finito define a dimensão de K_e para qualquer elemento finito, conforme explícito na Tabela 11.

Tabela 11: Dimensionamento da K_e para um elemento qualquer

Dimensão da Matriz K_e para um elemento qualquer			NÓ 1				NÓ 2				...	NÓ n			
			GL 1	GL 2	. . .	GL n	GL 1	GL 2	. . .	GL n	...	GL 1	GL 2	. . .	GL n
			1	2	.	.	.	.	.	.	.	.	.	.	N
NÓ 1	GL1	1													
	GL2	2													
	...	.													
	GL n	.													
NÓ 2	GL1	.													
	GL2	.													
	...	.													
	GL n	.													
…		.													
NÓ n	GL1	.													
	GL2	.													
	...	.													
	GL n	N													

Utilizando o sistema na Fig. 3 como exemplo, o elemento 1 possui os nós 1, 2, 3 com os graus de liberdade GL1, GL2, GL3, respectivamente, definindo, portanto, o seguinte dimensionamento para a

sua matriz K_e, conforme a Tabela 12:

Tabela 12: Exemplo de dimensionamento de K_e para o elemento 1

Dimensão da Matriz K_e para o elemento 1			NÓ1			NÓ2			NÓ3		
			GL1	GL2	GL3	GL1	GL2	GL3	GL1	GL2	GL3
			1	2	3	4	5	6	7	8	9
NÓ1	GL1	1									
	GL2	2									
	GL3	3									
NÓ2	GL1	4									
	GL2	5									
	GL3	6									
NÓ3	GL1	7									
	GL2	8									
	GL3	9									

Uma vez definida a dimensão de K_e para um determinado elemento, deve-se substituir seus índices pelos novos índices referentes aos graus de liberdade dos nós desse elemento definidos na matriz de graus de liberdade (Tabela 10).

No exemplo do elemento 1, a matriz K_e deve assumir os valores iniciais 1, 2, 3, 4, 5, 0, 0, 6, 7, conforme definição da matriz de graus de liberdade na Tabela 10:

Tabela 13: novos indices para a matriz K_e do elemento 1

Dimensão da Matriz K_e para o elemento 1			NÓ1			NÓ2			NÓ3		
			GL1	GL2	GL3	GL1	GL2	GL3	GL1	GL2	GL3
			1	2	3	4	5	0	0	6	7
NÓ1	GL1	1									
	GL2	2									
	GL3	3									
NÓ2	GL1	4									
	GL2	5									
	GL3	0									
NÓ3	GL1	0									
	GL2	6									
	GL3	7									

A atualização para os novos valores indiciais é crucial, pois eles definem exatamente as posições linha/coluna de $K_{Sistema}$ que receberão as informações contidas em cada célula de K_e de cada elemento finito.

Deve-se atentar que as células que possuírem índice de linha e/ou coluna 0 (zero), deverão ser descartadas. Isso ocorre devido à necessidade de observação às condições de contorno do sistema.

Essas condições definem que os valores ligados às restrições do sistema não poderão ser alterados. Isso quer dizer que os valores das células onde houver zero nas posições definidas pelo novo dimensionamento não serão inseridos em $K_{Sistema}$, ou seja, serão ignorados.

Outro ponto a ser definido é o da ocupação de valores de elementos diferentes nas mesmas células de $K_{Sistema}$. Quando isso ocorrer, os valores apenas serão adicionados aos valores já existentes na célula, e a mesma possuirá, cumulativamente, os valores de mais de um elemento.

Tabela 14: Matriz $K_{Sistema}$ baseada na matriz de graus de liberdade (Tabela 10) a ser preenchida com os dados das matrizes K_e do exemplo da Fig. 3.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1																		
2																		
3																		
4																		
5																		
6																		
7																		
8																		
9																		
10																		
11																		
12																		
13																		
14																		
15																		
16																		
17																		
18																		

Uma vez definida e preenchida completamente a matriz $K_{Sistema}$, utiliza-se a mesma lógica para a construção do vetor de solicitações e remoção das restrições no mesmo, o qual pode ser exemplificado com mais detalhes em Paraski[5], sendo possível realizar finalmente, a operação de solução de sistemas lineares definida na Eq (1).

Deve-se observar a existência de um padrão bem definido utilizado na a realização das operações descritas neste trabalho, o que leva, consequentemente, à definição de um algoritmo capaz de realizar a montagem de matrizes $K_{Sistema}$, independente do tipo de elemento, suas possíveis dimensões e possíveis combinações. Este algoritmo foi chamado de PP-Assembly e será apresentado a seguir.

2.2 Algoritmo PP-Assembly

O algoritmo PP-Assembly pode ser utilizado para a realização da montagem da matriz $K_{Sistema}$ e do vetor de solicitações, para qualquer tipo de sistema baseado em elementos finitos, com qualquer quantidade de graus de liberdade, independente de seu dimensionamento.

Os argumentos de entrada definidos no pseudocódigo representam estruturas e variáveis definidas na metodologia de elementos finitos. Paraski[2] define os argumentos de entrada do algoritmo:

- *Nnos* representando a quantidade de nós do sistema,
- *qtdGrauLib* sendo a quantidade de graus de liberdade do nó,
- *vetRest* é o vetor de restrições do sistema,
- *vetForce* representa o vetor de solicitações do sistema antes da remoção das condições de contorno,
- *nElem* é a quantidade de elementos do sistema,
- *conect* define a matriz de conectividade entre cada elemento e seus nós.

A matriz K_e, que representa a matriz de rigidez para cada elemento já calculada, deverá ser enviada durante a execução do algoritmo, pois os valores numéricos são diferentes para cada K_e de cada elemento.

O algoritmo retorna a matriz $K_{Sistema}$ já preenchida, assim como o vetor de solicitações, considerando as restrições do sistema que definem suas condições de contorno, prontos para serem calculados.

PP-Assembly(*Nnos, qtdGrauLib, vetRest, vetForce, nElem, conect*)

```
 1.   MgrauLib[ 1::Nnos, 1::qtdGrauLib + 1 ] = 0
 2.   contador  = 1
 3.   contador2 = 1
 4.   for i =1 to Nnos
 5.        MgrauLib[ i,1 ] = i
 6.        for j = 1 to qtdGrauLib
 7.             if (vetRest [ contador ] == 1)
 8.                  MgrauLib[ i, j+1] = contador2
 9.                  vetForceRed [contador2] = vetForce[contador]
10.                  contador2 = contador2 + 1
11.             end
12.        contador = contador + 1
13.        end
14.   end
15.   dimensao = contador2 - 1
16.   K[1..dimensao, 1..dimensao] = 0
17.   NosElem = length(conect [1, *] )
18.   indexGlobal [ qtdGrauLib*NosElem ]
19.   for i =1 to nElem
20.        lib = 1
21.        for j =1 to NosElem
22.             if (conect [ i,  j ] != 0)
23.                  no = conect [i, j ]
24.                  for contador=1 to qtdGrauLib
25.                       indexGlobal[lib] = MgrauLib[no, contador+1]
26.                       lib = lib + 1
27.                  end
28.             end
29.        end
30.       Ke = get(Ke, i)
31.        for j =1 to lenght(Ke)
32.             for l=1 to length(Ke)
33.                  if (indexGlobal [ j ] != 0 and indexGlobal[l] != 0)
34.                       jj = indexGlobal [ j ]
35.                       ll = indexGlobal [ l ]
36.                       K [jj, ll ] = K [jj, ll ] + Ke [j, l ]
37.                  end
38.             end
39.        end
40.   end
41.   return(K,vetForceRed)
```

O algoritmo acima recebe como argumentos de entrada as estruturas definidas neste trabalho, as quais são baseadas nos trabalhos de Paraski[2] e foram adaptadas para casos gerais de MEF. Através

dele, é realizada a construção da matriz $K_{Sistema}$ e são removidas as restrições do vetor de Solicitações, possibilitando o cálculo dos mesmos, através de solução de sistemas lineares.

O algoritmo apresentado neste trabalho é detalhado a seguir, conforme a numeração de suas linhas.

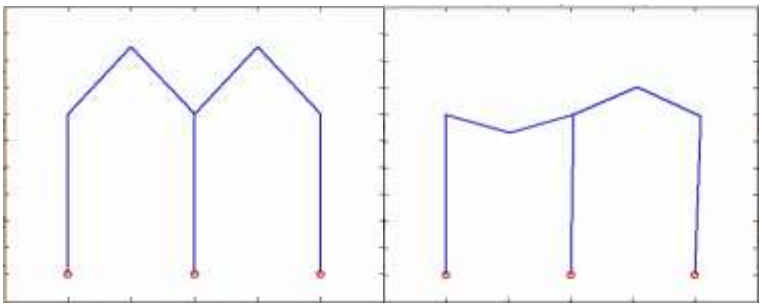
1. Cria uma matriz utilizando a quantidade de nós no sistema e a quantidade de graus de liberdade de um nó e inicializa a matriz com zeros. A quantidade de nós define as linhas da matriz e a quantidade de graus de liberdade define a quantidade de colunas Thomas[7].
2. A variável *contador* é inicializada e é utilizada para percorrer as estruturas originais, como o vetor de restrições.
3. A variável *contador2* também é inicializada e é responsável por enumerar os graus de liberdades válidos(sem restrições). É ele que irá definir a dimensão da matriz global $K_{Sistema}$
4. Inicializa a estrutura de repetição começando pelo índice número 1 e indo até a quantidade total de nós do sistema.
5. Está linha é responsável por identificar na primeira coluna da matriz de graus de liberdade os nós cujas as informações de numeração de graus de liberdade serão inseridas ao longo de cada linha.
6. Inicializa outra estrutura de repetição, dessa vez percorrendo a quantidade total dos graus de liberdade do nó.
7. Essa linha verifica se no vetor de restrições possua o valor 1, caso possua irá executar as linhas dentro da estrutura da decisão. Neste algoritmo foi definido que o valor 1 é para representar a ausência de restrição e o valor 0 a existência de restrição.
8. Nesta linha é inserido o valor numérico de grau de liberdade válido, que irá servir como índice a ser utilizado na transição da matriz K_e para a matriz $K_{Sistema}$.
9. Nesta linha a informação de força que não possui restrição é passada para o novo vetor de força, que deverá possuir a mesma dimensão da matriz $K_{Sistema}$.
10. Efetua a atualização do *contador2,* modificando-o para o próximo valor indicial da matriz $K_{Sistema}$ e do vetor de forças reduzido.
11. Finaliza a estrutura de decisão. (Linha 7).
12. Atualiza a variável *contador*, adicionando uma unidade para a verificação da próxima restrição.
13. Finaliza a estrutura de repetição. (Linha 6).
14. Finaliza o laço de repetição. (Linha 4).
15. Nesta linha, é determinada qual irá ser a dimensão da matriz de rigidez $K_{Sistema}$ e do vetor de forças reduzido utilizando como parâmetro a variável *dimensão*.
16. Utilizando a variável *dimensão* para definirmos a quantidade de linhas e colunas da matriz de rigidez da $K_{Sistema}$, com todos os elementos da matriz zerados.
17. Calcula a quantidade de nós do elemento utilizando a matriz de conectividade, passando por todas as colunas mas engatada na primeira linha.
18. Cria um vetor com todos os índices, usando as informações de quantidade de graus de liberdade multiplicando pela quantidade de nós do elemento, com isso é definida a quantidade de posições do vetor.
19. Nesta linha iniciamos o laço de repetição que irá percorrer todos os elementos do sistema.
20. A variável *lib* servirá para definir o vetor de índices de transição de K_e para $K_{Sistema}$.
21. Inicializa o laço para percorrer os nós do elemento atual.
22. Realiza a verificação se existe nó no elemento em questão, se for diferente de zero é porque existe um nó.
23. Se o nó existir, sua numeração de identificação é passada para a variável *no*.

24. Inicializa o laço de repetição para percorrer os graus de liberdade do nó que foi armazenado na variável *no*.
25. Constrói um novo vetor de índices de K_e para realizar a transição para $K_{Sistema}$.
26. Atualiza a variável *lib*, para posicionamento da inserção do novo índice de transição de K_e para $K_{Sistema}$.
27. Finaliza a estrutura de repetição. (Linha 24).
28. Finaliza a estrutura de decisão. (Linha 22).
29. Finaliza a estrutura de repetição. (Linha 21).
30. A função *get()* recebe a matriz de rigidez K_e do elemento *i* e a armazena na variável matricial *Ke*, para posteriormente efetuar a inserção de seus valores em $K_{Sistema}$.
31. Inicializa a estrutura de repetição começando com 1 indo até o tamanho total da matriz de rigidez local K_e, através da função *lenght*, que retorna um valor numérico com o tamanho da matriz.
32. A linha 32 faz exatamente a mesma coisa que a linha 31.
33. Verifica se os índices novos de linha e coluna são diferentes de zero para que a informação da célula possa ser passada para a matriz $K_{Sistema}$.
34. A variável *jj* recebe o dado do índice de transição para $K_{Sistema}$ do vetor *indexGlobal*, para facilitação da leitura do algoritmo.
35. A variável *ll* recebe o dado do índice de transição para $K_{Sistema}$ do vetor *indexGlobal* para facilitação da leitura do algoritmo.
36. A informação da célula analisada na matriz de rigidez local K_e é passada para posição definida para a matriz de rigidez Ksistema, sendo seu valor somado ao valor já existente em Ksistema, lembrando que a matriz foi inicializada com zero. (Linha 16).
37. Finaliza a estrutura de decisão. (Linha 33).
38. Finaliza a estrutura de repetição. (Linha 32).
39. Finaliza a estrutura de repetição. (Linha 31).
40. Finaliza a estrutura de repetição. (Linha 19).
41. Retorna as informações da matriz de rigidez global Ksistema e o vetor de forças reduzido para a realização da solução do sistema linear (eq. 1).

3 Resultados e Discussões

O algoritmo definido neste trabalho foi baseado na implementação computacional dos trabalhos de Paraski[2] e Chagas[1], os quais utilizaram formulações de elementos finitos distintas tanto com relação ao tipo de elemento, quanto em quantidade de graus de liberdade e dimensionamento, obtendo resultados consistentes. Paraski[2] utilizou elementos de viga-coluna bi-dimensionais compostos por 2 nós em suas extremidades e 3 graus de liberdade (x, y, rotação) para cada.

Figura 4. Deformação da Estrutura Multiconectada



Pode-se observar, na Fig 4, representando uma animação extraída da implementação computacional de Paraski[2], no lado esquerdo, a estrutura indeformada de um pórtico plano do tipo galpão duplo, contendo um nó central conectando 3 elementos. À direita, observa-se a estrutura deformada com flambagem evidente em sua coluna central, o que demonstra o sucesso da implementação computacional do algoritmo definido neste trabalho. Chagas[1], utilizou elementos finitos unidimensionais e bidimensionais quadriláteros compostos por 4 nós em seus vértices, o que demonstra que a variação do elemento finito, assim como a quantidade de graus de liberdade dos nós não influencia na performance do algoritmo.

4 Conclusões e trabalhos futuros

A importância de um algoritmo simples que resolva variados os casos de montagem se torna evidente ao passo que deixa de ser necessária a criação de um programa diferente para cada diferente elemento com diferentes graus de liberdade.

Apesar de baseado na codificação em linguagem *MatLab*, optou-se pela escrita deste algoritmo no formato de pseudo-código para facilitar a implementação computacional em qualquer linguagem de programação escolhida pelo usuário.

Deve-se observar que a utilização deste algoritmo, para resolver questões com elementos multiconectados, permite que sejam criadas estruturas mais complexas com as mesmas formulações de elementos finitos já estudadas e definidas, apenas se adaptando a parte da montagem do sistema.

A criação de uma nova matriz de conectividade entre nós e seus elementos, denominada neste trabalho por matriz de graus de liberdade define uma nova relação de conectividade no sistema, permitindo uma melhor visualização das regras que permeiam a montagem das matrizes de rigidez e vetores de solicitações orientados à célula.

Nos próximos trabalhos serão verificadas implementações para outros elementos finitos, variando-se as dimensões, quantidades de nós, graus de liberdade e aplicações finais.

Serão verificados e, se necessário, realizadas adaptações no algoritmo para casos que envolvam refinamento de malha, elementos finitos distintos no mesmo sistema e nós com variação de graus de liberdade, como encontrados nos casos de MEFG (método dos elementos finitos generalizados).

Referencia

[1] Chagas, B. O. Métodos de Elementos Finitos e Diferenças Finitas para a Equação de Helmholtz, Volta Redonda, 2013.
[2] Paraski, N. V. Análise Não Linear de Pórticos Planos via Matlab, Volta Redonda: Mestrado em Modelagem Computacional em Ciência e Tecnologia, EEIMVR, UFF, 2012.
[3] Galvão, A. S. Formulações geométricas não lineares de elementos finitos para análise de sistemas estruturais metálicos reticulados planos, dissertação de Mestrado, Ouro Preto: Programa de Pós-Graduação em Engenharia Civil, DE-CIV/Escola de Minas/UFOP, 2000.
[4] Japhet, C., Cuvelier, F., Scarella, G. An efficient way to perform the assembly of finite element matrices in Matlab and Octave. HAL-00785101v1. 2013
[5] Paraski, N. V. Estratégia Generalizada para Montagem de Matriz Global em Elementos Finitos, Volta Redonda, EEIMVR, UFF, 2013.
[6] F. Teixeira-Dias. Método dos Elementos Finitos - Técnicas de Simulação Numérica em Engenharia, n. 2, 2018.
[7] Thomas H. Cormen. Algoritmos - Teoria e Prática, n. 2, pg 91, 2012.
[8] Ferreira, Micael Rodrigues. Análise não linear por elementos finitos de vigas de betão armado à torçã, Covilhã, Departamento de Engenharia Civil e Arquitetura, Universidade da Beira Interior, 2016.