Università degli Studi di Padova
Dipartimento di Ingegneria Civile, Edile ed Ambientale
Corso di Laurea Magistrale in Ingegneria Civile
Indirizzo Strutture

Tesi di Laurea Magistrale

Costruzione di un codice non lineare agli elementi finiti per l'analisi di elementi shell a geometria esatta

Relatore:
Prof.ssa Valentina Salomoni

Laureando:
Stefano Danetti

Anno Accademico 2022-2023

Indice

Introduzione

L'analisi di strutture a guscio è ampiamente diffuso nell'ambito della meccanica strutturale, si può pensare, ad esempio, alle coperture a volta, alle condotte o ai silos, oppure, ancora, allo studio al dettaglio di travi costituite da profili a parete sottile, ampiemente utilizzate per le opere in acciaio.

Nonostante esistano soluzioni analitiche, esse sono relative solo ad alcuni casi particolari soddisfacenti determinate condizioni di simmetria; complicando l'analisi introducendo il calcolo non lineare per geometria, il numero si riduce ulteriormente. Per avere soluzioni rapide, efficaci, e relative a pressoché qualsiasi problema dell'ingegneria strutturale, è necessaria l'analisi numerica e, in particolare, al calcolo agli elementi finiti.

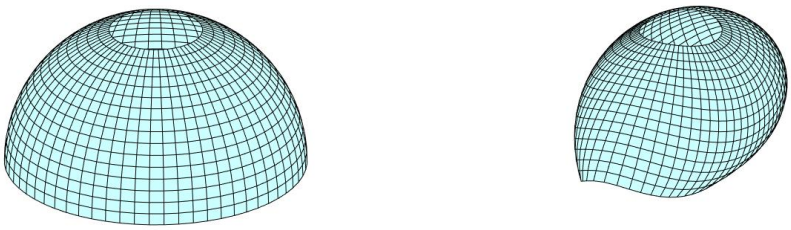


Figura 1: *A sinistra la configurazione iniziale, a destra quella deformata.*

Nella presente tesi viene descritto l'intero sviluppo, scritto in MATLAB, di un codice non lineare agli elementi finiti per elementi shell sulla base della serie di pubblicazioni *On a stress resultant geometrically exact shell model* (Simo, J. C. e coautori, 1989-1993) [10]-[16]; in particolare, di queste, sono considerate prevalentemente le prime tre [10], [11] e [12]. Il codice è inoltre testato e confrontato con alcuni risultati presenti in [12], presentando inoltre un'applicazione di eversione.

Con il termine *geometrically exact* (*a geometria esatta*, presente nel titolo), si intende che non sono previste ipotesi restrittive sulle quantità cinematiche presenti oltre all'ipotesi di guscio sottile: spostamenti, rotazioni e deformazioni non sono soggette a troncamenti di nessun ordine. A tal proposito, si può osservare la figura 1, la quale presenta la configurazione iniziale e deformata di una struttura a guscio semisferico, senza fattori di scala (in riferimento all'appicazione del codice su un'applicazione presente al capitolo 5). Il legame costitutivo considerato è il classico elastico, lineare, omogeneo ed isotropo.

Oltre al classico solutore in controllo di forza, vengono proposti un solutore in controllo di spostamento ed un solutore in controllo arc-length. Essi sono utili soprattutto quando il percorso tensionale della struttura non segue una curva monotona crescente, ma presenta fenomeni di *snap-back* e *snap-through*. Osservando la figura 2, la quale descrive il diagramma forza-spostamento di un'applicazione presente al capitolo 5, si osservano le diverse configurazioni raggiunte dalla struttura secondo il tipo di solutore scelto.

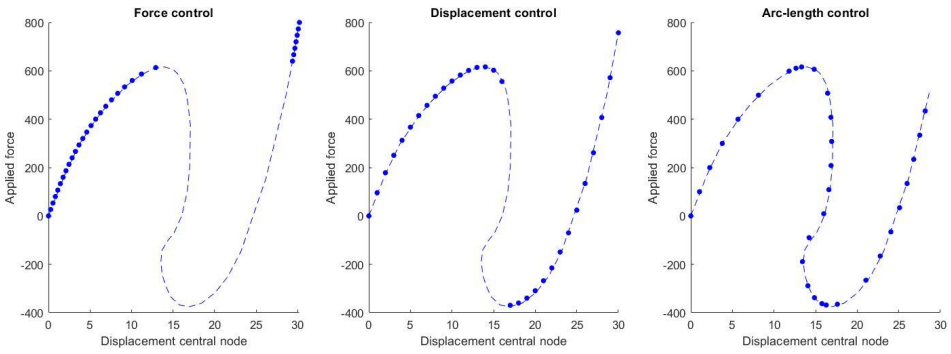


Figura 2: *Con la linea tratteggiata è riportato il percorso tensionale (in riferimento a [12]), con i marker i punti raggiunti dal codice della presente tesi alla convergenza di ogni step.*

Si segnala infine che nelle pubblicazioni [10], [11] e [12] l'analisi è svolta secondo una formulazione variazionale definita *mixed variational formulation*, inoltre, per evitare fenomeni di *shear locking*, le variabili di taglio sono tenute in considerazione secondo l'*assumed strain method*. Nella presente tesi, invece, è implementata una formulazione agli spostamenti definita *displacement formulation*, mentre per evitare fenomeni di *shear locking* si è ridotto il numero di punti di integrazione (considerandone uno solo) rispetto a quelli considerati per le variabili membranali e flessionali (per le quali ne sono previsti quattro).

Data l'analogia fra la geometria differenziale delle superfici e la teoria delle strutture a guscio, si è ritenuto utile esporre alcuni elementi fondamentali dell'algebra e del calcolo tensoriale nel primo capitolo, in modo da presentare in modo più ordinato alcune operazioni matematiche presenti nei capitoli successivi.

Nel secondo capitolo è descritta la teoria dei gusci sottili, sulla base di quanto descritto in [10], [17] e [18].

Nel terzo capitolo è descritta la procedura numerica agli elementi finiti necessaria alla scrittura del codice.

Infine, nel quinto capitolo vengono riportati i risultati derivanti dall'applicazione del codice su alcuni esempi classici in letteratura, confrontati con quanto riportato in [12].

Capitolo 1

Elementi di algebra e calcolo tensoriale in coordinate generali

In questo capitolo vengono descritti alcuni elementi di algebra e calcolo tensoriale al fine di introdurre al meglio alcune operazioni matematiche presenti nei capitoli successivi. Dopo una breve introduzione sul significato di base naturale covariante e controvariante di un sistema di coordinate in $\mathbb{R}^n$, si presentano i concetti fondamentali dell'algebra dei tensori doppi e alcune considerazioni utili sui tensori di ordine superiore. Successivamente, vengono ricavate alcune operazioni differenziali fondamentali per campi scalari, vettoriali e tensoriali definite in coordinate generali. La struttura del capitolo si basa fortemente su quanto descritto in [4] e [5].

1.1 Richiami di algebra vettoriale

1.1.1 Basi covarianti e controvarianti

Sia $\mathbb{R}^n$ uno spazio vettoriale e sia $\mathcal{G} = \{\mathbf{g}_i\}_{i=1,\dots,n}$ una base di $\mathbb{R}^n$ definita *covariante* e indicata con indici bassi. Si dimostra che esiste una base $\mathcal{G}^*$ di $\mathbb{R}^n$ detta base *duale* o *controvariante*, indicata con indici alti, definita dall'insieme di vettori $\{\mathbf{g}^i\}_{i=1,\dots,n}$ tali che:

$$\mathbf{g}_i \cdot \mathbf{g}^j = \delta_i^j \tag{1.1}$$

dove con il simbolo $\cdot$ si è indicato il prodotto scalare, mentre δ_i^j definisce il *delta di Kronecker*, ossia un numero pari a 1 se $i = j$, altrimenti pari a zero.

Sia $\mathbf{v} \in \mathbb{R}^n$ un vettore, esso può essere scritto secondo la base covariante o controvariante, dunque a componenti rispettivamente controvarianti o

covarianti:

$$\mathbf{v} = v^1\mathbf{g}_1 + \ldots + v^n\mathbf{g}_n = \sum_{i=1}^{n} v^i\mathbf{g}_i = v^i\mathbf{g}_i = v_i\mathbf{g}^i \tag{1.2}$$

dove negli ultimi membri si è omesso il simbolo di sommatoria richiamando la convenzione di Einstein (come si farà d'ora in poi, salvo i casi dove si riterrà necessario esplicitarla). Per calcolare la $i-esima$ componente controvariante del vettore $\mathbf{v}$ si può operare moltiplicando scalarmente ambo i membri della (1.2) per il vettore $\mathbf{g}^i$, infatti:

$$\mathbf{g}^i \cdot \mathbf{v} = \mathbf{g}^i \cdot (v^j\mathbf{g}_j) = v^j(\mathbf{g}^i \cdot \mathbf{g}_j) = v^j\delta^i_j = v^i \tag{1.3}$$

dove si è sfruttata la definizione (1.1). Con un ragionamento analogo la generica componente covariante risulta:

$$v_i = \mathbf{g}_i \cdot \mathbf{v} \tag{1.4}$$

Si definisce con $\mathcal{E} = \{\mathbf{e}_i\}_{i=1,\ldots,n}$ la base cartesiana di $\mathbb{R}^n$. Si verifica che la base cartesiana e la rispettiva duale coincidono (più in generale questo fatto è valido per ogni base ortonormale).

1.1.2 Metrica

Sia $\mathbf{v} = v^i\mathbf{g}_i \in \mathbb{R}^n$ un vettore espresso mediante la base covariante. Il quadrato della lunghezza del vettore corrisponde al quadrato della norma:

$$\|\mathbf{v}\|^2 = \mathbf{v} \cdot \mathbf{v} = v^i\mathbf{g}_i \cdot v^j\mathbf{g}_j = v^iv^jg_{ij} \tag{1.5}$$

avendo definito $\mathbf{g}_i \cdot \mathbf{g}_j = g_{ij}$. Il ragionamento esprimendo il vettore secondo la base controvariante è analogo:

$$\|\mathbf{v}\|^2 = v_iv_jg^{ij} \tag{1.6}$$

dove g^{ij} corrisponde a $\mathbf{g}^i \cdot \mathbf{g}^j$. I termini g_{ij} e g^{ij} vengono definiti *metriche*, rispettivamente *metrica covariante* e *metrica controvariante*, del sistema di coordinate.

Si prova ora a scrivere il generico vettore della base covariante $\mathbf{g}_i$ secondo la base duale: $\mathbf{g}_i = x_{ik}\mathbf{g}^k$, dove le componenti x_{ik}, incognite da ricavare, sono le componenti di $\mathbf{g}_i$ secondo la base duale. Si moltiplicano scalarmente ambo i membri per $\mathbf{g}_j$:

$$g_{ij} = \mathbf{g}_i \cdot \mathbf{g}_j = (x_{ik}\mathbf{g}^k) \cdot \mathbf{g}_j = x_{ik}\mathbf{g}^k \cdot \mathbf{g}_j = x_{ik}\delta^k_j = x_{ij}$$

di conseguenza le componenti incognite x_{ij} sono esattamente pari a g_{ij}, quindi:

$$\mathbf{g}_i = g_{ij}\mathbf{g}^j \tag{1.7}$$

Ripetendo il ragionamento col vettore $\mathbf{g}^i$ della base duale, si ottiene:

$$\mathbf{g}^i = g^{ij}\mathbf{g}_j \tag{1.8}$$

Ricordando la legge che lega ogni vettore al proprio duale (1.1), si osserva:

$$\mathbf{g}_i \cdot \mathbf{g}^j = \delta_j^i = (g_{ik}\mathbf{g}^k) \cdot (g^{jl}\mathbf{g}_l) = g_{ik}g^{jl}\mathbf{g}^k \cdot \mathbf{g}_l = g_{ik}g^{jl}\delta_l^k = g_{ik}g^{jk} = g_{ik}g^{kj}$$

dove nell'ultimo passaggio si è fatto uso dell'evidente relazione $g^{ij} = g^{ji}$, dovuta alla simmetria del prodotto scalare. Si ha dunque:

$$g_{ik}g^{kj} = \delta_j^i \tag{1.9}$$

Definendo ora la matrici $[\mathbf{G}]$ e $[\mathbf{G}^*]$, rispettivamente di componenti $[\mathbf{G}]_{ij} = g_{ij}$ e $[\mathbf{G}^*]_{ij} = g^{ij}$, evidentemente simmetriche, l'ultima relazione richiama il prodotto righe per colonne, dunque:

$$[\mathbf{G}]^{-1} = [\mathbf{G}^*] \tag{1.10}$$

e sono una l'inversa dell'altra.

1.2 Algebra dei tensori doppi

Si definisce *prodotto tensoriale* o *diade* la seguente applicazione lineare a partire da due vettori $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$:

$$\mathbf{a} \otimes \mathbf{b}(\mathbf{v}) = (\mathbf{b} \cdot \mathbf{v})\mathbf{a} \qquad \forall \mathbf{v} \in \mathbb{R}^n \tag{1.11}$$

Una combinazione lineare di diadi definisce un *tensore doppio* $\mathbf{T}$ (o *tensore di ordine* 2):

$$\mathbf{T} = T^{ij}\mathbf{g}_i \otimes \mathbf{g}_j \tag{1.12}$$

in questo caso il tensore è espresso secondo due basi covarianti (quindi le componenti hanno due indici controvarianti), tuttavia, è possibile descrivere un tensore secondo due basi controvarianti o miste:

$$\mathbf{T} = T^{ij}\mathbf{g}_i \otimes \mathbf{g}_j = T^i_{\ j}\,\mathbf{g}_i \otimes \mathbf{g}^j = T_i^{\ j}\mathbf{g}^i \otimes \mathbf{g}_j$$

Ogni tensore $\mathbf{T}$ può rappresentare una qualsiasi applicazione lineare $\mathbf{T}\colon \mathbb{R}^n \to \mathbb{R}^n$:

$$\begin{aligned} &\mathbf{T}(\mathbf{v} + \mathbf{w}) = \mathbf{T}(\mathbf{v}) + \mathbf{T}(\mathbf{w}) \qquad \forall \mathbf{v}, \mathbf{w} \in \mathbb{R}^n \\ &\mathbf{T}(\alpha\mathbf{v}) = \alpha\mathbf{T}(\mathbf{v}) \qquad \forall \mathbf{v} \in \mathbb{R}^n,\ \forall \alpha \in \mathbb{R} \end{aligned} \tag{1.13}$$

Lo spazio dei tensori doppi $\mathbf{T}\colon \mathbb{R}^n \to \mathbb{R}^n$ è definito Lin (o Lin_2).

Sia $\mathbf{T} \in \mathrm{Lin}$ un tensore doppio, per calcolare la generica componente T^{ij} si opera nel modo seguente: si applica il tensore al vettore $\mathbf{g}^j$:

$$\mathbf{T}(\mathbf{g}^j) = T^{kl}\mathbf{g}_k \otimes \mathbf{g}_l(\mathbf{g}^j) = T^{kl}(\mathbf{g}_l \cdot \mathbf{g}^j)\mathbf{g}_k = T^{kl}\delta^j_l \mathbf{g}_k = T^{kj}\mathbf{g}_k$$

Ora si moltiplica scalarmente il risultato per il vettore $\mathbf{v}^i$:

$$\mathbf{g}^i \cdot \mathbf{T}(\mathbf{g}^j) = \mathbf{g}^i \cdot T^{kj}\mathbf{g}_k = T^{kj}(\mathbf{g}^i \cdot \mathbf{g}_k) = T^{kj}\delta^i_k = T^{ij}$$

Riassumento il procedimento si è definito:

$$T^{ij} = \mathbf{g}^i \cdot \mathbf{T}(\mathbf{g}^j) \tag{1.14}$$

Nella presente tesi, l'applicazione di un tensore doppio ad un vettore viene indicata anche omettendo le parentesi tonde $\mathbf{Tv} = \mathbf{T}(\mathbf{v})$.

1.2.1 Composizione e prodotto scalare fra tensori doppi

L'operazione di composizione viene definita a partire da due tensori $\mathbf{A}, \mathbf{B} \in \mathrm{Lin}$ ed è l'applicazione tale per cui, per ogni $\mathbf{v} \in \mathbb{R}^n$, risulta:

$$\mathbf{AB}(\mathbf{v}) = \mathbf{A}(\mathbf{B}(\mathbf{v})) \tag{1.15}$$

Ad esempio, siano $\mathbf{A} = A^{ij}\mathbf{a}_i \otimes \mathbf{a}_j \in \mathrm{Lin}$, $\mathbf{B} = B^{kl}\mathbf{b}_k \otimes \mathbf{b}_l \in \mathrm{Lin}$, $\mathbf{v} \in \mathbb{R}^n$. La composizione dei tensori $\mathbf{A}$ e $\mathbf{B}$ applicata al vettore $\mathbf{v}$ risulta:

$$\begin{aligned}\mathbf{AB}(\mathbf{v}) &= \mathbf{A}(B^{kl}\mathbf{b}_k \otimes \mathbf{b}_l(\mathbf{v})) = \mathbf{A}(B^{kl}(\mathbf{b}_l \cdot \mathbf{v})\mathbf{b}_k) = \\ &= A^{ij}\mathbf{a}_i \otimes \mathbf{a}_j(B^{kl}(\mathbf{b}_l \cdot \mathbf{v})\mathbf{b}_k) = A^{ij}B^{kl}(\mathbf{b}_l \cdot \mathbf{v})(\mathbf{a}_j \cdot \mathbf{b}_k)\mathbf{a}_i\end{aligned}$$

la quale, dovendo essere valida per ogni $\mathbf{v} \in \mathbb{R}^n$, può essere riscritta:

$$\mathbf{AB}(\mathbf{v}) = A^{ij}B^{kl}(\mathbf{a}_j \cdot \mathbf{b}_k)\mathbf{a}_i \otimes \mathbf{b}_l(\mathbf{v})$$

Componendo i due tensori, dunque, si è ottenuto un nuovo tensore doppio di componenti $A^{ij}B^{kl}(\mathbf{a}_j \cdot \mathbf{b}_k)\mathbf{b}_l$ nella base $\mathbf{a}_i \otimes \mathbf{b}_l$:

$$\mathbf{AB} = A^{ij}B^{kl}(\mathbf{a}_j \cdot \mathbf{b}_k)\mathbf{a}_i \otimes \mathbf{b}_l \tag{1.16}$$

Si osserva che, generalmente, non vale la proprietà commutativa:

$$\mathbf{AB} \neq \mathbf{BA}$$

Il prodotto scalare fra due tensori doppi viene indicato, nella presente tesi, con il simbolo di *doppia contrazione* degli indici per il quale, a partire da due tensori doppi $\mathbf{A}, \mathbf{B} \in \mathrm{Lin}$, si ottiene lo scalare $\mathbf{A} : \mathbf{B} \in \mathbb{R}$ definito (considerando, ad esempio, senza perdita di generalità, la seguente definizione di $\mathbf{A}$ e $\mathbf{B}$):

$$\mathbf{A} : \mathbf{B} = (A^{ij}\mathbf{a}_i \otimes \mathbf{a}_j) : (B^{kl}\mathbf{b}_k \otimes \mathbf{b}_l) = A^{ij}B^{kl}(\mathbf{a}_i \cdot \mathbf{b}_k)(\mathbf{a}_j \cdot \mathbf{b}_l) \tag{1.17}$$

1.2.2 Particolari operazioni dei tensori doppi

Trasposizione
La proprietà che caratterizza l'operazione di *trasposizione* $(\bullet)^T$: Lin $\to$ Lin è quella di associare, ad ogni tensore doppio $\mathbf{T}$, l'unico tensore doppio *trasposto* $\mathbf{T}^T$ tale che:

$$\mathbf{T}(\mathbf{a}) \cdot \mathbf{b} = \mathbf{T}^T(\mathbf{b}) \cdot \mathbf{a} \qquad \forall \mathbf{a}, \mathbf{b} \in \mathbb{R}^n \tag{1.18}$$

Si verifica facilmente che tale operazione è lineare, inoltre, si verifica che, per $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$, si ha:

$$(\mathbf{u} \otimes \mathbf{v})^T = \mathbf{v} \otimes \mathbf{u} \tag{1.19}$$

infatti, $\forall \mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ù:

$$\mathbf{a} \cdot (\mathbf{v} \otimes \mathbf{u})\mathbf{b} = \mathbf{a} \cdot (\mathbf{u} \cdot \mathbf{b})\mathbf{v} = (\mathbf{a} \cdot \mathbf{v})(\mathbf{u} \cdot \mathbf{b}) = \mathbf{b} \cdot (\mathbf{u} \otimes \mathbf{v})\mathbf{a}$$

Dunque, esprimendo il tensore come combinazione lineare di diadi $\mathbf{T} = T^{ij}\mathbf{g}_i \otimes \mathbf{g}_j$, e per la linearità dell'operazione di trasposizione, si può scrivere:

$$\mathbf{T}^T = T^{ij}\mathbf{g}_j \otimes \mathbf{g}_i \tag{1.20}$$

Traccia
Nel linguaggio tensoriale, l'applicazione lineare *traccia* tr: Lin $\to \mathbb{R}$, a partire da un tensore doppio $\mathbf{T} = T^{ij}\mathbf{g}_i \otimes \mathbf{g}_j$, definisce l'unico numero reale $\mathrm{tr}(\mathbf{T}) \in \mathbb{R}$ tale che:

$$\mathrm{tr}(\mathbf{T}) = \mathrm{tr}(T^{ij}\mathbf{g}_i \otimes \mathbf{g}_j) = T^{ij}\mathbf{g}_i \cdot \mathbf{g}_j \tag{1.21}$$

Tensore identità
Si definisce *tensore indentità* il tensore $\mathbf{1} \in$ Lin tale che, per ogni $\mathbf{v} \in \mathbb{R}^n$, si ha:

$$\mathbf{1}(\mathbf{v}) = \mathbf{v} \tag{1.22}$$

Si può verificare facilmente che:

$$\mathbf{1} = \mathbf{g}_i \otimes \mathbf{g}^i = \mathbf{g}^i \otimes \mathbf{g}_i = g_{ij}\mathbf{g}^i \otimes \mathbf{g}^j = g^{ij}\mathbf{g}_i \otimes \mathbf{g}_j \tag{1.23}$$

infatti, verificando la prima (la seconda si verifica in modo analogo, mentre per verificare la terza e la quarta è sufficiente sostituire a $\mathbf{g}_i$ o $\mathbf{g}^i$ le espressioni (1.7), (1.8)):

$$\mathbf{g}_i \otimes \mathbf{g}^i(\mathbf{v}) = (\mathbf{g}^i \cdot \mathbf{v})\mathbf{g}_i = \left(\mathbf{g}^i \cdot v^k\mathbf{g}_k\right)\mathbf{g}_i = \left(v^k\delta^i_k\right)\mathbf{g}_i = v^i\mathbf{g}_i = \mathbf{v}$$

Tensore inverso
Dato un tensore doppio $\mathbf{A} \in \text{Lin}$, si definisce *tensore inverso* di $\mathbf{A}$ il tensore $\mathbf{A}^{-1} \in \text{Lin}$ tale che:

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{1} \tag{1.24}$$

Si dimostra che, dato $\mathbf{A} \in \text{Lin}$, se $\mathbf{A}^{-1} \in \text{Lin}$ esiste esso è unico. Si dimostra inoltre che:

$$(\mathbf{A}^{-1})^T = (\mathbf{A}^T)^{-1} = \mathbf{A}^{-T} \tag{1.25}$$

Determinante
Si definisce *determinante* di un tensore doppio $\mathbf{T}$, l'applicazione $\det\colon \text{Lin} \to \mathbb{R}$, pari al determinante della matrice delle componenti cartesiane di $\mathbf{T}$, definita $[\mathbf{T}]$:

$$\det\mathbf{T} = \det[\mathbf{T}] \tag{1.26}$$

Siano $\mathbf{A} \in \text{Lin}$, $\mathbf{B} \in \text{Lin}$. Si dimostra che:

$$\det(\mathbf{A}\mathbf{B}) = \det\mathbf{A}\det\mathbf{B} \qquad \det(\mathbf{A}^{-1}) = \det(\mathbf{A})^{-1} \tag{1.27}$$

Un'altra identità facilmente verificabile è la seguente:

$$\det\mathbf{T} = \mathbf{T}\mathbf{e}_1 \cdot \mathbf{T}\mathbf{e}_2 \times \mathbf{T}\mathbf{e}_3 \tag{1.28}$$

dove con il simbolo $\times$ si è indicato il prodotto vettoriale in $\mathbb{R}^3$.

Si osserva ora una proprietà utile: generalizzando la precendente a tre vettori generici della base cartesiana (quindi ripetuti o scambiati di posto) si ottiene:

$$(\mathbf{e}_i \cdot \mathbf{e}_j \times \mathbf{e}_k)\det\mathbf{T} = \mathbf{T}\mathbf{e}_i \cdot \mathbf{T}\mathbf{e}_j \times \mathbf{T}\mathbf{e}_k \tag{1.29}$$

Si considera ora il prodotto misto dell'immagine, mediante il tensore $\mathbf{T}$, di tre vettori aribitrari $\mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathbb{R}^3$, espressi secondo la base cartesiana: $\mathbf{u} = u^i\mathbf{e}_i$, $\mathbf{v} = v^i\mathbf{e}_i$, $\mathbf{w} = w^i\mathbf{e}_i$:

$$\begin{aligned}\mathbf{T}\mathbf{u} \cdot \mathbf{T}\mathbf{v} \times \mathbf{T}\mathbf{w} = \mathbf{T}(u^i\mathbf{e}_i) \cdot \mathbf{T}(v^j\mathbf{e}_j) \times \mathbf{T}(w^k\mathbf{e}_k) = u^i v^j w^k\, \mathbf{T}\mathbf{e}_i \cdot \mathbf{T}\mathbf{e}_j \times \mathbf{T}\mathbf{e}_k = \\ = u^i v^j w^k (\mathbf{e}_i \cdot \mathbf{e}_j \times \mathbf{e}_k)\det\mathbf{T} = (\mathbf{u} \cdot \mathbf{v} \times \mathbf{w})\det\mathbf{T}\end{aligned} \tag{1.30}$$

Dalla proprietà (1.18) di trasposizione si può scrivere:

$$\mathbf{T}\mathbf{u} \cdot \mathbf{T}\mathbf{v} \times \mathbf{T}\mathbf{w} = \mathbf{T}\mathbf{u} \cdot (\mathbf{T}\mathbf{v} \times \mathbf{T}\mathbf{w}) = \mathbf{T}^T(\mathbf{T}\mathbf{v} \times \mathbf{T}\mathbf{w}) \cdot \mathbf{u} \tag{1.31}$$

Confrontando la precedente con la (1.30):

$$\mathbf{u} \cdot \mathbf{T}^T(\mathbf{T}\mathbf{v} \times \mathbf{T}\mathbf{w}) = \mathbf{u} \cdot (\mathbf{v} \times \mathbf{w}) \det \mathbf{T}$$

Dall'arbitrarietà del vettore $\mathbf{u}$:

$$\mathbf{T}^T(\mathbf{T}\mathbf{v} \times \mathbf{T}\mathbf{w}) = (\mathbf{v} \times \mathbf{w}) \det \mathbf{T} \tag{1.32}$$

In particolare, supponendo che il tensore $\mathbf{T}$ sia invertibile, si ottiene infine:

$$\mathbf{T}\mathbf{v} \times \mathbf{T}\mathbf{w} = (\det \mathbf{T})\mathbf{T}^{-T}(\mathbf{v} \times \mathbf{w}) \tag{1.33}$$

valida $\forall \mathbf{v}, \mathbf{w} \in \mathbb{R}^3$.

1.3 Tensori di ordine superiore

Si può estendere il concetto di prodotto tensoriale (1.12) definendo un tensore di ordine p come una combinazione lineare di prodotti tensoriali di ordine p:

$$\mathbf{T} = T^{i_1 \dots i_p}\mathbf{g}_{i_1} \otimes \dots \otimes \mathbf{g}_{i_p} \tag{1.34}$$

Lo spazio dei tensori di ordine p si definisce Lin_p. Generalmente, due tensori si agiscono fra loro mediante applicazioni che, al contrario rispetto ai tensori doppi, devono essere definite opportunamente caso per caso. Ad esempio, un tensore del terzo ordine $\mathbf{T} \in \mathrm{Lin}_3$ può essere visto come un'applicazione lineare $\mathbf{T}\colon \mathbb{R}^n \to \mathrm{Lin}_2$, tale che, essendo $\mathbf{v} \in \mathbb{R}^n$, si ha:

$$\mathbf{T}(\mathbf{v}) = T^{ijk}\mathbf{g}_i \otimes \mathbf{g}_j \otimes \mathbf{g}_k(\mathbf{v}) = T^{ijk}(\mathbf{g}_k \cdot \mathbf{v})\mathbf{g}_i \otimes \mathbf{g}_j \tag{1.35}$$

oppure, $\mathbf{T}$ può essere inteso come un'applicazione $\mathrm{Lin}_2 \to \mathbb{R}^n$, $\mathbf{A} \in \mathrm{Lin}_2$:

$$\mathbf{T}(\mathbf{A}) = T^{ijk}\mathbf{g}_i \otimes \mathbf{g}_j \otimes \mathbf{g}_k(A^{lm}\mathbf{a}_l \otimes \mathbf{a}_m) = T^{ijk}A^{lm}(\mathbf{g}_k \cdot \mathbf{a}_m)\mathbf{g}_i \otimes \mathbf{g}_j \otimes \mathbf{a}_l$$

Per gli scopi del presente elaborato, i tensori tripli, presenti nella definizione di gradiente di un campo tensoriale doppio, verrano presentati come applicazioni lineari $\mathbf{T}\colon \mathbb{R}^n \to \mathrm{Lin}_2$, definiti secondo la (1.35).

Nell'elaborato saranno presenti anche tensori del quarto ordine, i quali verrano introdotti come applicazioni lineari $\mathbf{T}\colon \mathrm{Lin}_2 \to \mathrm{Lin}_2$, definiti secondo la seguente legge:

$$\mathbf{T}(\mathbf{A}) = T^{ijkl}\mathbf{g}_i \otimes \mathbf{g}_j \otimes \mathbf{g}_k \otimes \mathbf{g}_l(A^{ab}\mathbf{a}_a \otimes \mathbf{a}_b) = T^{ijkl}A_{ab}(\mathbf{g}_k \cdot \mathbf{a}_a)(\mathbf{g}_l \cdot \mathbf{a}_b)\mathbf{g}_i \otimes \mathbf{g}_j \tag{1.36}$$

Nella presente tesi, la precedente viene indicata anche con il simbolo di doppia contrazione $\mathbf{T}(\mathbf{A}) = \mathbf{T} : \mathbf{A}$.

Trasposzione di un tensore del quarto ordine

Ai fini del presente elaborato, si definisce l'operazione di trasposizione per un tensore del quarto ordine l'operazione $(\bullet)^T\colon \mathrm{Lin}_4 \to \mathrm{Lin}_4$ per la quale vengono scambiati di posto gli ultimi due vettori del prodotto tensoriale:

$$\mathbf{T}^T = (T^{ijkl}\mathbf{g}_i \otimes \mathbf{g}_j \otimes \mathbf{g}_k \otimes \mathbf{g}_l)^T = T^{ijkl}\mathbf{g}_i \otimes \mathbf{g}_j \otimes \mathbf{g}_l \otimes \mathbf{g}_k = \mathbf{T}^T \tag{1.37}$$

Tensore identità del quarto ordine
Si definisce il tensore identità del quarto ordine il tensore $\mathbb{I} \in \mathrm{Lin}_4$ tale per cui, per ogni $\mathbf{A} \in \mathrm{Lin}_2$:

$$\mathbb{I}(\mathbf{A}) = \mathbf{A} \tag{1.38}$$

Si verifica facilmente che il tensore identità del quarto ordine è dato da:

$$\begin{aligned} \mathbb{I} &= \mathbf{g}_i \otimes \mathbf{g}_j \otimes \mathbf{g}^i \otimes \mathbf{g}^j = \mathbf{g}^i \otimes \mathbf{g}^j \otimes \mathbf{g}_i \otimes \mathbf{g}_j = \\ &= g^{ik} g^{jl} \mathbf{g}_i \otimes \mathbf{g}_j \otimes \mathbf{g}_k \otimes \mathbf{g}_l = g_{ik} g_{jl} \mathbf{g}^i \otimes \mathbf{g}^j \otimes \mathbf{g}^k \otimes \mathbf{g}^l \end{aligned} \tag{1.39}$$

1.4 Gradiente di uno scalare e basi naturali

Base naturale covariante
Sia $\{\xi^i\}_{i=1,\dots,n}$ un sistema di coordinate per lo spazio $\mathbb{R}^n$ e sia $\mathbf{x} \in \mathbb{R}^n$ il vettore posizione. Si definisce *base naturale covariante* l'insieme di vettori $\{\mathbf{g}_i\}_{i=1,\dots,n}$ tali che:

$$\mathbf{g}_i = \frac{\partial \mathbf{x}}{\partial \xi^i} \tag{1.40}$$

Si dimostra che $\{\mathbf{g}_i\}_{i=1,\dots,n}$ costituisce una base di $\mathbb{R}^n$. Da un punto di vista geometrico, $\mathbf{g}_i$ è il vettore tangente alla curva nello spazio al variare della coordinata ξ^i tenendo fisse tutte le altre ξ^j (tali che $j \neq i$).

Ad esempio, indicando il vettore posizione mediante le coordinate cartesiane, si ha:

$$\mathbf{x} = x^i(\xi^1, \dots, \xi^n)\mathbf{e}_i \tag{1.41}$$

Di conseguenza, dato che la base cartesiana è costante nello spazio, dunque indipendente dalle coordinate ξ^i:

$$\mathbf{g}_i = \frac{\partial}{\partial \xi^i}(x^j \mathbf{e}_j) = \frac{\partial x^j}{\partial \xi^i}\mathbf{e}_j \tag{1.42}$$

Per invertire tale relazione si può operare come segue. Si esplicitano i vettori della base cartesiana attraverso i vettori della base naturale covariante:

$$\begin{cases} \mathbf{g}_1 = \dfrac{\partial x^1}{\partial \xi^1}\mathbf{e}_1 + \ldots + \dfrac{\partial x^n}{\partial \xi^1}\mathbf{e}_n \\ \vdots \\ \mathbf{g}_n = \dfrac{\partial x^1}{\partial \xi^n}\mathbf{e}_1 + \ldots + \dfrac{\partial x^n}{\partial \xi^n}\mathbf{e}_n \end{cases}$$

Volendo isolare l'i-esimo vettore della base cartesiana $\mathbf{e}_i$, moltiplicando ogni riga per $\partial\xi^j/\partial x^i$, si ottiene:

$$\begin{cases} \dfrac{\partial\xi^1}{\partial x^i}\mathbf{g}_1 = \dfrac{\partial\xi^1}{\partial x^i}\dfrac{\partial x^1}{\partial\xi^1}\mathbf{e}_1 + \ldots + \dfrac{\partial\xi^1}{\partial x^i}\dfrac{\partial x^n}{\partial\xi^1}\mathbf{e}_n \\ \vdots \\ \dfrac{\partial\xi^n}{\partial x^i}\mathbf{g}_n = \dfrac{\partial\xi^n}{\partial x^i}\dfrac{\partial x^1}{\partial\xi^n}\mathbf{e}_1 + \ldots + \dfrac{\partial\xi^n}{\partial x^i}\dfrac{\partial x^n}{\partial\xi^n}\mathbf{e}_n \end{cases}$$

Sommando le equazioni:

$$\begin{aligned}\Big(\frac{\partial\xi^1}{\partial x^i}\mathbf{g}_1 + \ldots + \frac{\partial\xi^n}{\partial x^i}\mathbf{g}_n\Big) = \Big(\frac{\partial\xi^1}{\partial x^i}\frac{\partial x^1}{\partial\xi^1} + \ldots + \frac{\partial\xi^n}{\partial x^i}\frac{\partial x^1}{\partial\xi^n}\Big)\mathbf{e}_1 + \ldots \\ \ldots + \Big(\frac{\partial\xi^1}{\partial x^i}\frac{\partial x^n}{\partial\xi^1} + \ldots + \frac{\partial\xi^n}{\partial x^i}\frac{\partial x^n}{\partial\xi^n}\Big)\mathbf{e}_n\end{aligned}$$

Riconoscendo la regola della catena all'interno delle parentesi del secondo membro:

$$\Big(\frac{\partial\xi^1}{\partial x^i}\mathbf{g}_1 + \ldots + \frac{\partial\xi^n}{\partial x^i}\mathbf{g}_n\Big) = \frac{\partial x^1}{\partial x^i}\mathbf{e}_1 + \ldots + \frac{\partial x^n}{\partial x^i}\mathbf{e}_n$$

Essendo $\partial x^j/\partial x^i = \delta^{ji}$ (infatti, per definizione, le coordinate sono fra loro indipendenti), l'unico termine diverso da zero al secondo membro è $\partial x^i/\partial x^i\mathbf{e}_i$, il quale si semplifica in $\mathbf{e}_i$. Si ha dunque, infine:

$$\Big(\frac{\partial\xi^1}{\partial x^i}\mathbf{g}_1 + \ldots + \frac{\partial\xi^n}{\partial x^i}\mathbf{g}_n\Big) = \mathbf{e}_i$$

Scrivendo il risultato nella notazione di Einstein, si ottiene:

$$\mathbf{e}_i = \frac{\partial\xi^j}{\partial x^i}\mathbf{g}_j \tag{1.43}$$

Gradiente di uno scalare

Sia $\varphi\colon \mathbb{R}^n \to \mathbb{R}$ un campo scalare. Volendolo linearizzare lungo una direzione fissata dal vettore $\mathbf{h}$, si definsce un'applicazione lineare $\nabla\varphi\colon \mathbb{R}^n \to \mathbb{R}^n$, chiamata *gradiente di* φ, la quale è calcolata nel punto $\mathbf{x}$, agisce sul vettore $\mathbf{h}$ ed è tale per cui:

$$\varphi(\mathbf{x} + t\mathbf{h}) = \varphi(\mathbf{x}) + \nabla\varphi(\mathbf{x})[t\mathbf{h}] + o(\|t\mathbf{h}\|) \tag{1.44}$$

dove $t \in \mathbb{R}$ e il simbolo o (definito *o-piccolo*), applicato ad uno scalare $x \in \mathbb{R}$, è tale per cui:

$$\lim_{x\to 0}\frac{o(x)}{x} = 0 \tag{1.45}$$

Riarrangiando i termini, sfruttando la linearità (per definizione) del gradiente e dividendo ambo i membri per $h \to 0$, ricordando la definizione di o-piccolo (1.45), si ottiene:

$$\nabla\varphi(\mathbf{x})[\mathbf{h}] = \lim_{t\to 0} \frac{1}{t}(\varphi(\mathbf{x} + t\mathbf{h}) - \varphi(\mathbf{x})) \tag{1.46}$$

L'applicazione $\nabla\varphi\colon \mathbb{R}^n \to \mathbb{R}$ mappa linearmente vettori da $\mathbb{R}^n$ in $\mathbb{R}$, può dunque essere considerato come un prodotto scalare fra vettori:

$$\nabla\varphi(\mathbf{x})[\mathbf{h}] = \nabla\varphi(\mathbf{x}) \cdot \mathbf{h} \tag{1.47}$$

Per non appesantire la scrittura, d'ora in avanti, se omesso, verrà sottointesa la dipendenza di $\nabla\varphi$ dal punto $\mathbf{x}$. Esplicitando il gradiente del campo vettoriale secondo la base controvariante:

$$\nabla\varphi = (\nabla\varphi)_i \mathbf{g}^i \tag{1.48}$$

se ne possono calcolare le componenti operando secondo la (1.4):

$$(\nabla\varphi)_i = \nabla\varphi \cdot \mathbf{g}_i \tag{1.49}$$

Osservando la (1.46), nel caso in cui $\mathbf{h}$ corrisponda ad un vettore della base naturale covariante $\mathbf{g}_i$, la generica componente covariante del gradiente corrisponde alla definizione di derivata direzionale lungo la direzione del vettore $\mathbf{g}_i$, di conseguenza, alla derivata parziale secondo la coordinata ξ^i:

$$(\nabla\varphi)_i \cdot \mathbf{g}_i = \nabla\varphi(\mathbf{g}_i) = \lim_{t\to 0} \frac{1}{t}(\varphi(\mathbf{x} + t\mathbf{g}_i) - \varphi(\mathbf{x})) = \frac{\partial\varphi}{\partial\xi^i} \tag{1.50}$$

Si ottiene quindi infine:

$$\nabla\varphi = \frac{\partial\varphi}{\partial\xi^i}\mathbf{g}^i \tag{1.51}$$

Operando nella base cartesiana, invece:

$$\nabla\varphi = \frac{\partial\varphi}{\partial x^i}\mathbf{e}^i \tag{1.52}$$

Base naturale controvariante

Osservando la (1.46) e calcolando il gradiente della generica coordinata ξ^i lungo la direzione data dal vettore $\mathbf{g}_i$, si ottiene δ^i_j:

$$\nabla\xi^i \cdot \mathbf{g}_j = \lim_{t\to 0} \frac{1}{t}(\xi^i(\mathbf{x} + t\mathbf{g}_j) - \xi^i(\mathbf{x})) = \frac{\partial\xi^i}{\partial\xi^j} = \delta^i_j \tag{1.53}$$

dato che $\{\xi^i\}_{i=1,\dots,n}$ è un sistema di coordinate, le quali sono dunque fra loro indipendenti. Il prodotto dato dalla (1.53) rispetta la condizione (1.1), i vettori della base duale (o naturale controvariante) sono dunque definiti:

$$\mathbf{g}^i = \nabla \xi^i \tag{1.54}$$

i quali risultano, espressi secondo una base cartesiana:

$$\mathbf{g}^i = \nabla \xi^i = \frac{\partial \xi^i}{\partial x^j} \mathbf{e}^j \tag{1.55}$$

Si nota infatti che:

$$\mathbf{g}_i \cdot \mathbf{g}^j = \frac{\partial x^k}{\partial \xi^i} \mathbf{e_k} \cdot \frac{\partial \xi^j}{\partial x^l} \mathbf{e}^l = \frac{\partial x^k}{\partial \xi^i} \frac{\partial \xi^j}{\partial x^l} (\mathbf{e}_k \cdot \mathbf{e}^j) = \frac{\partial x^k}{\partial \xi^i} \frac{\partial \xi^j}{\partial x^l} \delta^k_l = \frac{\partial x^k}{\partial \xi^i} \frac{\partial \xi^j}{\partial x^k} = \frac{\partial \xi^j}{\partial \xi^i} = \\ = \delta^j_i \tag{1.56}$$

corrispondente alla definizione (1.1).

Con un procedimento analogo a quanto scritto per la base naturale covariante, ma scritto in maniera compatta, si può invertire la relazione (1.55) moltiplicando ambo i membri per $\partial x^k / \partial \xi^i$ ed estendendo la sommatoria sull'indice i:

$$\frac{\partial x^k}{\partial \xi^i} \mathbf{g}^i = \frac{\partial x^k}{\partial \xi^i} \frac{\partial \xi^i}{\partial x^j} \mathbf{e}^j = \frac{\partial x^k}{\partial \xi^j} \mathbf{e}^j = \delta^{kj} \mathbf{e}^i = \mathbf{e}^j$$

Per cui, riordinando gli indici, si ottiene:

$$\mathbf{e}^i = \frac{\partial x^k}{\partial \xi^j} \mathbf{g}^j \tag{1.57}$$

1.5 Operazioni differenziali di un campo vettoriale

Gradiente di un campo vettoriale

Sia ora $\boldsymbol{\phi} \colon \mathbb{R}^n \to \mathbb{R}^n$ un campo vettoriale. Analogamente alla sezione precedente, si vuole introdurre un'applicazione lineare $\nabla \boldsymbol{\phi} \colon \mathbb{R}^n \to \mathbb{R}^n$ chiamata *gradiente di* $\boldsymbol{\phi}$, tale che:

$$\boldsymbol{\phi}(\mathbf{x} + t\mathbf{h}) = \boldsymbol{\phi}(\mathbf{x}) + \nabla \boldsymbol{\phi}(\mathbf{x})[t\mathbf{h}] + o(\|t\mathbf{h}\|) \tag{1.58}$$

Per cui, operando come nella sezione precedente, si ha:

$$\nabla \boldsymbol{\phi}(\mathbf{x})[\mathbf{h}] = \lim_{t \to 0} \frac{1}{t} (\boldsymbol{\phi}(\mathbf{x} + t\mathbf{h}) - \boldsymbol{\phi}(\mathbf{x})) \tag{1.59}$$

L'applicazione $\nabla\boldsymbol{\phi}\colon \mathbb{R}^n \to \mathbb{R}^n$ mappa linearmente vettori da $\mathbb{R}^n$ in $\mathbb{R}^n$ dunque, di fatto, può essere associata ad un tensore doppio.

Osservando la (1.59), nel caso $\mathbf{h}$ corrisponda ad un vettore della base naturale covariante $\mathbf{g}_i$, la definizione di gradiente corrisponde alla definizione di derivata parziale:

$$\nabla\boldsymbol{\phi}(\mathbf{x})[\mathbf{g}_i] = \lim_{t\to 0}\frac{1}{t}(\boldsymbol{\phi}(\mathbf{x}+t\mathbf{g}_i) - \boldsymbol{\phi}(\mathbf{x})) = \frac{\partial\boldsymbol{\phi}}{\partial\xi^i} \tag{1.60}$$

Di conseguenza, si verifica che il gradiente di un campo vettoriale può essere scritto:

$$\nabla\boldsymbol{\phi} = \frac{\partial\boldsymbol{\phi}}{\partial\xi^i}\otimes\mathbf{g}^i \tag{1.61}$$

Infatti, esso rispetta la precedente:

$$\nabla\boldsymbol{\phi}(\mathbf{g}_j) = \frac{\partial\boldsymbol{\phi}}{\partial\xi^i}\otimes\mathbf{g}^i(\mathbf{g}_j) = \frac{\partial\boldsymbol{\phi}}{\partial\xi^i}(\mathbf{g}^i\cdot\mathbf{g}_j) = \frac{\partial\boldsymbol{\phi}}{\partial\xi^i}\delta^i_j = \frac{\partial\boldsymbol{\phi}}{\partial\xi^j}$$

In coordinate cartesiane:

$$\nabla\boldsymbol{\phi} = \frac{\partial\boldsymbol{\phi}}{\partial x^j}\otimes\mathbf{e}^j = \frac{\partial(\phi\mathbf{e}_i)}{\partial x^j}\otimes\mathbf{e}^j = \frac{\partial\phi^i}{\partial x^j}\mathbf{e}_i\otimes\mathbf{e}^j \tag{1.62}$$

Gradiente del prodotto fra un campo scalare e un campo vettoriale

Il gradiente del prodotto fra un campo scalare $\varphi\colon \mathbb{R}^n \to \mathbb{R}$ e un campo vettoriale $\boldsymbol{\phi}\colon \mathbb{R}^n \to \mathbb{R}^n$ si calcola mediante la regola di derivazione del prodotto:

$$\nabla(\varphi\boldsymbol{\phi}) = \frac{\partial(\varphi\boldsymbol{\phi})}{\partial\xi^i}\otimes\mathbf{g}^i = \left(\frac{\partial\varphi}{\partial\xi^i}\boldsymbol{\phi} + \varphi\frac{\partial\boldsymbol{\phi}}{\partial\xi^i}\right)\otimes\mathbf{g}^i = \frac{\partial\varphi}{\partial\xi^i}\boldsymbol{\phi}\otimes\mathbf{g}^i + \varphi\frac{\partial\boldsymbol{\phi}}{\partial\xi^i}\otimes\mathbf{g}^i$$

riscrivendo il primo fattore del primo addendo come segue:

$$\frac{\partial\varphi}{\partial\xi^i}\boldsymbol{\phi}\otimes\mathbf{g}^i = \boldsymbol{\phi}\otimes\left(\frac{\partial\varphi}{\partial\xi^i}\mathbf{g}^i\right) = \boldsymbol{\phi}\otimes\nabla\varphi$$

e osservando che il secondo fattore del secondo addendo corrisponde al gradiente di $\boldsymbol{\phi}$, si ottiene infine:

$$\nabla(\varphi\boldsymbol{\phi}) = \boldsymbol{\phi}\otimes\nabla\varphi + \varphi\nabla\boldsymbol{\phi} \tag{1.63}$$

Divergenza di un campo vettoriale
Si definisce *divergenza* di un campo vettoriale $\boldsymbol{\phi}$ l'operatore $\mathrm{div}\colon \mathbb{R}^n \to \mathbb{R}$ per cui:

$$\mathrm{div}\boldsymbol{\phi}(\mathbf{x}) = \mathrm{tr}(\nabla\boldsymbol{\phi})(\mathbf{x}) \tag{1.64}$$

Applicando la definizione di traccia (1.21) alla (1.61):

$$\mathrm{div}\boldsymbol{\phi} = \frac{\partial\boldsymbol{\phi}}{\partial\xi^i}\cdot\mathbf{g}^i \tag{1.65}$$

Si ricorda che, in base al noto teorema della divergenza per campi vettoriali, dato un continuo $\Omega \subseteq \mathbb{R}^n$, sotto sufficienti ipotesi di regolarità, si ha:

$$\int_\Omega \mathrm{div}\boldsymbol{\phi}(\mathbf{x})\, d\Omega = \int_{\partial\Omega} \boldsymbol{\phi}(\mathbf{x})\cdot\mathbf{n}\, d\partial\Omega \tag{1.66}$$

dove $\mathbf{n}$ corrisponde al versore normale uscente alla (iper)superficie $\partial\Omega$.

1.6 Operazioni differenziali di un campo tensoriale

Gradiente di un campo tensoriale
Sia ora $\boldsymbol{\Phi}\colon \mathbb{R}^n \to \mathrm{Lin}$ un campo tensoriale del secondo ordine. Il gradiente di $\boldsymbol{\Phi}$ è l'applicazione lineare $\nabla\boldsymbol{\Phi}\colon \mathbb{R}^n \to \mathrm{Lin}$ tale che:

$$\boldsymbol{\Phi}(\mathbf{x}+t\mathbf{h}) = \boldsymbol{\Phi}(\mathbf{x}) + \nabla\boldsymbol{\Phi}(\mathbf{x})[t\mathbf{h}] + o(\|t\mathbf{h}\|) \tag{1.67}$$

Riarrangiando i termini, si ottiene:

$$\nabla\boldsymbol{\Phi}(\mathbf{x})[\mathbf{h}] = \lim_{t\to 0}\frac{1}{t}(\Phi(\mathbf{x}+t\mathbf{h}) - \boldsymbol{\Phi}(\mathbf{x})) \tag{1.68}$$

L'applicazione $\nabla\boldsymbol{\Phi}\colon \mathbb{R}^n \to \mathrm{Lin}$ mappa linearmente vettori da $\mathbb{R}^n$ in Lin dunque, di fatto, può essere associata ad un tensore triplo che agisce secondo la (1.35).

Si osserva che la derivata parziale di un campo tensoriale si calcola applicando la regola di definizione del prodotto, dato che il prodotto vettoriale gode della proprietà associativa, un'esempio sul campo tensoriale indicato in base controvariante $\boldsymbol{\Phi} = \Phi_{ij}\mathbf{g}^i \otimes \mathbf{g}^j$ risulta:

$$\frac{\partial\boldsymbol{\Phi}}{\partial\xi^k} = \frac{\partial}{\partial\xi^k}(\Phi_{ij}\mathbf{g}^i\otimes\mathbf{g}^j) = \frac{\partial\Phi_{ij}}{\partial\xi^k}\mathbf{g}^i\otimes\mathbf{g}^j + \Phi_{ij}\frac{\partial\mathbf{g}^i}{\partial\xi^k}\otimes\mathbf{g}^j + \Phi_{ij}\mathbf{g}^i\otimes\frac{\partial\mathbf{g}^j}{\partial\xi^k} \tag{1.69}$$

Analogamente a quanto visto nelle sezioni precedenti, applicando al tensore $\nabla\mathbf{\Phi}$ il vettore $\mathbf{g}_i$, si ottiene la definizione di derivata parziale del tensore lungo la coordinata ξ^i:

$$\nabla\mathbf{\Phi}(\mathbf{g}_i) = \frac{\partial\mathbf{\Phi}}{\partial\xi^i} \tag{1.70}$$

Dunque, si verifica che il gradiente di un tensore risulta:

$$\nabla\mathbf{\Phi} = \frac{\partial\mathbf{\Phi}}{\partial\xi^i} \otimes \mathbf{g}^i \tag{1.71}$$

Infatti, esso rispetta la precedente:

$$\nabla\mathbf{\Phi}(\mathbf{g}_j) = \frac{\partial\mathbf{\Phi}}{\partial\xi^i} \otimes \mathbf{g}^i(\mathbf{g}_j) = \frac{\partial\mathbf{\Phi}}{\partial\xi^i}(\mathbf{g}^i \cdot \mathbf{g}_j) = \frac{\partial\mathbf{\Phi}}{\partial\xi^i}\delta^i_j = \frac{\partial\mathbf{\Phi}}{\partial\xi^j}$$

Divergenza di un campo tensoriale
Si definisce divergenza di un tensore $\mathbf{\Phi}$ l'operatore $\mathrm{div}\mathbf{\Phi}\colon \mathrm{Lin} \to \mathbb{R}^n$ tale che, per ogni vettore costante $\mathbf{c} \in \mathbb{R}^n$, si ha:

$$c \cdot (\mathrm{div}\mathbf{\Phi}) = \mathrm{div}(\mathbf{\Phi}^T\mathbf{c}) \tag{1.72}$$

$\mathbf{\Phi}^T\mathbf{c}$ rappresenta un campo vettoriale del quale si può calcolarne la divergenza secondo la definizione (1.64), dunque:

$$\mathrm{div}(\mathbf{\Phi}^T\mathbf{c}) = \mathrm{div}(\Phi^{ij}\mathbf{g}_j \otimes \mathbf{g}_i(\mathbf{c})) = \mathrm{div}(\Phi^{ij}(\mathbf{g}_i \cdot \mathbf{c})\mathbf{g}_j) = \mathrm{tr}(\nabla(\Phi^{ij}(\mathbf{g}_i \cdot \mathbf{c})\mathbf{g}_j))$$

svolgendo i calcoli e ricordando il gradiente del prodotto fra un campo scalare e un campo vettoriale (1.63), si ottiene:

$$\begin{aligned}\mathrm{tr}(\nabla(\Phi^{ij}(\mathbf{g}_i \cdot \mathbf{c})\mathbf{g}_j)) &= \mathrm{tr}(\mathbf{g}_j \otimes \nabla(\Phi^{ij}(\mathbf{g}_i \cdot \mathbf{c})) + \Phi^{ij}(\mathbf{g}_i \cdot \mathbf{c})\nabla\mathbf{g}_j) = \\ &= \mathrm{tr}\Big(\mathbf{g}_j \otimes \frac{\partial}{\partial\xi^k}(\Phi^{ij}(\mathbf{g}_i \cdot \mathbf{c}))\mathbf{g}^k + \Phi^{ij}(\mathbf{g}_i \cdot \mathbf{c})\frac{\partial\mathbf{g}_j}{\partial\xi^k} \otimes \mathbf{g}^k\Big) = \\ &= (\mathbf{g}_j \cdot \mathbf{g}^k)\frac{\partial}{\partial\xi^k}(\Phi^{ij}(\mathbf{g}_i \cdot \mathbf{c})) + \Phi^{ij}(\mathbf{g}_i \cdot \mathbf{c})\Big(\frac{\partial\mathbf{g}_j}{\partial\xi^k} \cdot \mathbf{g}^k\Big)\end{aligned}$$

La costante $\mathbf{c}$ può essere estratta dal segno di derivazione e raccolta a fattor comune, risultando:

$$\mathrm{div}(\mathbf{\Phi}^T\mathbf{c}) = \Big[(\mathbf{g}_j \cdot \mathbf{g}^k)\frac{\partial}{\partial\xi^k}(\Phi^{ij}\mathbf{g}_i) + \Phi^{ij}\Big(\frac{\partial\mathbf{g}_j}{\partial\xi^k} \cdot \mathbf{g}^k\Big)\mathbf{g}_i\Big] \cdot \mathbf{c}$$

Si osserva che il termine all'interno della parentesi quadra corrisponde a:

$$(\mathbf{g}_j \cdot \mathbf{g}^k)\frac{\partial}{\partial\xi^k}(\Phi^{ij}\mathbf{g}_i) + \Phi^{ij}\Big(\frac{\partial\mathbf{g}_j}{\partial\xi^k} \cdot \mathbf{g}^k\Big)\mathbf{g}_i = \Big(\frac{\partial}{\partial\xi^k}(\Phi^{ij}\mathbf{g}_i) \otimes \mathbf{g}_j + \Phi^{ij}\mathbf{g}_i \otimes \frac{\partial\mathbf{g}_j}{\partial\xi^k}\Big)(\mathbf{g}^k)$$

il quale, dalla linearità del prodotto tensoriale:

$$\frac{\partial}{\partial \xi^k}(\Phi^{ij}\mathbf{g}_i) \otimes \mathbf{g}_j + \Phi^{ij}\mathbf{g}_i \otimes \frac{\partial \mathbf{g}_j}{\partial \xi^k} = \frac{\partial}{\partial \xi^k}(\Phi^{ij}\mathbf{g}_i \otimes \mathbf{g}_j)$$

ottenendo, dunque:

$$\mathrm{div}(\mathbf{\Phi}^T\mathbf{c}) = \frac{\partial}{\partial \xi^k}(\Phi^{ij}\mathbf{g}_i \otimes \mathbf{g}_j)(\mathbf{g}^k) \cdot \mathbf{c} = \frac{\partial \mathbf{\Phi}}{\partial \xi^k}\mathbf{g}^k \cdot \mathbf{c}$$

La divergenza di un campo tensoriale si può dunque scrivere:

$$\mathrm{div}\mathbf{\Phi} = \frac{\partial \mathbf{\Phi}}{\partial \xi^i}\mathbf{g}^i \tag{1.73}$$

Si dimostra che l'analoga alla (1.66) per campi tensoriali risulta:

$$\int_\Omega \mathrm{div}\mathbf{\Phi}(\mathbf{x})\, d\Omega = \int_{\partial\Omega} \mathbf{\Phi}(\mathbf{x})\mathbf{n}\, d\partial\Omega \tag{1.74}$$

dove i simboli hanno il significato introdotto precedentemente.

Capitolo 2

Teoria dei gusci sottili

In questo capitolo, in seguito ad una sezione che richiama alcuni concetti fondamentali della meccanica del continuo seguendo quanto descritto in [2], vengono definite le ipotesi cinematiche sulla quale si basa la teoria dei gusci (shell) sottili, introducendo le variabili generalizzate di tensione. Successivamente, dopo aver ricavato le equazioni di equilibrio, viene introdotta e linearizzata la formulazione variazionale, necessaria per i successivi sviluppi numerici.

2.1 Richiami di meccanica del continuo

Analisi della deformazione
Si considera una regione dello spazio $\mathcal{C}_0 \subseteq \mathbb{R}^3$, definita *configurazione iniziale*, ipotizzata sufficientemente regolare. I punti appartenenti a $\mathcal{C}_0$ vengono definiti *punti materiali* ed indicati con la lettera maiuscola $\mathbf{X} = X^i \mathbf{E}_i$, avendone esplicitato le componenti cartesiane. Si definisce deformazione $\boldsymbol{\chi}$ una funzione, sufficientemente regolare, che ad ogni punto nella configurazione iniziale associa un punto $\mathbf{x} = x^i \mathbf{e}_i \in \mathbb{R}^3$:

$$\mathbf{x} = \boldsymbol{\chi}(\mathbf{X}) \tag{2.1}$$

L'immagine di $\mathcal{C}_0$ mediante la deformazione $\boldsymbol{\chi}$ è definita *configurazione deformata* ed è indicata con $\mathcal{C}$, si fa riferimento all'immagine 2.1. Per evitare strappi e compenetrazioni di materia, si suppone che la funzione $\boldsymbol{\chi}$ sia bigettiva, di conseguenza invertibile.

Il gradiente della funzione $\boldsymbol{\chi}$, è definito *tensore gradiente di deformazione* ed è indicato con $\mathbf{F}$:

$$\mathbf{F} = \text{GRAD}\boldsymbol{\chi} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}} = \frac{\partial x^i}{\partial X^j} \mathbf{e}_i \otimes \mathbf{E}^j \tag{2.2}$$

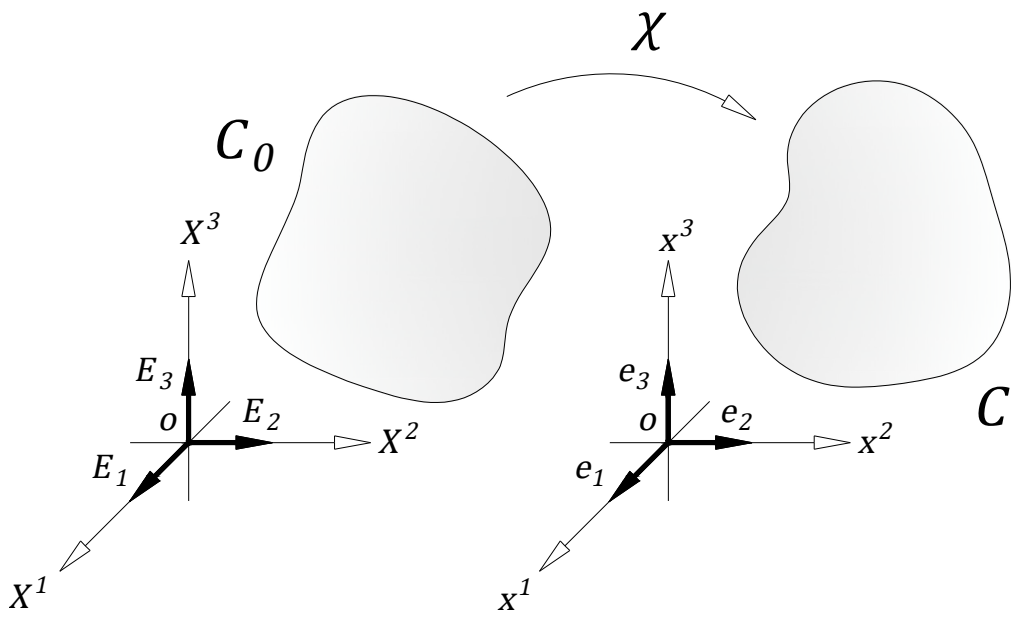


Figura 2.1

avendo indicato con GRAD il gradiente svolto rispetto alla configurazione ed avendo esplicitato le componenti cartesiane nell'ultimo passaggio (si è inoltre introdotta la notazione $\text{GRAD}(\bullet) = \frac{\partial(\bullet)}{\partial \mathbf{X}}$). Ricordando la definizione di gradiente di un campo vettoriale (1.58), si osserva:

$$\mathbf{x}(\mathbf{X} + t\mathbf{H}) = \mathbf{x}(\mathbf{X}) + \mathbf{F}(\mathbf{x})[t\mathbf{H}] + o(\|t\mathbf{H}\|)$$

Facendo tendere t a zero e indicando $d\mathbf{x} = \mathbf{x}(\mathbf{X} + t\mathbf{H}) - \mathbf{x}(\mathbf{X})$ e $d\mathbf{X} = t\mathbf{H}$, si può riarrangiare la precedente come:

$$d\mathbf{x} = \mathbf{F} d\mathbf{X} \tag{2.3}$$

Si osserva che il significato del tensore $\mathbf{F}$, dunque, è quello di mappare fibre di lunghezza infinitesima $d\mathbf{X}$ nella configurazione iniziale, in fibre di lunghezza infinitesima $d\mathbf{x}$ nella configurazione deformata.

Dalla condizione per la quale si richiede che un volume di materia rimanga positivo a deformazione avvenuta, un'ulteriore ipotesi è che il determinante di $\mathbf{F}$, indicato con J e definito *Jacobiano della deformazione*, sia sempre strettamente positivo, il che, secondo l'algebra lineare, garantisce l'invertibilità di $\mathbf{F}$.

Considerando una generica fibra infinitesima nella configurazione iniziale $d\mathbf{X}$ e le propria immagine nella configurazione deformata $d\mathbf{x}$, indicando il

quadrato delle rispettive lunghezze con $dL^2 = d\mathbf{X} \cdot d\mathbf{X}$ e $dl^2 = d\mathbf{x} \cdot d\mathbf{x}$, si verifica che la differenza fra queste due quantità risulta, ricordando la (2.3):

$$dl^2 - dL^2 = d\mathbf{x} \cdot d\mathbf{x} - d\mathbf{X} \cdot d\mathbf{X} = 2d\mathbf{X} \cdot \mathbf{E}d\mathbf{X}$$

avendo introdotto il tensore di deformazione materiale:

$$\mathbf{E} = \frac{1}{2}(\mathbf{F}^T\mathbf{F} - \mathbf{1}) \tag{2.4}$$

Introducendo lo *spostamento* di un punto fra la configurazione inziale e quella deformata $\mathbf{u} = \mathbf{x} - \mathbf{X}$, si definisce il *tensore di deformazione spaziale*:

$$\boldsymbol{\varepsilon} = \frac{1}{2}(\text{grad}\mathbf{u} + (\text{grad}\mathbf{u})^T) \tag{2.5}$$

avendo indicato con grad il gradiente calcolato rispetto ai punti della configurazione deformata. I tensori $\mathbf{E}$ e $\boldsymbol{\varepsilon}$ introducono delle specifiche *misure di deformazione*.

Si considera una superficie $\mathcal{S}_0$ nella configurazione deformata parametrizzata da una coppia di variabili (ξ^1, ξ^2), $\boldsymbol{\varphi}_0 \colon (\xi^1, \xi^2) \to \mathcal{S}_0$. Dall'analisi matematica, il vettore $\mathbf{n}_0$ normale alla superficie $\mathcal{S}_0$ moltiplicato l'elemento d'area infinitesimo dA_0, si può scrivere:

$$\mathbf{n}_0\, dA_0 = \frac{\partial \boldsymbol{\varphi}_0}{\partial \xi^1} \times \frac{\partial \boldsymbol{\varphi}_0}{\partial \xi^1}\, d\xi^1 d\xi^2 \tag{2.6}$$

Si considera l'immagine della superficie $\boldsymbol{\varphi}_0$ secondo la deformazione $\boldsymbol{\chi}$, $\boldsymbol{\varphi} = \boldsymbol{\chi}(\boldsymbol{\varphi}_0)$, sempre parametrizzata dalla coppia di variabili ξ^1, ξ^2, $\boldsymbol{\varphi} = \boldsymbol{\varphi}(\xi^1, \xi^2)$. A seguito di alcuni passaggi, dalla regola della catena per cui $\frac{\partial \mathbf{x}}{\partial \xi^i} = \mathbf{F}\frac{\partial \mathbf{X}}{\partial \xi^i}$ e sfruttando la relazione (1.33), si può definire la seguente legge, definita *formula di Nanson*:

$$\mathbf{n}\, dA = \frac{\partial \boldsymbol{\varphi}}{\partial \xi^1} \times \frac{\partial \boldsymbol{\varphi}}{\partial \xi^1}\, d\xi^1 d\xi^2 = J\mathbf{F}^{-T}\mathbf{n}_0\, dA_0 \tag{2.7}$$

Siano $\boldsymbol{\phi} \colon \mathbb{R}^3 \to \mathbb{R}^3$, $\boldsymbol{\Phi} \colon \mathbb{R}^3 \to \text{Lin}$, rispettivamente, un campo vettoriale e un campo tensoriale del secondo ordine. Dalla (2.7), si ottengono due importanti relazioni definite *trasformazioni di Piola*, che definiscono delle corrispondenze fra gli integrali di superficie nelle due configurazioni:

$$\begin{aligned} \int_{\mathcal{S}} \boldsymbol{\phi} \cdot \mathbf{n}\, dA &= \int_{\mathcal{S}_0} J\mathbf{F}^{-1}\boldsymbol{\phi} \cdot \mathbf{n}_0\, dA_0 \\ \int_{\mathcal{S}} \boldsymbol{\Phi}\mathbf{n}\, dA &= \int_{\mathcal{S}_0} J\mathbf{F}^{-T}\boldsymbol{\Phi}\mathbf{n}_0\, dA_0 \end{aligned} \tag{2.8}$$

Analisi della tensione ed equazioni indefinite di equilibrio
Nella meccanica del continuo si suppone che le interazioni fra il corpo e il mondo esterno vengano esercitate attraverso due tipologie di forze: le *forze di volume*, ad esempio la forza gravitazionale, agenti sul volume del solido attraverso una funzione, definita *densità delle forze di volume* ed indicata con $\mathbf{b}\colon \mathcal{C} \to \mathbb{R}^3$, e le *forze di superficie*, ad esempio le forze dovute al contatto fra il corpo e dei carichi fisici agenti sulla frontiera del continuo $\partial\mathcal{C}$ secondo la funzione *densità delle forze di superficie* $\mathbf{f}\colon \partial\mathcal{C} \to \mathbb{R}^3$. Secondo la meccanica del continuo, la funzione $\mathbf{f}$ è definita dall'azione di un tensore del secondo ordine $\boldsymbol{\sigma}$, che prende il nome di *tensore di Cauchy*, sulla normale esterna alla superficie $\mathbf{n}$:

$$\mathbf{f} = \boldsymbol{\sigma}\mathbf{n} \tag{2.9}$$

Le equazioni cardinali delle meccanica, in ambito statico, definiscono l'equilibrio del solido sotto l'azione di una distribuzione di forze di volume e di superficie. L'equilibrio alla traslazione e alla rotazione, rispettivamente, sono definite:

$$\begin{aligned} &\int_{\mathcal{C}} \mathbf{b}\, dV + \int_{\partial\mathcal{C}} \mathbf{f}\, dA = 0 \\ &\int_{\mathcal{C}} (\mathbf{x} - \mathbf{x}_0) \times \mathbf{b}\, dV + \int_{\partial\mathcal{C}} (\mathbf{x} - \mathbf{x}_0) \times \mathbf{f}\, dA = 0 \end{aligned} \tag{2.10}$$

avendo indicato con $\mathbf{x}_0$ un generico punto fisso nello spazio usato come polo per il calcolo dei momenti. Si dimostra che le precedenti sono soddisfatte per ogni parte del corpo se e solo se:

$$\begin{aligned} &\mathrm{div}\boldsymbol{\sigma} + \mathbf{b} = \mathbf{0} \\ &\boldsymbol{\sigma} = \boldsymbol{\sigma}^T \end{aligned} \tag{2.11}$$

Le precedenti sono definite, rispettivamente, *prima* e *seconda equazione indefinita di equilibrio in forma spaziale*. Si dimostra che esse sono equivalenti alle seguenti *prima* e *seconda equazione indefinita di equilibrio in forma materiale*:

$$\begin{aligned} &\mathrm{DIV}\mathbf{P} + J\mathbf{b} = \mathbf{0} \\ &\mathbf{P}\mathbf{F}^T = \mathbf{F}\mathbf{P}^T \end{aligned} \tag{2.12}$$

Avendo introdotto il *primo tensore di Piola-Kirchhoff*:

$$\mathbf{P} = J\boldsymbol{\sigma}\mathbf{F}^{-T} \tag{2.13}$$

Si introduce inoltre il *secondo tensore di Piola-Kirchhoff*, simmetrico:

$$\mathbf{S} = \mathbf{F}^{-1}\mathbf{P} = J\mathbf{F}^{-1}\boldsymbol{\sigma}\mathbf{F}^{-T} \tag{2.14}$$

I tensori $\boldsymbol{\sigma}$, $\mathbf{P}$ e $\mathbf{S}$ definiscono specifiche *misure di tensione*.

Forma variazionale delle equazioni di equilibrio

Si considera un insieme di funzioni $\mathbb{X}$ con dominio in $\mathbb{R}^n$ e, generalmente, a valori scalari, vettoriali o tensoriali. Si considera inoltre una generica funzione F, con dominio in $\mathbb{X}$ ed anch'essa, generalmente, a valori scalari, vettoriali o tensoriali. Si definisce *variazione* di F calcolata in $\mathbf{u}(\mathbf{X})$ e lungo la direzione $\mathbf{v}(\mathbf{X})$, con $\mathbf{u}(\mathbf{X}), \mathbf{v}(\mathbf{X}) \in \mathbb{X}$, la seguente:

$$D\mathsf{F}(\mathbf{u})[\mathbf{v}] = \frac{d}{d\epsilon}\Big(\mathsf{F}(\mathbf{u}+\epsilon\mathbf{v})\Big)\Big|_{\epsilon=0} \tag{2.15}$$

Si verifica che l'operatore $D(\bullet)$ segue le regole di derivazione. Per indicare una qualsiasi direzione $\mathbf{v}$ a partire da $\mathbf{u}$, seguendo la letteratura, la direzione $\mathbf{v}$ può essere scritta come $\delta\mathbf{u}$. In questo caso la variazione prende il nome di *variazione virtuale* e viene indicata con $\delta\mathsf{F}$:

$$\delta\mathsf{F} = \frac{d}{d\epsilon}\Big(\mathsf{F}(\mathbf{u}+\epsilon\delta\mathbf{u})\Big)\Big|_{\epsilon=0} \tag{2.16}$$

Si dimostra che le equazioni di equilibrio sono garantite se la seguente variazione virtuale del funzionale Π, funzione degli spostamenti $\mathbf{u}$, è nulla:

$$\delta\Pi = \delta\mathcal{L}_{\text{int}} - \delta\mathcal{L}_{\text{ext}} = \int_{\mathcal{C}} \boldsymbol{\sigma} : \delta\boldsymbol{\varepsilon}\, dV - \Big(\int_{\mathcal{C}} \mathbf{b}\cdot\delta\mathbf{u}\, dV + \int_{\mathcal{S}} \mathbf{f}\cdot\delta\mathbf{u}\, dA\Big) = 0 \tag{2.17}$$

dove il primo addendo definisce la variazione virtuale del *lavoro interno*, mentre il secondo la variazione virtuale del *lavoro esterno*.

La variazione del tensore di deformazione spaziale risulta:

$$\delta\boldsymbol{\varepsilon} = \frac{d}{dt}\Big\{\frac{1}{2}\Big[\Big(\frac{\partial}{\partial\mathbf{X}}(\mathbf{u}+t\delta\mathbf{u})\Big)^T \frac{\partial}{\partial\mathbf{X}}(\mathbf{u}+t\delta\mathbf{u})\Big]\Big\}\Big|_{t=0} \tag{2.18}$$

Svolgendo i calcoli:

$$\begin{aligned}\delta\boldsymbol{\varepsilon} &= \frac{1}{2}\frac{d}{dt}\Big[\Big(\frac{\partial\mathbf{u}}{\partial\mathbf{X}} + t\frac{\partial\delta\mathbf{u}}{\partial\mathbf{X}}\Big)^T\Big(\frac{\partial\mathbf{u}}{\partial\mathbf{X}} + t\frac{\partial\delta\mathbf{u}}{\partial\mathbf{X}}\Big)\Big]\Big|_{t=0} = \\ &= \frac{1}{2}\Big[\Big(\frac{\partial\delta\mathbf{u}}{\partial\mathbf{X}}\Big)^T\frac{\partial\mathbf{u}}{\partial\mathbf{X}} + \Big(\frac{\partial\mathbf{u}}{\partial\mathbf{X}}\Big)^T\frac{\partial\delta\mathbf{u}}{\partial\mathbf{X}}\Big]\end{aligned}$$

La (2.17) può essere scritta anche in forma materiale, si dimostra infatti, che risulta:

$$\delta\Pi = \delta\mathcal{L}_{\text{int}} - \delta\mathcal{L}_{\text{ext}} = \int_{\mathcal{C}_0} \mathbf{S} : \delta\mathbf{E}\, dV_0 - \Big(\int_{\mathcal{C}_0} J\mathbf{b}\cdot\delta\mathbf{u}\, dV_0 + \int_{\partial\mathcal{C}_0} J\mathbf{f}\cdot\delta\mathbf{u}\, dA_0\Big) = 0 \tag{2.19}$$

avendo, anche in questo caso, distinto la variazione virtuale del lavoro interno, per il primo addendo, e la variazione del lavoro esterno, per il secondo. Secondo passaggi analoghi alla (2.18) si verifica:

$$\delta\mathbf{E} = \frac{1}{2}(\delta\mathbf{F}^T\mathbf{F} + \delta\mathbf{F}^T\mathbf{F}) \tag{2.20}$$

Legame costitutivo elastico lineare omogeneo e isotropo
La risposta di un corpo deformabile soggetto ad un generico sistema di forze esterne non può essere definita senza tenere conto del materiale del quale è costituito. I *legami costitutivi* definiscono le leggi che legano le misure di tensione a delle specifiche misure di deformazione sotto le condizioni definite da determinati assiomi, per i quali, ad esempio, devono dare risultati invarianti sotto l'azione di moti rigidi.

Per la presente tesi si considera un legame costitutivo elastico lineare omogeneo e isotropo, il quale definisce il secondo tensore di Piola-Kirchhoff $\mathbf{S}$ tramite il tensore costitutivo del quarto ordine $\mathbb{H}$ agente sul tensore di deformazione $\mathbf{E}$ secondo la (1.36):

$$\mathbf{S} = \mathbb{H} : \mathbf{E} \tag{2.21}$$

Il tensore costitutivo $\mathbb{H}$ è definito:

$$\mathbb{H} = \frac{\nu E}{(1+\nu)(1-2\nu)} \mathbf{1} \otimes \mathbf{1} + \frac{E}{(1+\nu)} \mathbb{I}_S \tag{2.22}$$

dove $\mathbb{I}_S$ è il *tensore identità simmetrico del quarto ordine*, definito da $\mathbb{I} = (\mathbb{I} + \mathbb{I}^T)/2$, mentre E e ν, costanti, sono, rispettivamente, il modulo di Young e il coefficiente di Poisson del materiale.

2.2 Ipotesi cinematiche

Seguendo la teoria descritta in [10], in riferimento alla configurazione deformata dello shell $\mathcal{C}$, si definiscono due superfici differenziabili in $\mathbb{R}^3$, parametrizzate dalla coppia di coordinate $(\xi^1, \xi^2) \in \mathcal{A} \subseteq \mathbb{R}^2$. La prima individua la posizione della superficie del piano medio dello shell $\boldsymbol{\varphi} \colon \mathcal{A} \to \mathbb{R}^3$, la seconda definisce il vettore direttore $\mathbf{t} \colon \mathcal{A} \to \mathrm{S}^2$, ristretto alla condizione $\|\mathbf{t}\|$, che indica, punto per punto, la direzione della fibra dello shell, avendo indicato con S^2 l'insieme dei vettori in $\mathbb{R}^3$ con norma unitaria.

La superficie media dello shell nella configurazione deformata è indicata con $\mathcal{S}$, il proprio contorno con $\Gamma = \partial\mathcal{S}$.

Dall'ipotesi per la quale ogni fibra dello shell rimanga rettilinea a deformazione avvenuta, ogni punto del $\mathbf{x}$ è individuato attraverso una funzione $\boldsymbol{\Phi} \colon \mathcal{A} \times [h^-, h^+] \to \mathbb{R}^3$:

$$\mathbf{x} = \boldsymbol{\Phi}(\xi^1, \xi^2, \xi) = \boldsymbol{\varphi}(\xi^1, \xi^2) + \xi \mathbf{t}(\xi^1, \xi^2) \tag{2.23}$$

dove $-h^+$ e h^-, con $h^+ > -h^-$, sono le quote della superficie superiore e inferiore dello shell a partire dal piano medio; si è inoltre introdotta la

coordinata $\xi \in [h^-, h^+]$. Se la densità del materiale del quale è costituito lo shell è uniforme, come nel caso studiato nella presente tesi, si ha $h^- = -\frac{h}{2}$ e $h^+ = \frac{h}{2}$, avendo definito con h lo spessore dello shell.

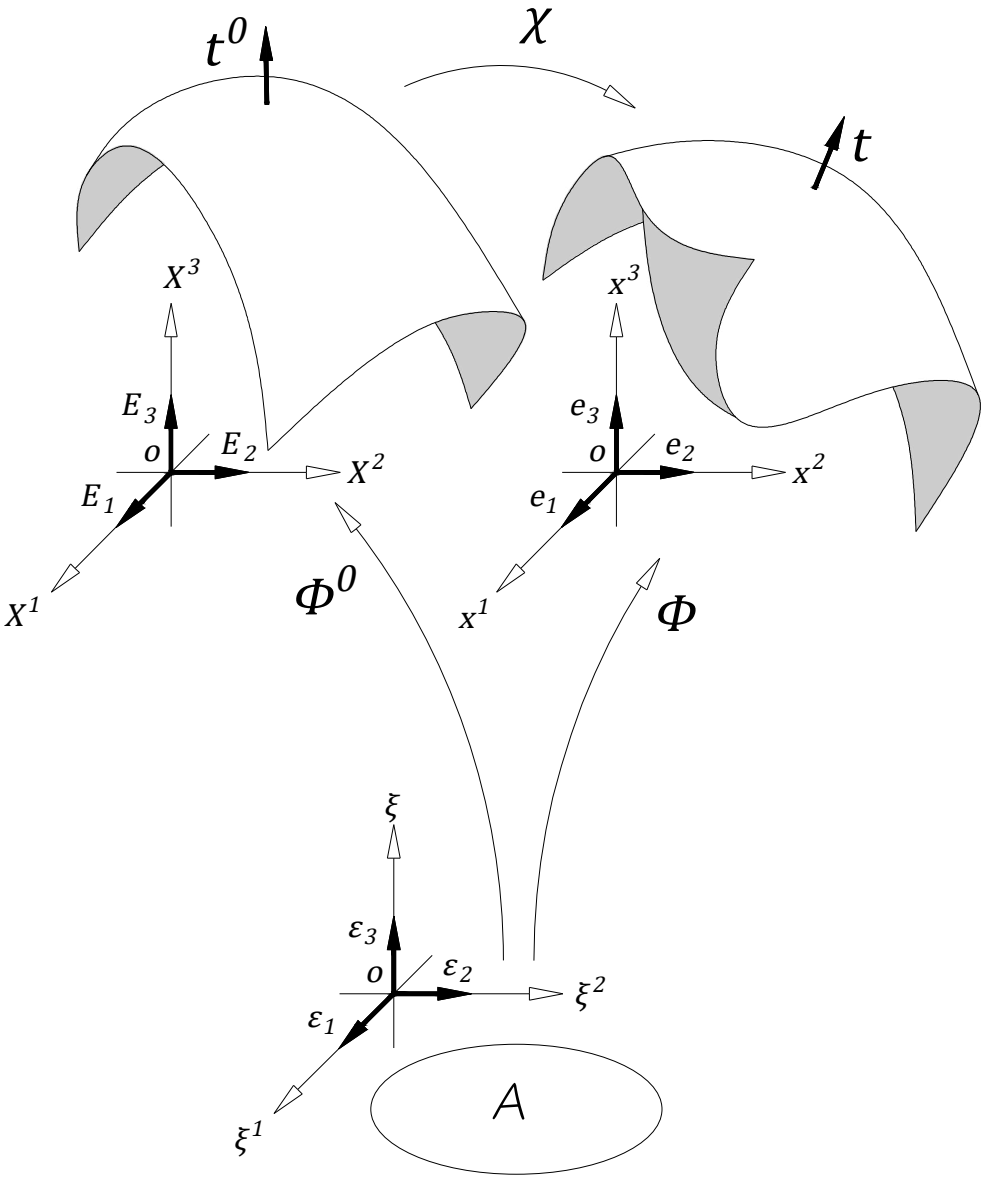


Figura 2.2: *Configurazioni dello shell e dominio delle coordinate parametrizzate, immagine fedele a quanto raffigurato in [10].*

In riferimento alla configurazione iniziale $\mathcal{C}_0$, invece, ogni punto $\mathbf{X}$ dello

shell è individuato dalla funzione $\mathbf{\Phi}^0 \colon \mathcal{A} \times [-\frac{h}{2}, \frac{h}{2}] \to \mathbb{R}^3$:

$$\mathbf{X} = \mathbf{\Phi}^0(\xi^1, \xi^2, \xi) = \boldsymbol{\varphi}^0(\xi^1, \xi^2) + \xi \mathbf{t}^0(\xi^1, \xi^2) \tag{2.24}$$

dove si distinguono le superficie differenziabili relative al piano medio medio $\boldsymbol{\varphi}^0 \colon \mathcal{A} \to \mathbb{R}^3$ e al direttore $\mathbf{t}^0 \colon \mathcal{A} \to \mathrm{S}^2$ relative alla configurazione iniziale, quest'ultima sempre costretta dalla condizione $\|\mathbf{t}^0\| = 1$.

La superficie media dello shell nella configurazione iniziale è indicata con $\mathcal{S}_0$, il proprio contorno con $\Gamma_0 = \partial\mathcal{S}_0$.

La funzione di deformazione $\boldsymbol{\chi} \colon \mathcal{C}_0 \to \mathcal{C}$ risulta dunque pari a $\boldsymbol{\chi} = \mathbf{\Phi} \circ (\mathbf{\Phi}^0)^{-1}$, dove con $\circ$ si è indicata la composizione di funzioni.

In riferimento alla figura 2.2, si identificano le due configurazioni, iniziale e deformata, e il dominio delle coordinate parametrizzate, avendo per quest'ultimo indicato con $\{\boldsymbol{\epsilon}_i\}_{i=1,\dots,3}$ i vettori della base cartesiana.

Essendo $\mathbf{x} = \mathbf{\Phi}(\xi^1, \xi^2, \xi)$ il vettore posizione nella configurazione deformata, i vettori della base naturale covariante nella configurazione deformata $\{\mathbf{g}_i\}_{i=1,\dots,3}$ risultano:

$$\mathbf{g}_1 = \frac{\partial \mathbf{\Phi}}{\partial \xi^1} = \boldsymbol{\varphi}_{,1} + \xi \mathbf{t}_{,1} \qquad \mathbf{g}_2 = \frac{\partial \mathbf{\Phi}}{\partial \xi^2} = \boldsymbol{\varphi}_{,2} + \xi \mathbf{t}_{,2} \qquad \mathbf{g}_3 = \frac{\partial \mathbf{\Phi}}{\partial \xi} = \mathbf{t} \tag{2.25}$$

dove con il simbolo $(\bullet)_{,i}$ si indica la derivata parziale rispetto alla coordinata ξ^i.

Analogamente, i vettori della base naturale covariante nella configurazione iniziale$\{\mathbf{G}_i\}_{i=1,\dots,3}$, invece, risultano:

$$\mathbf{G}_1 = \frac{\partial \mathbf{\Phi}^0}{\partial \xi^1} = \boldsymbol{\varphi}^0_{,1} + \xi \mathbf{t}^0_{,1} \qquad \mathbf{G}_2 = \frac{\partial \mathbf{\Phi}^0}{\partial \xi^2} = \boldsymbol{\varphi}^0_{,2} + \xi \mathbf{t}^0_{,2} \qquad \mathbf{G}_3 = \frac{\partial \mathbf{\Phi}^0}{\partial \xi} = \mathbf{t}^0 \tag{2.26}$$

I vettori delle base naturali covariante e controvariante nella configurazione iniziale, se calcolati in corrispondenza della superficie media, ossia per $\xi = 0$, vengono indicati con $\mathbf{a}^0_i = \mathbf{G}_i(\xi = 0)$ e $\mathbf{a}^i_0 = \mathbf{G}^i(\xi = 0)$; l'analogia con la configurazione deformata è analoga omettendo l'indice (pedice) $_0$ (^{0}).

Un'ulteriore ipotesi che si introduce è quella per cui la fibra dello shell nella configurazione iniziale risulti ortogonale alla superficie media. Introducendo le lettere greche per indicare gli indici 1, 2, l'ipotesi precedente si traduce nella relazione $\boldsymbol{\varphi}^0_{,\alpha} \cdot \mathbf{t}^0 = \mathbf{G}_\alpha \cdot \mathbf{G}_3 = 0$, osservando che nel secondo passaggio si è fatto uso della condizione $\|\mathbf{t}^0\| = 1$ dalla quale si ottiene $\mathbf{t}^0_{,\alpha} \cdot \mathbf{t}^0 = 0$ (analogamente per la configurazione deformata si ha $\mathbf{t}_{,\alpha} \cdot \mathbf{t} = 0$). Si fa riferimento alla figura 2.3.

Si definiscono le funzioni $\nabla\mathbf{\Phi} \colon \mathcal{A} \times [-\frac{h}{2}, \frac{h}{2}] \to \mathbb{R}^3$ e $\nabla\mathbf{\Phi}^0 \colon \mathcal{A} \times [-\frac{h}{2}, \frac{h}{2}] \to \mathbb{R}^3$:

$$\nabla\mathbf{\Phi} = \frac{\partial \mathbf{\Phi}}{\partial \xi^i} \otimes \boldsymbol{\epsilon}^i = \mathbf{g}_i \otimes \boldsymbol{\epsilon}^i \qquad \nabla\mathbf{\Phi}^0 = \frac{\partial \mathbf{\Phi}^0}{\partial \xi^i} \otimes \boldsymbol{\epsilon}^i = \mathbf{G}_i \otimes \boldsymbol{\epsilon}^i \tag{2.27}$$

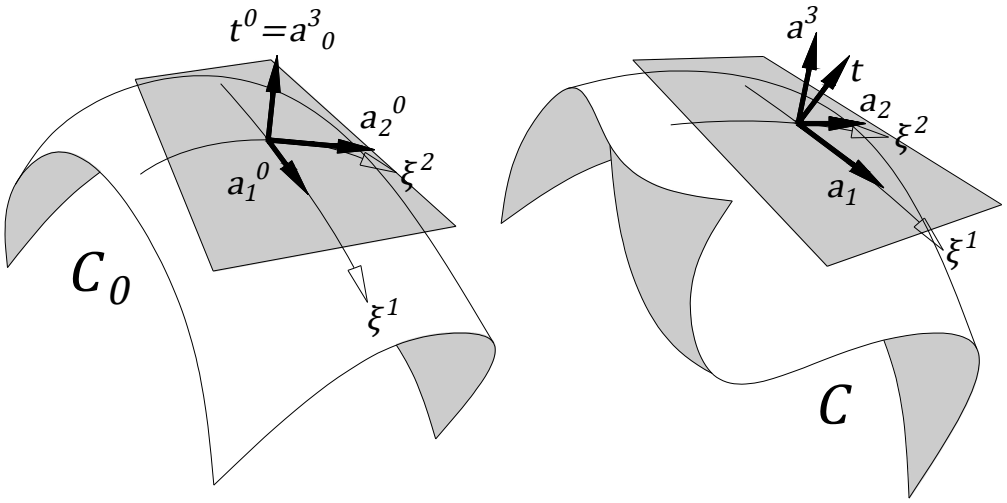


Figura 2.3: *Si osserva l'ortogonalità del vettore direttore nella configurazione iniziale con i vettori della base covariante sulla superficie media dello shell (a sinistra). Questo non è generalmente vero nella configurazione iniziale (a destra).*

Si verifica facilmente che $(\nabla\mathbf{\Phi}^0)^{-1} = \boldsymbol{\epsilon}_i \otimes \mathbf{G}^i$. Analogamente, si ha $(\nabla\mathbf{\Phi})^{-1} = \boldsymbol{\epsilon}_i \otimes \mathbf{g}^i$. Il tensore gradiente di deformazione $\mathbf{F}$ è definito dal gradiente della funzione $\boldsymbol{\chi}$ rispetto alla configurazione iniziale $\mathcal{C}_0$, definito dall'operatore GRAD:

$$\mathbf{F} = \text{GRAD}\boldsymbol{\chi} = \frac{\partial \mathbf{\Phi}}{\partial \xi^i} \otimes \mathbf{G}^i = \mathbf{g}_i \otimes \mathbf{G}^i \tag{2.28}$$

il quale, si verifica facilmente, è pari a $\mathbf{F} = \nabla\mathbf{\Phi}(\nabla\mathbf{\Phi}^0)^{-1}$. Con le relazioni precedenti si verifica:

$$\begin{aligned} \mathbf{g}_i &= \nabla\mathbf{\Phi}\boldsymbol{\epsilon}^i = \mathbf{F}\mathbf{G}_i \qquad & \mathbf{g}^i &= (\nabla\mathbf{\Phi})^{-T}\boldsymbol{\epsilon}^i = \mathbf{F}^{-T}\mathbf{G}^i \\ \mathbf{G}_i &= \nabla\mathbf{\Phi}^0\boldsymbol{\epsilon}^i = \mathbf{F}^{-1}\mathbf{g}_i \qquad & \mathbf{G}^i &= (\nabla\mathbf{\Phi}^0)^{-T}\boldsymbol{\epsilon}^i = \mathbf{F}^T\mathbf{g}^i \end{aligned} \tag{2.29}$$

Il Jacobiano della deformazione J è definito, sfruttando la (1.27), da:

$$\det\mathbf{F} = J = \frac{j}{j^0} \tag{2.30}$$

avendo definito con $j = \det\nabla\mathbf{\Phi} = \mathbf{g}_1 \times \mathbf{g}_2 \cdot \mathbf{t}$ e con $j^0 = \det\nabla\mathbf{\Phi}^0 = \mathbf{G}_1 \times \mathbf{G}_2 \cdot \mathbf{t}^0$, dove nei secondi passaggi si è ricordata la (1.30). I differenziali delle superfici medie, deformata ed iniziale, sono definiti, rispettivamente:

$$\bar{j} = \|\boldsymbol{\varphi}_{,1} \times \boldsymbol{\varphi}_{,2}\| \qquad \bar{j^0} = \|\boldsymbol{\varphi}^0_{,1} \times \boldsymbol{\varphi}^0_{,2}\| \tag{2.31}$$

Si osserva che, dalla condizione $\boldsymbol{\varphi}_{,\alpha} \cdot \mathbf{t} = 0$, per $\alpha = 1, 2$, si ottiene $j^0(\xi = 0) = \bar{j}^0$ (mentre, generalmente $j(\xi = 0) \neq \bar{j}$).

Il tensore di deformazione $\mathbf{E}$ è dato da:

$$\mathbf{E} = \frac{1}{2}(\mathbf{F}^T\mathbf{F} - \mathbf{1}) \tag{2.32}$$

di conseguenza, sostituendo la (2.28) ad $\mathbf{F}$ e la (1.23) per $\mathbf{1}$, svolgendo i calcoli si ottiene:

$$\mathbf{E} = \frac{1}{2}(g_{ij} - G_{ij})\mathbf{G}^i \otimes \mathbf{G}^j \tag{2.33}$$

Dalle (2.25) si osserva che:

$$\begin{aligned} g_{\alpha\beta} &= \boldsymbol{\varphi}_{,\alpha} \cdot \boldsymbol{\varphi}_{,\beta} + \xi(\boldsymbol{\varphi}_{,\alpha} \cdot \mathbf{t}_{,\beta} + \boldsymbol{\varphi}_{,\beta} \cdot \mathbf{t}_{,\alpha}) + (\xi)^2(\mathbf{t}_{,\alpha} \cdot \mathbf{t}_{,\beta}) \\ g_{3\alpha} &= g_{\alpha 3} = \boldsymbol{\varphi}_{,\alpha} \cdot \mathbf{t} + \xi \mathbf{t} \cdot \mathbf{t}_{,\alpha} = \boldsymbol{\varphi}_{,\alpha} \cdot \mathbf{t} \\ g_{33} &= 1 \end{aligned} \tag{2.34}$$

Analogamente, dalle (2.26):

$$\begin{aligned} G_{\alpha\beta} &= \boldsymbol{\varphi}^0_{,\alpha} \cdot \boldsymbol{\varphi}^0_{,\beta} + \xi(\boldsymbol{\varphi}^0_{,\alpha} \cdot \mathbf{t}^0_{,\beta} + \boldsymbol{\varphi}^0_{,\beta} \cdot \mathbf{t}^0_{,\alpha}) + (\xi)^2(\mathbf{t}^0_{,\alpha} \cdot \mathbf{t}^0_{,\beta}) \\ G_{3\alpha} &= G_{\alpha 3} = \boldsymbol{\varphi}^0_{,\alpha} \cdot \mathbf{t}^0 + \xi \mathbf{t}^0 \cdot \mathbf{t}^0_{,\alpha} = \boldsymbol{\varphi}^0_{,\alpha} \cdot \mathbf{t}^0 \\ G_{33} &= 1 \end{aligned} \tag{2.35}$$

(il quadrato della componente ξ è stato indicato con l'ausilio delle parentesi tonde per evitare la confusione con la coordinata ξ^2). Sostituendo le (2.34), (2.35) nella (2.33) e trascurando i termini in $(\xi)^2$ per l'ipotesi di shell sottile, si ottiene:

$$\begin{aligned} \mathbf{E} = &\frac{1}{2}(\boldsymbol{\varphi}_{,\alpha} \cdot \boldsymbol{\varphi}_{,\beta} - \boldsymbol{\varphi}^0_{,\alpha} \cdot \boldsymbol{\varphi}^0_{,\beta})\mathbf{G}^\alpha \otimes \mathbf{G}^\beta + \\ &+ \frac{1}{2}\xi[(\boldsymbol{\varphi}_{,\alpha} \cdot \mathbf{t}_{,\beta} - \boldsymbol{\varphi}^0_{,\alpha} \cdot \mathbf{t}^0_{,\beta}) + (\boldsymbol{\varphi}_{,\beta} \cdot \mathbf{t}_{,\alpha} - \boldsymbol{\varphi}^0_{,\beta} \cdot \mathbf{t}^0_{,\alpha})]\mathbf{G}^\alpha \otimes \mathbf{G}^\beta + \\ &+ \frac{1}{2}(\boldsymbol{\varphi}_{,\alpha} \cdot \mathbf{t} - \boldsymbol{\varphi}^0_{,\alpha} \cdot \mathbf{t}^0)(\mathbf{G}^\alpha \otimes \mathbf{G}^3 + \mathbf{G}^3 \otimes \mathbf{G}^\alpha) \end{aligned} \tag{2.36}$$

(in realtà, si ricorda che $\boldsymbol{\varphi}^0_{,\alpha} \cdot \mathbf{t}^0 = 0$). Introducendo le seguenti *variabili di deformazione*:

$$\begin{array}{lll} a_{\alpha\beta} = \boldsymbol{\varphi}_{,\alpha} \cdot \boldsymbol{\varphi}_{,\beta} & \kappa_{\alpha\beta} = \boldsymbol{\varphi}_{,\alpha} \cdot \mathbf{t}_{,\beta} & \gamma_\alpha = \boldsymbol{\varphi}_{,\alpha} \cdot \mathbf{t} \\ a^0_{\alpha\beta} = \boldsymbol{\varphi}^0_{,\alpha} \cdot \boldsymbol{\varphi}^0_{,\beta} & \kappa^0_{\alpha\beta} = \boldsymbol{\varphi}^0_{,\alpha} \cdot \mathbf{t}^0_{,\beta} & \gamma^0_\alpha = \boldsymbol{\varphi}^0_{,\alpha} \cdot \mathbf{t}^0 \end{array} \tag{2.37}$$

dalle quali si definiscono le seguenti *misure di deformazione*:

$$\varepsilon_{\alpha\beta} = \frac{1}{2}(a_{\alpha\beta} - a^0_{\alpha\beta}) \qquad \chi_{\alpha\beta} = \kappa_{\alpha\beta} - \kappa^0_{\alpha\beta} \qquad \rho_\alpha = \gamma_\alpha - \gamma^0_\alpha \tag{2.38}$$

il tensore di deformazione (2.36) può essere riscritto (si nota che i termini in $\mathbf{G}^3 \otimes \mathbf{G}^3$ si semplificano):

$$\mathbf{E} = (\varepsilon_{\alpha\beta} + \xi\chi_{(\alpha\beta)})\mathbf{G}^\alpha \otimes \mathbf{G}^\beta + \frac{1}{2}\rho_\alpha(\mathbf{G}^\alpha \otimes \mathbf{G}^3 + \mathbf{G}^3 \otimes \mathbf{G}^\alpha) \tag{2.39}$$

avendo definito le componenti simmetriche:

$$\chi_{(\alpha\beta)} = \kappa_{(\alpha\beta)} - \kappa^0_{(\alpha\beta)} \qquad \kappa_{(\alpha\beta)} = \frac{1}{2}(\kappa_{\alpha\beta} + \kappa_{\beta\alpha}) \qquad \kappa^0_{(\alpha\beta)} = \frac{1}{2}(\kappa^0_{\alpha\beta} + \kappa^0_{\beta\alpha})$$

2.2.1 Divergenza nelle due configurazioni

Si segue quanto formulato in [17] per scrivere in modo efficace l'operatore divergenza, rispettivamente di un campo tensoriale e di un campo vettoriale, nelle due configurazioni.

Derivate di j e j^0 lungo la generica coordinata ξ^i
In riferimento alla configurazione deformata, si presenta un'identità utile. Si osserva che, in base alla (1.30):

$$j = \det\nabla\boldsymbol{\Phi} = \nabla\boldsymbol{\Phi}\boldsymbol{\epsilon}_1 \cdot \nabla\boldsymbol{\Phi}\boldsymbol{\epsilon}_2 \times \nabla\boldsymbol{\Phi}\boldsymbol{\epsilon}_3 \tag{2.40}$$

Applicando ad ambo i membri la derivata parziale lungo la generica coordinata ξ^i:

$$\begin{aligned} j_{,i} = {} & \nabla\boldsymbol{\Phi}_{,i}\boldsymbol{\epsilon}_1 \cdot \nabla\boldsymbol{\Phi}\boldsymbol{\epsilon}_2 \times \nabla\boldsymbol{\Phi}\boldsymbol{\epsilon}_3 + \nabla\boldsymbol{\Phi}\boldsymbol{\epsilon}_1 \cdot \nabla\boldsymbol{\Phi}_{,i}\boldsymbol{\epsilon}_2 \times \nabla\boldsymbol{\Phi}\boldsymbol{\epsilon}_3 + \\ & + \nabla\boldsymbol{\Phi}\boldsymbol{\epsilon}_1 \cdot \nabla\boldsymbol{\Phi}\boldsymbol{\epsilon}_2 \times \nabla\boldsymbol{\Phi}_{,i}\boldsymbol{\epsilon}_3 \end{aligned}$$

Riscrivendo gli ultimi due addendi sfruttando l'identità (facilmente verificabile) $\mathbf{a}\cdot\mathbf{b}\times\mathbf{c} = \mathbf{a}\times\mathbf{b}\cdot\mathbf{c} = \mathbf{c}\times\mathbf{a}\cdot\mathbf{b}$, si ottiene:

$$\begin{aligned} j_{,i} = {} & \nabla\boldsymbol{\Phi}_{,i}\boldsymbol{\epsilon}_1 \cdot \nabla\boldsymbol{\Phi}\boldsymbol{\epsilon}_2 \times \nabla\boldsymbol{\Phi}\boldsymbol{\epsilon}_3 + \nabla\boldsymbol{\Phi}_{,i}\boldsymbol{\epsilon}_2 \cdot \nabla\boldsymbol{\Phi}\boldsymbol{\epsilon}_3 \times \nabla\boldsymbol{\Phi}\boldsymbol{\epsilon}_1 + \\ & + \nabla\boldsymbol{\Phi}_{,i}\boldsymbol{\epsilon}_3 \cdot \nabla\boldsymbol{\Phi}\boldsymbol{\epsilon}_1 \times \nabla\boldsymbol{\Phi}\boldsymbol{\epsilon}_2 \end{aligned}$$

Usando la (1.33) si ottiene:

$$\begin{aligned} j_{,i} = {} & \nabla\boldsymbol{\Phi}_{,i}\boldsymbol{\epsilon}_1 \cdot (\det\nabla\boldsymbol{\Phi})(\nabla\boldsymbol{\Phi})^{-T}(\boldsymbol{\epsilon}_2 \times \boldsymbol{\epsilon}_3) + \\ & + \nabla\boldsymbol{\Phi}_{,i}\boldsymbol{\epsilon}_2 \cdot (\det\nabla\boldsymbol{\Phi})(\nabla\boldsymbol{\Phi})^{-T}(\boldsymbol{\epsilon}_3 \times \boldsymbol{\epsilon}_1) + \\ & + \nabla\boldsymbol{\Phi}_{,i}\boldsymbol{\epsilon}_3 \cdot (\det\nabla\boldsymbol{\Phi})(\nabla\boldsymbol{\Phi})^{-T}(\boldsymbol{\epsilon}_1 \times \boldsymbol{\epsilon}_2) \end{aligned}$$

Per cui, svolgendo i calcoli e raccogliendo a fattor comune:

$$j_{,i} = j\big(\nabla\boldsymbol{\Phi}_{,i}\boldsymbol{\epsilon}_1 \cdot (\nabla\boldsymbol{\Phi})^{-T}\boldsymbol{\epsilon}_1 + \nabla\boldsymbol{\Phi}_{,i}\boldsymbol{\epsilon}_2 \cdot (\nabla\boldsymbol{\Phi})^{-T}\boldsymbol{\epsilon}_2 + \nabla\boldsymbol{\Phi}_{,i}\boldsymbol{\epsilon}_3 \cdot (\nabla\boldsymbol{\Phi})^{-T}\boldsymbol{\epsilon}_3\big)$$

Sfruttando la proprietà del trasposto (1.18):

$$j_{,i} = j\big(\boldsymbol{\epsilon}_1 \cdot (\nabla\boldsymbol{\Phi})^{-1}\nabla\boldsymbol{\Phi}_{,i}\boldsymbol{\epsilon}_1 + \boldsymbol{\epsilon}_2 \cdot (\nabla\boldsymbol{\Phi})^{-1}\nabla\boldsymbol{\Phi}_{,i}\boldsymbol{\epsilon}_2 + \boldsymbol{\epsilon}_3 \cdot (\nabla\boldsymbol{\Phi})^{-1}\nabla\boldsymbol{\Phi}_{,i}\boldsymbol{\epsilon}_3\big)$$

Svolgendo i calcoli:

$$\begin{aligned} j_{,i} = j\Big[&\boldsymbol{\epsilon}_1 \cdot (\boldsymbol{\epsilon}_j \otimes \mathbf{G}^j)\Big(\frac{\partial}{\partial\xi^i}\Big(\frac{\partial\boldsymbol{\Phi}}{\partial\xi^k}\Big) \otimes \boldsymbol{\epsilon}^k\Big)\boldsymbol{\epsilon}_1 + \\ &+ \boldsymbol{\epsilon}_2 \cdot (\boldsymbol{\epsilon}_j \otimes \mathbf{G}^j)\Big(\frac{\partial}{\partial\xi^i}\Big(\frac{\partial\boldsymbol{\Phi}}{\partial\xi^k}\Big) \otimes \boldsymbol{\epsilon}^k\Big)\boldsymbol{\epsilon}_2 + \\ &+ \boldsymbol{\epsilon}_3 \cdot (\boldsymbol{\epsilon}_j \otimes \mathbf{G}^j)\Big(\frac{\partial}{\partial\xi^i}\Big(\frac{\partial\boldsymbol{\Phi}}{\partial\xi^k}\Big) \otimes \boldsymbol{\epsilon}^k\Big)\boldsymbol{\epsilon}_3\Big] = \\ = j\Big(&\frac{\partial\xi^1}{\partial x^j}\frac{\partial^2 x^j}{\partial\xi^i\partial\xi^1} + \frac{\partial\xi^2}{\partial x^j}\frac{\partial^2 x^j}{\partial\xi^i\partial\xi^2} + \frac{\partial\xi^3}{\partial x^j}\frac{\partial^2 x^j}{\partial\xi^i\partial\xi^3}\Big) \end{aligned}$$

Infine, rinominando gli indici, si ottiene:

$$\frac{1}{j}j_{,i} = \frac{\partial\xi^j}{\partial x^k}\frac{\partial^2 x^k}{\partial\xi^i\partial\xi^j} \tag{2.41}$$

Ripercorrendo calcoli analoghi, si dimostra:

$$\frac{1}{j^0}j^0_{,i} = \frac{\partial\xi^j}{\partial X^k}\frac{\partial^2 X^k}{\partial\xi^i\partial\xi^j} \tag{2.42}$$

Divergenza di un vettore nelle due configurazioni
Si osserva ora che, dato un vettore $\boldsymbol{\phi} \in \mathbb{R}^3$, essendo costanti i vettori della base duale cartesiana, definendo con div l'operatore divergenza rispetto alla configurazione deformata, si ha:

$$\begin{aligned} \mathrm{div}\boldsymbol{\phi} = \frac{\partial\boldsymbol{\phi}}{\partial x^i}\cdot\mathbf{e}^i = \frac{\partial(\boldsymbol{\phi}\cdot\mathbf{e}^i)}{\partial x^i} = \frac{\partial}{\partial\xi^j}\Big(\boldsymbol{\phi}\cdot\frac{\partial x^i}{\partial\xi^k}\mathbf{g}^k\Big)\frac{\partial\xi^j}{\partial x^i} = \\ = \frac{\partial}{\partial\xi^j}\Big(\boldsymbol{\phi}\cdot\frac{\partial x^i}{\partial\xi^k}\mathbf{g}^k\frac{\partial\xi^j}{\partial x^i}\Big) - \boldsymbol{\phi}\cdot\frac{\partial x^i}{\partial\xi^k}\mathbf{g}^k\frac{\partial}{\partial\xi^j}\Big(\frac{\partial\xi^j}{\partial x^i}\Big) \end{aligned} \tag{2.43}$$

avendo fatto uso della relazione fra $\mathbf{e}^i$ e i vettori della base naturale controvariante $\mathbf{g}^k$ secondo la (1.57). Si osserva ora la seguente identità:

$$\frac{\partial\xi^j}{\partial\xi^k} = \frac{\partial\xi^j}{\partial x^i}\frac{\partial x^i}{\partial\xi^k} = \delta^{jk} \tag{2.44}$$

Derivando la precedente lungo ξ^j si ottiene:

$$\frac{\partial}{\partial\xi^j}\Big(\frac{\partial\xi^j}{\partial x^i}\frac{\partial x^i}{\partial\xi^k}\Big) = \frac{\partial}{\partial\xi^j}\Big(\frac{\partial\xi^j}{\partial x^i}\Big)\frac{\partial x^i}{\partial\xi^k} + \frac{\partial\xi^j}{\partial x^i}\frac{\partial}{\partial\xi^j}\Big(\frac{\partial x^i}{\partial\xi^k}\Big) = 0$$

la quale implica:

$$\frac{\partial}{\partial \xi^j}\Big(\frac{\partial \xi^j}{\partial x^i}\Big)\frac{\partial x^i}{\partial \xi^k} = -\frac{\partial \xi^j}{\partial x^i}\frac{\partial}{\partial \xi^j}\Big(\frac{\partial x^i}{\partial \xi^k}\Big) = -\frac{\partial \xi^j}{\partial x^i}\frac{\partial^2 x^i}{\partial \xi^j \partial \xi^k} \tag{2.45}$$

Sostituendo la precedente nella (2.43) e svolgendo i calcoli, rinominando opportunamente gli indici si ottiene:

$$\begin{aligned}\mathrm{div}\boldsymbol{\phi} &= \frac{\partial}{\partial \xi^j}\Big(\boldsymbol{\phi}\cdot\frac{\partial x^i}{\partial \xi^k}\mathbf{g}^k\frac{\partial \xi^j}{\partial x^i}\Big) + \frac{\partial \xi^j}{\partial x^i}\frac{\partial^2 x^i}{\partial \xi^j \partial \xi^k}\boldsymbol{\phi}\cdot\mathbf{g}^k = \\ &= \frac{\partial}{\partial \xi^j}(\boldsymbol{\phi}\cdot\mathbf{g}^j) + \frac{\partial \xi^j}{\partial x^i}\frac{\partial^2 x^i}{\partial \xi^j \partial \xi^k}\boldsymbol{\phi}\cdot\mathbf{g}^k\end{aligned}$$

Per cui, ricordando la (2.41):

$$\mathrm{div}\boldsymbol{\phi} = \frac{\partial}{\partial \xi^j}(\boldsymbol{\phi}\cdot\mathbf{g}^j) + \frac{1}{j}\frac{\partial j}{\partial \xi^k}\boldsymbol{\phi}\cdot\mathbf{g}^k = \frac{1}{j}\frac{\partial}{\partial \xi^i}(j\boldsymbol{\phi}\cdot\mathbf{g}^i) \tag{2.46}$$

La divergenza di $\boldsymbol{\phi}$ rispetto alla configurazione inziale, con un procedimento simile a quanto svolto per ottenere la precedente, risulta:

$$\mathrm{DIV}\boldsymbol{\phi} = \frac{1}{j^0}(j^0\boldsymbol{\phi}\cdot\mathbf{G}^i)_{,i} \tag{2.47}$$

Divergenza di un tensore del secondo ordine nelle due configurazioni

Sia ora $\boldsymbol{\Phi} \in \mathrm{Lin}$ un campo tensoriale del secondo ordine. Svolgendo calcoli analoghi alla (2.43) si ottiene:

$$\begin{aligned}\mathrm{div}\boldsymbol{\Phi} &= \frac{\partial \boldsymbol{\Phi}}{\partial x^i}\mathbf{e}^i = \frac{\partial(\boldsymbol{\Phi}\mathbf{e}^i)}{\partial x^i} = \frac{\partial}{\partial \xi^j}\Big(\boldsymbol{\Phi}\frac{\partial x^i}{\partial \xi^k}\mathbf{g}^k\Big)\frac{\partial \xi^j}{\partial x^i} = \\ &= \frac{\partial}{\partial \xi^j}\Big(\boldsymbol{\Phi}\frac{\partial x^i}{\partial \xi^k}\mathbf{g}^k\frac{\partial \xi^j}{\partial x^i}\Big) - \boldsymbol{\Phi}\frac{\partial x^i}{\partial \xi^k}\mathbf{g}^k\frac{\partial}{\partial \xi^j}\Big(\frac{\partial \xi^j}{\partial x^i}\Big)\end{aligned} \tag{2.48}$$

Per la (2.41) e svolgendo i calcoli si può scrivere:

$$\mathrm{div}\boldsymbol{\Phi} = \frac{\partial}{\partial \xi^j}(\boldsymbol{\Phi}\mathbf{g}^j) + \frac{1}{j}\frac{\partial j}{\partial \xi^k}\boldsymbol{\Phi}\mathbf{g}^k = \frac{1}{j}\frac{\partial}{\partial \xi^i}(j\boldsymbol{\Phi}\mathbf{g}^i) \tag{2.49}$$

Svolgendo calcoli analoghi, la divergenza di $\boldsymbol{\Phi}$ rispetto alla configurazione iniziale $\mathrm{DIV}\boldsymbol{\Phi}$ risulta:

$$\mathrm{DIV}\boldsymbol{\Phi} = \frac{1}{j^0}\frac{\partial}{\partial \xi^i}(j^0\boldsymbol{\Phi}\mathbf{G}^i) \tag{2.50}$$

Teorema della divergenza sulla superficie media dello shell
Sia $\boldsymbol{\psi}\colon \mathcal{A} \to \mathbb{R}^3$ un campo vettoriale tangente alla superficie dello shell nella configurazione deformata $\boldsymbol{\varphi}$, quindi della forma:

$$\boldsymbol{\psi} = \psi^\alpha(\xi^1, \xi^2)\mathbf{a}_\alpha(\xi^1, \xi^2) \tag{2.51}$$

Si indica con $\mathbf{N} = \frac{\mathbf{a}^3}{\|\mathbf{a}^3\|}$ il vettore normale alla superficie media dello shell e con $\boldsymbol{\nu}$ il vettore normale al contorno Γ appartenente al piano $\{\mathbf{a}_\alpha\}_{\alpha=1,2}$. Il vettore $\boldsymbol{\nu}$ risulta pari a $\frac{\mathbf{v}\times\mathbf{N}}{\|\mathbf{v}\times\mathbf{N}\|}$, avendo indicato con $\mathbf{v}$ il vettore tangente al contorno Γ. Parametrizzando Γ (e di conseguenza $\partial\mathcal{A}$) secondo una generica ascissa $s \in [0, l]$, si ha che Γ è identificata da $\boldsymbol{\varphi}(s)$ e $\mathbf{v}$ si può indicare come:

$$\mathbf{v} = \dot{\boldsymbol{\varphi}} = \frac{d\boldsymbol{\varphi}}{ds} = \frac{\partial\boldsymbol{\varphi}}{\partial\xi^\alpha}\frac{d\xi^\alpha}{ds} = \frac{d\xi^\alpha}{ds}\mathbf{a}_\alpha \tag{2.52}$$

Dunque, dalla regola del cambio di variabili:

$$\int_\Gamma \boldsymbol{\psi}\cdot\boldsymbol{\nu}\, d\Gamma = \int_0^l \boldsymbol{\psi}\cdot\frac{\dot{\boldsymbol{\varphi}}\times\mathbf{N}}{\|\dot{\boldsymbol{\varphi}}\times\mathbf{N}\|}\|\dot{\boldsymbol{\varphi}}\|\, ds \tag{2.53}$$

Essendo $\dot{\boldsymbol{\varphi}}$ appartenente al piano generato da $\{\mathbf{a}_\alpha\}_{\alpha=1,2}$, ed essendo $\mathbf{N}$ ortogonale allo stesso piano, si ha $\|\dot{\boldsymbol{\varphi}}\times\mathbf{N}\| = \|\dot{\boldsymbol{\varphi}}\|\cdot\|\mathbf{N}\|$; inoltre si ha che $\|\mathbf{N}\| = 1$. La precedente, dunque, si semplifica:

$$\begin{aligned}\int_0^l \boldsymbol{\psi}\cdot\frac{\dot{\boldsymbol{\varphi}}\times\mathbf{N}}{\|\dot{\boldsymbol{\varphi}}\times\mathbf{N}\|}\|\dot{\boldsymbol{\varphi}}\|\, ds = \int_0^l \boldsymbol{\psi}\cdot\dot{\boldsymbol{\varphi}}\times\mathbf{N}\, ds = \int_0^l \psi^\alpha\frac{d\xi^\beta}{ds}\mathbf{a}_\alpha\times\mathbf{a}_\beta\cdot\mathbf{N}\, ds = \\ = \int_0^l \Big(\psi^1\frac{d\xi^2}{ds} - \psi^2\frac{d\xi^1}{ds}\Big)\bar{j}\, ds\end{aligned}$$

avendo osservato che $\mathbf{a}_1\times\mathbf{a}_2\cdot\mathbf{N} = \|\mathbf{a}_1\times\mathbf{a}_2\| = \bar{j}$. Infine, dal teorema di Green, si può scrivere:

$$\int_0^l \Big(\psi^1\frac{d\xi^2}{ds} - \psi^2\frac{d\xi^1}{ds}\Big)\bar{j}\, ds = \int_{\mathcal{A}} (\bar{j}\psi^\alpha)_{,\alpha}\, d\xi^1 d\xi^2$$

e si ottiene, infine:

$$\int_\Gamma \boldsymbol{\psi}\cdot\boldsymbol{\nu}\, d\Gamma = \int_{\mathcal{A}} (\bar{j}\psi^\alpha)_{,\alpha}\, d\xi^1 d\xi^2 = \int_{\mathcal{S}} \frac{1}{\bar{j}}(\bar{j}\psi^\alpha)_{,\alpha}\, dS = \int_{\mathcal{S}} \mathrm{Div}_{\mathcal{S}}\boldsymbol{\psi}\, dS \tag{2.54}$$

avendo, nel terzo passaggio sfruttato la regola del cambio di variabili per gli integrali di superficie. Si è definita, dunque, la *divergenza sulla superficie* $\mathcal{S}$ del vettore $\boldsymbol{\psi}$:

$$\mathrm{Div}_{\mathcal{S}}\boldsymbol{\psi} = \frac{1}{\bar{j}}(\bar{j}\psi^\alpha)_{,\alpha} = \frac{1}{\bar{j}}(\bar{j}\boldsymbol{\psi}\cdot\mathbf{a}^\alpha)_{,\alpha} \tag{2.55}$$

La (2.54), in riferimento alla superficie iniziale, risulta:

$$\int_{\mathcal{S}_0} \mathrm{Div}_{\mathcal{S}_0}\boldsymbol{\psi}\, dS_0 = \int_{\Gamma_0} \boldsymbol{\psi}\cdot\boldsymbol{\nu}^0\, d\Gamma_0 \tag{2.56}$$

per la quale, la (2.57) diventa:

$$\mathrm{Div}_{\mathcal{S}_0}\boldsymbol{\psi} = \frac{1}{\bar{j}^0}(\bar{j}^0\psi^\alpha)_{,\alpha} = \frac{1}{\bar{j}^0}(\bar{j}^0\boldsymbol{\psi}\cdot\mathbf{a}_0^\alpha)_{,\alpha} \tag{2.57}$$

Si osserva che, in questo caso, la normale alla superficie media dello shell $\mathbf{N}^0 = \frac{\mathbf{a}_0^3}{\|\mathbf{a}_0^3\|} = \mathbf{a}_0^3 = \mathbf{t}^0$ e $\boldsymbol{\nu}^0$ risulta inoltre normale alla superficie laterale dello shell nella configurazione iniziale $\boldsymbol{\nu}^0 = \nu_\alpha^0\mathbf{a}_0^\alpha$ (avendo scritto, in questo caso, il vettore $\boldsymbol{\nu}^0$ secondo la base controvariante seguendo [18]; dalla condizione $\mathbf{a}_0^3 = \mathbf{t}^0$, infatti, il piano generato dai vettori $\{\mathbf{a}_\alpha^0\}_{\alpha=1,2}$ coincide con quello generato da $\{\mathbf{a}_0^\alpha\}_{\alpha=1,2}$).

Nel caso in cui $\boldsymbol{\Psi}$ fosse un campo tensoriale del secondo ordine, invece, ripercorrendo calcoli analoghi si ha:

$$\int_{\mathcal{S}} \mathrm{Div}_{\mathcal{S}}\boldsymbol{\Psi}\, dS = \int_{\Gamma} \boldsymbol{\Psi}\boldsymbol{\nu}\, d\Gamma \qquad \int_{\mathcal{S}_0} \mathrm{Div}_{\mathcal{S}_0}\boldsymbol{\Psi}\, dS = \int_{\Gamma_0} \boldsymbol{\Psi}\boldsymbol{\nu}^0\, d\Gamma_0 \tag{2.58}$$

per le quali, in questo caso, la divergenza di un campo tensoriale risulta:

$$\mathrm{Div}_{\mathcal{S}}\boldsymbol{\Psi} = \frac{1}{\bar{j}}(\bar{j}\boldsymbol{\Psi}\mathbf{a}^\alpha)_{,\alpha} \qquad \mathrm{Div}_{\mathcal{S}_0}\boldsymbol{\Psi} = \frac{1}{\bar{j}^0}(\bar{j}^0\boldsymbol{\Psi}\mathbf{a}_0^\alpha)_{,\alpha} \tag{2.59}$$

2.3 Equazioni indefinite di equilibrio e risultanti di tensione

Si ricavano le equazioni indefinite di equilibrio, rispettivamente alla traslazione e alla rotazione, seguendo il procedimento descritto in [18]. Una formulazione alternativa, a partire dalle due equazioni cardinali della dinamica, è presente in [10]. La formulazione variazionale e il successivo calcolo delle risultanti efficaci di tensione seguono i risultati illustrati in [10].

La prima equazione indefinita di equilibrio del continuo risulta:

$$\mathrm{div}\boldsymbol{\sigma} + \mathbf{b} = \mathbf{0} \tag{2.60}$$

dove $\boldsymbol{\sigma}$ è il tensore di Cauchy e $\mathbf{b}$ è il vettore delle forze di volume. Scrivendo il termine della divergenza secondo la (2.49), la precedente risulta:

$$\frac{1}{j}(j\boldsymbol{\sigma}\mathbf{g}^i)_{,i} + \mathbf{b} = \mathbf{0}$$

Per cui, moltiplicando ambo i membri per j e integrando lungo lo spessore dello shell, si ha:

$$\int_{-\frac{h}{2}}^{\frac{h}{2}} \frac{\partial}{\partial \xi^\alpha}(j\boldsymbol{\sigma}\mathbf{g}^\alpha)\,d\xi + \int_{-\frac{h}{2}}^{\frac{h}{2}} \frac{\partial}{\partial \xi}(j\boldsymbol{\sigma}\mathbf{g}^3)\,d\xi + \int_{-\frac{h}{2}}^{\frac{h}{2}} j\mathbf{b}\,d\xi = \mathbf{0}$$

Estraendo la derivata dall'integrale al primo addendo e calcolando il secondo si ottiene:

$$\frac{\partial}{\partial \xi^\alpha}\Big(\int_{-\frac{h}{2}}^{\frac{h}{2}} j\boldsymbol{\sigma}\mathbf{g}^\alpha\,d\xi\Big) + (j\boldsymbol{\sigma}\mathbf{g}^3)|_{\xi=-\frac{h}{2}}^{\xi=\frac{h}{2}} + \int_{-\frac{h}{2}}^{\frac{h}{2}} j\mathbf{b}\,d\xi = \mathbf{0} \tag{2.61}$$

Si introducono ora le *risultanti di sforzo per unità di superficie iniziale* $\mathbf{n}^\alpha$:

$$\mathbf{n}^\alpha = \frac{1}{\bar{j^0}}\int_{-\frac{h}{2}}^{\frac{h}{2}} j\boldsymbol{\sigma}\mathbf{g}^\alpha\,d\xi \tag{2.62}$$

e la *risultante di sforzo esterna per unità di superficie iniziale*:

$$\bar{\mathbf{n}} = \frac{1}{\bar{j^0}}(j\boldsymbol{\sigma}\mathbf{g}^3)|_{\xi=-\frac{h}{2}}^{\xi=\frac{h}{2}} + \frac{1}{\bar{j^0}}\int_{-\frac{h}{2}}^{\frac{h}{2}} j\mathbf{b}\,d\xi \tag{2.63}$$

Sostituendo le (2.62), (2.63) nella (2.61) e dividendo $\bar{j^0}$, essa diventa l'*equazione di equilibrio alla traslazione dello shell*:

$$\frac{1}{\bar{j^0}}(\bar{j^0}\mathbf{n}^\alpha)_{,\alpha} + \bar{\mathbf{n}} = \mathbf{0} \tag{2.64}$$

Ricominciando dalla prima equazione indefinita di equilibrio (2.60), moltiplicando ambo i membri per j e per ξ e integrando lungo lo spessore dello shell, si ha:

$$\int_{-\frac{h}{2}}^{\frac{h}{2}} \xi\frac{\partial}{\partial \xi^\alpha}(j\boldsymbol{\sigma}\mathbf{g}^\alpha)\,d\xi + \int_{-\frac{h}{2}}^{\frac{h}{2}} \xi\frac{\partial}{\partial \xi}(j\boldsymbol{\sigma}\mathbf{g}^3)\,d\xi + \int_{-\frac{h}{2}}^{\frac{h}{2}} j\xi\mathbf{b}\,d\xi = \mathbf{0}$$

La coordinata ξ può essere integrata all'interno del segno di derivazione lungo α, mentre, per il segno di derivazione lungo ξ, si sostituisce al secondo addendo l'identità:

$$\xi\frac{\partial}{\partial \xi}(j\boldsymbol{\sigma}\mathbf{g}^3) = \frac{\partial}{\partial \xi}(j\xi\boldsymbol{\sigma}\mathbf{g}^3) - j\boldsymbol{\sigma}\mathbf{g}^3$$

ottenendo:

$$\frac{\partial}{\partial \xi^\alpha}\Big(\int_{-\frac{h}{2}}^{\frac{h}{2}} j\xi\boldsymbol{\sigma}\mathbf{g}^\alpha\,d\xi\Big) + \int_{-\frac{h}{2}}^{\frac{h}{2}} \frac{\partial}{\partial \xi}(j\xi\boldsymbol{\sigma}\mathbf{g}^3)\,d\xi - \int_{-\frac{h}{2}}^{\frac{h}{2}} (j\xi\boldsymbol{\sigma}\mathbf{g}^3)\,d\xi + \int_{-\frac{h}{2}}^{\frac{h}{2}} j\xi\mathbf{b}\,d\xi = \mathbf{0}$$

Per cui, svolgendo i calcoli:

$$\frac{\partial}{\partial \xi^\alpha}\Big(\int_{-\frac{h}{2}}^{\frac{h}{2}} j\xi\boldsymbol{\sigma}\mathbf{g}^\alpha\, d\xi\Big) + (j\xi\boldsymbol{\sigma}\mathbf{g}^3)|_{\xi=-\frac{h}{2}}^{\xi=\frac{h}{2}} - \int_{-\frac{h}{2}}^{\frac{h}{2}} (j\xi\boldsymbol{\sigma}\mathbf{g}^3)\, d\xi + \int_{-\frac{h}{2}}^{\frac{h}{2}} j\xi\mathbf{b}\, d\xi = \mathbf{0} \tag{2.65}$$

Si introducono ora le *risultanti di momento per unità di superficie iniziale* $\tilde{\mathbf{m}}^\alpha$ e la *risultante di sforzo lungo lo spessore per unità di superficie iniziale* $\mathbf{l}$:

$$\tilde{\mathbf{m}}^\alpha = \frac{1}{\bar{j}^0}\int_{-\frac{h}{2}}^{\frac{h}{2}} j\xi\boldsymbol{\sigma}\mathbf{g}^\alpha\, d\xi \qquad \mathbf{l} = \frac{1}{\bar{j}^0}\int_{-\frac{h}{2}}^{\frac{h}{2}} j\boldsymbol{\sigma}\mathbf{g}^3\, d\xi \tag{2.66}$$

e la *risultante di momento esterna per unità di superficie iniziale*:

$$\bar{\tilde{\mathbf{m}}} = \frac{1}{\bar{j}^0}(j\xi\boldsymbol{\sigma}\mathbf{g}^3)|_{\xi=-\frac{h}{2}}^{\xi=\frac{h}{2}} + \frac{1}{\bar{j}^0}\int_{-\frac{h}{2}}^{\frac{h}{2}} j\xi\mathbf{b}\, d\xi \tag{2.67}$$

Sostituendo le (2.66), (2.67) nella (2.65) e dividendo $\bar{j}^0$, essa diventa l'*equazione di equilibrio alla rotazione dello shell*:

$$\frac{1}{\bar{j}^0}(\bar{j}^0\tilde{\mathbf{m}}^\alpha)_{,\alpha} + \bar{\tilde{\mathbf{m}}} - \mathbf{l} = \mathbf{0} \tag{2.68}$$

2.3.1 Condizione definita dalla seconda equazione indefinita di equilibrio

Dalla seconda equazione indefinita di equilibrio, che si traduce nella simmetria del tensore di Cauchy, si può affermare:

$$\mathbf{g}_i \times \boldsymbol{\sigma}\mathbf{g}^i = \mathbf{g}_i \times \sigma^{ij}\mathbf{g}_j = 0 \tag{2.69}$$

Di conseguenza:

$$\mathbf{g}_i \times \boldsymbol{\sigma}\mathbf{g}^i = (\boldsymbol{\varphi}_{,\alpha} + \xi\mathbf{t}_{,\alpha}) \times \boldsymbol{\sigma}\mathbf{g}^\alpha + \mathbf{t} \times \boldsymbol{\sigma}\mathbf{g}^3 = \boldsymbol{\varphi}_{,\alpha} \times \boldsymbol{\sigma}\mathbf{g}^\alpha + \mathbf{t}_{,\alpha} \times \xi\boldsymbol{\sigma}\mathbf{g}^\alpha + \mathbf{t} \times \boldsymbol{\sigma}\mathbf{g}^3 = 0$$

Moltiplicando ambo i membri per j, integrando lungo lo spessore dello shell (ricordando che i vettori $\boldsymbol{\varphi}$ e $\mathbf{t}$ sono indipendenti da ξ) e dividendo per $\bar{j}_0$, si ottiene:

$$\boldsymbol{\varphi}_{,\alpha} \times \Big(\frac{1}{\bar{j}_0}\int_{-\frac{h}{2}}^{\frac{h}{2}} j\boldsymbol{\sigma}\mathbf{g}^\alpha\, d\xi\Big) + \mathbf{t}_{,\alpha} \times \Big(\frac{1}{\bar{j}_0}\int_{-\frac{h}{2}}^{\frac{h}{2}} j\xi\boldsymbol{\sigma}\mathbf{g}^\alpha\, d\xi\Big) + \mathbf{t} \times \Big(\frac{1}{\bar{j}_0}\int_{-\frac{h}{2}}^{\frac{h}{2}} j\boldsymbol{\sigma}\mathbf{g}^3\, d\xi\Big) = 0$$

la quale, per mezzo delle (2.62), (2.66), si traduce in:

$$\boldsymbol{\varphi}_{,\alpha} \times \mathbf{n}^\alpha + \mathbf{t}_{,\alpha} \times \tilde{\mathbf{m}}^\alpha + \mathbf{t} \times \mathbf{l} = 0 \tag{2.70}$$

Si definiscono ora le componenti dei vettori $\mathbf{n}^\alpha$, $\tilde{\mathbf{m}}^\alpha$ e $\mathbf{t}_{,\alpha}$ lungo la base $\{\boldsymbol{\varphi}_{,1}, \boldsymbol{\varphi}_{,2}, \mathbf{t}\}$:

$$\mathbf{n}^\alpha = n^{\beta\alpha}\boldsymbol{\varphi}_{,\beta} + q^\alpha \mathbf{t} \qquad \tilde{\mathbf{m}}^\alpha = \tilde{m}^{\beta\alpha}\boldsymbol{\varphi}_{,\beta} + \tilde{m}^{3\alpha}\mathbf{t} \qquad \mathbf{t}_{,\alpha} = \lambda^\mu_\alpha \boldsymbol{\varphi}_{,\mu} + \lambda^3_\alpha \mathbf{t} \quad (2.71)$$

Sostituendo le relazioni (2.71) nella (2.70) si ottiene:

$$\boldsymbol{\varphi}_{,\alpha} \times (n^{\beta\alpha}\boldsymbol{\varphi}_{,\beta} + q^\alpha \mathbf{t}) + (\lambda^\mu_\alpha \boldsymbol{\varphi}_{,\mu} + \lambda^3_\alpha \mathbf{t}) \times (\tilde{m}^{\beta\alpha}\boldsymbol{\varphi}_{,\beta} + \tilde{m}^{3\alpha}\mathbf{t}) + \mathbf{t} \times \mathbf{l} = 0$$

Svolgendo i calcoli e rinominando opportunamente gli indici:

$$(n^{\beta\alpha} - \lambda^\beta_\mu \tilde{m}^{\alpha\mu})\boldsymbol{\varphi}_{,\alpha} \times \boldsymbol{\varphi}_{,\beta} + \mathbf{t} \times [\mathbf{l} - (q^\alpha + \lambda^\alpha_\mu \tilde{m}^{3\mu} - \lambda^3_\mu \tilde{m}^{\alpha\mu})\boldsymbol{\varphi}_{,\alpha}] = 0 \quad (2.72)$$

Moltiplicando ambo i membri per $\mathbf{t}$ il secondo addendo si annulla (infatti, definendo con $\mathbf{v}$ il termine all'interno della parentesi quadra, si ha $\mathbf{t} \cdot \mathbf{t} \times \mathbf{v} = \mathbf{t} \times \mathbf{t} \cdot \mathbf{v} = 0$) e la precedente si riduce:

$$(n^{\beta\alpha} - \lambda^\beta_\mu \tilde{m}^{\alpha\mu})\boldsymbol{\varphi}_{,\alpha} \times \boldsymbol{\varphi}_{,\beta} \cdot \mathbf{t} = 0$$

Svolgendo i calcoli:

$$j[(n^{21} - \lambda^2_\mu \tilde{m}^{1\mu}) - (n^{12} - \lambda^1_\mu \tilde{m}^{2\mu})] = 0$$

di conseguenza:

$$(n^{21} - \lambda^2_\mu \tilde{m}^{1\mu}) = (n^{12} - \lambda^1_\mu \tilde{m}^{2\mu})$$

Si definiscono dunque le *componenti di sforzo membranali* $\tilde{n}^{\beta\alpha}$, le quali risultano simmetriche:

$$\tilde{n}^{\beta\alpha} = n^{\beta\alpha} - \lambda^\beta_\mu \tilde{m}^{\alpha\mu} = \tilde{n}^{\alpha\beta} \quad (2.73)$$

Avendo provato la simmetria delle componenti $\tilde{n}^{\beta\alpha}$, il primo addendo della (2.72) è pari a zero. Introducendo ora le *componenti di sforzo taglianti*:

$$\tilde{q}^\alpha = q^\alpha - \lambda^3_\mu \tilde{m}^{\alpha\mu} \quad (2.74)$$

la (2.72) si riduce in:

$$\mathbf{t} \times [\mathbf{l} - (\tilde{q}^\alpha + \lambda^\alpha_\mu \tilde{m}^{3\mu})\boldsymbol{\varphi}_{,\alpha}] = 0$$

Di conseguenza, per rendere valida la precedente, il vettore $\mathbf{l}$ dovrà essere della forma:

$$\mathbf{l} = \lambda \mathbf{t} + (\tilde{q}^\alpha + \lambda^\alpha_\mu \tilde{m}^{3\mu})\boldsymbol{\varphi}_{,\alpha} \quad (2.75)$$

avendo introdotto la generica componente λ lungo $\mathbf{t}$.

2.4 Formulazione variazionale

Si assumono le seguenti condizioni al contorno sugli spostamenti. In riferimento alla superficie media e al vettore direttore, si ha $\boldsymbol{\varphi} - \boldsymbol{\varphi}^0 = \bar{\boldsymbol{\varphi}}$ per $(\xi^1, \xi^2) \in \partial\mathcal{A}_{\boldsymbol{\varphi}} \subseteq \mathcal{A}$ e $\mathbf{t} - \mathbf{t}^0 = \bar{\mathbf{t}}$ per $(\xi^1, \xi^2) \in \partial\mathcal{A}_{\mathbf{t}} \subseteq \mathcal{A}$, essendo noti i valori di $\bar{\boldsymbol{\varphi}}$ e $\bar{\mathbf{t}}$.

Si identifica con $\mathrm{T}_{\boldsymbol{\Phi}}\mathcal{C}$ lo spazio delle *variazioni ammissibili*, definito dallo spazio tangente alla configurazione deformata in corrispondenza della configurazione $\boldsymbol{\Phi}$:

$$\mathrm{T}_{\boldsymbol{\Phi}}\mathcal{C} = \{(\delta\boldsymbol{\varphi}, \delta\mathbf{t})\colon \mathcal{A} \to \mathbb{R}^3 \times \mathrm{T}_{\mathbf{t}}\mathrm{S}^2 \quad \text{t.c.} \quad \delta\boldsymbol{\varphi}|_{\partial\mathcal{A}_{\boldsymbol{\varphi}}} = \mathbf{0} \quad \delta\mathbf{t}|_{\partial\mathcal{A}_{\mathbf{t}}} = \mathbf{0}\} \tag{2.76}$$

avendo indicato con $\mathrm{T}_{\mathbf{t}}\mathrm{S}^2$ lo spazio tangente alla sfera unitaria S^2 in corrispondenza del vettore $\mathbf{t}$, definito dall'insieme di vettori in $\mathbf{v} \in \mathbb{R}^3$ tali che, fissato $\mathbf{t} \in \mathrm{S}^2$, si ha $\mathbf{v} \cdot \mathbf{t} = 0$. I vettori appartenenti a $\mathrm{T}_{\mathbf{t}}\mathrm{S}^2$, possono essere scritti nella forma $\mathbf{v} = \boldsymbol{\theta} \times \mathbf{t}$, avendo introdotto il vettore $\boldsymbol{\theta} \in \mathbb{R}^3$ tale che $\boldsymbol{\theta} \cdot \mathbf{t} = 0$. Si definisce, dunque, $\delta\mathbf{t} = \delta\boldsymbol{\theta} \times \mathbf{t}$.

Moltiplicando scalarmente la (2.64) per $\delta\boldsymbol{\varphi}$, la (2.68) per $\delta\mathbf{t}$ si ha:

$$\Big(\frac{1}{\bar{j}_0}(\bar{j}_0\mathbf{n}^\alpha)_{,\alpha} + \bar{\mathbf{n}}\Big) \cdot \delta\boldsymbol{\varphi} + \Big(\frac{1}{\bar{j}^0}(\bar{j}^0\tilde{\mathbf{m}}^\alpha)_{,\alpha} + \bar{\tilde{\mathbf{m}}} - \mathbf{l}\Big) \cdot \delta\mathbf{t} = \mathbf{0}$$

Integrando la precedente sulla superficie media nella configurazione iniziale $\mathcal{S}_0$:

$$\int_{\mathcal{S}_0} \Big(\frac{1}{\bar{j}_0}(\bar{j}_0\mathbf{n}^\alpha)_{,\alpha} \cdot \delta\boldsymbol{\varphi} + \bar{\mathbf{n}} \cdot \delta\boldsymbol{\varphi} + \frac{1}{\bar{j}^0}(\bar{j}^0\tilde{\mathbf{m}}^\alpha)_{,\alpha} \cdot \delta\mathbf{t} + \bar{\tilde{\mathbf{m}}} \cdot \delta\mathbf{t} - \mathbf{l} \cdot \delta\mathbf{t}\Big)\, dS_0 = \mathbf{0} \tag{2.77}$$

Si considera ora la (2.57) sul campo vettoriale $(\mathbf{n}^\alpha \cdot \delta\boldsymbol{\varphi})\mathbf{a}^0_\alpha$:

$$\begin{aligned} \mathrm{Div}_{\mathcal{S}_0}[(\mathbf{n}^\alpha \cdot \delta\boldsymbol{\varphi})\mathbf{a}^0_\alpha] &= \frac{1}{\bar{j}^0}[\bar{j}^0(\mathbf{n}^\alpha \cdot \delta\boldsymbol{\varphi})\mathbf{a}^0_\alpha \cdot \mathbf{a}^\beta_0]_{,\beta} = \frac{1}{\bar{j}^0}[\bar{j}^0(\mathbf{n}^\alpha \cdot \delta\boldsymbol{\varphi})]_{,\alpha} = \\ &= \frac{1}{\bar{j}^0}(\bar{j}^0\mathbf{n}^\alpha)_{,\alpha} \cdot \delta\boldsymbol{\varphi} + \mathbf{n}^\alpha \cdot \delta\boldsymbol{\varphi}_{,\alpha} \end{aligned} \tag{2.78}$$

Considerando, invece, il campo vettoriale $(\tilde{\mathbf{m}}^\alpha \cdot \delta\mathbf{t})\mathbf{a}^0_\alpha$, a seguito di passaggi analoghi alla precedente:

$$\mathrm{Div}_{\mathcal{S}_0}[(\tilde{\mathbf{m}}^\alpha \cdot \delta\mathbf{t})\mathbf{a}^0_\alpha] = \frac{1}{\bar{j}^0}(\bar{j}^0\tilde{\mathbf{m}}^\alpha)_{,\alpha} \cdot \delta\mathbf{t} + \tilde{\mathbf{m}}^\alpha \cdot \delta\mathbf{t}_{,\alpha} \tag{2.79}$$

Considerando le (2.78), (2.79) nella (2.77):

$$\begin{aligned} \int_{\mathcal{S}_0} &(\mathrm{Div}_{\mathcal{S}_0}[(\mathbf{n}^\alpha \cdot \delta\boldsymbol{\varphi})\mathbf{a}^0_\alpha] - \mathbf{n}^\alpha \cdot \delta\boldsymbol{\varphi}_{,\alpha} + \bar{\mathbf{n}} \cdot \delta\boldsymbol{\varphi} + \\ &+ \mathrm{Div}_{\mathcal{S}_0}[(\tilde{\mathbf{m}}^\alpha \cdot \delta\mathbf{t})\mathbf{a}^0_\alpha] - \tilde{\mathbf{m}}^\alpha \cdot \delta\mathbf{t}_{,\alpha} + \bar{\tilde{\mathbf{m}}} \cdot \delta\mathbf{t} - \mathbf{l} \cdot \delta\mathbf{t})\, dS_0 = \mathbf{0} \end{aligned} \tag{2.80}$$

Per il teorema della divergenza (2.56), introducendo il vettore normale esterno alla superficie laterale dello shell nella configurazione di riferimento $\boldsymbol{\nu}^0 = \nu^0_\beta \mathbf{a}^\beta_0$, si può scrivere:

$$\begin{aligned}
\int_{\mathcal{S}_0} \mathrm{Div}_{\mathcal{S}_0}[(\mathbf{n}^\alpha \cdot \delta\boldsymbol{\varphi})\mathbf{a}^0_\alpha]\, dS_0 &= \int_{\Gamma_0} (\mathbf{n}^\alpha \cdot \delta\boldsymbol{\varphi})\mathbf{a}^0_\alpha \cdot \boldsymbol{\nu}^0\, d\Gamma_0 = \\
&= \int_{\Gamma_0} (\mathbf{n}^\alpha \nu^0_\alpha) \cdot \delta\boldsymbol{\varphi}\, d\Gamma_0 = \int_{\Gamma_0} \bar{\bar{\mathbf{n}}} \cdot \delta\boldsymbol{\varphi}\, d\Gamma_0 \\
\int_{\mathcal{S}_0} \mathrm{Div}_{\mathcal{S}_0}[(\tilde{\mathbf{m}}^\alpha \cdot \delta\mathbf{t})\mathbf{a}^0_\alpha]\, dS_0 &= \int_{\Gamma_0} (\tilde{\mathbf{m}}^\alpha \cdot \delta\mathbf{t})\mathbf{a}^0_\alpha \cdot \boldsymbol{\nu}^0\, d\Gamma_0 = \\
&= \int_{\Gamma_0} (\tilde{\mathbf{m}}^\alpha \nu^0_\alpha) \cdot \delta\mathbf{t}\, d\Gamma_0 = \int_{\Gamma_0} \bar{\bar{\tilde{\mathbf{m}}}} \cdot \delta\mathbf{t}\, d\Gamma_0
\end{aligned} \tag{2.81}$$

avendo introdotto la *risultante di sforzo esterna* $\bar{\bar{\mathbf{n}}}$ e la *risultante di momento esterna* $\bar{\bar{\tilde{\mathbf{m}}}}$:

$$\bar{\bar{\mathbf{n}}} = \mathbf{n}^\alpha \nu^\alpha_0 \qquad \bar{\bar{\tilde{\mathbf{m}}}} = \tilde{\mathbf{m}}^\alpha \nu^\alpha_0 \tag{2.82}$$

Sostituendo le precedenti nella (2.80) si ottiene la *forma debole delle equazioni di equilibrio*:

$$\begin{aligned}
&\int_{\mathcal{S}_0} (\mathbf{n}^\alpha \cdot \delta\boldsymbol{\varphi}_{,\alpha} + \tilde{\mathbf{m}}^\alpha \cdot \delta\mathbf{t}_{,\alpha} + \mathbf{l} \cdot \delta\mathbf{t})\, dS_0 = \\
&\quad = \int_{\Gamma_0} (\bar{\bar{\mathbf{n}}} \cdot \boldsymbol{\varphi} + \bar{\bar{\tilde{\mathbf{m}}}} \cdot \mathbf{t})\, d\Gamma_0 + \int_{\mathcal{S}_0} (\bar{\mathbf{n}} \cdot \delta\boldsymbol{\varphi} + \bar{\tilde{\mathbf{m}}} \cdot \delta\mathbf{t})\, dS_0
\end{aligned} \tag{2.83}$$

avendo introdotto la variazione virtuale di lavoro interno ed esterno, rispettivamente:

$$\begin{aligned}
\delta\mathcal{L}_{\mathrm{int}} &= \int_{\mathcal{S}_0} (\mathbf{n}^\alpha \cdot \delta\boldsymbol{\varphi}_{,\alpha} + \tilde{\mathbf{m}}^\alpha \cdot \delta\mathbf{t}_{,\alpha} + \mathbf{l} \cdot \delta\mathbf{t})\, dS_0 \\
\delta\mathcal{L}_{\mathrm{ext}} &= \int_{\Gamma_0} (\bar{\bar{\mathbf{n}}} \cdot \delta\boldsymbol{\varphi} + \bar{\bar{\tilde{\mathbf{m}}}} \cdot \delta\mathbf{t})\, d\Gamma_0 + \int_{S_0} (\bar{\mathbf{n}} \cdot \delta\boldsymbol{\varphi} + \bar{\tilde{\mathbf{m}}} \cdot \delta\mathbf{t})\, dS_0
\end{aligned} \tag{2.84}$$

2.4.1 Calcolo delle componenti della variazione virtuale del lavoro interno

Dalla (2.37), si procede ora al calcolo delle *variazioni virtuali delle variabili di deformazione*. In riferimento a $a_{\alpha\beta}$, ricordando che l'operatore $\delta(\bullet)$ segue le regole di derivazione, si ha:

$$\delta a_{\alpha\beta} = \delta(\boldsymbol{\varphi}_{,\alpha} \cdot \boldsymbol{\varphi}_{,\beta}) = \delta\boldsymbol{\varphi}_{,\alpha} \cdot \boldsymbol{\varphi}_{,\beta} + \boldsymbol{\varphi}_{,\alpha} \cdot \delta\boldsymbol{\varphi}_{,\beta} \tag{2.85}$$

Mentre, con passaggi analoghi, si verifica:

$$\delta\gamma_\alpha = \delta\boldsymbol{\varphi}_{,\alpha}\cdot\mathbf{t} + \boldsymbol{\varphi}_{,\alpha}\cdot\delta\mathbf{t} \qquad \delta\kappa_{\alpha\beta} = \delta\boldsymbol{\varphi}_{,\alpha}\cdot\mathbf{t}_{,\beta} + \boldsymbol{\varphi}_{,\alpha}\cdot\delta\mathbf{t}_{,\beta} \tag{2.86}$$

La variazione virtuale del lavoro interno $\delta\mathcal{L}_{\text{int}}$ può essere messa in funzione delle componenti $\tilde{n}^{\beta\alpha}$, $\tilde{m}^{\beta\alpha}$ e q^α. Sostituendo (2.84) l'espressione di $\mathbf{n}^\alpha$ introdotta nella (2.71) si ottiene:

$$\delta\mathcal{L}_{\text{int}} = \int_{\mathcal{S}_0} (n^{\beta\alpha}\boldsymbol{\varphi}_{,\beta}\cdot\delta\boldsymbol{\varphi}_{,\alpha} + q^\alpha\mathbf{t}\cdot\delta\boldsymbol{\varphi}_{,\alpha} + \tilde{m}^{\beta\alpha}\boldsymbol{\varphi}_{,\beta}\cdot\mathbf{t}_{,\alpha} + \tilde{m}^{3\alpha}\mathbf{t}\cdot\delta\mathbf{t}_{,\alpha} + \mathbf{l}\cdot\delta\mathbf{t})\, dS_0 \tag{2.87}$$

applicando la variazione virtuale ad ambo i membri della relazione $\mathbf{t}\cdot\mathbf{t}_{,\alpha} = 0$, si ottiene $\delta\mathbf{t}\cdot\mathbf{t}_\alpha = -\mathbf{t}\cdot\delta\mathbf{t}_\alpha$. Sfruttando tale relazione e ricordando l'espressione (2.71) per $\mathbf{t}_{,\alpha}$ e la (2.75) per $\mathbf{l}$, si osserva che:

$$\begin{aligned}
&\tilde{m}^{3\alpha}\mathbf{t}\cdot\delta\mathbf{t}_{,\alpha} + \mathbf{l}\cdot\delta\mathbf{t} = -\tilde{m}^{3\alpha}\mathbf{t}_{,\alpha}\cdot\delta\mathbf{t} + \mathbf{l}\cdot\delta\mathbf{t} = \\
&\quad = -\tilde{m}^{3\alpha}(\lambda^\mu_\alpha\boldsymbol{\varphi}_{,\mu} + \lambda^3_\alpha\mathbf{t})\cdot\delta\mathbf{t} + (\lambda\mathbf{t} + \tilde{q}^\alpha\boldsymbol{\varphi}_{,\alpha} + \lambda^\alpha_\mu\tilde{m}^{3\mu}\boldsymbol{\varphi}_{,\alpha})\cdot\delta\mathbf{t} = \\
&\quad = \tilde{q}^\alpha\boldsymbol{\varphi}_{,\alpha}\cdot\delta\mathbf{t}
\end{aligned}$$

avendo rinominato in modo opportuno gli indici. L'integranda della (2.87) può dunque essere definita:

$$(\tilde{n}^{\beta\alpha} + \lambda^\beta_\mu\tilde{m}^{\alpha\mu})\boldsymbol{\varphi}_{,\beta}\cdot\delta\boldsymbol{\varphi}_{,\alpha} + (\tilde{q}^\alpha + \lambda^\mu_3\tilde{m}^{\alpha\mu})\mathbf{t}\cdot\delta\boldsymbol{\varphi}_{,\alpha} + \tilde{m}^{\beta\alpha}\boldsymbol{\varphi}_{,\beta}\cdot\delta\mathbf{t}_{,\alpha} + \tilde{q}^\alpha\boldsymbol{\varphi}_{,\alpha}\cdot\delta\mathbf{t}$$

avendo inoltre aggiunto le relazioni (2.71). Il termine $\tilde{m}^{\beta\alpha}\boldsymbol{\varphi}_{,\beta}\cdot\delta\mathbf{t}_{,\alpha}$ può essere riscritto come:

$$\tilde{m}^{\beta\alpha}\boldsymbol{\varphi}_{,\beta}\cdot\delta\mathbf{t}_{,\alpha} = \tilde{m}^{\beta\alpha}\delta\kappa_{\beta\alpha} - \tilde{m}^{\beta\alpha}\mathbf{t}_{,\alpha}\cdot\delta\boldsymbol{\varphi}_{,\beta} = \tilde{m}^{\beta\alpha}\delta\kappa_{\beta\alpha} - \tilde{m}^{\beta\alpha}(\lambda^\mu_\alpha\boldsymbol{\varphi}_{,\mu} + \lambda^3_\alpha\mathbf{t})\cdot\delta\boldsymbol{\varphi}_{,\beta}$$

Rinominando opportunamente gli indici, si osserva che l'ultimo termine della precedente si semplifica con $\lambda^\beta_\mu\tilde{m}^{\alpha\mu}\boldsymbol{\varphi}_{,\beta}\cdot\delta\boldsymbol{\varphi}_{,\alpha}$ e $\lambda^\mu_3\tilde{m}^{\alpha\mu}\mathbf{t}\cdot\delta\boldsymbol{\varphi}_{,\alpha}$. Inoltre, grazie alla simmetria del termine $\tilde{n}^{\beta\alpha}$:

$$\tilde{n}^{\beta\alpha}\boldsymbol{\varphi}_{,\beta}\cdot\delta\boldsymbol{\varphi}_\alpha = \frac{1}{2}(\tilde{n}^{\beta\alpha}\boldsymbol{\varphi}_{,\beta}\cdot\delta\boldsymbol{\varphi}_\alpha + \tilde{n}^{\beta\alpha}\boldsymbol{\varphi}_{,\alpha}\cdot\delta\boldsymbol{\varphi}_\beta) = \frac{1}{2}\tilde{n}^{\beta\alpha}\delta a_{\beta\alpha}$$

Infine, riconoscendo il termine $\tilde{q}^\alpha(\mathbf{t}\cdot\delta\boldsymbol{\varphi}_{,\alpha} + \boldsymbol{\varphi}_{,\alpha}\cdot\delta\mathbf{t}) = \tilde{q}^\alpha\delta\gamma_\alpha$, la (2.87) diventa:

$$\delta\mathcal{L}_{\text{int}} = \int_{S_0}\left(\frac{1}{2}\tilde{n}^{\alpha\beta}\delta a_{\alpha\beta} + \tilde{q}^\alpha\delta\gamma_\alpha + \tilde{m}^{\alpha\beta}\delta\kappa_{\alpha\beta}\right) dS_0$$

Seguendo [10] e [18], si sceglierà di scrivere la variazione virtuale del lavoro interno considerando solamente la parte simmetrica del momento $\tilde{m}^{(\alpha\beta)} = \frac{1}{2}(\tilde{m}^{\alpha\beta} + \tilde{m}^{\beta\alpha})$ e della variazione della misura di deformazione $\delta\kappa_{\alpha\beta}$, commentendo un errore trascurabile. Di conseguenza, la precedente diventa:

$$\delta\mathcal{L}_{\text{int}} = \int_{S_0}\left(\frac{1}{2}\tilde{n}^{\alpha\beta}\delta a_{\alpha\beta} + \tilde{q}^\alpha\delta\gamma_\alpha + \tilde{m}^{(\alpha\beta)}\delta\kappa_{(\alpha\beta)}\right) dS_0 \tag{2.88}$$

Componenti di sforzo efficaci

Richiamando dalla (2.62) la risultante di sforzo per unità di superficie iniziale $\mathbf{n}^\alpha$ e svolgendo i calcoli considerando l'espressione esplicita per i vettori della base covariante (2.25), essa risulta pari a:

$$\mathbf{n}^\alpha = \frac{1}{\bar{j}_0}\int_{-\frac{h}{2}}^{\frac{h}{2}} j\sigma^{i\alpha}\mathbf{g}_\alpha\, d\xi = \frac{1}{\bar{j}_0}\int_{-\frac{h}{2}}^{\frac{h}{2}} j\sigma^{\beta\alpha}(\boldsymbol{\varphi}_{,\alpha} + \xi\mathbf{t}_{,\alpha})\, d\xi + \frac{1}{\bar{j}_0}\int_{-\frac{h}{2}}^{\frac{h}{2}} j\sigma^{3\alpha}\mathbf{t}\, d\xi$$

Introducendo la (2.71) per $\mathbf{t}_{,\alpha}$ si ottiene:

$$\mathbf{n}^\alpha = \frac{1}{\bar{j}_0}\int_{-\frac{h}{2}}^{\frac{h}{2}} j\sigma^{\beta\alpha}(\boldsymbol{\varphi}_{,\alpha} + \xi(\lambda^\mu_\beta\boldsymbol{\varphi}_{,\mu} + \lambda^3_\beta\mathbf{t}))\, d\xi + \frac{1}{\bar{j}_0}\int_{-\frac{h}{2}}^{\frac{h}{2}} j\sigma^{3\alpha}\mathbf{t}\, d\xi$$

Per cui, infine, svolgendo i calcoli, rinominando opportunamente gli indici ed estraendo dagli integrali le grandezze vettoriali, tutte indipendenti dalla coordinata ξ, si ottiene:

$$\begin{aligned}\mathbf{n}^\alpha &= \Big[\frac{1}{\bar{j}_0}\int_{-\frac{h}{2}}^{\frac{h}{2}} j(\sigma^{\beta\alpha} + \xi\lambda^\beta_\mu\sigma^{\mu\alpha})\, d\xi\Big]\boldsymbol{\varphi}_{,\beta} + \Big[\frac{1}{\bar{j}_0}\int_{-\frac{h}{2}}^{\frac{h}{2}} j(\sigma^{3\alpha} + \xi\lambda^3_\mu\sigma^{\mu\alpha})\, d\xi\Big]\mathbf{t} = \\ &= n^{\beta\alpha}\boldsymbol{\varphi}_{,\beta} + q^\alpha\mathbf{t}\end{aligned} \tag{2.89}$$

avendo ricavato, in funzione di $\boldsymbol{\sigma}$, i termini $n^{\beta\alpha}$ e q^α, introdotti nella (2.71). Con passaggi analoghi a quanto svolto per ricavare la (2.89), la risultante di momento per unità di superficie inziale $\tilde{\mathbf{m}}^\alpha$, introdotta nella (2.66), diventa:

$$\begin{aligned}\tilde{\mathbf{m}}^\alpha = &\Big[\frac{1}{\bar{j}_0}\int_{-\frac{h}{2}}^{\frac{h}{2}} j(\xi\sigma^{\beta\alpha} + (\xi)^2\lambda^\beta_\mu\sigma^{\mu\alpha})\, d\xi\Big]\boldsymbol{\varphi}_{,\beta} + \\ &+ \Big[\frac{1}{\bar{j}_0}\int_{-\frac{h}{2}}^{\frac{h}{2}} j(\sigma^{3\alpha} + (\xi)^2\lambda^3_\mu\sigma^{\mu\alpha})\, d\xi\Big]\mathbf{t} = \tilde{m}^{\beta\alpha}\boldsymbol{\varphi}_{,\beta} + \tilde{m}^{3\alpha}\mathbf{t}\end{aligned} \tag{2.90}$$

avendo definito i termini $\tilde{m}^{\beta\alpha}$ e $\tilde{m}^{3\alpha}$ introdotti nella (2.71). In particolare, riordinando gli indici, $\tilde{m}^{\alpha\beta}$ risulta:

$$\tilde{m}^{\alpha\beta} = \frac{1}{\bar{j}_0}\int_{-\frac{h}{2}}^{\frac{h}{2}} j(\xi\sigma^{\alpha\beta} + (\xi)^2\lambda^\alpha_\mu\sigma^{\mu\beta})\, d\xi \tag{2.91}$$

Sostituendo le (2.89), (2.90) all'interno della (2.73), si ottengono le componenti $\tilde{n}^{\beta\alpha}$ in funzione di $\boldsymbol{\sigma}$. Accorpando tutto all'interno dell'integrale:

$$\tilde{n}^{\beta\alpha} = n^{\beta\alpha} - \lambda^\beta_\mu\tilde{m}^{\alpha\mu} = \frac{1}{\bar{j}_0}\int_{-\frac{h}{2}}^{\frac{h}{2}} j\big[\sigma^{\beta\alpha} + \xi\lambda^\beta_\mu\sigma^{\mu\alpha} - \lambda^\beta_\mu(\xi\sigma^{\alpha\mu} + (\xi)^2\lambda^\alpha_\gamma\sigma^{\gamma\mu})\big]\, d\xi$$

Rinominando opportunamente gli indici e sfruttando la simmetria delle componenti $\sigma^{\mu\alpha}$, si osserva che i termini moltiplicati per ξ si semplificano, risultando, infine:

$$\tilde{n}^{\beta\alpha} = \frac{1}{\bar{j}_0} \int_{-\frac{h}{2}}^{\frac{h}{2}} j(\sigma^{\beta\alpha} + (\xi)^2 \lambda^{\alpha}_{\gamma} \sigma^{\mu\gamma})\, d\xi \tag{2.92}$$

Analogamente, si calcola la componente $\tilde{q}^{\alpha}$:

$$\tilde{q}^{\alpha} = q^{\alpha} - \lambda^3_{\mu} \tilde{m}^{\alpha\mu} = \frac{1}{\bar{j}_0} \int_{-\frac{h}{2}}^{\frac{h}{2}} j\left[\sigma^{3\alpha} + \xi \lambda^3_{\mu} \sigma^{\mu\alpha} - \lambda^3_{\mu}(\xi \sigma^{\alpha\mu} + (\xi)^2 \lambda^{\alpha}_{\gamma} \sigma^{\gamma\mu})\right] d\xi$$

Di nuovo, sfruttando la simmetria delle componenti $\sigma^{\mu\alpha}$, i termini moltiplicati per ξ si semplificano, la precedente di conseguenza diventa:

$$\tilde{q}^{\alpha} = \frac{1}{\bar{j}_0} \int_{-\frac{h}{2}}^{\frac{h}{2}} j(\sigma^{3\alpha} - (\xi)^2 \lambda^3_{\mu} \lambda^{\alpha}_{\gamma} \sigma^{\gamma\mu})\, d\xi \tag{2.93}$$

Dopo semplici passaggi sfruttando le (2.29), (2.30), si verifica che le componenti $\tilde{n}^{\alpha\beta}$, $\tilde{m}^{\alpha\beta}$ e $\tilde{q}^{\alpha}$, espresse rispettivamente dalle (2.92), (2.91) e (2.93), possono essere dunque messe in relazione con le componenti del secondo tensore di Piola-Kirchhoff secondo la (2.14) e, dall'ipotesi di guscio sottile per la quale si trascurano i termini in $(\xi)^2$ e si approssima j^0 a $\bar{j^0}$ (indipendente da ξ), si ottiene:

$$\tilde{n}^{\alpha\beta} = \int_{-\frac{h}{2}}^{\frac{h}{2}} S^{\alpha\beta}\, d\xi \qquad \tilde{m}^{\alpha\beta} = \tilde{m}^{(\alpha\beta)} = \int_{-\frac{h}{2}}^{\frac{h}{2}} \xi S^{\alpha\beta}\, d\xi \qquad \tilde{q}^{\alpha} = \int_{-\frac{h}{2}}^{\frac{h}{2}} S^{3\alpha}\, d\xi \tag{2.94}$$

avendo scritto $m^{\alpha\beta} = m^{(\alpha\beta)}$, dalla simmetria delle componenti $S^{\alpha\beta}$.

2.4.2 Corrispondenza col la teoria del continuo mediante il principio dei lavori virtuali

In questa sezione si procede al calcolo della forma variazionale (2.83) a partire dal principio dei lavori virtuali nel continuo (2.19).

Inizialmente, si calcola la variazione del tensore gradiente di deformazione $\mathbf{F}$, ricordando la (2.28):

$$\delta\mathbf{F} = \delta(\mathbf{g}_i(\mathbf{\Phi}) \otimes \mathbf{G}^i) = (\delta\mathbf{g}_i) \otimes \mathbf{G}^i = (\delta\boldsymbol{\varphi}_{,\alpha} + \xi\delta\mathbf{t}_{,\alpha}) \otimes \mathbf{G}^{\alpha} + \delta\mathbf{t} \otimes \mathbf{G}^3 \tag{2.95}$$

A questo punto si può calcolare la variazione virtuale del lavoro interno, ricordando la (2.19):

$$\delta\mathcal{L}_{\text{int}} = \int_{\mathcal{C}_0} \mathbf{S} : \delta\mathbf{E}\, dV_0$$

L'integranda della precedente si può scrivere:

$$\mathbf{S} : \delta\mathbf{E} = \frac{1}{2}(\mathbf{S} \cdot \delta\mathbf{F}^T\mathbf{F} + \mathbf{S} \cdot \mathbf{F}^T\delta\mathbf{F})$$

Tenendo in considerazione la simmetria di $\mathbf{S}$:

$$\begin{aligned}\mathbf{S} : \delta\mathbf{E} &= \frac{1}{2}(\mathbf{F}\mathbf{S}^T \cdot \delta\mathbf{F} + \mathbf{F}\mathbf{S} \cdot \delta\mathbf{F}) = \mathbf{F}\mathbf{S} \cdot \delta\mathbf{F} = \\ &= \mathbf{F}\mathbf{S} \cdot ((\delta\boldsymbol{\varphi}_{,\alpha} + \xi\delta\mathbf{t}_{,\alpha}) \otimes \boldsymbol{\epsilon}^\alpha + \delta\mathbf{t} \otimes \boldsymbol{\epsilon}^3)(\nabla\boldsymbol{\Phi}_0^{-1}) = \\ &= \mathbf{F}(J\mathbf{F}^{-1}\boldsymbol{\sigma}\mathbf{F}^{-T}) \cdot ((\delta\boldsymbol{\varphi}_{,\alpha} + \xi\delta\mathbf{t}_{,\alpha}) \otimes \boldsymbol{\epsilon}^\alpha + \delta\mathbf{t} \otimes \boldsymbol{\epsilon}^3)(\nabla\boldsymbol{\Phi}_0^{-1})\end{aligned}$$

Avendo ricordato la relazione fra il secondo tensore di Piola-Kirchhoff ed il tensore di Cauchy (2.14). Dopo semplici passaggi algebrici, si verifica:

$$\begin{aligned}&\mathbf{F}(J\mathbf{F}^{-1}\boldsymbol{\sigma}\mathbf{F}^{-T}) \cdot ((\delta\boldsymbol{\varphi}_{,\alpha} + \xi\delta\mathbf{t}_{,\alpha}) \otimes \boldsymbol{\epsilon}^\alpha + \delta\mathbf{t} \otimes \boldsymbol{\epsilon}^3)(\nabla\boldsymbol{\Phi}_0^{-1}) = \\ &\quad = J\boldsymbol{\sigma}[\mathbf{F}^{-T}\mathbf{G}^\alpha \cdot (\delta\boldsymbol{\varphi}_{,\alpha} + \xi\delta\mathbf{t}_{,\alpha}) + \mathbf{F}^{-T}\mathbf{G}^3 \cdot \delta\mathbf{t}] = \\ &\quad = J\boldsymbol{\sigma}(\mathbf{g}^\alpha \cdot (\delta\boldsymbol{\varphi}_{,\alpha} + \xi\delta\mathbf{t}_{,\alpha}) + J\boldsymbol{\sigma}\mathbf{g}^3 \cdot \delta\mathbf{t}\end{aligned}$$

avendo sfruttato la (2.29) nell'ultimo passaggio. La variazione virtuale del lavoro interno può dunque scriversi:

$$\begin{aligned}\delta\mathcal{L}_{\text{int}} = \int_{\mathcal{C}_0} \mathbf{S} : \delta\mathbf{E}\, dV_0 = \int_{\mathcal{A}} \frac{1}{\bar{j}^0}\Big[\Big(\int_{-\frac{h}{2}}^{+\frac{h}{2}} j\boldsymbol{\sigma}\mathbf{g}^\alpha\, d\xi\Big) \cdot \boldsymbol{\varphi}_{,\alpha} + \\ + \Big(\int_{-\frac{h}{2}}^{+\frac{h}{2}} \xi j\boldsymbol{\sigma}\mathbf{g}^\alpha\, d\xi\Big) \cdot \mathbf{t}_{,\alpha} + \Big(\int_{-\frac{h}{2}}^{+\frac{h}{2}} \xi j\boldsymbol{\sigma}\mathbf{g}^3\, d\xi\Big) \cdot \delta\mathbf{t}\Big]\, \bar{j}^0 d\xi^1 d\xi^2\end{aligned}$$

Ricordando le (2.62) e (2.66), la precedente può riassumersi:

$$\delta\mathcal{L}_{\text{int}} = \int_{\mathcal{A}} (\mathbf{n}^\alpha \cdot \delta\boldsymbol{\varphi}_{,\alpha} + \tilde{\mathbf{m}}^\alpha \cdot \delta\mathbf{t}_{,\alpha} + \mathbf{l} \cdot \delta\mathbf{t})\, \bar{j}^0 d\xi^1 d\xi^2 \tag{2.96}$$

Analogamente, si può partire dalla variazione del lavoro esterno, ricordando la (2.17):

$$\delta\mathcal{L}_{\text{ext}} = \int_{\partial\mathcal{C}} \mathbf{f} \cdot \delta\boldsymbol{\Phi}\, dA + \int_{\mathcal{C}} \mathbf{b} \cdot \delta\boldsymbol{\Phi}\, dV \tag{2.97}$$

Il contributo delle forze di volume risulta:

$$\begin{aligned}&\int_{\mathcal{C}} \mathbf{b} \cdot \delta\boldsymbol{\Phi}\, dV = \\ &\quad = \int_{\mathcal{A}} \Big[\Big(\int_{-\frac{h}{2}}^{\frac{h}{2}} j\mathbf{b}\, d\xi + \Big) \cdot \delta\boldsymbol{\varphi}\Big]\, d\xi^1 d\xi^2 + \int_{\mathcal{A}} \Big[\Big(\int_{-\frac{h}{2}}^{\frac{h}{2}} \xi j\mathbf{b}\, d\xi\Big) \cdot \delta\mathbf{t}\Big]\, d\xi^1 d\xi^2\end{aligned}$$

Il contributo delle forze di superficie si può scrivere, ricordando la (2.9):

$$\int_{\partial\mathcal{C}} \mathbf{f}\cdot\delta\mathbf{\Phi}\,dA = \int_{\partial\mathcal{C}} \boldsymbol{\sigma}\mathbf{n}\cdot\delta\mathbf{\Phi}\,dA$$

avendo indicato con $\mathbf{n}$ la normale uscente dalla superficie esterna dello shell. Seguendo il procedimento descritto in [10], ricordando la formula di Nanson (2.7):

$$\mathbf{n}\,dA = j\nabla\mathbf{\Phi}^{-T}\boldsymbol{\mu}\,d\sigma \tag{2.98}$$

avendo indicato con $\boldsymbol{\mu} = \mu^i\boldsymbol{\epsilon}_i$ la normale uscente alla superficie esterna del dominio $\partial\mathcal{A}\times[-\frac{h}{2},\frac{h}{2}]\cup\mathcal{A}\times\{-\frac{h}{2},\frac{h}{2}\}$, e con $d\sigma$ il differenziale della superficie esterna. Spezzando l'integrale sui sottodomini, e indicando con $\mu_\alpha\boldsymbol{\epsilon}^\alpha$ la normale esterna al contorno $\partial\mathcal{A}$:

$$\begin{aligned}
\int_{\partial\mathcal{C}} \boldsymbol{\sigma}\mathbf{n}\cdot\delta\mathbf{\Phi}\,dA &= \int_{\partial\mathcal{A}}\Big[\Big(\int_{-\frac{h}{2}}^{\frac{h}{2}}(j\boldsymbol{\sigma}\nabla\mathbf{\Phi}^{-T}\mu_\alpha\boldsymbol{\epsilon}^\alpha)\,d\xi\Big)\cdot(\delta\boldsymbol{\varphi}+\xi\delta\mathbf{t})\Big]\,ds+\\
&+\int_{\mathcal{A}}\Big[(j\boldsymbol{\sigma}\nabla\mathbf{\Phi}^{-T}\boldsymbol{\epsilon}^3)|_{\xi=-\frac{h}{2}}^{\xi=\frac{h}{2}}\,d\xi^1d\xi^2\Big]\cdot(\delta\boldsymbol{\varphi}+\xi\delta\mathbf{t}) =\\
&=\int_{\partial\mathcal{A}}\Big[\Big(\mu_\alpha\int_{-\frac{h}{2}}^{\frac{h}{2}}(j\boldsymbol{\sigma}\mathbf{g}^\alpha)\,d\xi\Big)\cdot\delta\boldsymbol{\varphi}+\Big(\mu_\alpha\int_{-\frac{h}{2}}^{\frac{h}{2}}(\xi j\boldsymbol{\sigma}\mathbf{g}^\alpha)\,d\xi\Big)\cdot\delta\mathbf{t})\Big]\,ds+\\
&+\int_{\mathcal{A}}\Big[\Big((j\boldsymbol{\sigma}\mathbf{g}^3)|_{\xi=-\frac{h}{2}}^{\xi=\frac{h}{2}}\Big)\cdot\delta\boldsymbol{\varphi}+\Big((\xi\boldsymbol{\sigma}\mathbf{g}^3)|_{\xi=-\frac{h}{2}}^{\xi=\frac{h}{2}}\Big)\cdot\delta\mathbf{t}\Big]\,d\xi^1d\xi^2
\end{aligned}$$

Per il teorema di Green, e richiamando il teorema della divergenza per la superficie dello shell (2.54), si osserva:

$$\begin{aligned}
&\int_{\partial\mathcal{A}}\Big[\Big(\mu_\alpha\int_{-\frac{h}{2}}^{\frac{h}{2}}(j\boldsymbol{\sigma}\mathbf{g}^\alpha)\,d\xi\Big)\cdot\delta\boldsymbol{\varphi}\Big]\,ds=\int_{\partial\mathcal{A}}[(\bar{j}^0\mathbf{n}^\alpha\cdot\delta\boldsymbol{\varphi})\mu_\alpha]\,ds=\\
&=\int_{\mathcal{A}}(\bar{j}^0\mathbf{n}^\alpha\cdot\delta\boldsymbol{\varphi})_{,\alpha}\,d\xi^1d\xi^2=\int_{\mathcal{S}_0}\mathrm{Div}_{\mathcal{S}_0}(\mathbf{n}^\alpha\cdot\delta\boldsymbol{\varphi})\,dS_0=\int_{\Gamma_0}(\mathbf{n}^\alpha\cdot\delta\boldsymbol{\varphi})\nu_0^\alpha\,d\Gamma_0=\\
&=\int_{\Gamma_0}\bar{\bar{\mathbf{n}}}\cdot\delta\boldsymbol{\varphi}\,d\Gamma_0
\end{aligned}$$

Con passaggi analoghi, si verifica inoltre:

$$\int_{\partial\mathcal{A}}\Big[\Big(\mu_\alpha\int_{-\frac{h}{2}}^{\frac{h}{2}}(\xi j\boldsymbol{\sigma}\mathbf{g}^\alpha)\,d\xi\Big)\cdot\delta\mathbf{t})\Big]\,ds=\int_{\Gamma_0}\bar{\bar{\tilde{\mathbf{m}}}}\cdot\delta\mathbf{t}\,d\Gamma_0$$

Sommando il contributo delle forze di volume con quello delle forze di superficie, ricordando le (2.63), (2.67) e (2.82) la variazione del lavoro esterno risulta:

$$\delta\mathcal{L}_{\text{ext}} = \int_{\Gamma_0} (\bar{\bar{\mathbf{n}}} \cdot \delta\boldsymbol{\varphi} + \bar{\bar{\tilde{\mathbf{m}}}} \cdot \delta\mathbf{t})\, d\Gamma_0 + \int_{\mathcal{A}} (\bar{\mathbf{n}} \cdot \delta\boldsymbol{\varphi} + \bar{\tilde{\mathbf{m}}} \cdot \delta\mathbf{t})\, \bar{j}^0 d\xi^1 d\xi^2 \qquad (2.99)$$

Dalla differenza fra la (2.96) e la (2.99) si ottiene, dunque, la forma variazionale (2.83).

2.4.3 Linearizzazione della forma variazionale

Seguendo [10], indicando con $\boldsymbol{\Phi} = (\boldsymbol{\varphi}, \mathbf{t}) \in \mathcal{C}$ e $\Delta\boldsymbol{\Phi} = (\Delta\boldsymbol{\varphi}, \Delta\mathbf{t}) \in \mathrm{T}_{\boldsymbol{\Phi}}\mathcal{C}$, rispettivamente, la configurazione assunta dallo shell e le variazioni corrispondenti, si parametrizza $\boldsymbol{\Phi}$ secondo una variabile $\varepsilon \in \mathbb{R} \to \boldsymbol{\Phi}_\varepsilon = (\boldsymbol{\varphi}_\varepsilon, \mathbf{t}_\varepsilon)$, in modo da soddisfare i seguenti requisiti:

$$\boldsymbol{\Phi}_\varepsilon|_{\varepsilon=0} = \boldsymbol{\Phi} \qquad \left.\frac{d\boldsymbol{\Phi}_\varepsilon}{d\varepsilon}\right|_{\varepsilon=0} = \Delta\boldsymbol{\Phi} \qquad (2.100)$$

A tal proposito, per la parametrizzazione della superficie media e del vettore direttore si assume:

$$\boldsymbol{\varphi}_\varepsilon = \boldsymbol{\varphi} + \varepsilon\Delta\boldsymbol{\varphi} \qquad \mathbf{t}_\varepsilon = \exp_{\mathbf{t}}(\varepsilon\Delta\mathbf{t}) \qquad (2.101)$$

dove, con $\exp\colon \mathrm{T}_{\mathbf{t}}\mathrm{S}^2 \to \mathrm{S}^2$, si è introdotta la funzione *mappa esponenziale*, descritta a partire dal seguente capoverso. Si osserva inoltre che, dato che $\Delta\mathbf{t} \in \mathrm{T}_{\mathbf{t}}\mathrm{S}^2$, si può scrivere $\Delta\mathbf{t} = \Delta\boldsymbol{\theta} \times \mathbf{t}$.

Siano $\mathbf{x}_0 \in \mathbb{R}^3$ un vettore e $\mathbf{n} \in \mathbb{R}^3$ un vettore tale che $\|\mathbf{n}\| = 1$. Seguendo la procedura descritta in [7], in riferimento alla figura 2.4, il vettore $\mathbf{x} \in \mathbb{R}^3$, definito dalla rotazione di $\mathbf{x}_0 \in \mathbb{R}^3$ attorno all'asse definito da $\mathbf{n}$ per un angolo θ, risulta:

$$[\cos\theta\mathbf{1} + \sin\theta\hat{\mathbf{n}} + (1 - \cos\theta)\mathbf{n} \otimes \mathbf{n}]\mathbf{x} = \boldsymbol{\Lambda}\mathbf{x} \qquad (2.102)$$

avendo introdotto il tensore $\boldsymbol{\Lambda}$ ed il tensore $\hat{\mathbf{n}}$, di componenti cartesiane $\hat{n}^{ij}\mathbf{e}_i \otimes \mathbf{e}_j$, le quali sono raccolte nella seguente matrice $[\hat{\mathbf{n}}]$:

$$[\hat{\mathbf{n}}] = \begin{bmatrix} 0 & -n^3 & n^2 \\ n^3 & 0 & -n^1 \\ -n^2 & n^1 & 0 \end{bmatrix} \qquad (2.103)$$

dove n^i sono le componenti cartesiane del vettore $\mathbf{n} = n^i\mathbf{e}_i$. Si verifica facilmente che $\hat{\mathbf{n}}\mathbf{x} = \mathbf{n} \times \mathbf{x}$.

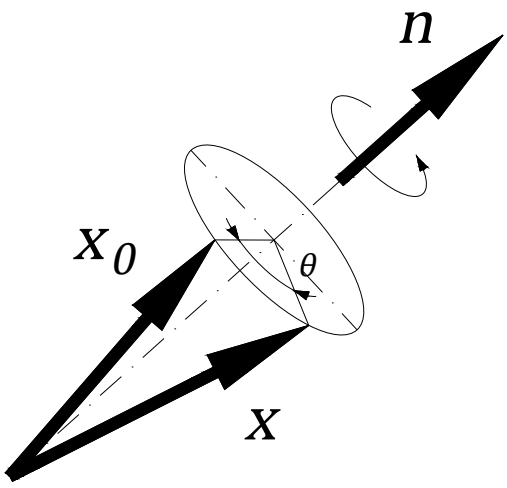


Figura 2.4: *Immagine fedele a quanto raffigurato in [7].*

Si osserva che, identificando la direzione $\mathbf{n}$ con il vettore $\frac{\varepsilon\Delta\mathbf{t}}{\|\varepsilon\Delta\mathbf{t}\|}$, l'angolo θ con $\|\varepsilon\Delta\mathbf{t}\|$ e il generico vettore $\mathbf{x}$ con $\mathbf{t}$, la (2.102) diventa la rotazione del direttore $\mathbf{t}$ lungo la direzione definita dal vettore $\varepsilon\Delta\mathbf{t}$ per un angolo pari a $\|\varepsilon\Delta\mathbf{t}\|$. Inoltre, dalla condizione $\Delta\mathbf{t}\cdot\mathbf{t}=0$, la (2.102) si semplifica:

$$\exp_{\mathbf{t}}(\varepsilon\Delta\mathbf{t}) = \cos\|\varepsilon\Delta\mathbf{t}\|\mathbf{t} + \frac{\sin\|\varepsilon\Delta\mathbf{t}\|}{\|\varepsilon\Delta\mathbf{t}\|}\varepsilon\Delta\mathbf{t} \tag{2.104}$$

Si fa riferimento alla figura 2.5.

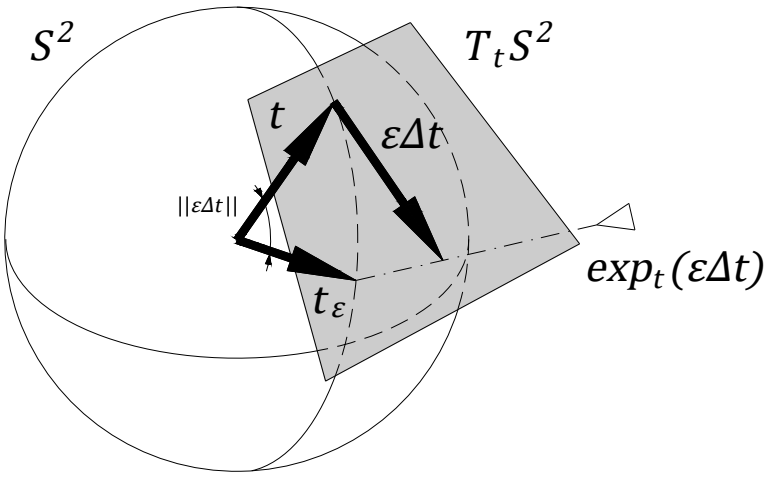


Figura 2.5: *Immagine fedele a quanto raffigurato in [10].*

Si verifica facilmente che:

$$\boldsymbol{\varphi}_\varepsilon\big|_{\varepsilon=0} = \boldsymbol{\varphi} \qquad \frac{d\boldsymbol{\varphi}_\varepsilon}{d\varepsilon}\Big|_{\varepsilon=0} = \Delta\boldsymbol{\varphi}$$
$$\mathbf{t}_\varepsilon\big|_{\varepsilon=0} = \mathbf{t} \qquad \frac{d\mathbf{t}_\varepsilon}{d\varepsilon}\Big|_{\varepsilon=0} = \Delta\mathbf{t}$$

che garantiscono le condizioni (2.100). Si verifica, inoltre:

$$\frac{d}{d\varepsilon}\boldsymbol{\varphi}_{\varepsilon,\alpha}\Big|_{\varepsilon=0} = \Delta\boldsymbol{\varphi}_{,\alpha} \qquad \frac{d}{d\varepsilon}\mathbf{t}_{\varepsilon,\alpha}\Big|_{\varepsilon=0} = \Delta\mathbf{t}_{,\alpha} \tag{2.105}$$

Seguendo il procedimento descritto in [10], si definisce il vettore direttore come la rotazione di $\mathbf{e}_3$ secondo un opportuno tensore di rotazione $\boldsymbol{\Lambda}$:

$$\mathbf{t} = \boldsymbol{\Lambda}\mathbf{e}_3 \tag{2.106}$$

Definendo con $\Delta\boldsymbol{\theta} = \boldsymbol{\Lambda}\Delta\boldsymbol{\Theta}$, si ha, ricordando la (1.32) e sfruttando la proprietà per la quale $\det\boldsymbol{\Lambda} = 1$ e $\boldsymbol{\Lambda}^{-1} = \boldsymbol{\Lambda}^T$:

$$\Delta\mathbf{t} = \Delta\boldsymbol{\theta} \times \mathbf{t} = \boldsymbol{\Lambda}\Delta\boldsymbol{\Theta} \times \boldsymbol{\Lambda}\mathbf{e}^3 = \boldsymbol{\Lambda}(\Delta\boldsymbol{\Theta} \times \mathbf{E}^3) = \boldsymbol{\Lambda}\Delta\mathbf{T} \tag{2.107}$$

avendo definito il vettore $\Delta\mathbf{T} = \Delta\boldsymbol{\Theta} \times \mathbf{E}^3$. Si osserva che, dalla condizione $\Delta\boldsymbol{\theta}\cdot\mathbf{t} = 0$ si ha, per la proprietà di trasposizione (1.18), $\boldsymbol{\Lambda}\Delta\boldsymbol{\Theta}\cdot\Lambda\mathbf{E}_3 = \Delta\boldsymbol{\Theta}\cdot(\boldsymbol{\Lambda}^T\boldsymbol{\Lambda})\mathbf{E_3} = \Delta\boldsymbol{\Theta}\cdot\mathbf{E_3} = 0$ e, di conseguenza, $\Delta\mathbf{T}\cdot\mathbf{E}_3 = \mathbf{E}_3\times\mathbf{E}_3\cdot\Delta\boldsymbol{\Theta} = 0$. Il vettore $\Delta\mathbf{T} = \Delta T^1\mathbf{E}_1 + \Delta T^2\mathbf{E}_2$, dunque, possiede solo due gradi di libertà.

In riferimento alla forma debole delle equazioni di equilibrio (2.80), si definisce la funzione $\mathcal{G}(\boldsymbol{\Phi}_\varepsilon, \delta\boldsymbol{\Phi})$, pari alla differenza fra la variazione di lavoro interno $\delta\mathcal{L}_{\text{int}}$ con la variazione del lavoro esterno $\delta\mathcal{L}_{\text{ext}}$ (ricordando che $\tilde{m}^{\alpha\beta} = \tilde{m}^{(\alpha\beta)}$):

$$\begin{aligned}\mathcal{G} = \mathcal{G}(\boldsymbol{\Phi}_\varepsilon, \delta\boldsymbol{\Phi}) &= \delta\mathcal{L}_{\text{int}}(\boldsymbol{\Phi}_\varepsilon, \delta\boldsymbol{\Phi}) - \delta\mathcal{L}_{\text{ext}}(\delta\boldsymbol{\Phi}) = \\ &= \int_{S_0}\Big(\frac{1}{2}\tilde{n}^{\alpha\beta}\delta a_{\alpha\beta} + \tilde{q}^\alpha\delta\gamma_\alpha + \tilde{m}^{\alpha\beta}\delta\kappa_{(\alpha\beta)}\Big)\,dS_0 + \\ &\quad - \Big(\int_{\Gamma_0}(\bar{\bar{\mathbf{n}}}\cdot\boldsymbol{\varphi} + \bar{\bar{\bar{\mathbf{m}}}}\cdot\mathbf{t})\,d\Gamma_0 + \int_{S_0}(\bar{\mathbf{n}}\cdot\delta\boldsymbol{\varphi} + \bar{\bar{\mathbf{m}}}\cdot\delta\mathbf{t})\,dS_0\Big) = 0\end{aligned} \tag{2.108}$$

la quale è un'equazione non lineare nella variabile $\boldsymbol{\Phi}_\varepsilon$. La precedente può essere linearizzata arrestando lo sviluppo al primo ordine, lungo la direzione $\Delta\boldsymbol{\Phi}$:

$$\mathcal{G} = \mathcal{G}(\boldsymbol{\Phi}_\varepsilon, \delta\boldsymbol{\Phi}) + D\mathcal{G}(\boldsymbol{\Phi}_\varepsilon, \delta\boldsymbol{\Phi})[\Delta\boldsymbol{\Phi}] + o(\|\Delta\boldsymbol{\Phi}\|) = 0 \tag{2.109}$$

Si calcola dunque la variazione di $\mathcal{G}$, a partire dalla configurazione $\boldsymbol{\Phi}_\varepsilon$ lungo la direzione $\Delta\boldsymbol{\Phi}$:

$$D\mathcal{G}(\boldsymbol{\Phi}_\varepsilon, \delta\boldsymbol{\Phi})[\Delta\boldsymbol{\Phi}] = \frac{d}{d\varepsilon}\big(\mathcal{G}(\boldsymbol{\Phi}_\varepsilon, \delta\boldsymbol{\Phi})\big)\big|_{\varepsilon=0} \tag{2.110}$$

Variazioni delle misure di deformazione e delle relative variazioni virtuali

Al fine del calcolo di (2.110), si determinano le *variazioni delle variabili di deformazione*. In riferimento a $a_{\alpha\beta}$, si ha, parametrizzando la superficie media ed il vettore direttore secondo le (2.100) e (2.101), si ha:

$$a_{\alpha\beta}(\boldsymbol{\Phi}_\varepsilon) = (\boldsymbol{\varphi}_\varepsilon)_{,\alpha} \cdot (\boldsymbol{\varphi}_\varepsilon)_{,\beta} \tag{2.111}$$

Per cui, la variazione della precedente risulta, nominando l'operatore $\Delta(\bullet)$ in modo analogo alla (2.15):

$$\begin{aligned} \Delta a_{\alpha\beta} &= \frac{d}{d\varepsilon}\big(a_{\alpha\beta}(\boldsymbol{\Phi}_\varepsilon)\big)\Big|_{\varepsilon=0} = \frac{d}{d\varepsilon}\big((\boldsymbol{\varphi}_\varepsilon)_{,\alpha} \cdot (\boldsymbol{\varphi}_\varepsilon)_{,\beta}\big)\Big|_{\varepsilon=0} = \\ &= \Big(\frac{d}{d\varepsilon}\boldsymbol{\varphi}_{\varepsilon,\alpha} \cdot \boldsymbol{\varphi}_{\varepsilon,\beta} + \boldsymbol{\varphi}_{\varepsilon,\alpha} \cdot \frac{d}{d\varepsilon}\boldsymbol{\varphi}_{\varepsilon,\beta}\Big)\Big|_{\varepsilon=0} = \Delta\boldsymbol{\varphi}_{,\alpha} \cdot \boldsymbol{\varphi}_{,\beta} + \boldsymbol{\varphi}_{,\alpha}\Delta\boldsymbol{\varphi}_{,\beta} \end{aligned} \tag{2.112}$$

Mentre, con passaggi analoghi, si verifica:

$$\begin{aligned} \Delta\gamma_\alpha &= \Delta\boldsymbol{\varphi}_{,\alpha} \cdot \mathbf{t} + \boldsymbol{\varphi}_{,\alpha} \cdot \Delta\mathbf{t} \\ \Delta\kappa_{(\alpha\beta)} &= \frac{1}{2}(\Delta\boldsymbol{\varphi}_{,\alpha} \cdot \mathbf{t}_{,\beta} + \boldsymbol{\varphi}_{,\alpha} \cdot \Delta\mathbf{t}_{,\beta} + \Delta\boldsymbol{\varphi}_{,\beta} \cdot \mathbf{t}_{,\alpha} + \boldsymbol{\varphi}_{,\beta} \cdot \Delta\mathbf{t}_{,\alpha}) \end{aligned} \tag{2.113}$$

Si procede ora al calcolo delle variazioni delle variazioni virtuali delle variazioni delle variabili di deformazione introdotte nella (2.37). Si verifica che esse risultano:

$$\begin{aligned} \Delta\delta a_{\alpha\beta} &= \delta\boldsymbol{\varphi}_\alpha \cdot \Delta\boldsymbol{\varphi}_{,\beta} + \Delta\boldsymbol{\varphi}_{,\alpha} \cdot \delta\boldsymbol{\varphi}_\beta \\ \Delta\delta\gamma_\alpha &= \Delta\boldsymbol{\varphi}_{,\alpha} \cdot \delta\mathbf{t} + \delta\boldsymbol{\varphi}_{,\alpha} \cdot \Delta\mathbf{t} + \delta\boldsymbol{\varphi}_{,\alpha} \cdot \Delta\delta\mathbf{t} \\ \Delta\delta\kappa_{(\alpha\beta)} &= \frac{1}{2}(\Delta\boldsymbol{\varphi}_{,\alpha} \cdot \delta\mathbf{t}_{,\beta} + \delta\boldsymbol{\varphi}_{,\alpha} \cdot \Delta\mathbf{t}_{,\beta} + \delta\boldsymbol{\varphi}_{,\alpha} \cdot \Delta\delta\mathbf{t}_{,\beta} + \\ &\qquad + \Delta\boldsymbol{\varphi}_{,\beta} \cdot \delta\mathbf{t}_{,\alpha} + \delta\boldsymbol{\varphi}_{,\beta} \cdot \Delta\mathbf{t}_{,\alpha} + \delta\boldsymbol{\varphi}_{,\beta} \cdot \Delta\delta\mathbf{t}_{,\alpha}) \end{aligned} \tag{2.114}$$

Si osserva che $\Delta\delta\mathbf{t}_\varepsilon \neq \mathbf{0}$, infatti, dalla definizione per la quale $\delta\mathbf{t}_\varepsilon = \delta\boldsymbol{\theta} \times \mathbf{t}_\varepsilon$ per un opportuno $\delta\boldsymbol{\theta}$:

$$\begin{aligned} \Delta\delta\mathbf{t}_\varepsilon &= \frac{d}{d\varepsilon}\big(\delta\boldsymbol{\theta} \times \mathbf{t}_\varepsilon\big)\Big|_{\varepsilon=0} = \delta\boldsymbol{\theta} \times \frac{d}{d\varepsilon}\mathbf{t}_\varepsilon\Big|_{\varepsilon=0} = \\ &= \delta\boldsymbol{\theta} \times \Delta\mathbf{t} = \delta\mathbf{t} \times (\Delta\mathbf{t} \times \mathbf{t}) = -\mathbf{t}(\delta\mathbf{t} \cdot \Delta\mathbf{t}) \end{aligned}$$

avendo negli ultimi due passaggi sfruttato le proprietà del prodotto triplo e del prodotto misto.

Variazione della forma variazionale
Calcolate le variazioni delle misure di deformazione, è possibile calcolare la variazione della forma variazionale (2.110).

$$D\mathcal{G}(\mathbf{\Phi}_\varepsilon, \delta\mathbf{\Phi})[\Delta\mathbf{\Phi}] = \frac{d}{d\varepsilon}\big(\mathcal{G}(\mathbf{\Phi}_\varepsilon)\big)\big|_{\varepsilon=0}$$

Secondo sufficienti ipotesi di regolarità, si porta la derivata all'interno dell'integrale:

$$\begin{aligned}
&\frac{d}{d\varepsilon}\big(\mathcal{G}(\mathbf{\Phi}_\varepsilon, \delta\mathbf{\Phi})\big)\Big|_{\varepsilon=0} = \\
&\quad= \frac{d}{d\varepsilon}\Big(\int_{S_0}\Big(\frac{1}{2}\tilde{n}^{\alpha\beta}\delta a_{\alpha\beta} + \tilde{q}^{\alpha}\delta\gamma_{\alpha} + \tilde{m}^{\alpha\beta}\delta\kappa_{(\alpha\beta)}\Big)\,dS_0\Big)\Big|_{\varepsilon=0} = \\
&\quad= \int_{\mathcal{S}_0}\Big(\frac{1}{2}\frac{d}{d\varepsilon}\tilde{n}^{\alpha\beta}\Big|_{\varepsilon=0}\delta a_{\alpha\beta} + \frac{d}{d\varepsilon}\tilde{q}^{\alpha}\Big|_{\varepsilon=0}\delta\gamma^{\alpha} + \frac{d}{d\varepsilon}\tilde{m}^{\alpha\beta}\Big|_{\varepsilon=0}\delta\kappa_{(\alpha\beta)}\Big)\,dS_0 + \\
&\qquad+ \int_{\mathcal{S}_0}\Big(\frac{1}{2}\tilde{n}^{\alpha\beta}\frac{d}{d\varepsilon}\delta a_{\alpha\beta}\Big|_{\varepsilon=0} + \tilde{q}^{\alpha}\frac{d}{d\varepsilon}\delta\gamma^{\alpha}\Big|_{\varepsilon=0} + \tilde{m}^{\alpha\beta}\frac{d}{d\varepsilon}\delta\kappa_{(\alpha\beta)}\Big|_{\varepsilon=0}\Big)\,dS_0 = \\
&\quad= \int_{\mathcal{S}_0}\Big(\frac{1}{2}\Delta\tilde{n}^{\alpha\beta}\delta a_{\alpha\beta} + \Delta\tilde{q}^{\alpha}\delta\gamma^{\alpha} + \Delta\tilde{m}^{\alpha\beta}\delta\kappa_{(\alpha\beta)}\Big)\,dS_0 + \\
&\qquad+ \int_{\mathcal{S}_0}\Big(\frac{1}{2}\tilde{n}^{\alpha\beta}\Delta\delta a_{\alpha\beta} + \tilde{q}^{\alpha}\Delta\delta\gamma^{\alpha} + \tilde{m}^{\alpha\beta}\Delta\delta\kappa_{(\alpha\beta)}\Big)\,dS_0 = \\
&\quad= D_M\mathcal{G}(\mathbf{\Phi}_\varepsilon, \delta\mathbf{\Phi})[\Delta\mathbf{\Phi}] + D_G\mathcal{G}(\mathbf{\Phi}_\varepsilon, \delta\mathbf{\Phi})[\Delta\mathbf{\Phi}]
\end{aligned} \tag{2.115}$$

avendo definito la parte *materiale* e *geometrica* della variazione.

2.5 Legame costitutivo elastico lineare omogeneo e isotropo

Richiamando il legame costitutivo elastico, lineare, omogeneo e isotropo introdotto dalle (2.21), (2.22), sostituendo la (1.23) per $\mathbf{1}$, la (1.39) per $\mathbb{I}$ e ricordando l'operazione di trasposizione per i tensori del quarto ordine definita dalla (1.37), si può scrivere il tensore $\mathbb{H}$ secondo la base naturale covariante dello shell nella configurazione inziale:

$$\mathbb{H} = \Big(\frac{\nu E}{(1+\nu)(1-2\nu)}G^{ij}G^{kl} + \frac{E}{2(1+\nu)}(G^{ik}G^{jl} + G^{il}G^{jk})\Big)\mathbf{G}_i\otimes\mathbf{G}_j\otimes\mathbf{G}_k\otimes\mathbf{G}_l \tag{2.116}$$

avendo introdotto le componenti H^{ijkl}, le quali, si verifica facilmente, sono soggette alle seguenti simmetrie:

$$H^{ijkl} = H^{jikl} = H^{ijlk} = H^{klij} \tag{2.117}$$

Dalle condizioni $G_{\alpha 3} = G_{3\alpha} = 0$ e $G_{33} = 1$, si ottiene che la matrice $[\mathbf{G}]$, di componenti $[\mathbf{G}]_{ij} = G_{ij}$ risulta nulla nelle entrate 13, 23, 31 e 32, mentre l'entrata 33 risulta pari a 1. Essendo la matrice $[\mathbf{G}^*]$, di componenti $[\mathbf{G}^*]_{ij} = G^{ij}$, pari a $[\mathbf{G}]^{-1}$ per la (1.10), anche le entrate 13, 23, 31, 32 della matrice $[\mathbf{G}^*]$ sono nulle, mentre l'entrata 33 risulta pari a 1. Da queste considerazioni, svolgendo i calcoli secondo la (2.116), si ottiene:

$$H^{\alpha\beta 3\gamma} = H^{3\alpha\gamma\delta} = H^{3\alpha 33} = 0 \tag{2.118}$$

risultato che si estende considerando anche le simmetrie delle componenti (2.117). Nello studio dei gusci sottili, per ottenere misure di tensione più realistiche, è di norma forzare l'anullamento della componente S^{33} del secondo tensore di Piola-Kirchhoff:

$$\begin{aligned} S^{33} = H^{33kl}E_{kl} &= H^{33\alpha\beta}E_{\alpha\beta} + H^{333\alpha}E_{3\alpha} + H^{33\alpha 3}E_{\alpha 3} + H^{3333}E_{33} = \\ &= H^{33\alpha\beta}E_{\alpha\beta} + H^{3333}E_{33} = 0 \end{aligned}$$

avendo considerato la (2.118) con le relative simmetrie. La precedente si traduce nell'imporre, ai fini del calcolo del legame costitutivo:

$$E_{33} = -\frac{H^{33\alpha\beta}}{H^{3333}} \tag{2.119}$$

Di conseguenza:

$$\begin{aligned} S^{\alpha\beta} &= H^{\alpha\beta\gamma\delta}E_{\gamma\delta} + H^{\alpha\beta 3\gamma}E_{3\gamma} + H^{\alpha\beta\gamma 3}E_{\gamma 3} + H^{\alpha\beta 33}E_{33} = \\ &\quad = \Big(H^{\alpha\beta\gamma\delta} - \frac{H^{\alpha\beta 33}H^{33\gamma\delta}}{H^{3333}}\Big)E_{\gamma\delta} \\ S^{3\alpha} &= H^{3\alpha\gamma\delta}E_{\gamma\delta} + H^{3\alpha 3\gamma}E_{3\gamma} + H^{3\alpha\gamma 3}E_{\gamma 3} + H^{3\alpha 33}E_{33} = 2H^{3\alpha 3\gamma}E_{3\gamma} \\ S^{33} &= 0 \end{aligned} \tag{2.120}$$

Le componenti del tensore $\mathbf{E}$ risultano, ricordando la (2.39):

$$E_{\alpha\beta} = \varepsilon_{\alpha\beta} + \xi\chi_{(\alpha\beta)} \qquad E_{3\alpha} = \frac{1}{2}\rho_\alpha \qquad E_{33} = 0 \tag{2.121}$$

Dall'ipotesi di guscio sottile si possono approssimare i prodotti G^{ij}, presenti nella (2.116), con $a_0^{ij} = \mathbf{a}_0^i \cdot \mathbf{a}_0^j$. Sostituendo le precedenti alla (2.120) e svolgendo i calcoli, si ottiene:

$$\begin{aligned} S^{\alpha\beta} &= \frac{E}{1-\nu^2}\Big(\nu a_0^{\alpha\beta}a_0^{\gamma\delta} + \frac{1-\nu}{2}(a_0^{\alpha\gamma}a_0^{\beta\gamma} + a_0^{\alpha\delta}a_0^{\beta\gamma})\Big)(\varepsilon_{\gamma\delta} + \xi\chi_{(\gamma\delta)}) \\ S^{3\alpha} &= \frac{E}{1-\nu^2}\frac{1-\nu}{2}a_0^{\alpha\gamma}\rho_\gamma \\ S^{33} &= 0 \end{aligned} \tag{2.122}$$

Sostituendo alla (2.94) le (2.122):

$$\begin{aligned}
\tilde{n}^{\alpha\beta} &= \frac{Eh}{1-\nu^2}\Big(\nu a_0^{\alpha\beta}a_0^{\gamma\delta} + \frac{1-\nu}{2}(a_0^{\alpha\gamma}a_0^{\beta\gamma} + a_0^{\alpha\delta}a_0^{\beta\gamma})\Big)\varepsilon_{\gamma\delta} \\
\tilde{m}^{\alpha\beta} &= \frac{Eh^3}{12(1-\nu^2)}\Big(\nu a_0^{\alpha\beta}a_0^{\gamma\delta} + \frac{1-\nu}{2}(a_0^{\alpha\gamma}a_0^{\beta\gamma} + a_0^{\alpha\delta}a_0^{\beta\gamma})\Big)\chi_{(\gamma\delta)} \qquad (2.123) \\
\tilde{q}^{\alpha} &= \frac{\kappa E}{1-\nu^2}\frac{1-\nu}{2}a_0^{\alpha\gamma}\rho_\gamma
\end{aligned}$$

avendo fatto uso delle relazioni:

$$\int_{-\frac{h}{2}}^{\frac{h}{2}} d\xi = h \qquad \int_{-\frac{h}{2}}^{\frac{h}{2}} \xi\, d\xi = 0 \qquad \int_{-\frac{h}{2}}^{\frac{h}{2}} \xi^2\, d\xi = \frac{h^3}{12}$$

ed avendo introdotto il termine κ, solitamente pari a $5/6$, nell'espressione di $\tilde{q}^{\alpha}$, dalla classica teoria dei gusci e delle piastre sottili.

Capitolo 3

Formulazione agli elementi finiti e descrizione dell'algoritmo risolutivo

In questo capitolo vengono descritti gli aspetti numerici più significativi implementati nella presente tesi. Inizialmente, viene presentato l'elemento finito considerato, citando inoltre alcuni aspetti fondamentali relativi all'integrazione di Gauss. Successivamente, viene descritta la procedura di interpolazione dei gradi di libertà relativi alla superficie media $\boldsymbol{\varphi}$ e del vettore direttore $\mathbf{t}$, presentando inoltre la mappa esponenziale per considerare in modo efficace le rotazioni finite. Infine, viene descritto l'algoritmo non lineare di soluzione della forma variazionale (2.108) per il quale, a partire da una generica configurazione $\boldsymbol{\Phi}^{(k)}$, calcolando la forma variazionale linearizzata (2.109) lungo la direzione $\Delta\boldsymbol{\Phi}^{(k)}$ ed imponendola pari a zero, si ottiene:

$$\mathcal{G}^{(k)} = \mathcal{G}(\boldsymbol{\Phi}^{(k)}, \delta\boldsymbol{\Phi}^{(k)}) + D\mathcal{G}(\boldsymbol{\Phi}^{(k)}, \delta\boldsymbol{\Phi}^{(k)})[\Delta\boldsymbol{\Phi}^{(k)}] + o(\|\Delta\boldsymbol{\Phi}^{(k)}\|) = 0 \tag{3.1}$$

Mediante il procedimento descritto nel presente capitolo, la precendente può essere invertita per ottenere l'incremento $\Delta\boldsymbol{\Phi}^{(k)}$, trascurando $o(\|\Delta\boldsymbol{\Phi}^{(k)}\|)$:

$$\Delta\boldsymbol{\Phi}^{(k)} = -(D\mathcal{G}(\boldsymbol{\Phi}^{(k)}, \delta\boldsymbol{\Phi}^{(k)})[\Delta\boldsymbol{\Phi}^{(k)}])^{-1}\mathcal{G}(\boldsymbol{\Phi}^{(k)}, \delta\boldsymbol{\Phi}^{(k)}) \tag{3.2}$$

per poi aggiornare la configurazione all'iterazione successiva:

$$\boldsymbol{\Phi}^{(k+1)} = \text{update}(\boldsymbol{\Phi}^{(k)}, \Delta\boldsymbol{\Phi}^{(k)}) \tag{3.3}$$

Ripetendo il processo, si itera fino a convergenza.

3.1 Elemento finito bilineare a quattro nodi

Seguendo la procedura descritta in [11], si definisce il dominio $\square$ nello spazio delle coordinate (ξ^1, ξ^2):

$$\square := \{(\xi^1, \xi^2) \in [-1, 1] \times [-1, 1]\} \tag{3.4}$$

Inoltre, si definiscono *le funzioni di forma* $N^I : \square \to \mathbb{R}$, con $I = 1, \ldots, 4$:

$$N^I(\xi^1, \xi^2) = \frac{1}{4}(1 + \xi^1 \xi^1_I)(1 + \xi^2 \xi^2_I) \tag{3.5}$$

le quali, in riferimento alla funzione N^I, sono pari a 1 quando $(\xi^1, \xi^2) = (\xi^1_I, \xi^2_I)$, e pari a zero quando $(\xi^1, \xi^2) = (\xi^1_J, \xi^2_J)$ con $J \neq I$.

Interpolazione della superficie media
In riferimento alla figura 3.1, seguendo la classica trattazione degli elementi finiti isoparametrici, ogni elemento finito approssima una porzione della superficie media dello shell alla generica iterazione k (per la quale con $k = 0$ si intende la configurazione iniziale) definita da quattro vertici $\boldsymbol{\varphi}^{(k)}_{I=1,\ldots,4}$, che definiscono i quattro nodi afferenti all'elemento. Il numero totale di nodi ed elementi presenti nella mesh è indicato, rispettivamente, con $\mathrm{N_N}$ e $\mathrm{N_E}$. Nella numerazione totale, i nodi afferenti all'elemento sono inidicati con I, J, K e L e corrispondono, rispettivamente, al nodo 1, 2, 3, e 4 nel sistema di riferimento $\square$.

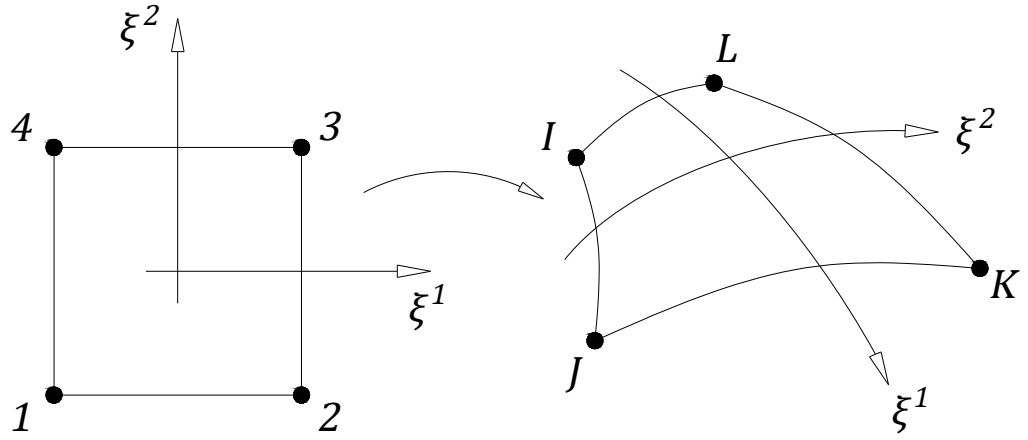


Figura 3.1

Il vettore posizione ristretto all'elemento finito risulta, dunque, una funzione $\boldsymbol{\varphi}^{(k)} \colon \square \to \mathbb{R}^3$:

$$\boldsymbol{\varphi}^{(k)} = \sum_{I=1}^{4} N^I(\xi^1, \xi^2)\boldsymbol{\varphi}_I^{(k)} \tag{3.6}$$

e la corrispondente derivata lungo la generica variabile ξ^α risulta:

$$\boldsymbol{\varphi}_{,\alpha}^{(k)} = \sum_{I=1}^{4} N_{,\alpha}^I(\xi^1, \xi^2)\boldsymbol{\varphi}_I^{(k)} \tag{3.7}$$

Anche la variazione all'iterazione k e la variazione virtuale della superficie media, sempre ristrette al singolo elemento finito, vengono approssimate seguendo lo stesso schema:

$$\begin{aligned} \Delta\boldsymbol{\varphi}^{(k)} &= \sum_{I=1}^{4} N^I(\xi^1, \xi^2)\Delta\boldsymbol{\varphi}_I^{(k)} \\ \delta\boldsymbol{\varphi} &= \sum_{I=1}^{4} N^I(\xi^1, \xi^2)\delta\boldsymbol{\varphi}_I \end{aligned} \tag{3.8}$$

Ad ogni iterazione, dunque, il valore della superficie media nella configurazione deformata di ogni nodo viene aggiornato semplicemente sommando al valore della superficie media all'iterazione precedente la variazione corrispondente, la quale è ottenuta, in ogni nodo, dal procedimento inteso dalla (3.3):

$$\boldsymbol{\varphi}_I^{(k+1)} = \boldsymbol{\varphi}_I^{(k)} + \Delta\boldsymbol{\varphi}_I^{(k)} \tag{3.9}$$

Interpolazione del vettore direttore ed aggiornamento nei nodi
Dalla condizione per la quale $\|\mathbf{t}^{(k)}\| = 1$, lo schema per l'interpolazione del vettore direttore risulta più complesso rispetto a quello per la superficie media. Inizialmente, sono noti i valori del vettore direttore nei nodi della mesh $\mathbf{t}_I^0$, i quali corrispondono alla normale alla superficie media nella configurazione iniziale della superficie della quale si svole svolgere l'analisi (nella presente formulazione, sono esclusi gli studi di superfici non differenziabili, come, ad esempio, le superfici che presentano spigoli per i quali non è possibile definire la normale; per una formulazione che tiene in considerazione questa eventualità si può fare riferimento a [16]).

Per ogni nodo I della mesh è possibile, dunque, calcolare il tensore di rotazione $\boldsymbol{\Lambda}_I^0$ che ruota il vettore cartesiano $\mathbf{E}_3$ in $\mathbf{t}_I^0$. Richiamando la mappa esponenziale (2.5), si ha:

$$\boldsymbol{\Lambda}_I^0 = (\mathbf{E}_3 \cdot \mathbf{t}_I^0)\mathbf{1} + \widehat{(\mathbf{E}_3 \times \mathbf{t}_I^0)} + \frac{1}{1 + \mathbf{E}_3 \cdot \mathbf{t}_I^0}(\mathbf{E}_3 \times \mathbf{t}_I^0) \otimes (\mathbf{E}_3 \times \mathbf{t}_I^0) \tag{3.10}$$

L'interpolazione del vettore direttore nella configurazione iniziale su ogni elemento finito risulta:

$$\mathbf{t}^0 = \frac{\sum_{I=1}^4 N^I(\xi^1,\xi^2)\mathbf{t}_I^0}{\|\sum_{I=1}^4 N^I(\xi^1,\xi^2)\mathbf{t}_I^0\|} = \frac{\tilde{\mathbf{t}}^0}{\|\tilde{\mathbf{t}}^0\|} \tag{3.11}$$

avendo definito il vettore $\tilde{\mathbf{t}}^0$. La derivata di $\mathbf{t}^0$ rispetto alla generica coordinata ξ^α, utilizzando la regola della catena per la quale si ha $\frac{\partial\|\tilde{\mathbf{t}}^0\|}{\partial\xi^\alpha} = \frac{1}{\|\tilde{\mathbf{t}}^0\|}(\tilde{\mathbf{t}}^0 \cdot \tilde{\mathbf{t}}^0_{,\alpha}) = \mathbf{t}^0 \cdot \tilde{\mathbf{t}}^0_{,\alpha}$, si verifica facilmente che è pari a:

$$\mathbf{t}^0_{,\alpha} = \frac{1}{\|\tilde{\mathbf{t}}^0\|}[\tilde{\mathbf{t}}^0_{,\alpha} - (\mathbf{t}^0 \cdot \tilde{\mathbf{t}}^0_{,\alpha})\mathbf{t}^0] \tag{3.12}$$

Ad ogni iterazione, il vettore $\Delta\mathbf{T}_I^{(k)}$ è ottenuto nei nodi dal procedimento inteso dalla (3.3). A questo punto, è possibile ottenere il vettore $\Delta\mathbf{t}_I^{(k)}$ attraverso il tensore $\mathbf{\Lambda}_I^{(k)}$ secondo la relazione (2.107). Dalla mappa esponenziale, successivamente, si ottengono i valori di $\mathbf{t}_I^{(k+1)}$ nei nodi all'iterazione successiva:

$$\mathbf{t}_I^{(k+1)} = \cos\|\Delta\mathbf{t}_I^{(k)}\| + \frac{\sin\|\Delta\mathbf{t}_I^{(k)}\|}{\|\Delta\mathbf{t}_I^{(k)}\|}\Delta\mathbf{t}_I^{(k)} = \Delta\mathbf{\Lambda}_I^{(k)}\mathbf{t}_I^{(k)} \tag{3.13}$$

avendo introdotto, per ogni nodo della mesh, il tensore $\Delta\mathbf{\Lambda}_I^{(k)}$:

$$\begin{aligned}\Delta\mathbf{\Lambda}_I^{(k)} = &\cos\|\Delta\mathbf{t}_I^{(k)}\|\mathbf{1} + \frac{\sin\|\Delta\mathbf{t}_I^{(k)}\|}{\|\Delta\mathbf{t}_I^{(k)}\|}(\mathbf{t}_I^{(k)} \times \Delta\mathbf{t}_I^{(k)}) + \\ &+ \frac{1-\cos\|\Delta\mathbf{t}_I^{(k)}\|}{\|\Delta\mathbf{t}_I^{(k)}\|^2}(\mathbf{t}_I^{(k)} \times \Delta\mathbf{t}_I^{(k)}) \otimes (\mathbf{t}_I^{(k)} \times \Delta\mathbf{t}_I^{(k)})\end{aligned} \tag{3.14}$$

Ricordando la relazione (2.106), dunque, il tensore $\mathbf{\Lambda}_I^{(k+1)}$, associato a $\mathbf{t}_I^{(k+1)}$, risulta:

$$\mathbf{\Lambda}_I^{(k+1)} = \Delta\mathbf{\Lambda}_I^{(k)}\mathbf{\Lambda}_I^{(k)} \tag{3.15}$$

Si osserva che la relazione (3.13) presenta una singolarità per $\|\Delta\mathbf{t}_I^{(k)}\| = 0$. Tuttavia, dalla doppia implicazione per la quale $\|\Delta\mathbf{t}_I^{(k)}\| = 0 \iff \Delta\mathbf{t}_I^{(k)} = \mathbf{0}$, per $\|\Delta\mathbf{t}_I^{(k)}\| = 0$ il vettore $\mathbf{t}_I^{(k)}$ non subisce alcuna rotazione e si ha, semplicemente, $\mathbf{t}_I^{(k+1)} = \mathbf{t}_I^{(k)}$.

Aggiornamento del vettore direttore nei punti Gauss

Come definito in [12], mentre lo schema di aggiornamento del vettore direttore nei nodi può essere definito canonico, diversi sono gli schemi di interpolazione e di aggiornamento del direttore all'interno dell'elemento nelle iterazioni successive a quella iniziale. In particolare, ad ogni iterazione, viene calcolato $\mathbf{t}^{(k)}$ in corrispondenza di specifici punti all'interno dell'elemento, definiti *punti Gauss*, descritti nel prossimo paragrafo, a partire dall'interpolazione di $\Delta\mathbf{t}^{(k)}$, $\Delta\boldsymbol{\theta}^{(k)}$ o $\Delta\mathbf{T}^{(k)}$. Sempre in riferimento a [12], nella presente tesi è stata sviluppata ed implementata la *continuum consistent interpolation*, per la quale, alla generica iterazione k, la quantità interpolata è la variazione del vettore direttore:

$$\Delta\mathbf{t}^{(k)} = \sum_{I=1}^{4} N^I(\xi^1, \xi^2)\Delta\mathbf{t}_I^{(k)} \tag{3.16}$$

L'aggiormento del direttore nei punti Gauss è sempre svolto secondo la mappa esponenziale:

$$\mathbf{t}^{(k+1)} = \cos\|\Delta\mathbf{t}^{(k)}\|\mathbf{t}^{(k)} + \frac{\sin\|\Delta\mathbf{t}^{(k)}\|}{\|\Delta\mathbf{t}^{(k)}\|}\Delta\mathbf{t}^{(k)} \tag{3.17}$$

La derivata rispetto alla generica coordinata ξ^α del vettore direttore nel punto Gauss, si verifica che risulta, dalla precedente:

$$\begin{aligned}\mathbf{t}_{,\alpha}^{(k+1)} = &\cos\|\Delta\mathbf{t}^{(k)}\|\mathbf{t}_{,\alpha}^{(k)} + \Big[\frac{\sin\|\Delta\mathbf{t}^{(k)}\|}{\|\Delta\mathbf{t}^{(k)}\|}(\mathbf{1} - \mathbf{t}^{(k)}\otimes\mathbf{t}^{(k)}) + \\ &+ \frac{1}{\|\Delta\mathbf{t}^{(k)}\|^2}\Big(\cos\|\Delta\mathbf{t}^{(k)}\| - \frac{\sin\|\Delta\mathbf{t}^{(k)}\|}{\|\Delta\mathbf{t}^{(k)}\|}\Big)\Delta\mathbf{t}^{(k)}\otimes\Delta\mathbf{t}^{(k)}\Big]\Delta\mathbf{t}_{,\alpha}^{(k)}\end{aligned} \tag{3.18}$$

Come citato in [12], considerando l'interpolazione (3.16), si verifica che, all'interno dell'elemento, la condizione $\mathbf{t}^{(k)}\cdot\Delta\mathbf{t}^{(k)} = 0$ non è generalmente soddisfatta; questo porta ad ottenere, generalmente, $\|\mathbf{t}^{(k+1)}\| \neq 1$ e $\mathbf{t}^{(k+1)}\cdot\mathbf{t}_{,\alpha}^{(k+1)} \neq 0$. Tuttavia, sempre citando [12], si verifica, dai confronti con altre formulazioni, che questo non porta una perdità di accuratezza nella convergenza della soluzione.

Seguendo l'interpolazione (3.16), la variazione virtuale del vettore direttore è interpolata in modo analogo:

$$\delta\mathbf{t} = \sum_{I=1}^{4} N^I(\xi^1, \xi^2)\delta\mathbf{t}_I \tag{3.19}$$

Integrazione di Gauss

Seguendo [6], si presentano gli aspetti più significativi per la presenti tesi dell'integrazione di Gauss. Sia $f(x)\colon [a,b] \to \mathbb{R}$. Al fine di calcolare l'integrale di

$f(x)$ nell'intervallo $[a, b]$, l'idea alla base dell'integrazione di Gauss è quella di sostituire ad $f(x)$ un polinomio di grado $2n+1$ interpolante $n+1$ punti d'appoggio appartenenti ad $[a, b]$. Volendo costruire una struttura di integrazione indipendente dall'intervallo $[a, b]$, definendo il cambio di variabili:

$$x(t) = \frac{2}{b-a}\Big(x - \frac{a+b}{2}\Big) \tag{3.20}$$

i punti di interpolazione del polinomio risultano $x(t_A)$, dove t_A sono specifici punti appartenenti ad $[-1, 1]$. Seguendo la procedura descritta in [Citazione], a seguito di opportuni passaggi, si ha:

$$\int_a^b f(x)\,dx = \varepsilon + \frac{b-a}{2}\sum_{i=0}^{n} A_A f(x(t_A)) \tag{3.21}$$

dove $A_A^{(n)}$ sono determinati coefficienti in funzione del grado del polinomio approssimante scelto ed ε, definito *errore*, è pari a:

$$\varepsilon = \frac{f^{(2n+2)}(x_\varepsilon)}{(2n+2)!}\int_a^b F^2(x)\,dx \tag{3.22}$$

avendo indicato con x_ε un generico punto appartenente ad $[a, b]$, con l'apice $^{(2n+2)}$ l'operazione di derivata $(2n+2)$-esima e con $F(x)$ il polinomio di grado $n+1$:

$$F(x) = \prod_{k=0}^{n}(x - x_k) \tag{3.23}$$

Si osserva che, in base alle (3.22) e (3.23), nel caso in cui la funzione integranda $f(x)$ sia un polinomio di grado pari o minore di $2n+1$, l'errore ε risulta pari a zero e l'integrale si riduce alla sola sommatoria nel secondo membro della (3.21). Per un polinomio di interpolazione lineare, quindi per n pari a 0, si ha un singolo punti di appoggio, al quale sono associati i seguenti valori di t_0 e A_0:

$$t_0 = 0 \quad A_0 = 2$$

Invece, per un polinomio di interpolazione parabolico, dunque con n pari a 1, si ha:

$$t_0 = -\frac{\sqrt{3}}{3} \quad A_0 = 1 \qquad t_1 = \frac{\sqrt{3}}{3} \quad A_1 = 1$$

La (3.21) è facilmente estendibile anche ad integrali bidimensionali. Restingendo il caso in cui si abbia un dominio di integrazione di forma quadrangolare Ω racchiuso da quattro vertici $(x_I, y_I)_{I=1,\dots,4}$, ricordando le funzioni di

forma definite dalla (3.5), operando la trasformazione:

$$x(\xi^1,\xi^2)=\sum_{I=1}^{4}N^I(\xi^1,\xi^2)x_I \qquad y(\xi^1,\xi^2)=\sum_{I=1}^{4}N^I(\xi^1,\xi^2)y_I \tag{3.24}$$

si può operare la regola del cambio di variabili per gli integrali doppi ad una generica funzione $f(x,y)\colon \Omega\to\mathbb{R}$ integrata su Ω:

$$\int_\Omega f(x,y)\,dxdy=\int_{-1}^{1}\int_{-1}^{1}f(x(\xi^1,\xi^2),y(\xi^1,\xi^2))|\mathrm{det}J(\xi^1,\xi^2)|\,d\xi^1d\xi^2 \tag{3.25}$$

dove J è la matrice Jacobiana associata alla trasformazione (3.24):

$$J(\xi^1,\xi^2)=\begin{bmatrix}\dfrac{\partial x}{\partial\xi^1} & \dfrac{\partial y}{\partial\xi^1}\\[2ex] \dfrac{\partial x}{\partial\xi^2} & \dfrac{\partial y}{\partial\xi^2}\end{bmatrix} \tag{3.26}$$

A questo punto, è possibile calcolare la (3.25) secondo due integrazioni di Gauss monodimensionali, ad esempio, prima lungo ξ^1 e successivamente lungo ξ^2. Dunque, considerando, senza perdita di generalità, lo stesso numero di punti Gauss $\mathrm{N_{GP}}$ lungo ξ^1 e ξ^2, l'integrazione di Gauss della (3.25), trascurando l'errore ε, risulta:

$$\begin{aligned}&\int_{-1}^{1}\int_{-1}^{1}f(x(\xi^1,\xi^2),y(\xi^1,\xi^2))|\mathrm{det}J(\xi^1,\xi^2)|\,d\xi^1d\xi^2\approx\\ &\quad\approx\int_{-1}^{1}\Big(\sum_{A=0}^{\mathrm{N_{GP}}}A_Af(x(\xi_A^1,\xi^2),y(\xi_A^1,\xi^2))|\mathrm{det}J(\xi_A^1,\xi^2)|\Big)\,d\xi^2=\\ &\quad=\sum_{B=0}^{\mathrm{N_{GP}}}\sum_{A=0}^{\mathrm{N_{GP}}}A_BA_Af(x(\xi_A^1,\xi_B^2),y(\xi_A^1,\xi_B^2))|\mathrm{det}J(\xi_A^1,\xi_B^2)|\end{aligned} \tag{3.27}$$

dove $\xi_A^\alpha=t_A$, essendo il dominio di integrazione pari a $[-1,1]$.

Le varie combinazioni di coordinate (ξ_A^1,ξ_B^2) possono essere identificate con degli specifici punti $(\xi_A^1,\xi_A^2)\in\square$. Così facendo, la precedente si può scrivere:

$$\begin{aligned}&\int_{-1}^{1}\int_{-1}^{1}f(x(\xi^1,\xi^2),y(\xi^1,\xi^2))|\mathrm{det}J(\xi^1,\xi^2)|\,d\xi^1d\xi^2\approx\\ &\quad\approx\sum_{A=0}^{\mathrm{N_{GP}}}w_Af(x(\xi_A^1,\xi_A^2),y(\xi_A^1,\xi_A^2))|\mathrm{det}J(\xi_A^1,\xi_A^2)|\end{aligned} \tag{3.28}$$

per la quale, le quantità (ξ_A^1, ξ_A^2) e w_A risultano, scegliendo un punto Gauss in ξ^1 e un punto Gauss in ξ^2 ($\mathrm{N_{GP}} = 1$):

$$(\xi_1^1, \xi_1^2) = (0,0) \quad w_1 = 4 \tag{3.29}$$

mentre, scegliendo due punti Gauss in entrambe le direzioni ($\mathrm{N_{GP}} = 4$):

$$\begin{aligned}
&(\xi_1^1, \xi_1^2) = \Big(-\frac{\sqrt{3}}{3}, -\frac{\sqrt{3}}{3}\Big) \quad w_1 = 1 \qquad && (\xi_2^1, \xi_2^2) = (+\frac{\sqrt{3}}{3}, -\frac{\sqrt{3}}{3}) \quad w_2 = 1 \\
&(\xi_3^1, \xi_3^2) = \Big(+\frac{\sqrt{3}}{3}, +\frac{\sqrt{3}}{3}\Big) \quad w_3 = 1 \qquad && (\xi_4^1, \xi_4^2) = (-\frac{\sqrt{3}}{3}, +\frac{\sqrt{3}}{3}) \quad w_4 = 1
\end{aligned} \tag{3.30}$$

3.1.1 Algoritmo di aggiornamento

Riassumendo, si riporta di seguito uno schema dell'algoritmo di aggiornamento. Le quantità richiamate implicitamente con **get** si intendono calcolate mediante i metodi precedentemente descritti.

input for every Node: $(I = 1, \ldots, \mathrm{N_N})$: $\boldsymbol{\varphi}_I^{(k)}; \mathbf{t}_I^{(k)}; \boldsymbol{\Lambda}_I^{(k)}; \Delta\boldsymbol{\varphi}_I^{(k)}; \Delta\mathbf{T}_I^{(k)}$;
input for every Element **in every** Gauss Point: $\mathbf{t}^{(k)}; \mathbf{t}_{,\alpha}^{(k)}$;
for every Node **do**
  $\boldsymbol{\varphi}_I^{(k+1)} = \boldsymbol{\varphi}_I^{(k)} + \Delta\boldsymbol{\varphi}_I^{(k)}$;
  $\Delta\mathbf{t}_I^{(k)} = \boldsymbol{\Lambda}_I^{(k)} \Delta\mathbf{T}_I^{(k)}$;
  if $\Delta\mathbf{t}^{(k+1)} \neq 0$; **then**

$$\mathbf{t}_I^{(k+1)} = \cos\|\Delta\mathbf{t}_I^{(k)}\| + \frac{\sin\|\Delta\mathbf{t}_I^{(k)}\|}{\|\Delta\mathbf{t}_I^{(k)}\|}\Delta\mathbf{t}_I^{(k)};$$

$$\Delta\boldsymbol{\Lambda}_I^{(k)} = \cos\|\Delta\mathbf{t}_I^{(k)}\|\mathbf{1} + \frac{\sin\|\Delta\mathbf{t}_I^{k}\|}{\|\Delta\mathbf{t}_I^{k}\|}(\mathbf{t}_I^k \times \Delta\mathbf{t}_I^k) + \\ + \frac{1-\cos\|\Delta\mathbf{t}_I^{k}\|}{\|\Delta\mathbf{t}_I^{(k)}\|^2}(\mathbf{t}_I^{(k)} \times \Delta\mathbf{t}_I^{(k)}) \otimes (\mathbf{t}_I^{(k)} \times \Delta\mathbf{t}_I^{(k)});$$

  else

$$\mathbf{t}_I^{(k+1)} = \cos\|\Delta\mathbf{t}_I^{(k)}\| + \frac{\sin\|\Delta\mathbf{t}_I^{(k)}\|}{\|\Delta\mathbf{t}_I^{(k)}\|}\Delta\mathbf{t}_I^{(k)};$$

$$\Delta\boldsymbol{\Lambda}_I^{(k)} = \mathbf{1};$$

  end if
  $\boldsymbol{\Lambda}_I^{(k+1)} = \Delta\boldsymbol{\Lambda}_I^{(k)} \boldsymbol{\Lambda}_I^{(k)}$;
end for
for every Element **do**
  get $(I = 1, \ldots, 4)$: $\Delta\mathbf{t}_I^{(k+1)}$;
  for every Gauss Point **do**

get: $\mathbf{t}^{(k)}; \mathbf{t}^{(k)}_{,\alpha}$;
$\Delta\mathbf{t}^{(k+1)} = N^I \Delta\mathbf{t}^{(k+1)}$;
$\Delta\mathbf{t}^{(k+1)}_{,\alpha} = N^I_{,\alpha} \Delta\mathbf{t}^{(k+1)}$;
if $\Delta\mathbf{t}^{(k+1)} \neq 0$; **then**

$$\mathbf{t}^{(k+1)} = \exp(\Delta\mathbf{t}^{(k)}) = \cos\|\Delta\mathbf{t}^{(k)}\|\mathbf{t}^{(k)} + \frac{\sin\|\Delta\mathbf{t}^{(k)}\|}{\|\Delta\mathbf{t}^{(k)}\|}\Delta\mathbf{t}^{(k)};$$

$$\mathbf{t}^{(k+1)}_{,\alpha} = \cos\|\Delta\mathbf{t}^{(k)}\|\mathbf{t}^{(k)}_{,\alpha} + \Big[\frac{\sin\|\Delta\mathbf{t}^{(k)}\|}{\|\Delta\mathbf{t}^{(k)}\|}(\mathbf{1} - \mathbf{t}^{(k)} \otimes \mathbf{t}^{(k)}) + \\ + \cos\|\Delta\mathbf{t}^{(k)}\| - \frac{\sin\|\Delta\mathbf{t}^{(k)}\|}{\|\Delta\mathbf{t}^{(k)}\|}\Big)\Delta\mathbf{t}^{(k)} \otimes \Delta\mathbf{t}^{(k)}\Big]\Delta\mathbf{t}^{(k)}_{,\alpha};$$

else
$\mathbf{t}^{(k+1)} = \mathbf{t}^{(k)}$;
$\mathbf{t}^{(k+1)}_{,\alpha} = \mathbf{t}^{(k)}_{,\alpha}$;
end if
end for
end for
output for every Node ($I = 1, \ldots, \mathrm{N_N}$): $\boldsymbol{\varphi}^{(k+1)}_I; \mathbf{t}^{(k+1)}_I; \boldsymbol{\Lambda}^{(k+1)}_I$;
output for every Element **in every** Gauss Point: $\mathbf{t}^{(k+1)}; \mathbf{t}^{(k+1)}_{,\alpha}$;

3.2 Costruzione dell'algoritmo risolutivo

Alla generica iterazione k, per ogni elemento finito, dall'algoritmo di aggiornamento all'iterazione precedente, sono noti:

- il vettore posizione della superficie media nei quattro nodi afferenti all'elemento: $\boldsymbol{\varphi}^{(k)}_{I=1,\ldots,4}$;
- il vettore direttore nei quattro nodi afferenti all'elemento: $\mathbf{t}^{(k)}_{I=1,\ldots,4}$ ed il corrispondente tensore di rotazione $\boldsymbol{\Lambda}^{(k)}_{I=1,\ldots,4}$;
- il vettore direttore nei punti Gauss dell'elemento: $\mathbf{t}^{(k)}$;
- la derivata del vettore direttore, rispetto alle due coordinate $\xi^{\alpha=1,2}$, nei punti Gauss dell'elemento: $\mathbf{t}^{(k)}_{,\alpha}$;

Le stesse quantità sono inoltre note in riferimento alla configurazione iniziale $\boldsymbol{\varphi}^0_{I=1,\ldots,4}$, $\mathbf{t}^0_{I=1,\ldots,4}$, $\boldsymbol{\Lambda}^0_{I=1,\ldots,4}$, $\mathbf{t}^0$, $\mathbf{t}^0_{,\alpha}$.

Al fine di svolgere un'integrazione di Gauss, per ogni punto Gauss, si calcolano $\boldsymbol{\varphi}^{(k)}$, $\boldsymbol{\varphi}^{(k)}_{,\alpha}$, $\boldsymbol{\varphi}^0$ $\boldsymbol{\varphi}^0_{,\alpha}$, dalle quantità nodali corrispondenti, per $\alpha = 1, 2$, secondo le (3.6) e (3.7).

In ogni punto Gauss, i vettori $\mathbf{a}_1^0 = \boldsymbol{\varphi}_{,1}^0$, $\mathbf{a}_2^0 = \boldsymbol{\varphi}_{,2}^0$ e $\mathbf{a}_0^3 = \mathbf{t}^0$ definiscono la base naturale covariante rispetto al sistema di coordinate ξ^1, ξ^2, ξ sulla superficie media (quindi per $\xi = 0$). Si possono dunque calcolare le componenti cartesiane dei vettori della base controvariante secondo la relazione (1.1):

$$\begin{bmatrix}\mathbf{a}_0^1 & \mathbf{a}_0^2 & \mathbf{a}_0^3\end{bmatrix} = \begin{bmatrix}\boldsymbol{\varphi}_{,1}^0 & \boldsymbol{\varphi}_{,2}^0 & \mathbf{t}^0\end{bmatrix}^{-T}$$

Dunque, dopo avere svolto i prodotti $a_0^{11} = \mathbf{a}_0^1 \cdot \mathbf{a}_0^1$, $a_0^{22} = \mathbf{a}_0^2 \cdot \mathbf{a}_0^2$ e $a_0^{12} = \mathbf{a}_0^1 \cdot \mathbf{a}_0^2$, si calcolano le componenti del legame costitutivo (2.123), le quali sono salvate nelle matrici $[\mathbb{H}_m]$, $[\mathbb{H}_b]$ e $[\mathbb{H}_s]$:

$$[\mathbb{H}_m] = \frac{Eh}{1-\nu^2}[\tilde{\mathbb{H}}_{mb}] \qquad [\mathbb{H}_b] = \frac{Eh^3}{12(1-\nu^2)}[\tilde{\mathbb{H}}_{mb}] \qquad [\mathbb{H}_s] = \frac{\kappa Eh}{2(1+\nu)}[\tilde{\mathbb{H}}_s] \tag{3.31}$$

avendo indicato con:

$$[\tilde{\mathbb{H}}_{mb}] = \begin{bmatrix} a_0^{11}a_0^{11} & \begin{pmatrix}\nu a_0^{11}a_0^{22}+\\(1-\nu)a_0^{12}a_0^{12}\end{pmatrix} & a_0^{11}a_0^{12} \\ \begin{pmatrix}\nu a_0^{11}a_0^{22}+\\(1-\nu)a_0^{12}a_0^{12}\end{pmatrix} & a_0^{22}a_0^{22} & a_0^{22}a_0^{12} \\ a_0^{11}a_0^{12} & a_0^{22}a_0^{12} & \begin{pmatrix}\frac{1-\nu}{2}a_0^{11}a_0^{22}+\\\frac{1+\nu}{2}a_0^{12}a_0^{12}\end{pmatrix} \end{bmatrix}$$

$$[\tilde{\mathbb{H}}_s] = \begin{bmatrix} a_0^{11} & a_0^{12} \\ a_0^{12} & a_0^{22} \end{bmatrix}$$

Al fine di svolgere l'integrazione di Gauss secondo la trasformazione delle coordinate dalla superficie iniziale al dominio $\mathcal{A}$, è necessario, inoltre, calcolare il valore del determinante di superficie $\bar{j}^0$:

$$\bar{j}^0 = \|\boldsymbol{\varphi}_{,1}^0 \times \boldsymbol{\varphi}_{,2}^0\| \tag{3.32}$$

3.2.1 Costruzione della matrice di rigidezza

Sempre in corrispondenza di ogni punto Gauss, si procede calcolando le misure di deformazione (2.38):

$$\varepsilon_{\alpha\beta}^{(k)} = \frac{1}{2}(\boldsymbol{\varphi}_{,\alpha}^{(k)} \cdot \boldsymbol{\varphi}_{,\beta}^{(k)} - \boldsymbol{\varphi}_{,\alpha}^{0} \cdot \boldsymbol{\varphi}_{,\beta}^{0})$$

$$\chi_{(\alpha\beta)}^{(k)} = \frac{1}{2}(\boldsymbol{\varphi}_{,\alpha}^{(k)} \cdot \mathbf{t}_{,\beta}^{(k)} - \boldsymbol{\varphi}_{,\alpha}^{0(k)} \cdot \mathbf{t}_{,\beta}^{0(k)}) + \frac{1}{2}(\boldsymbol{\varphi}_{,\beta}^{(k)} \cdot \mathbf{t}_{,\alpha}^{(k)} - \boldsymbol{\varphi}_{,\beta}^{0(k)} \cdot \mathbf{t}_{,\alpha}^{0(k)})$$

$$\rho_{\alpha}^{(k)} = \boldsymbol{\varphi}_{,\alpha}^{(k)} \cdot \mathbf{t}^{(k)} - \boldsymbol{\varphi}_{,\alpha}^{(0)} \cdot \mathbf{t}^{(0)}$$

con le quali, si calcolano le componenti di sforzo efficaci (2.123):

$$\begin{aligned}
\tilde{n}^{\alpha\beta(k)} &= \frac{Eh}{1-\nu^2}\Big(\nu a_0^{\alpha\beta}a_0^{\gamma\delta} + \frac{1-\nu}{2}(a_0^{\alpha\gamma}a_0^{\beta\gamma} + a_0^{\alpha\delta}a_0^{\beta\gamma})\Big)\varepsilon_{\gamma\delta}^{(k)} \\
\tilde{m}^{\alpha\beta(k)} &= \frac{Eh^3}{12(1-\nu^2)}\Big(\nu a_0^{\alpha\beta}a_0^{\gamma\delta} + \frac{1-\nu}{2}(a_0^{\alpha\gamma}a_0^{\beta\gamma} + a_0^{\alpha\delta}a_0^{\beta\gamma})\Big)\chi_{(\gamma\delta)}^{(k)} \\
\tilde{q}^{\alpha(k)} &= \frac{\kappa Eh}{1-\nu^2}\frac{1-\nu}{2}a_0^{\alpha\gamma}\rho_\gamma^{(k)}
\end{aligned} \tag{3.33}$$

Richiamando le componenti delle matrici (3.31), queste risultano, in modo esplicito:

$$\begin{aligned}
\tilde{n}^{11(k)} &= [\mathbb{H}_m]_{11}\varepsilon_{11}^{(k)} + [\mathbb{H}_m]_{12}\varepsilon_{22}^{(k)} + [\mathbb{H}_m]_{13}(2\varepsilon_{12}^{(k)}) \\
\tilde{n}^{22(k)} &= [\mathbb{H}_m]_{21}\varepsilon_{11}^{(k)} + [\mathbb{H}_m]_{22}\varepsilon_{22}^{(k)} + [\mathbb{H}_m]_{23}(2\varepsilon_{12}^{(k)}) \\
\tilde{n}^{12(k)} &= [\mathbb{H}_m]_{31}\varepsilon_{11}^{(k)} + [\mathbb{H}_m]_{32}\varepsilon_{22}^{(k)} + [\mathbb{H}_m]_{33}(2\varepsilon_{12}^{(k)}) \\
\tilde{m}^{11(k)} &= [\mathbb{H}_b]_{11}\chi_{(11)}^{(k)} + [\mathbb{H}_b]_{12}\chi_{(22)}^{(k)} + [\mathbb{H}_b]_{13}(2\chi_{(12)}^{(k)}) \\
\tilde{m}^{22(k)} &= [\mathbb{H}_b]_{21}\chi_{(11)}^{(k)} + [\mathbb{H}_b]_{22}\chi_{(22)}^{(k)} + [\mathbb{H}_b]_{23}(2\chi_{(12)}^{(k)}) \\
\tilde{m}^{12(k)} &= [\mathbb{H}_b]_{31}\chi_{(11)}^{(k)} + [\mathbb{H}_b]_{32}\chi_{(22)}^{(k)} + [\mathbb{H}_b]_{33}(2\chi_{(12)}^{(k)}) \\
\tilde{q}^{1(k)} &= [\mathbb{H}_s]_{11}\rho_1^{(k)} + [\mathbb{H}_s]_{12}\rho_2^{(k)} \\
\tilde{q}^{2(k)} &= [\mathbb{H}_s]_{21}\rho_1^{(k)} + [\mathbb{H}_s]_{22}\rho_2^{(k)}
\end{aligned} \tag{3.34}$$

Dato che le variabili di deformazione nella configurazione inziale sono indipendenti dalla configurazione $\mathbf{\Phi}$, si ha $\Delta\varepsilon_{\alpha\beta} = \frac{1}{2}\Delta a_{\alpha\beta}$, $\Delta\chi_{(\alpha\beta)} = \Delta\kappa_{(\alpha\beta)}$ e $\Delta\rho_\alpha = \Delta\gamma_\alpha$. Di conseguenza, le variazioni delle misure di deformazione risultano:

$$\begin{aligned}
\Delta\varepsilon_{\alpha\beta}^{(k)} = \frac{1}{2}\Delta a_{\alpha\beta}^{(k)} &= \frac{1}{2}[\Delta\boldsymbol{\varphi}_{,\alpha}^{(k)}\cdot\boldsymbol{\varphi}_{,\beta}^{(k)} + \boldsymbol{\varphi}_{,\alpha}^{(k)}\cdot\Delta\boldsymbol{\varphi}_{,\beta}^{(k)}] = \\
&= \frac{1}{2}[N_{,\alpha}^I(\Delta\boldsymbol{\varphi}_I^{(k)}\cdot\boldsymbol{\varphi}_{,\beta}^{(k)}) + N_{,\beta}^J(\boldsymbol{\varphi}_{,\alpha}^{(k)}\cdot\Delta\boldsymbol{\varphi}_J^{(k)})] \\
\Delta\chi_{(\alpha\beta)}^{(k)} = \Delta\kappa_{(\alpha\beta)}^{(k)} &= \frac{1}{2}(\Delta\boldsymbol{\varphi}_{,\alpha}^{(k)}\cdot\mathbf{t}_{,\beta}^{(k)} + \boldsymbol{\varphi}_{,\alpha}^{(k)}\cdot\Delta\mathbf{t}_{,\beta}^{(k)} + \\
&\quad + \Delta\boldsymbol{\varphi}_{,\beta}^{(k)}\cdot\mathbf{t}_{,\alpha}^{(k)} + \quad \boldsymbol{\varphi}_{,\beta}^{(k)}\cdot\Delta\mathbf{t}_{,\alpha}^{(k)}) = \\
&= \frac{1}{2}[N_{,\alpha}^I(\Delta\boldsymbol{\varphi}_I^{(k)}\cdot\mathbf{t}_{,\beta}^{(k)} + \boldsymbol{\varphi}_{,\beta}^{(k)}\cdot\mathbf{\Lambda}_J^{(k)}\Delta\mathbf{T}_J^{(k)}) + \\
&\quad + N_{,\beta}^J(\Delta\boldsymbol{\varphi}_I^{(k)}\cdot\mathbf{t}_{,\alpha}^{(k)} + \boldsymbol{\varphi}_{,\alpha}^{(k)}\cdot\mathbf{\Lambda}_J^{(k)}\Delta\mathbf{T}_J^{(k)})] \\
\Delta\rho_\alpha^{(k)} = \Delta\gamma_\alpha^{(k)} &= \Delta\boldsymbol{\varphi}_{,\alpha}^{(k)}\cdot\mathbf{t}^{(k)} + \boldsymbol{\varphi}_{,\alpha}^{(k)}\cdot\Delta\mathbf{t}^{(k)} = \\
&= N_{,\alpha}^I(\Delta\boldsymbol{\varphi}_I^{(k)}\cdot\mathbf{t}^{(k)}) + N^J(\boldsymbol{\varphi}_{,\alpha}^{(k)}\cdot\mathbf{\Lambda}_J^{(k)}\Delta\mathbf{T}_J^{(k)})
\end{aligned} \tag{3.35}$$

Da queste, è possibile calcolare il valore della variazione delle componenti generalizzate di tensione. Dalla linearità dell'operatore $\mathbb{H}$, costante, si ha:

$$\begin{aligned}
\Delta\tilde{n}^{\alpha\beta(k)} &= \frac{Eh}{1-\nu^2}\Big(\nu a_0^{\alpha\beta}a_0^{\gamma\delta} + \frac{1-\nu}{2}(a_0^{\alpha\gamma}a_0^{\beta\gamma} + a_0^{\alpha\delta}a_0^{\beta\gamma})\Big)\tfrac{1}{2}\Delta a_{\gamma\delta}^{(k)} \\
\Delta\tilde{m}^{\alpha\beta(k)} &= \frac{Eh^3}{12(1-\nu^2)}\Big(\nu a_0^{\alpha\beta}a_0^{\gamma\delta} + \frac{1-\nu}{2}(a_0^{\alpha\gamma}a_0^{\beta\gamma} + a_0^{\alpha\delta}a_0^{\beta\gamma})\Big)\Delta\kappa_{(\gamma\delta)}^{(k)} \qquad (3.36) \\
\Delta\tilde{q}^{\alpha(k)} &= \frac{\kappa Eh}{1-\nu^2}\frac{1-\nu}{2}a_0^{\alpha\gamma}\Delta\gamma_\gamma^{(k)}
\end{aligned}$$

Di nuovo, in modo esplicito:

$$\begin{aligned}
\Delta\tilde{n}^{11(k)} &= [\mathbb{H}_m]_{11}(\tfrac{1}{2}\Delta a_{11}^{(k)}) + [\mathbb{H}_m]_{12}(\tfrac{1}{2}\Delta a_{22}^{(k)}) + [\mathbb{H}_m]_{13}(2\tfrac{1}{2}\Delta a_{12}^{(k)}) \\
\Delta\tilde{n}^{22(k)} &= [\mathbb{H}_m]_{21}(\tfrac{1}{2}\Delta a_{11}^{(k)}) + [\mathbb{H}_m]_{22}(\tfrac{1}{2}\Delta a_{22}^{(k)}) + [\mathbb{H}_m]_{23}(2\tfrac{1}{2}\Delta a_{12}^{(k)}) \\
\Delta\tilde{n}^{12(k)} &= [\mathbb{H}_m]_{31}(\tfrac{1}{2}\Delta a_{11}^{(k)}) + [\mathbb{H}_m]_{32}(\tfrac{1}{2}\Delta a_{22}^{(k)}) + [\mathbb{H}_m]_{33}(2\tfrac{1}{2}\Delta a_{12}^{(k)}) \\
\Delta\tilde{m}^{11(k)} &= [\mathbb{H}_b]_{11}(\Delta\kappa_{(11)}^{(k)}) + [\mathbb{H}_b]_{12}(\Delta\kappa_{(22)}^{(k)}) + [\mathbb{H}_b]_{13}(2\Delta\kappa_{(12)}^{(k)}) \\
\Delta\tilde{m}^{22(k)} &= [\mathbb{H}_b]_{21}(\Delta\kappa_{(11)}^{(k)}) + [\mathbb{H}_b]_{22}(\Delta\kappa_{(22)}^{(k)}) + [\mathbb{H}_b]_{23}(2\Delta\kappa_{(12)}^{(k)}) \\
\Delta\tilde{m}^{12(k)} &= [\mathbb{H}_b]_{31}(\Delta\kappa_{(11)}^{(k)}) + [\mathbb{H}_b]_{32}(\Delta\kappa_{(22)}^{(k)}) + [\mathbb{H}_b]_{33}(2\Delta\kappa_{(12)}^{(k)}) \\
\Delta\tilde{q}^{1(k)} &= [\mathbb{H}_s]_{11}(\Delta\gamma_1^{(k)}) + [\mathbb{H}_s]_{12}(\Delta\gamma_2^{(k)}) \\
\Delta\tilde{q}^{2(k)} &= [\mathbb{H}_s]_{21}(\Delta\gamma_1^{(k)}) + [\mathbb{H}_s]_{22}(\Delta\gamma_2^{(k)})
\end{aligned}$$

Le variazioni virtuali delle variabili di deformazioni risultano:

$$\begin{aligned}
\delta a_{\alpha\beta}^{(k)} &= \delta\boldsymbol{\varphi}_{,\alpha}^{(k)}\cdot\boldsymbol{\varphi}_{,\beta}^{(k)} + \boldsymbol{\varphi}_{,\alpha}^{(k)}\cdot\delta\boldsymbol{\varphi}_{,\beta}^{(k)} = \\
&= N_{,\alpha}^I(\delta\boldsymbol{\varphi}_I^{(k)}\cdot\boldsymbol{\varphi}_{,\beta}^{(k)}) + N_{,\beta}^J(\boldsymbol{\varphi}_{,\alpha}^{(k)}\cdot\delta\boldsymbol{\varphi}_J^{(k)}) \\
\delta\kappa_{(\alpha\beta)}^{(k)} &= \frac{1}{2}(\delta\boldsymbol{\varphi}_{,\alpha}^{(k)}\cdot\mathbf{t}_{,\beta}^{(k)} + \boldsymbol{\varphi}_{,\alpha}^{(k)}\cdot\delta\mathbf{t}_{,\beta}^{(k)} + \delta\boldsymbol{\varphi}_{,\beta}^{(k)}\cdot\mathbf{t}_{,\alpha}^{(k)} + \boldsymbol{\varphi}_{,\beta}^{(k)}\cdot\delta\mathbf{t}_{,\alpha}^{(k)}) = \\
&= \frac{1}{2}[N_{,\alpha}^I(\delta\boldsymbol{\varphi}_I^{(k)}\cdot\mathbf{t}_{,\beta}^{(k)} + \boldsymbol{\varphi}_{,\beta}^{(k)}\cdot\boldsymbol{\Lambda}_J^{(k)}\delta\mathbf{T}_J^{(k)}) + \qquad (3.37) \\
&\quad + N_{,\beta}^J(\delta\boldsymbol{\varphi}_I^{(k)}\cdot\mathbf{t}_{,\alpha}^{(k)} + \boldsymbol{\varphi}_{,\alpha}^{(k)}\cdot\boldsymbol{\Lambda}_J^{(k)}\delta\mathbf{T}_J^{(k)})] \\
\delta\gamma_\alpha^{(k)} &= \delta\boldsymbol{\varphi}_{,\alpha}^{(k)}\cdot\mathbf{t}^{(k)} + \boldsymbol{\varphi}_{,\alpha}^{(k)}\cdot\delta\mathbf{t}^{(k)} = \\
&= N_{,\alpha}^I(\delta\boldsymbol{\varphi}_I^{(k)}\cdot\mathbf{t}^{(k)}) + N^J(\boldsymbol{\varphi}_{,\alpha}^{(k)}\cdot\boldsymbol{\Lambda}_J^{(k)}\delta\mathbf{T}_J^{(k)})
\end{aligned}$$

Mentre, le variazioni virtuali delle variazioni delle variabili di deformazione risultano:

$$\begin{aligned}
\Delta\delta a_{\alpha\beta}^{(k)} &= \delta\boldsymbol{\varphi}_{,\alpha}^{(k)} \cdot \Delta\boldsymbol{\varphi}_{,\beta}^{(k)} + \Delta\boldsymbol{\varphi}_{,\alpha}^{(k)} \cdot \delta\boldsymbol{\varphi}_{,\beta}^{(k)} = \\
&= N_{,\alpha}^{I} N_{,\beta}^{J} (\delta\boldsymbol{\varphi}_{I}^{(k)} \cdot \Delta\boldsymbol{\varphi}_{J}^{(k)}) + N_{,\alpha}^{I} N_{,\beta}^{J} (\Delta\boldsymbol{\varphi}_{I}^{(k)} \cdot \delta\boldsymbol{\varphi}_{J}^{(k)}) \\
\Delta\delta\kappa_{(\alpha\beta)}^{(k)} &= \frac{1}{2}(\delta\boldsymbol{\varphi}_{,\alpha}^{(k)} \cdot \Delta\mathbf{t}_{,\beta}^{(k)} + \Delta\boldsymbol{\varphi}_{,\alpha}^{(k)} \cdot \delta\mathbf{t}_{,\beta}^{(k)} + \boldsymbol{\varphi}_{,\alpha}^{(k)} \cdot \Delta\delta\mathbf{t}_{,\beta}^{(k)}) + \\
&\quad + \frac{1}{2}(\delta\boldsymbol{\varphi}_{,\beta}^{(k)} \cdot \Delta\mathbf{t}_{,\alpha}^{(k)} + \Delta\boldsymbol{\varphi}_{,\beta}^{(k)} \cdot \delta\mathbf{t}_{,\alpha}^{(k)} + \boldsymbol{\varphi}_{,\beta}^{(k)} \cdot \Delta\delta\mathbf{t}_{,\alpha}^{(k)}) = \\
&\quad = \frac{1}{2}(N_{,\alpha}^{I} N_{,\beta}^{J} + N_{,\beta}^{I} N_{,\alpha}^{J})(\delta\boldsymbol{\varphi}_{I}^{(k)} \cdot \boldsymbol{\Lambda}_{J}^{(k)} \Delta\mathbf{T}_{J}^{(k)}) + \\
&\quad + \frac{1}{2}(N_{,\alpha}^{I} N_{,\beta}^{J} + N_{,\beta}^{I} N_{,\alpha}^{J})(\Delta\boldsymbol{\varphi}_{I}^{(k)} \cdot \boldsymbol{\Lambda}_{J}^{(k)} \delta\mathbf{T}_{J}^{(k)}) + \\
&\quad + \frac{1}{2}(N_{,\beta}^{I} \boldsymbol{\varphi}_{,\alpha}^{(k)} \cdot \mathbf{t}^{(k)} + N_{,\alpha}^{I} \boldsymbol{\varphi}_{,\beta}^{(k)} \cdot \mathbf{t}^{(k)})(\boldsymbol{\Lambda}_{I}^{(k)} \Delta\mathbf{T}_{I}^{(k)} \cdot \boldsymbol{\Lambda}_{J}^{(k)} \Delta\mathbf{T}_{J}^{(k)}) \\
\Delta\delta\gamma_{\alpha}^{(k)} &= \delta\boldsymbol{\varphi}_{,\alpha}^{(k)} \cdot \Delta\mathbf{t}^{(k)} + \Delta\boldsymbol{\varphi}_{,\alpha}^{(k)} \cdot \delta\mathbf{t}^{(k)} + \boldsymbol{\varphi}_{,\alpha}^{(k)} \cdot \Delta\delta\mathbf{t}^{(k)} = \\
&= N_{,\alpha}^{I} (\delta\boldsymbol{\varphi}_{I}^{(k)} \cdot \mathbf{t}^{(k)}) + N_{,\alpha}^{I} N^{J} (\boldsymbol{\varphi}_{I}^{(k)} \cdot \boldsymbol{\Lambda}_{I}^{(k)} \delta\mathbf{T}_{J}^{(k)}) + \\
&\quad - N_{,\beta}^{I} (\boldsymbol{\varphi}^{(k)} \cdot \boldsymbol{\Lambda}_{I}^{(k)} \Delta\mathbf{T}_{I}^{(k)})(\boldsymbol{\Lambda}_{I}^{(k)} \Delta\mathbf{T}_{I}^{(k)} \cdot \boldsymbol{\Lambda}_{I}^{(k)} \delta\mathbf{T}_{I}^{(k)})
\end{aligned} \tag{3.38}$$

In riferimento al generico elemento finito, definendo i vettori delle componenti cartesiane delle variazioni e delle variazioni virtuali dei gradi di libertà alla generica iterazione k:

$$\begin{aligned}
[\Delta\boldsymbol{\Phi}_{\mathrm{E}}^{(k)}] &= \begin{bmatrix} \Delta\boldsymbol{\varphi}_{1}^{(k)} & \Delta\boldsymbol{\varphi}_{2}^{(k)} & \Delta\boldsymbol{\varphi}_{3}^{(k)} & \Delta\boldsymbol{\varphi}_{4}^{(k)} & \Delta\mathbf{T}_{1}^{(k)} & \Delta\mathbf{T}_{2}^{(k)} & \Delta\mathbf{T}_{3}^{(k)} & \Delta\mathbf{T}_{4}^{(k)} \end{bmatrix}^{T} \\
[\delta\boldsymbol{\Phi}_{\mathrm{E}}^{(k)}] &= \begin{bmatrix} \delta\boldsymbol{\varphi}_{1}^{(k)} & \delta\boldsymbol{\varphi}_{2}^{(k)} & \delta\boldsymbol{\varphi}_{3}^{(k)} & \delta\boldsymbol{\varphi}_{4}^{(k)} & \delta\mathbf{T}_{1}^{(k)} & \delta\mathbf{T}_{2}^{(k)} & \delta\mathbf{T}_{3}^{(k)} & \delta\mathbf{T}_{4}^{(k)} \end{bmatrix}^{T}
\end{aligned} \tag{3.39}$$

dove ogni vettore considera le componenti cartesiane, in riferimento al generico nodo I:

$$\begin{aligned}
\Delta\boldsymbol{\varphi}_{I}^{(k)} &= \begin{bmatrix} \Delta\varphi_{I}^{1(k)} & \Delta\varphi_{I}^{2(k)} & \Delta\varphi_{I}^{3(k)} \end{bmatrix}^{T} \qquad \Delta\mathbf{T}_{I}^{(k)} = \begin{bmatrix} \Delta T_{I}^{1(k)} & \Delta T_{I}^{2(k)} \end{bmatrix}^{T} \\
\delta\boldsymbol{\varphi}_{I}^{(k)} &= \begin{bmatrix} \delta\varphi_{I}^{1(k)} & \delta\varphi_{I}^{2(k)} & \delta\varphi_{I}^{3(k)} \end{bmatrix}^{T} \qquad \delta\mathbf{T}_{I}^{(k)} = \begin{bmatrix} \delta T_{I}^{1(k)} & \delta T_{I}^{2(k)} \end{bmatrix}^{T}
\end{aligned}$$

avendo considerato solamente le componenti diverse da zero delle variazioni $\Delta\mathbf{T}_{I}^{(k)}, \delta\mathbf{T}_{I}^{(k)}$. In questo modo, salvando le componenti cartesiane delle variazioni del vettore direttore in modo analogo alle precedenti:

$$\Delta\mathbf{t}_{I}^{(k)} = \begin{bmatrix} \Delta t_{I}^{1(k)} & \Delta t_{I}^{2(k)} & \Delta t_{I}^{3(k)} \end{bmatrix}^{T} \qquad \delta\mathbf{t}_{I}^{(k)} = \begin{bmatrix} \delta t_{I}^{1(k)} & \delta t_{I}^{2(k)} & \delta t_{I}^{3(k)} \end{bmatrix}^{T} \tag{3.40}$$

la (2.107) può essere scritta:

$$\Delta \mathbf{t}_I^{(k)} = \bar{\mathbf{\Lambda}}_I^{(k)} \Delta \mathbf{T}_I^{(k)} \qquad \Delta \mathbf{t}_I^{(k)} = \bar{\mathbf{\Lambda}}_I^{(k)} \Delta \mathbf{T}_I^{(k)} \tag{3.41}$$

avendo intorodotto la matrice $\bar{\mathbf{\Lambda}}_I^{(k)}$, la quale considera solamente le componenti cartesiane efficaci del tensore $\mathbf{\Lambda}_I^{(k)}$ in riferimento alla relazione precedente:

$$\bar{\mathbf{\Lambda}}_I^{(k)} = \begin{bmatrix} \Lambda_{I,11}^{(k)} & \Lambda_{I,12}^{(k)} \\ \Lambda_{I,21}^{(k)} & \Lambda_{I,22}^{(k)} \\ \Lambda_{I,31}^{(k)} & \Lambda_{I,32}^{(k)} \end{bmatrix} \tag{3.42}$$

Si definiscono le matrici (con $\mathbf{0}_{i\times j}$ si è indicata la matrice, di dimensioni $i \times j$ con tutti zeri):

- $[\frac{1}{2} a_{11}^{(k)}] = \begin{bmatrix} [\frac{1}{2} a_{11}^{(k)}]_{12\times 1} \\ \mathbf{0}_{8\times 1} \end{bmatrix}$, tale che:

  $$\tfrac{1}{2}\Delta a_{11}^{(k)} = [\tfrac{1}{2} a_{11}^{(k)}]^T [\Delta \mathbf{\Phi}_{\mathrm{E}}^{(k)}] \qquad \tfrac{1}{2}\delta a_{11}^{(k)} = [\delta \mathbf{\Phi}_{\mathrm{E}}^{(k)}]^T [\tfrac{1}{2} a_{11}^{(k)}]^T$$

  avendo indicato con:

  - $[\frac{1}{2} a_{11}^{(k)}]_{12\times 1} = \begin{bmatrix} N_{,1}^1 \boldsymbol{\varphi}_{,1}^{(k)} \\ N_{,1}^2 \boldsymbol{\varphi}_{,1}^{(k)} \\ N_{,1}^3 \boldsymbol{\varphi}_{,1}^{(k)} \\ N_{,1}^4 \boldsymbol{\varphi}_{,1}^{(k)} \end{bmatrix}$

- $[\frac{1}{2} a_{22}^{(k)}] = \begin{bmatrix} [\frac{1}{2} a_{22}^{(k)}]_{12\times 1} \\ \mathbf{0}_{8\times 1} \end{bmatrix}$, tale che:

  $$\tfrac{1}{2}\Delta a_{22}^{(k)} = [\tfrac{1}{2} a_{22}^{(k)}]^T [\Delta \mathbf{\Phi}_{\mathrm{E}}^{(k)}] \qquad \tfrac{1}{2}\delta a_{22}^{(k)} = [\delta \mathbf{\Phi}_{\mathrm{E}}^{(k)}]^T [\tfrac{1}{2} a_{22}^{(k)}]^T$$

  avendo indicato con:

  - $[\frac{1}{2} a_{22}^{(k)}]_{12\times 1} = \begin{bmatrix} N_{,2}^1 \boldsymbol{\varphi}_{,2}^{(k)} \\ N_{,2}^2 \boldsymbol{\varphi}_{,2}^{(k)} \\ N_{,2}^3 \boldsymbol{\varphi}_{,2}^{(k)} \\ N_{,2}^4 \boldsymbol{\varphi}_{,2}^{(k)} \end{bmatrix}$

- $[2\frac{1}{2}a_{12}^{(k)}] = \begin{bmatrix} [2\frac{1}{2}a_{12}^{(k)}]_{12\times 1} \\ \mathbf{0}_{8\times 1} \end{bmatrix}$, tale che:

$$2\tfrac{1}{2}\Delta a_{12}^{(k)} = [2\tfrac{1}{2}a_{12}^{(k)}]^T[\Delta\boldsymbol{\Phi}_{\mathrm{E}}^{(k)}] \qquad 2\tfrac{1}{2}\delta a_{12}^{(k)} = [\delta\boldsymbol{\Phi}_{\mathrm{E}}^{(k)}]^T[2\tfrac{1}{2}a_{12}^{(k)}]$$

avendo indicato con:

  - $[2\frac{1}{2}a_{12}^{(k)}]_{12\times 1} = \begin{bmatrix} N_{,1}^1\boldsymbol{\varphi}_{,2}^{(k)} + N_{,2}^1\boldsymbol{\varphi}_{,1}^{(k)} \\ N_{,1}^2\boldsymbol{\varphi}_{,2}^{(k)} + N_{,2}^2\boldsymbol{\varphi}_{,1}^{(k)} \\ N_{,1}^3\boldsymbol{\varphi}_{,2}^{(k)} + N_{,2}^3\boldsymbol{\varphi}_{,1}^{(k)} \\ N_{,1}^4\boldsymbol{\varphi}_{,2}^{(k)} + N_{,2}^4\boldsymbol{\varphi}_{,1}^{(k)} \end{bmatrix}$

- $[\kappa_{(11)}^{(k)}] = \begin{bmatrix} [\kappa_{(11)}^{(k)}]_{12\times 1} \\ [\kappa_{(11)}^{(k)}]_{8\times 1} \end{bmatrix}$, tale che:

$$\Delta\kappa_{(11)}^{(k)} = [\kappa_{(11)}^{(k)}]^T[\Delta\boldsymbol{\Phi}_{\mathrm{E}}^{(k)}] \qquad \delta\kappa_{(11)}^{(k)} = [\delta\boldsymbol{\Phi}_{\mathrm{E}}^{(k)}]^T[\kappa_{(11)}^{(k)}]^T$$

avendo indicato con:

  - $[\kappa_{(11)}^{(k)}]_{12\times 1} = \begin{bmatrix} N_{,1}^1\mathbf{t}_{,1}^{(k)} \\ N_{,1}^2\mathbf{t}_{,1}^{(k)} \\ N_{,1}^3\mathbf{t}_{,1}^{(k)} \\ N_{,1}^4\mathbf{t}_{,1}^{(k)} \end{bmatrix}$

  - $[\kappa_{(11)}^{(k)}]_{8\times 1} = \begin{bmatrix} N_{,1}^1\bar{\boldsymbol{\Lambda}}_1^{(k)T}\boldsymbol{\varphi}_{,1}^{(k)} \\ N_{,1}^2\bar{\boldsymbol{\Lambda}}_2^{(k)T}\boldsymbol{\varphi}_{,1}^{(k)} \\ N_{,1}^3\bar{\boldsymbol{\Lambda}}_3^{(k)T}\boldsymbol{\varphi}_{,1}^{(k)} \\ N_{,1}^4\bar{\boldsymbol{\Lambda}}_4^{(k)T}\boldsymbol{\varphi}_{,1}^{(k)} \end{bmatrix}$

- $[\kappa_{(22)}^{(k)}] = \begin{bmatrix} [\kappa_{(22)}^{(k)}]_{12\times 1} \\ [\kappa_{(22)}^{(k)}]_{8\times 1} \end{bmatrix}$, tale che:

$$\Delta\kappa_{(22)}^{(k)} = [\kappa_{(22)}^{(k)}]^T[\Delta\boldsymbol{\Phi}_{\mathrm{E}}^{(k)}] \qquad \delta\kappa_{(22)}^{(k)} = [\delta\boldsymbol{\Phi}_{\mathrm{E}}^{(k)}]^T[\kappa_{(22)}^{(k)}]^T$$

avendo indicato con:

- $[\kappa_{(22)}^{(k)}]_{12\times1} = \begin{bmatrix} N_{,2}^1 \mathbf{t}_{,2}^{(k)} \\ N_{,2}^2 \mathbf{t}_{,2}^{(k)} \\ N_{,2}^3 \mathbf{t}_{,2}^{(k)} \\ N_{,2}^4 \mathbf{t}_{,2}^{(k)} \end{bmatrix}$

- $[\kappa_{(22)}^{(k)}]_{8\times1} = \begin{bmatrix} N_{,2}^1 \bar{\mathbf{\Lambda}}_1^{(k)T} \boldsymbol{\varphi}_{,2}^{(k)} \\ N_{,2}^2 \bar{\mathbf{\Lambda}}_2^{(k)T} \boldsymbol{\varphi}_{,2}^{(k)} \\ N_{,2}^3 \bar{\mathbf{\Lambda}}_3^{(k)T} \boldsymbol{\varphi}_{,2}^{(k)} \\ N_{,2}^4 \bar{\mathbf{\Lambda}}_4^{(k)T} \boldsymbol{\varphi}_{,2}^{(k)} \end{bmatrix}$

- $[2\kappa_{(12)}^{(k)}] = \begin{bmatrix} [2\kappa_{(12)}^{(k)}]_{12\times1} \\ [2\kappa_{(12)}^{(k)}]_{8\times1} \end{bmatrix}$, tale che:

$$2\Delta\kappa_{(12)}^{(k)} = [2\kappa_{(12)}^{(k)}]^T [\Delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}] \qquad 2\delta\kappa_{(12)}^{(k)} = [\delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}]^T [2\kappa_{(12)}^{(k)}]^T$$

avendo indicato con:

- $[2\kappa_{(12)}^{(k)}]_{12\times1} = \begin{bmatrix} N_{,1}^1 \mathbf{t}_{,2}^{(k)} + N_{,2}^1 \mathbf{t}_{,1}^{(k)} \\ N_{,1}^2 \mathbf{t}_{,2}^{(k)} + N_{,2}^2 \mathbf{t}_{,1}^{(k)} \\ N_{,1}^3 \mathbf{t}_{,2}^{(k)} + N_{,2}^3 \mathbf{t}_{,1}^{(k)} \\ N_{,1}^4 \mathbf{t}_{,2}^{(k)} + N_{,2}^4 \mathbf{t}_{,1}^{(k)} \end{bmatrix}$

- $[2\kappa_{(12)}^{(k)}]_{8\times1} = \begin{bmatrix} N_{,2}^1 \bar{\mathbf{\Lambda}}_1^{(k)T} \boldsymbol{\varphi}_{,1}^{(k)} + N_{,1}^1 \bar{\mathbf{\Lambda}}_1^{(k)T} \boldsymbol{\varphi}_{,2}^{(k)} \\ N_{,2}^2 \bar{\mathbf{\Lambda}}_2^{(k)T} \boldsymbol{\varphi}_{,1}^{(k)} + N_{,1}^2 \bar{\mathbf{\Lambda}}_2^{(k)T} \boldsymbol{\varphi}_{,2}^{(k)} \\ N_{,2}^3 \bar{\mathbf{\Lambda}}_3^{(k)T} \boldsymbol{\varphi}_{,1}^{(k)} + N_{,1}^3 \bar{\mathbf{\Lambda}}_3^{(k)T} \boldsymbol{\varphi}_{,2}^{(k)} \\ N_{,2}^4 \bar{\mathbf{\Lambda}}_4^{(k)T} \boldsymbol{\varphi}_{,1}^{(k)} + N_{,1}^4 \bar{\mathbf{\Lambda}}_4^{(k)T} \boldsymbol{\varphi}_{,2}^{(k)} \end{bmatrix}$

- $[\gamma_1^{(k)}] = \begin{bmatrix} [\gamma_1^{(k)}]_{12\times1} \\ [\gamma_1^{(k)}]_{8\times1} \end{bmatrix}$, tale che:

$$\Delta\gamma_1^{(k)} = [\gamma_1^{(k)}]^T [\Delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}] \qquad \delta\gamma_1^{(k)} = [\delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}]^T [\gamma_1^{(k)}]^T$$

avendo indicato con:

- $[\gamma_1^{(k)}]_{12\times 1} = \begin{bmatrix} N^1_{,1}\mathbf{t}^{(k)} \\ N^2_{,1}\mathbf{t}^{(k)} \\ N^3_{,1}\mathbf{t}^{(k)} \\ N^4_{,1}\mathbf{t}^{(k)} \end{bmatrix}$

- $[\gamma_1^{(k)}]_{8\times 1} = \begin{bmatrix} N^1\bar{\mathbf{\Lambda}}_1^{(k)T}\boldsymbol{\varphi}^{(k)}_{,1} \\ N^2\bar{\mathbf{\Lambda}}_2^{(k)T}\boldsymbol{\varphi}^{(k)}_{,1} \\ N^3\bar{\mathbf{\Lambda}}_3^{(k)T}\boldsymbol{\varphi}^{(k)}_{,1} \\ N^4\bar{\mathbf{\Lambda}}_4^{(k)T}\boldsymbol{\varphi}^{(k)}_{,1} \end{bmatrix}$

- $[\gamma_2^{(k)}] = \begin{bmatrix} [\gamma_2^{(k)}]_{12\times 1} \\ [\gamma_2^{(k)}]_{8\times 1} \end{bmatrix}$, tale che:

$$\Delta\gamma_2^{(k)} = [\gamma_2^{(k)}]^T[\Delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}] \qquad \delta\gamma_2^{(k)} = [\delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}]^T[\gamma_2^{(k)}]^T$$

avendo indicato con:

- $[\gamma_1^{(k)}]_{12\times 1} = \begin{bmatrix} N^1_{,2}\mathbf{t}^{(k)} \\ N^2_{,2}\mathbf{t}^{(k)} \\ N^3_{,2}\mathbf{t}^{(k)} \\ N^4_{,2}\mathbf{t}^{(k)} \end{bmatrix}$

- $[\gamma_1^{(k)}]_{8\times 1} = \begin{bmatrix} N^1\bar{\mathbf{\Lambda}}_1^{(k)T}\boldsymbol{\varphi}^{(k)}_{,2} \\ N^2\bar{\mathbf{\Lambda}}_2^{(k)T}\boldsymbol{\varphi}^{(k)}_{,2} \\ N^3\bar{\mathbf{\Lambda}}_3^{(k)T}\boldsymbol{\varphi}^{(k)}_{,2} \\ N^4\bar{\mathbf{\Lambda}}_4^{(k)T}\boldsymbol{\varphi}^{(k)}_{,2} \end{bmatrix}$

- $[\tilde{n}^{11(k)}]$, tale che:

$$\Delta\tilde{n}^{11(k)} = [\tilde{n}^{11(k)}]^T[\Delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}]$$

avendo indicato con:

$$[\tilde{n}^{11(k)}] = [\mathbb{H}_m]_{11}[\tfrac{1}{2}a_{11}^{(k)}] + [\mathbb{H}_m]_{12}[\tfrac{1}{2}a_{22}^{(k)}] + [\mathbb{H}_m]_{13}[2\tfrac{1}{2}a_{12}^{(k)}]$$

- $[\tilde{n}^{22(k)}]$, tale che:

$$\Delta\tilde{n}^{22(k)} = [\tilde{n}^{22(k)}]^T[\Delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}]$$

avendo indicato con:

$$[\tilde{n}^{22(k)}] = [\mathbb{H}_m]_{21}[\tfrac{1}{2}a_{11}^{(k)}] + [\mathbb{H}_m]_{22}[\tfrac{1}{2}a_{22}^{(k)}] + [\mathbb{H}_m]_{23}[2\tfrac{1}{2}a_{12}^{(k)}]$$

- $[\tilde{n}^{12(k)}]$, tale che:

$$\Delta\tilde{n}^{12(k)} = [\tilde{n}^{12(k)}]^T[\Delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}]$$

avendo indicato con:

$$[\tilde{n}^{12(k)}] = [\mathbb{H}_m]_{31}[\tfrac{1}{2}a_{11}^{(k)}] + [\mathbb{H}_m]_{32}[\tfrac{1}{2}a_{22}^{(k)}] + [\mathbb{H}_m]_{33}[2\tfrac{1}{2}a_{12}^{(k)}]$$

- $[\tilde{m}^{11(k)}]$, tale che:

$$\Delta\tilde{m}^{11(k)} = [\tilde{m}^{11(k)}]^T[\Delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}]$$

avendo indicato con:

$$[\tilde{m}^{11(k)}] = [\mathbb{H}_b]_{11}[\kappa_{(11)}^{(k)}] + [\mathbb{H}_b]_{12}[\kappa_{(22)}^{(k)}] + [\mathbb{H}_b]_{13}[2\kappa_{(12)}^{(k)}]$$

- $[\tilde{m}^{22(k)}]$, tale che:

$$\Delta\tilde{m}^{22(k)} = [\tilde{m}^{22(k)}]^T[\Delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}]$$

avendo indicato con:

$$[\tilde{m}^{22(k)}] = [\mathbb{H}_b]_{21}[\kappa_{(11)}^{(k)}] + [\mathbb{H}_b]_{22}[\kappa_{(22)}^{(k)}] + [\mathbb{H}_b]_{23}[2\kappa_{(12)}^{(k)}]$$

- $[\tilde{m}^{12(k)}]$, tale che:

$$\Delta\tilde{m}^{12(k)} = [\tilde{m}^{12(k)}]^T[\Delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}]$$

avendo indicato con:

$$[\tilde{m}^{12(k)}] = [\mathbb{H}_b]_{31}[\kappa_{(11)}^{(k)}] + [\mathbb{H}_b]_{32}[\kappa_{(22)}^{(k)}] + [\mathbb{H}_b]_{33}[2\kappa_{(12)}^{(k)}]$$

- $[\tilde{q}^{1(k)}]$, tale che:

$$\Delta\tilde{q}^{1(k)} = [\tilde{q}^{1(k)}]^T[\Delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}]$$

avendo indicato con:

$$[\tilde{q}^{1(k)}] = [\mathbb{H}_s]_{11}[\gamma_1^{(k)}] + [\mathbb{H}_s]_{12}[\gamma_2^{(k)}]$$

- $[\tilde{q}^{2(k)}]$, tale che:

$$\Delta\tilde{q}^{2(k)} = [\tilde{q}^{2(k)}]^T[\Delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}]$$

avendo indicato con:

$$[\tilde{q}^{2(k)}] = [\mathbb{H}_s]_{21}[\gamma_1^{(k)}] + [\mathbb{H}_s]_{22}[\gamma_2^{(k)}]$$

- $[\frac{1}{2}\Delta\delta a_{11}^{(k)}] = \begin{bmatrix} [\frac{1}{2}\Delta\delta a_{11}^{(k)}]_{12\times12} & \mathbf{0}_{12\times8} \\ \mathbf{0}_{8\times12} & \mathbf{0}_{8\times8} \end{bmatrix}$, tale che:

$$\frac{1}{2}\Delta\delta a_{11}^{(k)} = [\delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}]^T[\frac{1}{2}\Delta\delta a_{11}^{(k)}][\Delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}]$$

avendo indicato con:

  - $[\frac{1}{2}\Delta\delta a_{11}^{(k)}]_{12\times12} = \begin{bmatrix} (\bullet)_{11} & (\bullet)_{12} & (\bullet)_{13} & (\bullet)_{14} \\ (\bullet)_{21} & (\bullet)_{22} & (\bullet)_{23} & (\bullet)_{24} \\ (\bullet)_{31} & (\bullet)_{32} & (\bullet)_{33} & (\bullet)_{34} \\ (\bullet)_{41} & (\bullet)_{42} & (\bullet)_{43} & (\bullet)_{44} \end{bmatrix}$

    $(\bullet)_{IJ} = N_{,1}^I N_{,1}^J \mathbf{1}$

- $[\frac{1}{2}\Delta\delta a_{22}^{(k)}] = \begin{bmatrix} [\frac{1}{2}\Delta\delta a_{22}^{(k)}]_{12\times12} & \mathbf{0}_{12\times8} \\ \mathbf{0}_{8\times12} & \mathbf{0}_{8\times8} \end{bmatrix}$, tale che:

$$\frac{1}{2}\Delta\delta a_{22}^{(k)} = [\delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}]^T[\frac{1}{2}\Delta\delta a_{22}^{(k)}][\Delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}]$$

avendo indicato con:

  - $[\frac{1}{2}\Delta\delta a_{22}^{(k)}]_{12\times12} = \begin{bmatrix} (\bullet)_{11} & (\bullet)_{12} & (\bullet)_{13} & (\bullet)_{14} \\ (\bullet)_{21} & (\bullet)_{22} & (\bullet)_{23} & (\bullet)_{24} \\ (\bullet)_{31} & (\bullet)_{32} & (\bullet)_{33} & (\bullet)_{34} \\ (\bullet)_{41} & (\bullet)_{42} & (\bullet)_{43} & (\bullet)_{44} \end{bmatrix}$

    $(\bullet)_{IJ} = N_{,2}^I N_{,2}^J \mathbf{1}$

- $[2\frac{1}{2}\Delta\delta a_{12}^{(k)}] = \begin{bmatrix} [2\frac{1}{2}\Delta\delta a_{12}^{(k)}]_{12\times12} & \mathbf{0}_{12\times8} \\ \mathbf{0}_{8\times12} & \mathbf{0}_{8\times8} \end{bmatrix}$, tale che:

$$2\frac{1}{2}\Delta\delta a_{12}^{(k)} = [\delta\mathbf{\Phi}^{(k)}]^T[2\frac{1}{2}\Delta\delta a_{12}^{(k)}][\Delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}]$$

avendo indicato con:

  - $[2\frac{1}{2}\Delta\delta a_{12}^{(k)}]_{12\times12} = \begin{bmatrix} (\bullet)_{11} & (\bullet)_{12} & (\bullet)_{13} & (\bullet)_{14} \\ (\bullet)_{21} & (\bullet)_{22} & (\bullet)_{23} & (\bullet)_{24} \\ (\bullet)_{31} & (\bullet)_{32} & (\bullet)_{33} & (\bullet)_{34} \\ (\bullet)_{41} & (\bullet)_{42} & (\bullet)_{43} & (\bullet)_{44} \end{bmatrix}$

    $(\bullet)_{IJ} = (N_{,1}^I N_{,2}^J + N_{,2}^I N_{,1}^J)\mathbf{1}$

- $[\Delta\delta\kappa^{(k)}_{(11)}] = \begin{bmatrix} \mathbf{0}_{12\times12} & [\Delta\delta\kappa^{(k)}_{(11)}]_{12\times8} \\ [\Delta\delta\kappa^{(k)}_{(11)}]_{8\times12} & [\Delta\delta\kappa^{(k)}_{(11)}]_{8\times8} \end{bmatrix}$, tale che:

$$\Delta\delta\kappa^{(k)}_{(11)} = [\delta\mathbf{\Phi}^{(k)}_{\mathrm{E}}]^T[\Delta\delta\kappa^{(k)}_{(11)}][\Delta\mathbf{\Phi}^{(k)}_{\mathrm{E}}]$$

avendo indicato con:

- $[\Delta\delta\kappa^{(k)}_{(11)}]_{12\times8} = [\Delta\delta\kappa^{(k)}_{(11)}]^T_{8\times12} = \begin{bmatrix} (\bullet)_{11} & (\bullet)_{12} & (\bullet)_{13} & (\bullet)_{14} \\ (\bullet)_{21} & (\bullet)_{22} & (\bullet)_{23} & (\bullet)_{24} \\ (\bullet)_{31} & (\bullet)_{32} & (\bullet)_{33} & (\bullet)_{34} \\ (\bullet)_{41} & (\bullet)_{42} & (\bullet)_{43} & (\bullet)_{44} \end{bmatrix}$

  $(\bullet)_{IJ} = N^I_{,1}N^J_{,1}\bar{\mathbf{\Lambda}}^{(k)}_I$

- $[\Delta\delta\kappa^{(k)}_{(11)}]_{8\times8} = \begin{bmatrix} (\bullet)_{11} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} \\ \mathbf{0}_{2\times2} & (\bullet)_{22} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} \\ \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & (\bullet)_{33} & \mathbf{0}_{2\times2} \\ \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & (\bullet)_{44} \end{bmatrix}$

  $(\bullet)_{IJ} = -N^I_{,1}(\boldsymbol{\varphi}^{(k)}_{,1}\cdot\mathbf{t}^{(k)}_I)\bar{\mathbf{\Lambda}}^{(k)T}_I\bar{\mathbf{\Lambda}}^{(k)}_I$

- $[\Delta\delta\kappa^{(k)}_{(22)}] = \begin{bmatrix} \mathbf{0}_{12\times12} & [\Delta\delta\kappa^{(k)}_{(22)}]_{12\times8} \\ [\Delta\delta\kappa^{(k)}_{(22)}]_{8\times12} & [\Delta\delta\kappa^{(k)}_{(22)}]_{8\times8} \end{bmatrix}$, tale che:

$$\Delta\delta\kappa^{(k)}_{(22)} = [\delta\mathbf{\Phi}^{(k)}_{\mathrm{E}}]^T[\Delta\delta\kappa^{(k)}_{(22)}][\Delta\mathbf{\Phi}^{(k)}_{\mathrm{E}}]$$

avendo indicato con:

- $[\Delta\delta\kappa^{(k)}_{(22)}]_{12\times8} = [\Delta\delta\kappa^{(k)}_{(22)}]^T_{8\times12} = \begin{bmatrix} (\bullet)_{11} & (\bullet)_{12} & (\bullet)_{13} & (\bullet)_{14} \\ (\bullet)_{21} & (\bullet)_{22} & (\bullet)_{23} & (\bullet)_{24} \\ (\bullet)_{31} & (\bullet)_{32} & (\bullet)_{33} & (\bullet)_{34} \\ (\bullet)_{41} & (\bullet)_{42} & (\bullet)_{43} & (\bullet)_{44} \end{bmatrix}$

  $(\bullet)_{IJ} = N^I_{,2}N^J_{,2}\bar{\mathbf{\Lambda}}^{(k)}_I$

- $[\Delta\delta\kappa^{(k)}_{(22)}]_{8\times8} = \begin{bmatrix} (\bullet)_{11} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} \\ \mathbf{0}_{2\times2} & (\bullet)_{22} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} \\ \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & (\bullet)_{33} & \mathbf{0}_{2\times2} \\ \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & (\bullet)_{44} \end{bmatrix}$

  $(\bullet)_{IJ} = -N^I_{,2}(\boldsymbol{\varphi}^{(k)}_{,2}\cdot\mathbf{t}^{(k)}_I)\bar{\mathbf{\Lambda}}^{(k)T}_I\bar{\mathbf{\Lambda}}^{(k)}_I$

- $[2\Delta\delta\kappa_{(12)}^{(k)}] = \begin{bmatrix} \mathbf{0}_{12\times12} & [\Delta\delta\kappa_{(12)}^{(k)}]_{12\times8} \\ [\Delta\delta\kappa_{(12)}^{(k)}]_{8\times12} & [\Delta\delta\kappa_{(12)}^{(k)}]_{8\times8} \end{bmatrix}$, tale che:

$$2\Delta\delta\kappa_{(12)}^{(k)} = [\delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}]^T[2\Delta\delta\kappa_{(12)}^{(k)}][\Delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}]$$

avendo indicato con:

  - $[2\Delta\delta\kappa_{(12)}^{(k)}]_{12\times8} = [2\Delta\delta\kappa_{(12)}^{(k)}]_{8\times12}^T = \begin{bmatrix} (\bullet)_{11} & (\bullet)_{12} & (\bullet)_{13} & (\bullet)_{14} \\ (\bullet)_{21} & (\bullet)_{22} & (\bullet)_{23} & (\bullet)_{24} \\ (\bullet)_{31} & (\bullet)_{32} & (\bullet)_{33} & (\bullet)_{34} \\ (\bullet)_{41} & (\bullet)_{42} & (\bullet)_{43} & (\bullet)_{44} \end{bmatrix}$

    $(\bullet)_{IJ} = (N_{,1}^I N_{,2}^J + N_{,2}^I N_{,1}^J)\bar{\mathbf{\Lambda}}_I^{(k)}$

  - $[2\Delta\delta\kappa_{(22)}^{(k)}]_{8\times8} = \begin{bmatrix} (\bullet)_{11} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} \\ \mathbf{0}_{2\times2} & (\bullet)_{22} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} \\ \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & (\bullet)_{33} & \mathbf{0}_{2\times2} \\ \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & (\bullet)_{44} \end{bmatrix}$

    $(\bullet)_{II} = -[N_{,1}^I(\boldsymbol{\varphi}_{,2}^{(k)} \cdot \mathbf{t}_I^{(k)}) + N_{,2}^I(\boldsymbol{\varphi}_{,1}^{(k)} \cdot \mathbf{t}_I^{(k)})]\bar{\mathbf{\Lambda}}_I^{(k)T}\bar{\mathbf{\Lambda}}_I^{(k)}$

- $[\Delta\delta\gamma_1^{(k)}] = \begin{bmatrix} \mathbf{0}_{12\times12} & [\Delta\delta\gamma_1^{(k)}]_{12\times8} \\ [\Delta\delta\gamma_1^{(k)}]_{8\times12} & [\Delta\delta\gamma_1^{(k)}]_{8\times8} \end{bmatrix}$, tale che:

$$\Delta\delta\gamma_1^{(k)} = [\delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}][\Delta\delta\gamma_1^{(k)}]^T[\Delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}]$$

avendo indicato con:

  - $[\Delta\delta\gamma_1^{(k)}]_{12\times8} = [\Delta\delta\gamma_1^{(k)}]_{8\times12}^T = \begin{bmatrix} (\bullet)_{11} & (\bullet)_{12} & (\bullet)_{13} & (\bullet)_{14} \\ (\bullet)_{21} & (\bullet)_{22} & (\bullet)_{23} & (\bullet)_{24} \\ (\bullet)_{31} & (\bullet)_{32} & (\bullet)_{33} & (\bullet)_{34} \\ (\bullet)_{41} & (\bullet)_{42} & (\bullet)_{43} & (\bullet)_{44} \end{bmatrix}$

    $(\bullet)_{IJ} = N_{,1}^I N^J \bar{\mathbf{\Lambda}}_I^{(k)}$

  - $[\Delta\delta\gamma_1^{(k)}]_{8\times8} = \begin{bmatrix} (\bullet)_{11} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} \\ \mathbf{0}_{2\times2} & (\bullet)_{22} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} \\ \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & (\bullet)_{33} & \mathbf{0}_{2\times2} \\ \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & (\bullet)_{44} \end{bmatrix}$

    $(\bullet)_{IJ} = -N^I(\boldsymbol{\varphi}_{,1}^{(k)} \cdot \mathbf{t}_I^{(k)})\bar{\mathbf{\Lambda}}_I^{(k)T}\bar{\mathbf{\Lambda}}_I^{(k)}$

- $[\Delta\delta\gamma_2^{(k)}] = \begin{bmatrix} \mathbf{0}_{12\times12} & [\Delta\delta\gamma_2^{(k)}]_{12\times8} \\ [\Delta\delta\gamma_2^{(k)}]_{8\times12} & [\Delta\delta\gamma_2^{(k)}]_{8\times8} \end{bmatrix}$, tale che:

$$\Delta\delta\gamma_2^{(k)} = [\delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}]^T[\Delta\delta\gamma_2^{(k)}][\Delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}]$$

avendo indicato con:

  - $[\Delta\delta\gamma_2^{(k)}]_{12\times8} = [\Delta\delta\gamma_2^{(k)}]^T_{8\times12} = \begin{bmatrix} (\bullet)_{11} & (\bullet)_{12} & (\bullet)_{13} & (\bullet)_{14} \\ (\bullet)_{21} & (\bullet)_{22} & (\bullet)_{23} & (\bullet)_{24} \\ (\bullet)_{31} & (\bullet)_{32} & (\bullet)_{33} & (\bullet)_{34} \\ (\bullet)_{41} & (\bullet)_{42} & (\bullet)_{43} & (\bullet)_{44} \end{bmatrix}$

    $(\bullet)_{IJ} = N^I_{,2}N^J\bar{\mathbf{\Lambda}}_I^{(k)}$

  - $[\Delta\delta\gamma_2^{(k)}]_{8\times8} = \begin{bmatrix} (\bullet)_{11} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} \\ \mathbf{0}_{2\times2} & (\bullet)_{22} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} \\ \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & (\bullet)_{33} & \mathbf{0}_{2\times2} \\ \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} & (\bullet)_{44} \end{bmatrix}$

    $(\bullet)_{IJ} = -N^I(\boldsymbol{\varphi}_{,2}^{(k)} \cdot \mathbf{t}_I^{(k)})\bar{\mathbf{\Lambda}}_I^{(k)T}\bar{\mathbf{\Lambda}}_I^{(k)}$

Si procede calcolando le matrici di rigidezza materiale e geometrica. In riferimento alla (2.115), la componente materiale risulta:

$$D_M\mathcal{G}^{(k)} = \int_{\mathcal{A}} \left(\frac{1}{2}\Delta\tilde{n}^{\alpha\beta(k)}\delta a_{\alpha\beta}^{(k)} + \Delta\tilde{q}^{\alpha(k)}\delta\gamma^{\alpha} + \Delta\tilde{m}^{\alpha\beta(k)}\delta\kappa_{(\alpha\beta)}^{(k)}\right)\bar{j}^0 d\xi^1 d\xi^2$$

Essendo le quantità approssimate con funzioni di forma definite a tratti per ogni elemento, l'integrale precedente viene calcolato:

$$\begin{aligned} D_M\mathcal{G}^{(k)} &= \int_{\mathcal{A}} \left(\frac{1}{2}\Delta\tilde{n}^{\alpha\beta(k)}\delta a_{\alpha\beta}^{(k)} + \Delta\tilde{q}^{\alpha(k)}\delta\gamma_{\alpha}^{(k)} + \Delta\tilde{m}^{\alpha\beta(k)}\delta\kappa_{(\alpha\beta)}^{(k)}\right)\bar{j}^0 d\xi^1 d\xi^2 = \\ &= \sum_{\mathrm{E}=1}^{\mathrm{N_E}} \int_{\square} \left(\frac{1}{2}\Delta\tilde{n}^{\alpha\beta(k)}\delta a_{\alpha\beta}^{(k)} + \Delta\tilde{q}^{\alpha(k)}\delta\gamma^{\alpha} + \Delta\tilde{m}^{\alpha\beta(k)}\delta\kappa_{(\alpha\beta)}^{(k)}\right)\bar{j}^0 d\xi^1 d\xi^2 = \\ &= \sum_{\mathrm{E}=1}^{\mathrm{N_E}} D_M\mathcal{G}_{\mathrm{E}}^{(k)} \end{aligned} \tag{3.43}$$

Sfruttando le matrici introdotte nel paragrafo precedente, quest'ultima può essere scritta (con un abuso di notazione sugli indici $_{\alpha\beta}$, come si farà d'ora

in avanti, senza appesantire la notazione):

$$
\begin{aligned}
D_M\mathcal{G}_{\mathrm{E}}^{(k)} = \int_{\square} \Big(& [\delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}]^T([\tfrac{1}{2}a_{\alpha\beta}^{(k)}][\tilde{n}^{\alpha\beta(k)}]^T)[\Delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}] + \\
& + [\delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}]^T([\kappa_{(\alpha\beta)}^{(k)}][\tilde{m}^{\alpha\beta(k)}]^T)[\Delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}]^T + \\
& + [\delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}]^T([\gamma_{\alpha}^{(k)}][q^{\alpha(k)}]^T)[\Delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}]\Big)\,\bar{j}^0 d\xi^1 d\xi^2 = \\
= [\delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}]^T \int_{\square} \Big(& [\tfrac{1}{2}a_{\alpha\beta}^{(k)}][\tilde{n}^{\alpha\beta(k)}]^T + [\kappa_{(\alpha\beta)}^{(k)}][\tilde{m}^{\alpha\beta(k)}]^T + \\
& + [\gamma_{\alpha}^{(k)}][q^{\alpha(k)}]^T\Big)\,\bar{j}^0 d\xi^1 d\xi^2 [\Delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}] = [\delta\mathbf{\Phi}^{(k)}]^T K_{M,\mathrm{E}}^{(k)} [\Delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}]
\end{aligned}
$$

avendo estratto dall'integrale le matrici $[\delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}]$ e $[\Delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}]$, contenenti valori sui nodi, indipendenti da ξ^α, ed introdotto la *matrice di rigidezza materiale ristretta all'elemento* $K_{M,\mathrm{E}}^{(k)}$, la quale, nella presente tesi, è calcolata mediante un'integrazione di Gauss:

$$
\begin{aligned}
K_{M,\mathrm{E}}^{(k)} &= \int_{\square} \Big([\tfrac{1}{2}a_{\alpha\beta}^{(k)}][\tilde{n}^{\alpha\beta(k)}]^T + [\kappa_{(\alpha\beta)}^{(k)}][\tilde{m}^{\alpha\beta(k)}]^T + [\gamma_{\alpha}^{(k)}][q^{\alpha(k)}]^T\Big)\,\bar{j}^0 d\xi^1 d\xi^2 = \\
&= \sum_{\mathrm{GP}=1}^{\mathrm{N_{GP}}} \Big([\tfrac{1}{2}a_{\alpha\beta}^{(k)}][\tilde{n}^{\alpha\beta(k)}]^T + [\kappa_{(\alpha\beta)}^{(k)}][\tilde{m}^{\alpha\beta(k)}]^T + [\gamma_{\alpha}^{(k)}][q^{\alpha(k)}]^T\Big)\bar{j}^0 w
\end{aligned}
$$

La componente geometrica della (2.115) ristretta al generico elemento, risulta:

$$
D_G\mathcal{G}_{\mathrm{E}}^{(k)} = \int_{\square} \Big(\frac{1}{2}\tilde{n}^{\alpha\beta(k)}\Delta\delta a_{\alpha\beta}^{(k)} + \tilde{q}^{\alpha(k)}\Delta\delta\gamma_{\alpha}^{(k)} + \tilde{m}^{\alpha\beta(k)}\Delta\delta\kappa_{(\alpha\beta)}^{(k)}\Big)\,\bar{j}^0 d\xi^1 d\xi^2
$$

Con un procedimento analogo, introducendo le matrici precedentemente definite, si ottiene la *matrice di rigidezza geometrica ristretta all'elemento* $K_{G,\mathrm{E}}^{(k)}$, tale che $D_G\mathcal{G}_{\mathrm{E}}^{(k)} = [\delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}]^T K_{G,\mathrm{E}}^{(k)} [\Delta\mathbf{\Phi}_{\mathrm{E}}^{(k)}]$ la quale, considerando sempre un'integrazione di Gauss, risulta:

$$
\begin{aligned}
K_{G,\mathrm{E}}^{(k)} &= \int_{\square} \Big(\tilde{n}^{(k)\alpha\beta}[\tfrac{1}{2}\Delta\delta a_{\alpha\beta}^{(k)}] + \tilde{m}^{(k)\alpha\beta}[\Delta\delta\kappa_{(\alpha\beta)}^{(k)}]] + \tilde{q}^{(k)\alpha}[\Delta\delta\gamma_{\alpha}^{(k)}]\Big)\,\bar{j}^0 d\xi^1 d\xi^2 = \\
&= \sum_{\mathrm{GP}=1}^{\mathrm{N_{GP}}} \Big(\tilde{n}^{(k)\alpha\beta}[\tfrac{1}{2}\Delta\delta a_{\alpha\beta}^{(k)}] + \tilde{m}^{(k)\alpha\beta}[\Delta\delta\kappa_{(\alpha\beta)}^{(k)}] + \tilde{q}^{(k)\alpha}[\Delta\delta\gamma_{\alpha}^{(k)}]\Big)\bar{j}^0 w
\end{aligned}
$$

La somma dei due contributi, (3.2.1) e (3.2.1), restituisce la *matrice di rigidezza ristretta all'elemento* $K_{\mathrm{E}}^{(k)}$:

$$
K_{\mathrm{E}}^{(k)} = K_{M,\mathrm{E}}^{(k)} + K_{G,\mathrm{E}}^{(k)}
$$

la quale è una matrice 20×20 e, in particolare, in riferimento al generico elemento di nodi I, J, K, e L, presenta la seguente forma:

$$K_{\mathrm{E}}^{(k)} = \begin{bmatrix} K_{12\times12}^{\mathrm{E}(k)} & K_{12\times8}^{\mathrm{E}(k)} \\ K_{8\times12}^{\mathrm{E}(k)} & K_{8\times8}^{\mathrm{E}(k)} \end{bmatrix}$$

e ogni sottomatrice risulta:

$$\begin{bmatrix} (\bullet)_{II} & (\bullet)_{IJ} & (\bullet)_{IK} & (\bullet)_{IL} \\ (\bullet)_{JI} & (\bullet)_{JJ} & (\bullet)_{JK} & (\bullet)_{JL} \\ (\bullet)_{KI} & (\bullet)_{KJ} & (\bullet)_{KK} & (\bullet)_{KL} \\ (\bullet)_{LI} & (\bullet)_{LJ} & (\bullet)_{LK} & (\bullet)_{LL} \end{bmatrix}$$

dove ogni $(\bullet)$ è una matrice 3×3, 3×2, 2×3 e 2×2, in riferimento, rispettivamente, a $K_{12\times12}^{\mathrm{E}(k)}$, $K_{12\times8}^{\mathrm{E}(k)}$, $K_{8\times12}^{\mathrm{E}(k)}$ e $K_{8\times8}^{\mathrm{E}(k)}$.

Assemblaggio della matrice di rigidezza globale

Definendo le matrici delle componenti cartesiane delle variazioni e delle variazioni virtuali dei gradi di libertà alla generica iterazione k:

$$\begin{aligned} [\Delta\mathbf{\Phi}^{(k)}] &= \begin{bmatrix} \Delta\boldsymbol{\varphi}_1^{(k)} & \cdots & \Delta\boldsymbol{\varphi}_{\mathrm{N_N}}^{(k)} & \Delta\mathbf{T}_1^{(k)} & \cdots & \Delta\mathbf{T}_{\mathrm{N_N}}^{(k)} \end{bmatrix}^T \\ [\delta\mathbf{\Phi}^{(k)}] &= \begin{bmatrix} \delta\boldsymbol{\varphi}_1^{(k)} & \cdots & \delta\boldsymbol{\varphi}_{\mathrm{N_N}}^{(k)} & \delta\mathbf{T}_1^{(k)} & \cdots & \delta\mathbf{T}_{\mathrm{N_N}}^{(k)} \end{bmatrix}^T \end{aligned} \tag{3.44}$$

si verifica che l'integrale generale (2.115) può scriversi:

$$[\delta\mathbf{\Phi}]^T K^{(k)} [\Delta\mathbf{\Phi}^{(k)}] \tag{3.45}$$

avendo introdotto la *matrice di rigidezza globale* $K^{(k)}$, di dimensioni $5\mathrm{N_N} \times 5\mathrm{N_N}$, della seguente forma:

$$K^{(k)} = \begin{bmatrix} K_{3\mathrm{N_N}\times3\mathrm{N_N}}^{(k)} & K_{3\mathrm{N_N}\times2\mathrm{N_N}}^{(k)} \\ K_{2\mathrm{N_N}\times3\mathrm{N_N}}^{(k)} & K_{2\mathrm{N_N}\times2\mathrm{N_N}}^{(k)} \end{bmatrix}$$

Analogamente alla matrice di rigidezza locale, ogni sottomatrice risulta:

$$\begin{bmatrix} (\bullet)_{11} & \cdots & (\bullet)_{1\mathrm{N_N}} \\ \vdots & & \\ (\bullet)_{\mathrm{N_N}1} & \cdots & (\bullet)_{\mathrm{N_N}\mathrm{N_N}} \end{bmatrix}$$

dove ogni $(\bullet)$ è una matrice 3×3, 3×2, 2×3 e 2×2, in riferimento, rispettivamente, a $K_{3\mathrm{N_N}\times3\mathrm{N_N}}^{(k)}$, $K_{3\mathrm{N_N}\times2\mathrm{N_N}}^{(k)}$, $K_{2\mathrm{N_N}\times3\mathrm{N_N}}^{(k)}$ e $K_{2\mathrm{N_N}\times2\mathrm{N_N}}^{(k)}$.

Il generico contributo $(\bullet)_{IJ}$ della matrice di rigidezza globale, dunque, è dato dalla somma dei contributi $(\bullet)_{IJ}$ delle matrici di rigidezza locali, secondo il classico schema di assemblaggio.

La variazione di $\mathcal{G}^{(k)}$ può così essere scritta:

$$D\mathcal{G}(\boldsymbol{\Phi}^{(k)}, \delta\boldsymbol{\Phi}^{(k)})[\Delta\boldsymbol{\Phi}^{(k)}] = [\delta\boldsymbol{\Phi}^{(k)}]^T K^{(k)}[\Delta\boldsymbol{\Phi}^{(k)}] \tag{3.46}$$

3.2.1.1 Algoritmo di costruzione

Si riporta di seguito uno schema dell'algoritmo per la costruzione della matrice di rigidezza. Con **set** si è indicata l'inizializzazione di una variabile. Si pone attenzione al fatto che le sommatorie sugli indici $\alpha\beta$ sono intese considerando le componenti simmetriche come nei metodi descritti precedentemente. Si fa inoltre attenzione alla schematizzazione della procedura di assemblaggio per la quale la corrispondenza fra il generico nodo nel sistema di riferimento locale $i = 1, \ldots, 4$ e il corrispondente nel sistema di riferimento globale $I = 1, \ldots, \mathrm{N_N}$ viene indicata con $i \to I$.

input: $h; E; \nu; \kappa;$
set: $K_M^{(k)} = \mathbf{0}_{5\mathrm{N_N}\times 5\mathrm{N_N}}; K_G^{(k)} = \mathbf{0}_{5\mathrm{N_N}\times 5\mathrm{N_N}};$
for every Element **do**
  get $(I = 1, \ldots, 4)$: $\boldsymbol{\varphi}_I^{(k)}; \mathbf{t}_I^{(k)}; \boldsymbol{\Lambda}_I; \boldsymbol{\varphi}_I^0; \mathbf{t}_I^0;$
  set: $K_{M,\mathrm{E}}^{(k)} = \mathbf{0}_{20\times 20}; K_{G,\mathrm{E}}^{(k)} = \mathbf{0}_{20\times 20};$
  for every GaussPoint **do**
    get: $(\xi^1, \xi^2); w;$
    get: $\mathbf{t}^{(k)}; \mathbf{t}_{,\alpha=1,2}^{(k)}; \mathbf{t}^0; \mathbf{t}_{,\alpha=1,2}^0;$
    get $(I = 1, \ldots, 4), (\alpha = 1, 2)$: $N^I; N^I_{,\alpha};$
    $\boldsymbol{\varphi}^{(k)} = N^I \boldsymbol{\varphi}_I^{(k)}; \boldsymbol{\varphi}_{,\alpha}^{(k)} = N^I_{,\alpha} \boldsymbol{\varphi}_I^{(k)};$
    $\boldsymbol{\varphi}^0 = N^I \boldsymbol{\varphi}_I^0; \boldsymbol{\varphi}_{,\alpha}^0 = N^I_{,\alpha} \boldsymbol{\varphi}_I^0;$
    $\begin{bmatrix}\mathbf{a}_0^1 & \mathbf{a}_0^2 & \mathbf{a}_0^3\end{bmatrix} = \begin{bmatrix}\boldsymbol{\varphi}_{,1}^0 & \boldsymbol{\varphi}_{,2}^0 & \mathbf{t}^0\end{bmatrix}^{-T};$
    $a^{\alpha\beta} = \mathbf{a}_0^\alpha \cdot \mathbf{a}_0^\beta;$
    $\bar{j}^0 = \|\boldsymbol{\varphi}_{,1}^0 \times \boldsymbol{\varphi}_{,2}^0\|;$
    $\varepsilon_{\alpha\beta}^{(k)} = \dfrac{1}{2}(\boldsymbol{\varphi}_{,\alpha}^{(k)} \cdot \boldsymbol{\varphi}_{,\beta}^{(k)} - \boldsymbol{\varphi}_{,\alpha}^0 \cdot \boldsymbol{\varphi}_{,\beta}^0);$
    $\chi_{(\alpha\beta)}^{(k)} = \dfrac{1}{2}(\boldsymbol{\varphi}_{,\alpha}^{(k)} \cdot \mathbf{t}_{,\beta}^{(k)} - \boldsymbol{\varphi}_{,\alpha}^0 \cdot \mathbf{t}_{,\beta}^0) + \dfrac{1}{2}(\boldsymbol{\varphi}_{,\beta}^{(k)} \cdot \mathbf{t}_{,\alpha}^{(k)} - \boldsymbol{\varphi}_{,\beta}^0 \cdot \mathbf{t}_{,\alpha}^0);$
    $\rho_\alpha^{(k)} = \boldsymbol{\varphi}_{,\alpha}^{(k)} \cdot \mathbf{t}^{(k)} - \boldsymbol{\varphi}_{,\alpha}^0 \cdot \mathbf{t}^0;$
    $\tilde{n}^{\alpha\beta(k)} = \dfrac{Eh}{1-\nu^2}\Big(\nu a_0^{\alpha\beta} a_0^{\gamma\delta} + \dfrac{1-\nu}{2}(a_0^{\alpha\gamma} a_0^{\beta\gamma} + a_0^{\alpha\delta} a_0^{\beta\gamma})\Big)\varepsilon_{\gamma\delta}^{(k)};$
    $\tilde{m}^{\alpha\beta(k)} = \dfrac{Eh^3}{12(1-\nu^2)}\Big(\nu a_0^{\alpha\beta} a_0^{\gamma\delta} + \dfrac{1-\nu}{2}(a_0^{\alpha\gamma} a_0^{\beta\gamma} + a_0^{\alpha\delta} a_0^{\beta\gamma})\Big)\chi_{(\gamma\delta)}^{(k)};$

$\tilde{q}^{\alpha(k)} = \dfrac{\kappa E h}{1-\nu^2}\dfrac{1-\nu}{2} a_0^{\alpha\gamma}\rho_\gamma^{(k)}$;
get: $[\frac{1}{2}a_{\alpha\beta}^{(k)}]; [\kappa_{(\alpha\beta)}^{(k)}]; [\gamma_\alpha^{(k)}]; [\tilde{n}^{\alpha\beta(k)}]; [\tilde{m}^{\alpha\beta(k)}]; [q^{\alpha(k)}]$;
get: $[\frac{1}{2}\Delta\delta a_{\alpha\beta}^{(k)}]; [\Delta\delta\kappa_{(\alpha\beta)}^{(k)}]; [\Delta\delta\gamma_\alpha^{(k)}]$;
$K_{M,\mathrm{E}}^{(k)} = K_{M,\mathrm{E}}^{(k)} + \big([\frac{1}{2}a_{\alpha\beta}^{(k)}][\tilde{n}^{\alpha\beta(k)}]^T + [\kappa_{(\alpha\beta)}^{(k)}][\tilde{m}^{\alpha\beta(k)}]^T +$
$+ [\gamma_\alpha^{(k)}][q^{\alpha(k)}]^T\big)\bar{j}^0 w$
$K_{G,\mathrm{E}}^{(k)} = K_{G,\mathrm{E}}^{(k)} + \big(\tilde{n}^{(k)\alpha\beta}[\frac{1}{2}\Delta\delta a_{\alpha\beta}^{(k)}] + \tilde{m}^{(k)\alpha\beta}[\Delta\delta\kappa_{(\alpha\beta)}^{(k)}] +$
$+ \tilde{q}^{(k)\alpha}[\Delta\delta\gamma_\alpha^{(k)}]\big)\bar{j}^0 w$;
end for
$(i = 1, \ldots, 4), (j = 1, \ldots, 4)$;
$(i \to I; j \to J)$;
$K_{M,3\mathrm{N_N}\times 3\mathrm{N_N},IJ}^{(k)} = K_{M,3\mathrm{N_N}\times 3\mathrm{N_N},IJ}^{(k)} + K_{M,12\times 12,ij}^{\mathrm{E}(k)}$;
$K_{M,3\mathrm{N_N}\times 2\mathrm{N_N},IJ}^{(k)} = K_{M,3\mathrm{N_N}\times 2\mathrm{N_N},IJ}^{(k)} + K_{M,12\times 8,ij}^{\mathrm{E}(k)}$;
$K_{M,2\mathrm{N_N}\times 3\mathrm{N_N},IJ}^{(k)} = K_{M,2\mathrm{N_N}\times 3\mathrm{N_N},IJ}^{(k)} + K_{M,8\times 12,ij}^{\mathrm{E}(k)}$;
$K_{M,2\mathrm{N_N}\times 2\mathrm{N_N},IJ}^{(k)} = K_{M,2\mathrm{N_N}\times 2\mathrm{N_N},IJ}^{(k)} + K_{M,8\times 8,ij}^{\mathrm{E}(k)}$;
$K_{G,3\mathrm{N_N}\times 3\mathrm{N_N},IJ}^{(k)} = K_{G,3\mathrm{N_N}\times 3\mathrm{N_N},IJ}^{(k)} + K_{G,12\times 12,ij}^{\mathrm{E}(k)}$;
$K_{G,3\mathrm{N_N}\times 2\mathrm{N_N},IJ}^{(k)} = K_{G,3\mathrm{N_N}\times 2\mathrm{N_N},IJ}^{(k)} + K_{G,12\times 8,ij}^{\mathrm{E}(k)}$;
$K_{G,2\mathrm{N_N}\times 3\mathrm{N_N},IJ}^{(k)} = K_{G,2\mathrm{N_N}\times 3\mathrm{N_N},IJ}^{(k)} + K_{G,8\times 12,ij}^{\mathrm{E}(k)}$;
$K_{G,2\mathrm{N_N}\times 2\mathrm{N_N},IJ}^{(k)} = K_{G,2\mathrm{N_N}\times 2\mathrm{N_N},IJ}^{(k)} + K_{G,8\times 8,ij}^{\mathrm{E}(k)}$;
end for
$K^{(k)} = K_M^{(k)} + K_G^{(k)}$;
output: $K^{(k)}$;

3.2.2 Costruzione del vettore residuo

Alla generica iterazione k, la funzione $\mathcal{G}^{(k)}$ risulta:

$$\begin{aligned}\mathcal{G}^{(k)} &= \mathcal{G}(\mathbf{\Phi}^{(k)}, \delta\mathbf{\Phi}^{(k)}) = \delta\mathcal{L}_{\mathrm{int}}^{(k)} - \delta\mathcal{L}_{\mathrm{ext}} = \\ &= \int_{\mathcal{A}} \Big(\frac{1}{2}\tilde{n}^{\alpha\beta(k)}\delta a_{\alpha\beta} + \tilde{q}^{\alpha(k)}\delta\gamma_\alpha + \tilde{m}^{\alpha\beta(k)}\delta\kappa_{(\alpha\beta)}\Big)\bar{j}^0 d\xi^1 d\xi^2 + \\ &\quad - \Big(\int_{\Gamma_0} (\bar{\bar{\mathbf{n}}}\cdot\delta\boldsymbol{\varphi} + \bar{\bar{\tilde{\mathbf{m}}}}\cdot\delta\mathbf{t})\, d\Gamma_0 + \int_{\mathcal{A}} (\bar{\mathbf{n}}\cdot\delta\boldsymbol{\varphi} + \bar{\tilde{\mathbf{m}}}\cdot\delta\mathbf{t})\,\bar{j}^0 d\xi^1 d\xi^2\Big)\end{aligned} \tag{3.47}$$

Vettore delle forze interne
Richiamando le matrici introdotte precedentemente, la variazione virtuale

del lavoro interno può scriversi, per il singolo elemento finito:

$$\begin{aligned}\delta\mathcal{L}^{\mathrm{E}}_{\mathrm{int}} &= \int_{\square}\Big(\tilde{n}^{\alpha\beta(k)}[\delta\mathbf{\Phi}_{\mathrm{E}}]^T[\tfrac{1}{2}a^{(k)}_{\alpha\beta}] + \tilde{m}^{\alpha\beta(k)}[\delta\mathbf{\Phi}_{\mathrm{E}}]^T[\kappa^{(k)}_{(\alpha\beta)}]+\\ &\quad + \tilde{q}^{\alpha(k)}[\delta\mathbf{\Phi}_{\mathrm{E}}]^T[\gamma^{(k)}_{\alpha}]\Big)\,\bar{j}^0 d\xi^1 d\xi^2 =\\ &= [\delta\mathbf{\Phi}_{\mathrm{E}}]^T\int_{\square}\Big(\tilde{n}^{\alpha\beta(k)}[\tfrac{1}{2}a^{(k)}_{\alpha\beta}] + \tilde{m}^{\alpha\beta(k)}[\kappa^{(k)}_{(\alpha\beta)}] + \tilde{q}^{\alpha(k)}[\gamma^{(k)}_{\alpha}]\Big)\,\bar{j}^0 d\xi^1 d\xi^2 =\\ &= [\delta\mathbf{\Phi}_{\mathrm{E}}]^T[\mathbf{F}^{\mathrm{E}(k)}_{\mathrm{int}}]\end{aligned} \tag{3.48}$$

avendo portato $[\delta\mathbf{\Phi}_{\mathrm{E}}]^T$ fuori dall'integrale, essendo indipendente da ξ^1, ξ^2, ed introdotto il *vettore delle forze interne ristretto all'elemento* alla k-esima iterazione $[\mathbf{F}^{\mathrm{E}(k)}_{\mathrm{int}}]$, il quale può essere calcolato con un'integrazione di Gauss:

$$\begin{aligned}[\mathbf{F}^{\mathrm{E}(k)}_{\mathrm{int}}] &= \int_{\square}\Big(\tilde{n}^{\alpha\beta}[\tfrac{1}{2}a^{(k)}_{\alpha\beta}] + \tilde{m}^{\alpha\beta}[\kappa^{(k)}_{(\alpha\beta)}] + \tilde{q}^{\alpha}[\gamma^{(k)}_{\alpha}]\Big)\,\bar{j}^0 d\xi^1 d\xi^2\\ &= \sum_{\mathrm{GP}=1}^{\mathrm{N_{GP}}}\Big(\tilde{n}^{\alpha\beta(k)}[\tfrac{1}{2}a^{(k)}_{\alpha\beta}] + \tilde{m}^{\alpha\beta(k)}[\kappa^{(k)}_{(\alpha\beta)}] + \tilde{q}^{\alpha(k)}[\gamma^{(k)}_{\alpha}]\Big)\bar{j}^0 w\end{aligned} \tag{3.49}$$

e presenta la forma:

$$[\mathbf{F}^{\mathrm{E}(k)}_{\mathrm{int}}] = \begin{bmatrix}\mathbf{n}^{(k)}_I & \mathbf{n}^{(k)}_J & \mathbf{n}^{(k)}_K & \mathbf{n}^{(k)}_L & \mathbf{m}^{(k)}_I & \mathbf{m}^{(k)}_J & \mathbf{m}^{(k)}_K & \mathbf{m}^{(k)}_L\end{bmatrix} \tag{3.50}$$

in riferimento ai nodi I, J, K, e L afferenti all'elemento finito considerato.

Analogamente a quanto descritto per il passaggio dalla matrice di rigidezza locale a quella globale, per il vettore delle forze interne si introduce il *vettore delle forze interne globale* $[\mathbf{F}^{(k)}_{\mathrm{int}}]$, per il quale, la variazione virtuale del lavoro interno nella (3.47) risulta $\delta\mathcal{L}_{\mathrm{int}} = [\delta\mathbf{\Phi}]^T[\mathbf{F}^{\mathrm{E}(k)}_{\mathrm{int}}]$:

$$[\mathbf{F}^{(k)}_{\mathrm{int}}] = \begin{bmatrix}\mathbf{n}^{(k)}_1 & \cdots & \mathbf{n}^{(k)}_{\mathrm{N_N}} & \mathbf{m}^{(k)}_1 & \cdots & \mathbf{m}^{(k)}_{\mathrm{N_N}}\end{bmatrix} \tag{3.51}$$

In riferimento a $[\mathbf{F}^{(k)}_{\mathrm{int}}]$ e $[\mathbf{F}^{\mathrm{E}(k)}_{\mathrm{int}}]$, ogni vettore $\mathbf{n}^{(k)}_I$ è di dimensioni 3×1, mentre ogni vettore $\mathbf{m}^{(k)}_I$ è di dimensioni 2×1. Ogni componente $\mathbf{n}^{(k)}_I$ del vettore globale è dato dalla somma delle componenti $\mathbf{n}^{(k)}_I$ delle componenti locali del vettore delle forze interne.

Vettore delle forze esterne

Introducendo le interpolazioni (3.7) e (3.19), la variazione virtuale del lavoro esterno può scriversi:

$$\begin{aligned}\delta\mathcal{L}_{\mathrm{ext}} &= \int_{\Gamma_0}\left(N^I\bar{\bar{\mathbf{n}}}\cdot\delta\boldsymbol{\varphi}_I + N^I\bar{\bar{\tilde{\mathbf{m}}}}\cdot\delta\mathbf{t}_I\right)d\Gamma_0 +\\ &\quad + \int_{\mathcal{A}}\left(N^I\bar{\mathbf{n}}\cdot\delta\boldsymbol{\varphi}_I + N^I\bar{\tilde{\mathbf{m}}}\cdot\delta\mathbf{t}_I\right)\bar{j}^0 d\xi^1 d\xi^2\end{aligned}$$

Le variabili $\delta\boldsymbol{\varphi}_I$ e $\delta\mathbf{t}_I$ possono essere estratte dall'integrale, dunque:

$$\begin{aligned}\delta\mathcal{L}_{\text{ext}} &= \Big(\int_\gamma (N^I\bar{\bar{\mathbf{n}}})\,\bar{j}^0\,ds + \int_{\mathcal{A}} (N^I\bar{\bar{\mathbf{n}}})\,\bar{j}^0\,d\xi^1 d\xi^2\Big)\cdot\delta\boldsymbol{\varphi}_I + \\ &\quad + \Big(\int_\gamma (N^I\bar{\bar{\tilde{\mathbf{m}}}})\,\bar{j}^0\,ds + \int_{\mathcal{A}} (N^I\bar{\tilde{\mathbf{m}}})\,\bar{j}^0\,d\xi^1 d\xi^2\Big)\cdot\delta\mathbf{t}_I \\ &= \mathbf{n}^I_{\text{ext}}\cdot\delta\boldsymbol{\varphi}_I + \mathbf{m}^I_{\text{ext}}\cdot\delta\mathbf{t}_I = \mathbf{n}^I_{\text{ext}}\cdot\delta\boldsymbol{\varphi}_I + \bar{\mathbf{m}}^I_{\text{ext}}\cdot\delta\mathbf{T}_I\end{aligned} \tag{3.52}$$

avendo introdotto, in riferimento al nodo I, lo *sforzo nodale esterno* $\mathbf{n}^I_{\text{ext}}$, il *momento nodale esterno* $\mathbf{m}^I_{\text{ext}}$ ed il *momento nodale esterno in riferimento a* $\delta\mathbf{T}$ $\bar{\mathbf{m}}^I_{\text{ext}} = \boldsymbol{\Lambda}_I^T\mathbf{m}^I_{\text{ext}}$, ricordando la (2.107). Si fa presente che gli integrali (3.52) devono essere suddivisi opportunamente fra gli elementi afferenti al nodo I, data la struttura a tratti delle funzioni di forma N^I.

Si nota che, nel caso di uno sforzo concentrato $\mathbf{n}^I_{\text{ext}}$ in corripondenza del generico nodo I, il quale è ipotizzabile come una concentrazione localizzata di tensioni, alla componente I-esima del vettore $[\mathbf{F}_{\text{ext}}]$ viene semplicemente aggiunto il valore $\mathbf{n}^I_{\text{ext}}$ (la procedura è analoga nel caso di una coppia concentrata $\mathbf{m}^I_{\text{ext}}$).

3.2.2.1 Algoritmo di costruzione

Si riporta di seguito uno schema dell'algoritmo per la costruzione del vettore delle forze interne.

input: $h; E; \nu; \kappa$;
set: $[\mathbf{F}^{(k)}_{\text{int}}] = \mathbf{0}_{5\text{N}_\text{N}\times 1}$;
for every Element **do**
  get $(I = 1,\dots,4)$: $\boldsymbol{\varphi}^{(k)}_I; \mathbf{t}^{(k)}_I; \boldsymbol{\Lambda}_I; \boldsymbol{\varphi}^0_I; \mathbf{t}^0_I$;
  $[\mathbf{F}^{E(k)}_{\text{int}}] = \mathbf{0}_{20\times 1}$;
  for every GaussPoint **do**
    get: $(\xi^1,\xi^2); w$;
    get: $\mathbf{t}^{(k)}; \mathbf{t}^{(k)}_{,\alpha=1,2}; \mathbf{t}^0; \mathbf{t}^0_{,\alpha=1,2}$;
    get $(I = 1,\dots,4), (\alpha = 1,2)$: $N^I; N^I_{,\alpha}$;
    $\boldsymbol{\varphi}^{(k)} = N^I\boldsymbol{\varphi}^{(k)}_I; \boldsymbol{\varphi}^{(k)}_{,\alpha} = N^I_{,\alpha}\boldsymbol{\varphi}^{(k)}_I$;
    $\boldsymbol{\varphi}^0 = N^I\boldsymbol{\varphi}^0_I; \boldsymbol{\varphi}^0_{,\alpha} = N^I_{,\alpha}\boldsymbol{\varphi}^0_I$;
    $\begin{bmatrix}\mathbf{a}^1_0 & \mathbf{a}^2_0 & \mathbf{a}^3_0\end{bmatrix} = \begin{bmatrix}\boldsymbol{\varphi}^0_{,1} & \boldsymbol{\varphi}^0_{,2} & \mathbf{t}^0\end{bmatrix}^{-T}$;
    $a^{\alpha\beta} = \mathbf{a}^\alpha_0\cdot\mathbf{a}^\beta_0$;
    $\bar{j}^0 = \|\boldsymbol{\varphi}^0_{,1}\times\boldsymbol{\varphi}^0_{,2}\|$;
    $\varepsilon^{(k)}_{\alpha\beta} = \dfrac{1}{2}(\boldsymbol{\varphi}^{(k)}_{,\alpha}\cdot\boldsymbol{\varphi}^{(k)}_{,\beta} - \boldsymbol{\varphi}^0_{,\alpha}\cdot\boldsymbol{\varphi}^0_{,\beta})$;
    $\chi^{(k)}_{(\alpha\beta)} = \dfrac{1}{2}(\boldsymbol{\varphi}^{(k)}_{,\alpha}\cdot\mathbf{t}^{(k)}_{,\beta} - \boldsymbol{\varphi}^0_{,\alpha}\cdot\mathbf{t}^0_{,\beta}) + \dfrac{1}{2}(\boldsymbol{\varphi}^{(k)}_{,\beta}\cdot\mathbf{t}^{(k)}_{,\alpha} - \boldsymbol{\varphi}^0_{,\beta}\cdot\mathbf{t}^0_{,\alpha})$;

$\rho_\alpha^{(k)} = \boldsymbol{\varphi}_{,\alpha}^{(k)} \cdot \mathbf{t}^{(k)} - \boldsymbol{\varphi}_{,\alpha}^{0} \cdot \mathbf{t}^{0}$;
$\tilde{n}^{\alpha\beta(k)} = \dfrac{Eh}{1-\nu^2}\Big(\nu a_0^{\alpha\beta} a_0^{\gamma\delta} + \dfrac{1-\nu}{2}(a_0^{\alpha\gamma} a_0^{\beta\gamma} + a_0^{\alpha\delta} a_0^{\beta\gamma})\Big)\varepsilon_{\gamma\delta}^{(k)}$;
$\tilde{m}^{\alpha\beta(k)} = \dfrac{Eh^3}{12(1-\nu^2)}\Big(\nu a_0^{\alpha\beta} a_0^{\gamma\delta} + \dfrac{1-\nu}{2}(a_0^{\alpha\gamma} a_0^{\beta\gamma} + a_0^{\alpha\delta} a_0^{\beta\gamma})\Big)\chi_{(\gamma\delta)}^{(k)}$;
$\tilde{q}^{\alpha(k)} = \dfrac{\kappa Eh}{1-\nu^2}\dfrac{1-\nu}{2} a_0^{\alpha\gamma} \rho_\gamma^{(k)}$;
get: $[\frac{1}{2} a_{\alpha\beta}^{(k)}]; [\kappa_{(\alpha\beta)}^{(k)}]; [\gamma_\alpha^{(k)}]$;
get: $[\frac{1}{2}\Delta\delta a_{\alpha\beta}^{(k)}]; [\Delta\delta\kappa_{(\alpha\beta)}^{(k)}]; [\Delta\delta\gamma_\alpha^{(k)}]$;
$[\mathbf{F}_{\text{int}}^{E(k)}] = [\mathbf{F}_{\text{int}}^{E(k)}] + \big(\tilde{n}^{\alpha\beta(k)}[\frac{1}{2} a_{\alpha\beta}^{(k)}] + \tilde{m}^{\alpha\beta(k)}[\kappa_{(\alpha\beta)}^{(k)}] + \tilde{q}^{\alpha(k)}[\gamma_\alpha^{(k)}]\big)\bar{j}^0 w$;
end for
$(i = 1, \ldots, 4)$;
$(i \to I)$;
$[\mathbf{F}_{\text{int}}^{(k)}]_I = [\mathbf{F}_{\text{int}}^{(k)}]_I + [\mathbf{F}_{\text{int}}^{\text{E}(k)}]_i$;
end for
output: $[\mathbf{F}_{\text{int}}^{(k)}]$;

3.2.3 Soluzione del sistema

La (3.52) può essere scritta:

$$\delta\mathcal{L}_{\text{ext}} = [\delta\boldsymbol{\Phi}]^T [\mathbf{F}_{\text{ext}}] \tag{3.53}$$

avendo introdotto il *vettore delle forze esterne*, di dimensioni $5\text{N}_\text{N} \times 1$, contenente i carichi esterni in corrispondenza dei gradi di libertà:

$$[\mathbf{F}_{\text{ext}}] = \begin{bmatrix} \mathbf{n}_{\text{ext}}^1 & \cdots & \mathbf{n}_{\text{ext}}^{\text{N}_\text{N}} & \bar{\mathbf{m}}_{\text{ext}}^1 & \cdots & \bar{\mathbf{m}}_{\text{ext}}^{\text{N}_\text{N}} \end{bmatrix}^T \tag{3.54}$$

La forma variazionale (3.47) alla k-esima iterazione risulta, dunque:

$$\mathcal{G}^{(k)} = [\delta\boldsymbol{\Phi}]^T([\mathbf{F}_{\text{int}}^{(k)}] - [\mathbf{F}_{\text{ext}}]) = [\delta\boldsymbol{\Phi}]^T [\mathbf{R}^{(k)}] \tag{3.55}$$

avendo introdotto il *vettore residuo* alla k-esima iterazione. La precedente deve essere valida per ogni variazione $\delta\boldsymbol{\Phi}^{(k)}$ (corrispondente a $[\delta\boldsymbol{\Phi}^{(k)}]$):

$$\mathcal{G}^{(k)} = [\mathbf{F}_{\text{int}}^{(k)}] - [\mathbf{F}_{\text{ext}}] = [\mathbf{R}^{(k)}] = [\mathbf{0}] \tag{3.56}$$

Costruita la matrice di rigidezza ed il vettore residuo, ad ogni iterazione la forma variazionale linearizzata imposta pari a zero (3.1) risulta:

$$[\delta\boldsymbol{\Phi}^{(k)}]^T [\mathbf{R}^{(k)}] + [\delta\boldsymbol{\Phi}^{(k)}]^T K^{(k)} [\Delta\boldsymbol{\Phi}^{(k)}] = [\mathbf{0}] \tag{3.57}$$

Di nuovo, dovendo essere valida per ogni variazione virtuale, la precedente diventa:

$$[\mathbf{R}^{(k)}] + K^{(k)}[\Delta\mathbf{\Phi}^{(k)}] = [\mathbf{0}] \tag{3.58}$$

L'incremento dei gradi di libertà è dato dunque dalla soluzione del sistema:

$$[\Delta\mathbf{\Phi}^{(k)}] = -K^{(k)-1}[\mathbf{R}^{(k)}] \tag{3.59}$$

La precedente esplicita la (3.2).

Ricavando le variazioni dei gradi di libertà dei nodi corrispondenti relativi alla superficie media $\Delta\boldsymbol{\varphi}_I^{(k)}$ e al vettore direttore $\Delta\mathbf{T}_I^{(k)}$ dal vettore $[\Delta\mathbf{\Phi}^{(k)}]$ è possibile aggiornare le configurazioni assunte dai gradi di libertà secondo quanto descritto dalla sezione 3.1. Iterando fino alla convergenza si ha la soluzione.

Si fa presente che, seguendo la letteratura, per evitare fenomeni di *locking* della soluzione, si sono considerati quattro punti Gauss per l'integrazione delle variabili membranali e flessionali, per l'integrazione delle variabili taglianti, invece, è stato considerato un unico punto Gauss.

3.2.3.1 Algoritmo risolutivo

Riassumendo, dunque, si riporta uno schema dell'algoritmo risolutivo. In riferimento a $K^{(k)}$ e $[\mathbf{F}_{\text{int}}^{(k)}]$, con **get** si è indicato la procedura di calcolo della matrice di rigidezza e del vettore delle forze interne, rispettivamente. Con la stringa update, invece, si è indicata la procedura descritta dalla (3.3). Al fine di facilitare la convergenza, il problema viene risolto per step di carico uniformi, aumentando passo passo il vettore delle forze esterne $[\mathbf{F}_{\text{ext}}]$. Il numero degli step di carico considerati è indicato con LoadSteps.

input: $[\mathbf{F}_{\text{ext}}]$; Toll; MaxIter; LoadSteps;
$[\Delta\mathbf{F}_{\text{ext}}] = \dfrac{[\mathbf{F}_{\text{ext}}]}{\text{LoadSteps}}$;
for every LoadStep **do**
  $[\Delta\mathbf{F}_{\text{ext}}] = \text{Step} \cdot [\Delta\mathbf{F}_{\text{ext}}]$;
  set: $k = 0$; Err $= +\infty$;
  while Err $>$ Toll **and** $k <$ MaxIter **do**
    $k = k + 1$;
    get: $K^{(k)}, [\mathbf{F}_{\text{int}}^{(k)}]$;
    $[\mathbf{R}^{(k)}] = [\mathbf{F}_{\text{int}}^{(k)}] - [\mathbf{F}_{\text{ext}}]$;
    Err $= \|[\mathbf{R}^{(k)}]\|$;
    $[\Delta\mathbf{\Phi}^{(k)}] = -K^{(k)-1}[\mathbf{R}^{(k)}]$;
    $[\mathbf{\Phi}^{(k+1)}] = \text{update}([\mathbf{\Phi}^{(k)}], [\Delta\mathbf{\Phi}^{(k)}])$;
  end while

end for

L'algoritmo presentato definisce un'analisi in *controllo di forza* (*load control*).

Capitolo 4

Il codice MATLAB

In questo capitolo vengono descritte le variabili principali e le diverse classi e function caratterizzanti il codice, scritto in MATLAB, il quale è interamente riportato.

4.1 Descrizione del codice

Per facilitare il richiamo ordinato delle variabili all'interno del codice, sono state create diverse classi, contenute nella classe `Global`, costituite dalle seguenti variabili:

- `Mesh`:
  - `TotSpaceDimension`: 3, le dimensioni dello spazio fisico;
  - `TotNodes`: numero totale dei nodi presenti nella mesh;
  - `TotElements`: numero totale degli elementi presenti nella mesh;
  - `TotDof`: numero totale dei gradi di libertà presenti nella mesh;
  - `TotDofNode`: 5, numero totale dei gradi di libertà per ogni nodo;
  - `TotNodesElement`: 4, numero totale dei nodi per ogni elemento;
  - `TotDofSurfaceElement`: 12, numero totale dei gradi di libertà relativi alla superficie media presenti in ogni elemento;
  - `TotDofDirectorElement`: 8, numero totale dei gradi di libertà relativi al vettore direttore presenti in ogni elemento;
  - `TotDofElement`: 20, numero totale dei gradi di libertà presenti in ogni elemento;
  - `TotDofSurface`: numero totale dei gradi di libertà relativi alla superficie media presenti nella mesh;

- `TotDofDirector`: numero totale dei gradi di libertà relativi al vettore direttore presenti nella mesh;
- `Connectivity`: matrice di dimensioni `TotNodes`$\times 4$, dove l'I-esima riga contiene la numerazione dei nodi afferenti all'elemento I-esimo);

- `Element`:
  - `TotGaussPointsMembraneAndBending`: 4, il numero di punti Gauss per ogni elemento relativi alla variabili membranali e flessionali;
  - `TotGaussPointsShear`: 1, il numero di punti Gauss per ogni elemento relativi alla variabili taglianti;

- `Geometry`:
  - `Height`: lo spessore dello shell h;
  - `ShearFactor`: $5/6$, il fattore di correzione per il taglio κ;

- `Elastic`:
  - `ElasticModule`: modulo di Young E del materiale;
  - `PoissonCoefficient`: coefficiente di Poisson ν del materiale;

- `Configuration`:
  - `x0Nodes`: matrice di dimensioni `TotNodes`$\times 3$, dove l'I-esima riga contiene le componenti cartesiane nella configurazione iniziale, rispettivamente nella prima, seconda e terza colonna, relative al vettore posizione della superficie media $\boldsymbol{\varphi}_I^0 = \varphi_I^{0i}\mathbf{E}_i$ del nodo numero I;

$$\texttt{x0Nodes} = \begin{bmatrix} \vdots \\ [\boldsymbol{\varphi}_I^0]^T \\ \vdots \end{bmatrix}$$

$$[\boldsymbol{\varphi}_I^0] = \begin{bmatrix} \varphi_I^{01} & \varphi_I^{02} & \varphi_I^{03} \end{bmatrix}^T$$

  - `t0Nodes`: matrice di dimensioni `TotNodes`$\times 3$, dove l'I-esima riga contiene le componenti cartesiane nella configurazione iniziale, rispettivamente nella prima, seconda e terza colonna, relative al

vettore direttore $\mathbf{t}_I = t_I^{0i}\mathbf{E}_i$ del nodo numero I;

$$\texttt{t0Nodes} = \begin{bmatrix} \vdots \\ [\mathbf{t}_I^0]^T \\ \vdots \end{bmatrix}$$

$$[\mathbf{t}_I^0] = \begin{bmatrix} t_I^{01} & t_I^{02} & t_I^{03} \end{bmatrix}^T$$

- `t0GaussPointsMembraneAndBending`: matrice che salva le componenti cartesiane nella configurazione iniziale del vettore direttore $\mathbf{t}^0 = t^{0i}\mathbf{E}_i$ su ogni punto Gauss relativo alle variabili membranali e flessionali di ogni elemento. L'ordine delle righe, a gruppi di tre, definisce l'ordine degli elementi (pedice E), l'ordine delle colonne definisce l'ordine dei punti Gauss relativi all'elemento (pedice GP);

$$\texttt{t0GaussPointsMembraneAndBending} =$$

$$= \begin{bmatrix} & \vdots & \\ [\mathbf{t}^0]_{\mathrm{E=I,GP=1}} & \cdots & [\mathbf{t}^0]_{\mathrm{E=I,GP=4}} \\ & \vdots & \end{bmatrix}$$

$$[\mathbf{t}^0] = \begin{bmatrix} t^{01} & t^{02} & t^{03} \end{bmatrix}^T$$

- `t0GaussPointsShear`: matrice che salva le componenti cartesiane nella configurazione iniziale del vettore direttore $\mathbf{t}^0 = t^{0i}\mathbf{E}_i$ su ogni punto Gauss relativo alle variabili taglianti di ogni elemento. L'ordine di righe e colonne è analogo a quello definito per la matrice `t0GaussPointsMembraneAndBending` (in questo caso, è presente un unico punto Gauss per ogni elemento);

$$\texttt{t0GaussShear} = \begin{bmatrix} \vdots \\ [\mathbf{t}^0]_{\mathrm{E=I,GP=1}} \\ \vdots \end{bmatrix}$$

$$[\mathbf{t}^0] = \begin{bmatrix} t^{01} & t^{02} & t^{03} \end{bmatrix}^T$$

- `t0_1GaussPointsMembraneAndBending`: matrice che salva le componenti cartesiane nella configurazione iniziale della derivata lungo ξ^1 del vettore direttore $\mathbf{t}_{,1}^0 = t_{,1}^{0i}\mathbf{E}_i$ su ogni punto Gauss relativo alle variabili membranali e flessionali di ogni elemento. L'ordine di righe e colonne è analogo a quello definito per la matrice

`t0GaussPointsMembraneAndBending`;

$$\texttt{t0\_1GaussPointsMembraneAndBending} =$$

$$= \begin{bmatrix} & \vdots & \\ [\mathbf{t}^0_{,1}]_{\text{E=I,GP=1}} & \cdots & [\mathbf{t}^0_{,1}]_{\text{E=I,GP=4}} \\ & \vdots & \end{bmatrix}$$

$$[\mathbf{t}^0_{,1}] = \begin{bmatrix} t^{01}_{,1} & t^{02}_{,1} & t^{03}_{,1} \end{bmatrix}^T$$

- `t0_2GaussPointsMembraneAndBending`: matrice che salva le componenti cartesiane nella configurazione iniziale della derivata lungo ξ^2 del vettore direttore $\mathbf{t}^0_{,2} = t^{0i}_{,2}\mathbf{E}_i$ su ogni punto Gauss relativo alle variabili membranali e flessionali di ogni elemento. L'ordine di righe e colonne è analogo a quello definito per la matrice `t0GaussPointsMembraneAndBending`;

$$\texttt{t0\_2GaussPointsMembraneAndBending} =$$

$$= \begin{bmatrix} & \vdots & \\ [\mathbf{t}^0_{,2}]_{\text{E=I,GP=1}} & \cdots & [\mathbf{t}^0_{,2}]_{\text{E=I,GP=4}} \\ & \vdots & \end{bmatrix}$$

$$[\mathbf{t}^0_{,2}] = \begin{bmatrix} t^{01}_{,2} & t^{02}_{,2} & t^{03}_{,2} \end{bmatrix}^T$$

- `R0Nodes`: matrice che salva le componenti cartesiane nella configurazione iniziale del tensore $\mathbf{\Lambda}^0_I = \Lambda^{0ij}_I \mathbf{E}_i \otimes \mathbf{E}_j$ su ogni nodo della mesh. L'ordine delle righe, a gruppi di tre, definisce l'ordine dei nodi;

$$\texttt{R0Nodes} = \begin{bmatrix} \vdots \\ [\mathbf{\Lambda}^0_I] \\ \vdots \end{bmatrix}$$

$$[\mathbf{\Lambda}^0_I] = \begin{bmatrix} \Lambda^{0,11}_I & \Lambda^{0,12}_I & \Lambda^{0,13}_I \\ \Lambda^{0,21}_I & \Lambda^{0,22}_I & \Lambda^{0,23}_I \\ \Lambda^{0,31}_I & \Lambda^{0,32}_I & \Lambda^{0,33}_I \end{bmatrix}$$

- `xNodes`: matrice analoga a `x0Nodes`, ma in riferimento alla configurazione deformata;

$$\texttt{xNodes} = \begin{bmatrix} \vdots \\ [\boldsymbol{\varphi}_I]^T \\ \vdots \end{bmatrix}$$

$$[\boldsymbol{\varphi}] = \begin{bmatrix} \varphi^1_I & \varphi^2_I & \varphi^3_I \end{bmatrix}^T$$

- `tNodes`: matrice analoga a `t0Nodes`, ma in riferimento alla configurazione deformata;

$$\texttt{tNodes} = \begin{bmatrix} \vdots \\ [\mathbf{t}_I]^T \\ \vdots \end{bmatrix}$$

$$[\mathbf{t}_I] = \begin{bmatrix} t_I^1 & t_I^2 & t_I^3 \end{bmatrix}^T$$

- `tGaussPointsMembraneAndBending`: matrice analoga a `t0GaussPointsMembraneAndBending`, ma in riferimento alla configurazione deformata;

$$\texttt{tGaussPointsMembraneAndBending} =$$

$$= \begin{bmatrix} & \vdots & \\ [\mathbf{t}]_{\mathrm{E=I,GP=1}} & \cdots & [\mathbf{t}]_{\mathrm{E=I,GP=4}} \\ & \vdots & \end{bmatrix}$$

$$[\mathbf{t}] = \begin{bmatrix} t^1 & t^2 & t^3 \end{bmatrix}^T$$

- `tGaussPointsShear`: matrice analoga a `t0GaussPointsShear`, ma in riferimento alla configurazione deformata;

$$\texttt{tGaussShear} = \begin{bmatrix} \vdots \\ [\mathbf{t}]_{\mathrm{E=I,GP=1}} \\ \vdots \end{bmatrix}$$

$$[\mathbf{t}] = \begin{bmatrix} t^1 & t^2 & t^3 \end{bmatrix}^T$$

- `t_1GaussPointsMembraneAndBending`: matrice analoga a `t0_1GaussPointsMembraneAndBending`, ma in riferimento alla configurazione deformata;

$$\texttt{t\_1GaussPointsMembraneAndBending} =$$

$$= \begin{bmatrix} & \vdots & \\ [\mathbf{t}_{,1}]_{\mathrm{E=I,GP=1}} & \cdots & [\mathbf{t}_{,1}]_{\mathrm{E=I,GP=4}} \\ & \vdots & \end{bmatrix}$$

$$[\mathbf{t}_{,1}] = \begin{bmatrix} t_{,1}^1 & t_{,1}^2 & t_{,1}^3 \end{bmatrix}^T$$

- `t_2GaussPointsMembraneAndBending`: matrice analoga a `t0_2GaussPointsMembraneAndBending`, ma in riferimento alla con-

figurazione deformata;

$$\texttt{t\_2GaussPointsMembraneAndBending} = \\ = \begin{bmatrix} & \vdots & \\ [\mathbf{t}_{,2}]_{\mathrm{E=I,GP=1}} & \cdots & [\mathbf{t}_{,2}]_{\mathrm{E=I,GP=4}} \\ & \vdots & \end{bmatrix} \\ [\mathbf{t}_{,2}] = \begin{bmatrix} t^1_{,2} & t^2_{,2} & t^3_{,2} \end{bmatrix}^T$$

- `RNodes`: matrice analoga a `R0Nodes`, ma in riferimento alla configurazione deformata;

$$\texttt{RNodes} = \begin{bmatrix} \vdots \\ [\mathbf{\Lambda}_I] \\ \vdots \end{bmatrix} \\ [\mathbf{\Lambda}_I] = \begin{bmatrix} \Lambda_I^{11} & \Lambda_I^{12} & \Lambda_I^{13} \\ \Lambda_I^{21} & \Lambda_I^{22} & \Lambda_I^{23} \\ \Lambda_I^{31} & \Lambda_I^{32} & \Lambda_I^{33} \end{bmatrix}$$

- `External`:
  - `FreeDof`: matrice colonna contenente la numerazione dei gradi di libertà della mesh non vincolati. La numerazione globale dei gradi di libertà considera, secondo la numerazione globale dei nodi, prima tutti quelli relativi alla superficie media, per poi considerare tutti quelli relativi al vettore direttore;
  - `ExternalForces`: matrice colonna contenente le componenti cartesiane delle forze estene $\mathbf{n}^I_{\mathrm{ext}} = n^{Ii}_{\mathrm{ext}}\mathbf{E}_i$, $\bar{\mathbf{m}}^I_{\mathrm{ext}} = \bar{m}^{Ii}_{\mathrm{ext}}\mathbf{E}_i$, applicate ai gradi di libertà;

$$\texttt{ExternalForces} = \begin{bmatrix} [\mathbf{n}^1_{\mathrm{ext}}]^T & \cdots & [\mathbf{n}^{\mathrm{N_N}}_{\mathrm{ext}}]^T & [\bar{\mathbf{m}}^1_{\mathrm{ext}}]^T & \cdots & [\bar{\mathbf{m}}^{\mathrm{N_N}}_{\mathrm{ext}}]^T \end{bmatrix}^T \\ [\mathbf{n}^I_{\mathrm{ext}}] = \begin{bmatrix} n^{I1}_{\mathrm{ext}} & n^{I2}_{\mathrm{ext}} & n^{I3}_{\mathrm{ext}} \end{bmatrix} \qquad [\bar{\mathbf{m}}^I_{\mathrm{ext}}] = \begin{bmatrix} \bar{m}^{I1}_{\mathrm{ext}} & \bar{m}^{I2}_{\mathrm{ext}} \end{bmatrix}$$

  - `ExternalForce`: valore massimo della forza esterna considerato.
- `Solver`:
  - `Tollerance`: tolleranza per la soluzione del sistema non linerare;
  - `MaxIterations`: numero massimo di iterazioni concesse alla soluzione iterativa dei sistemi non lineari;

    - `LoadSteps`: numero di step di carico.

- `Plot`:

    - `Movie`: struttura che salva i frame di ogni deformata per visualizzarne l'animazione;
    - `SavexNodesx`: matrice che, in ogni riga, salva il vettore della posizione lungo x di ogni nodo per ogni step di carico al fine del plottaggio dei grafici finali;
    - `SavexNodesy`: matrice che, in ogni riga, salva il vettore della posizione lungo y di ogni nodo per ogni step di carico al fine del plottaggio dei grafici finali;
    - `SavexNodesz`: matrice che, in ogni riga, salva il vettore della posizione lungo z di ogni nodo per ogni step di carico al fine del plottaggio dei grafici finali;
    - `NodePlot`: grado di libertà scelto per il plottaggio finale del diagramma forza-spostamento;
    - `DisplacementPlot`: spostamento del grado di libertà `NodePlot` per ogni step di carico;
    - `ForcePlot`: valore della forza esterna considerata per ogni step di carico;

Per ogni applicazione, presentate nel capitolo successivo, è stato implementato un script che risolve il problema proposto. Le function implementate nello script sono le seguenti:

- `Generate`: inizializza i dati presenti nelle varie classi a partire dai dati inseriti nell' `InputFile`;
- `NonLinearSolver`: risolve il sistema non lineare secondo lo schema iterativo di Newton-Raphson;
- `GaussPoint`: restituisce le coordinate dei punti Gauss nel sistema di riferimento locale;
- `ShapeFunction`: resituisce il valore delle funzioni di forma N in funzione delle coordinate ξ^1, ξ^2;
- `DerShapeFunction`: resituisce il valore della derivata delle funzioni di forma rispetto alle coordinate ξ^1, ξ^2 in funzione delle coordinate ξ^1, ξ^2;

- `StiffnessMatrix`: calcola la matrice di rigidezza K_{E} relativa ad un singolo elemento;
- `InternalForces`: calcola il vettore delle forze interne $[\mathbf{F}^{\mathrm{E}}_{\mathrm{int}}]$ relativo ad un singolo elemento;
- `AssembleStiffnessMatrix`: posiziona le componenti della matrice di rigidezza relativa ad un elemento nelle componenti della matrice di rigidezza globale K;
- `AssembleInternalForces`: posiziona le componenti del vettore delle forze interne relativa ad un elemento nelle componenti del vettore delle forze interne globale $[\mathbf{F}_{\mathrm{int}}]$;
- `UpdateSurface`: ad ogni iterazione, esegue l'aggiornamento della variabile $\boldsymbol{\varphi}_I$ in ogni nodo della mesh;
- `UpdateDirector`: ad ogni iterazione, esegue l'aggiornamento delle variabili $\mathbf{t}_I$, $\boldsymbol{\Lambda}_I$ in ogni nodo della mesh, e delle variabili $\mathbf{t}$, $\mathbf{t}_{,1}$, $\mathbf{t}_{,2}$ in ogni punto Gauss per ogni elemento;
- `ExpMap`: esegue la mappa esponenziale relativa alla rotazione del vettore direttore $\mathbf{t}$;
- `DerExpMap`: esegue la mappa esponenziale relativa alle derivate del vettore direttore $\mathbf{t}_{,\alpha=1,2}$;
- `ElasticCovariantTensor`: calcola le componenti del tensore costitutivo covariante $\mathbb{H}$;
- `PlotConfigurations`: plotta la mesh deformata ad ogni step di carico.

4.2 Il codice di calcolo

4.2.1 Class

Global

```
classdef Global < handle
    properties
        Geometry;
        Element;
        Elastic;
        Mesh;
        Configuration;
        External;
        Solver;
```

```
        Plot;
    end
    methods
        function obj = Global()
        end
    end
end
```

Mesh

```
classdef Mesh < handle
    properties
        TotSpaceDimensions;
        TotNodes;
        TotElements;
        TotDof;
        TotDofSurfaceNode;
        TotDofDirectorNode;
        TotDofNode;
        TotNodesElement;
        TotDofSurfaceElement;
        TotDofDirectorElement;
        TotDofElement;
        TotDofSurface;
        TotDofDirector;
        Connectivity;
    end
    methods
        function obj = Mesh()
        end
    end
end
```

Element

```
classdef Element < handle
    properties
        TotGaussPointsMembraneAndBending;
        TotGaussPointsShear;
    end
    methods
        function obj = Element()
        end
    end
end
```

Geometry

```
classdef Geometry < handle
    properties
        Height;
        ShearFactor;
    end
    methods
        function obj = Geometry()
        end
    end
end
```

Elastic

```
classdef Elastic < handle
    properties
        ElasticModule;
        PoissonCoefficient;
    end
    methods
        function obj = Elastic()
        end
    end
end
```

Configuration

```
classdef Configuration < handle
    properties
        x0Nodes;
        t0Nodes;
        t0GaussPointsMembraneAndBending;
        t0GaussPointsShear;
        t0_1GaussPointsMembraneAndBending;
        t0_2GaussPointsMembraneAndBending;
        R0Nodes;
        xNodes;
        tNodes;
        tGaussPointsMembraneAndBending;
        tGaussPointsShear;
        t_1GaussPointsMembraneAndBending;
        t_2GaussPointsMembraneAndBending;
        RNodes;
    end
    methods
        function obj = Configuration()
        end
    end
end
```

External

```
classdef External < handle
    properties
        FreeDof;
        ExternalForces;
        ExternalForce;
    end
    methods
        function obj = External()
        end
    end
end
```

Solver

```
classdef Solver < handle
    properties
        Tollerance;
        MaxIterations;
        LoadSteps;
    end
```

```
    methods
        function obj = Solver()
        end
    end
end
```

Plot

```
classdef Plot < handle
        properties
        Movie;
        SavexNodesx;
        SavexNodesy;
        SavexNodesz;
        NodePlot;
        DisplacementPlot;
        ForcePlot;
    end
    methods
        function obj = Plot()
        end
    end
end
```

4.2.2 Function

Generate

```
function [GenerateGlobal] = Generate(InputFile)
%-------------------------------------------------------------------------------------------%
eval(InputFile);
%-------------------------------------------------------------------------------------------%
GenerateGlobal = Global();
%-------------------------------------------------------------------------------------------%
GenerateGlobal.Mesh = Mesh();
GenerateGlobal.Mesh.TotSpaceDimensions = TotSpaceDimensions;
GenerateGlobal.Mesh.TotNodes = TotNodes;
GenerateGlobal.Mesh.TotElements = TotElements;
GenerateGlobal.Mesh.TotDof = TotDof;
GenerateGlobal.Mesh.TotDofSurfaceNode = TotDofSurfaceNode;
GenerateGlobal.Mesh.TotDofDirectorNode = TotDofDirectorNode;
GenerateGlobal.Mesh.TotDofNode = TotDofNode;
GenerateGlobal.Mesh.TotNodesElement = TotNodesElement;
GenerateGlobal.Mesh.TotDofSurfaceElement = TotDofSurfaceElement;
GenerateGlobal.Mesh.TotDofDirectorElement = TotDofDirectorElement;
GenerateGlobal.Mesh.TotDofElement = TotDofElement;
GenerateGlobal.Mesh.TotDofSurface = TotDofSurface;
GenerateGlobal.Mesh.TotDofDirector = TotDofDirector;
GenerateGlobal.Mesh.Connectivity = Connectivity;
%-------------------------------------------------------------------------------------------%
GenerateGlobal.Element = Element();
GenerateGlobal.Element.TotGaussPointsMembraneAndBending = TotGaussPointsMembraneAndBending;
GenerateGlobal.Element.TotGaussPointsShear = TotGaussPointsShear;
%-------------------------------------------------------------------------------------------%
GenerateGlobal.Elastic = Elastic();
GenerateGlobal.Elastic.ElasticModule = ElasticModule;
GenerateGlobal.Elastic.PoissonCoefficient = PoissonCoefficient;
%-------------------------------------------------------------------------------------------%
GenerateGlobal.Geometry = Geometry();
GenerateGlobal.Geometry.Height = Height;
```

```
GenerateGlobal.Geometry.ShearFactor = ShearFactor;
%-------------------------------------------------------------------------------------------%
GenerateGlobal.Configuration = Configuration();
GenerateGlobal.Configuration.x0Nodes = x0Nodes;
GenerateGlobal.Configuration.t0Nodes = t0Nodes;
GenerateGlobal.Configuration.t0GaussPointsMembraneAndBending = ...
    t0GaussPointsMembraneAndBending;
GenerateGlobal.Configuration.t0GaussPointsShear = t0GaussPointsShear;
GenerateGlobal.Configuration.t0_1GaussPointsMembraneAndBending = ...
    t0_1GaussPointsMembraneAndBending;
GenerateGlobal.Configuration.t0_2GaussPointsMembraneAndBending = ...
    t0_2GaussPointsMembraneAndBending;
GenerateGlobal.Configuration.R0Nodes = R0Nodes;
GenerateGlobal.Configuration.xNodes = xNodes;
GenerateGlobal.Configuration.tNodes = tNodes;
GenerateGlobal.Configuration.tGaussPointsMembraneAndBending = ...
    tGaussPointsMembraneAndBending;
GenerateGlobal.Configuration.tGaussPointsShear = tGaussPointsShear;
GenerateGlobal.Configuration.t_1GaussPointsMembraneAndBending = ...
    t_1GaussPointsMembraneAndBending;
GenerateGlobal.Configuration.t_2GaussPointsMembraneAndBending = ...
    t_2GaussPointsMembraneAndBending;
GenerateGlobal.Configuration.RNodes = RNodes;
%-------------------------------------------------------------------------------------------%
GenerateGlobal.External = External();
GenerateGlobal.External.FreeDof = FreeDof;
GenerateGlobal.External.ExternalForces = ExternalForces;
GenerateGlobal.External.ExternalForce = ExternalForce;
%-------------------------------------------------------------------------------------------%
GenerateGlobal.Solver = Solver();
GenerateGlobal.Solver.Tollerance = Tollerance;
GenerateGlobal.Solver.MaxIterations = MaxIterations;
GenerateGlobal.Solver.LoadSteps = LoadSteps;
%-------------------------------------------------------------------------------------------%
GenerateGlobal.Plot = Plot();
GenerateGlobal.Plot.Movie = Movie;
GenerateGlobal.Plot.SavexNodesx = SavexNodesx;
GenerateGlobal.Plot.SavexNodesy = SavexNodesy;
GenerateGlobal.Plot.SavexNodesz = SavexNodesz;
GenerateGlobal.Plot.NodePlot = NodePlot;
GenerateGlobal.Plot.DisplacementPlot = DisplacementPlot;
GenerateGlobal.Plot.ForcePlot = ForcePlot;
%-------------------------------------------------------------------------------------------%
end
```

NonLinearSolver

```
function NonLinearSolver(Global)
%-------------------------------------------------------------------------------------------%
fExtTotal = Global.External.ExternalForces;
DfExt = fExtTotal/Global.Solver.LoadSteps;
FreeDof = Global.External.FreeDof;
Du = zeros(Global.Mesh.TotDof,1);
%-------------------------------------------------------------------------------------------%
for Step = 1:Global.Solver.LoadSteps
    Iteration = 0;
    fExt = DfExt*Step;
    %---------------------------------------------------------------------------------------%
    Iteration = Iteration+1;
    fInt = zeros(Global.Mesh.TotDof,1);
    K = zeros(Global.Mesh.TotDof,Global.Mesh.TotDof);
```

```
    for ElementNumber = 1:Global.Mesh.TotElements
        KElement = StiffnessMatrix(Global,ElementNumber);
        fIntElement = InternalForces(Global,ElementNumber);
        K = AssembleStiffnessMatrix(K,KElement,Global,ElementNumber);
        fInt = AssembleInternalForces(fInt,fIntElement,Global,ElementNumber);
    end
    Res = fInt-fExt; Err = norm(Res(FreeDof),2);
    Du(FreeDof,1) = -K(FreeDof,FreeDof)\Res(FreeDof);
    DxNodes = Du(1:Global.Mesh.TotDofSurface);
    DTNodes = Du(Global.Mesh.TotDofSurface+1:Global.Mesh.TotDof,1);
    UpdateSurface(Global,DxNodes); UpdateDirector(Global,DTNodes);
    while Err > Global.Solver.Tollerance && Iteration < Global.Solver.MaxIterations
        Iteration = Iteration+1;
        fInt = zeros(Global.Mesh.TotDof,1);
        K = zeros(Global.Mesh.TotDof,Global.Mesh.TotDof);
        for ElementNumber = 1:Global.Mesh.TotElements
            KElement = StiffnessMatrix(Global,ElementNumber);
            fIntElement = InternalForces(Global,ElementNumber);
            K = AssembleStiffnessMatrix(K,KElement,Global,ElementNumber);
            fInt = AssembleInternalForces(fInt,fIntElement,Global,ElementNumber);
        end
        Res = fInt-fExt; Err = norm(Res(FreeDof),2);
        Du(FreeDof,1) = -K(FreeDof,FreeDof)\Res(FreeDof);
        DxNodes = Du(1:Global.Mesh.TotDofSurface);
        DTNodes = Du(Global.Mesh.TotDofSurface+1:Global.Mesh.TotDof,1);
        UpdateSurface(Global,DxNodes); UpdateDirector(Global,DTNodes);
    end
    if Iteration >= Global.Solver.MaxIterations
        error('Reached max number of iterations');
    end
    %-----------------------------------------------------------------------------------%
    Global.Plot.SavexNodesx(:,Step+1) = Global.Configuration.xNodes(:,1);
    Global.Plot.SavexNodesy(:,Step+1) = Global.Configuration.xNodes(:,2);
    Global.Plot.SavexNodesz(:,Step+1) = Global.Configuration.xNodes(:,3);
    %-----------------------------------------------------------------------------------%
    Global.Plot.ForcePlot(Step+1,1) = ...
        Global.External.ExternalForce/Global.Solver.LoadSteps*Step;
    Global.Plot.DisplacementPlot(Step+1,1) = ...
        -(Global.Configuration.xNodes(Global.Plot.NodePlot(1,1),3) ...
        -Global.Configuration.x0Nodes(Global.Plot.NodePlot(1,1),3));
    %-----------------------------------------------------------------------------------%
end
%-----------------------------------------------------------------------------------%
end
```

GaussPoint

```
function [Point,Weight] = GaussPoint(GaussPointNumber,TotGaussPoints)
%-----------------------------------------------------------------------------------%
switch TotGaussPoints
    case(1)
        Point = [0 0]';
        Weight = 4;
    case(4)
        sqrt3on3 = sqrt(3)/3;
        Weight = 1;
        switch(GaussPointNumber)
            case(1)
                Point = [-sqrt3on3 -sqrt3on3]';
            case(2)
                Point = [sqrt3on3 -sqrt3on3]';
```

```
            case(3)
                Point = [sqrt3on3 sqrt3on3]';
            case(4)
                Point = [-sqrt3on3 sqrt3on3]';
        end
end
%-------------------------------------------------------------------------------------------%
end
```

ShapeFunction

```
function sf = ShapeFunction(sfNumber,Point)
%-------------------------------------------------------------------------------------------%
xi = Point(1,1); eta = Point(2,1);
switch(sfNumber)
    case(1)
        sf = 0.25*(1-xi)*(1-eta);
    case(2)
        sf = 0.25*(1+xi)*(1-eta);
    case(3)
        sf = 0.25*(1+xi)*(1+eta);
    case(4)
        sf = 0.25*(1-xi)*(1+eta);
end
%-------------------------------------------------------------------------------------------%
end
```

DerShapeFunction

```
function Dsf = DerShapeFunction(sfNumber,Direction,Point)
%-------------------------------------------------------------------------------------------%
xi = Point(1,1); eta = Point(2,1);
switch(sfNumber)
    case(1)
        switch(Direction)
            case(1)
                Dsf = -0.25*(1-eta);
            case(2)
                Dsf = -0.25*(1-xi);
        end
    case(2)
        switch(Direction)
            case(1)
                Dsf = 0.25*(1-eta);
            case(2)
                Dsf = -0.25*(1+xi);
        end
    case(3)
        switch(Direction)
            case(1)
                Dsf = 0.25*(1+eta);
            case(2)
                Dsf = 0.25*(1+xi);
        end
    case(4)
        switch(Direction)
            case(1)
                Dsf = -0.25*(1+eta);
            case(2)
                Dsf = 0.25*(1-xi);
```

```
        end
end
%-------------------------------------------------------------------------------------------%
end
```

StiffnessMatrix

```
function K = StiffnessMatrix(Global,ElementNumber)
%-------------------------------------------------------------------------------------------%
x0Nodes = Global.Configuration.x0Nodes;
xNodes = Global.Configuration.xNodes;
tNodes = Global.Configuration.tNodes;
RNodes = Global.Configuration.RNodes;
Connectivity = Global.Mesh.Connectivity;
Node1 = Connectivity(ElementNumber,1); Node2 = Connectivity(ElementNumber,2);
Node3 = Connectivity(ElementNumber,3); Node4 = Connectivity(ElementNumber,4);
x01 = x0Nodes(Node1,:)'; x02 = x0Nodes(Node2,:)';
x03 = x0Nodes(Node3,:)'; x04 = x0Nodes(Node4,:)';
x1 = xNodes(Node1,:)'; x2 = xNodes(Node2,:)'; x3 = xNodes(Node3,:)'; x4 = xNodes(Node4,:)';
t1 = tNodes(Node1,:)'; t2 = tNodes(Node2,:)'; t3 = tNodes(Node3,:)'; t4 = tNodes(Node4,:)';
R1 = RNodes(3*(Node1-1)+1:3*Node1,:); R2 = RNodes(3*(Node2-1)+1:3*Node2,:);
R3 = RNodes(3*(Node3-1)+1:3*Node3,:); R4 = RNodes(3*(Node4-1)+1:3*Node4,:);
Rbar1 = R1(:,1:2); Rbar2 = R2(:,1:2); Rbar3 = R3(:,1:2); Rbar4 = R4(:,1:2);
t0GPMB = Global.Configuration.t0GaussPointsMembraneAndBending ...
    (3*(ElementNumber-1)+1:3*ElementNumber,:);
t0GPS = Global.Configuration.t0GaussPointsShear(3*(ElementNumber-1)+1:3*ElementNumber,:);
t0_1GPMB = Global.Configuration.t0_1GaussPointsMembraneAndBending ...
    (3*(ElementNumber-1)+1:3*ElementNumber,:);
t0_2GPMB = Global.Configuration.t0_2GaussPointsMembraneAndBending ...
    (3*(ElementNumber-1)+1:3*ElementNumber,:);
tGPS = Global.Configuration.tGaussPointsShear(3*(ElementNumber-1)+1:3*ElementNumber,:);
t_1GPMB = Global.Configuration.t_1GaussPointsMembraneAndBending ...
    (3*(ElementNumber-1)+1:3*ElementNumber,:);
t_2GPMB = Global.Configuration.t_2GaussPointsMembraneAndBending ...
    (3*(ElementNumber-1)+1:3*ElementNumber,:);
TotGPMB = Global.Element.TotGaussPointsMembraneAndBending;
TotGPS = Global.Element.TotGaussPointsShear;
I = eye(3); O2x2 = zeros(2,2); O8x8 = zeros(8,8);
O12x12 = zeros(12,12); O12x8 = zeros(12,8); O8x12 = zeros(8,12);
KgMB = zeros(20,20); KmMB = zeros(20,20); KgS = zeros(20,20); KmS = zeros(20,20);
%-------------------------------------------------------------------------------------------%
% Membrane and bending contribution
%-------------------------------------------------------------------------------------------%
for GPNumber = 1:TotGPMB
    [GP,Weight] = GaussPoint(GPNumber,TotGPMB);
    N1_1 = DerShapeFunction(1,1,GP); N1_2 = DerShapeFunction(1,2,GP);
    N2_1 = DerShapeFunction(2,1,GP); N2_2 = DerShapeFunction(2,2,GP);
    N3_1 = DerShapeFunction(3,1,GP); N3_2 = DerShapeFunction(3,2,GP);
    N4_1 = DerShapeFunction(4,1,GP); N4_2 = DerShapeFunction(4,2,GP);
    x_1 = x1*N1_1+x2*N2_1+x3*N3_1+x4*N4_1;
    x_2 = x1*N1_2+x2*N2_2+x3*N3_2+x4*N4_2;
    x0_1 = x01*N1_1+x02*N2_1+x03*N3_1+x04*N4_1;
    x0_2 = x01*N1_2+x02*N2_2+x03*N3_2+x04*N4_2;
    t_1 = t_1GPMB(:,GPNumber);
    t_2 = t_2GPMB(:,GPNumber);
    t0 = t0GPMB(:,GPNumber);
    t0_1 = t0_1GPMB(:,GPNumber);
    t0_2 = t0_2GPMB(:,GPNumber);
    j0bar = norm(cross(x0_1,x0_2),2);
    %-------------------------------------------------------------------------------------------%
    % Elastic covariant tensor
```

```
%-------------------------------------------------------------------------------------------%
[Cn,Cm,~] = ElasticCovariantTensor(x0_1,x0_2,t0,Global);
Cn11 = Cn(1,1); Cn12 = Cn(1,2); Cn13 = Cn(1,3);
Cn21 = Cn(2,1); Cn22 = Cn(2,2); Cn23 = Cn(2,3);
Cn31 = Cn(3,1); Cn32 = Cn(3,2); Cn33 = Cn(3,3);
Cm11 = Cm(1,1); Cm12 = Cm(1,2); Cm13 = Cm(1,3);
Cm21 = Cm(2,1); Cm22 = Cm(2,2); Cm23 = Cm(2,3);
Cm31 = Cm(3,1); Cm32 = Cm(3,2); Cm33 = Cm(3,3);
%-------------------------------------------------------------------------------------------%
% Strain
%-------------------------------------------------------------------------------------------%
e11 = 0.5*(x_1'*x_1-x0_1'*x0_1);
e22 = 0.5*(x_2'*x_2-x0_2'*x0_2);
e12 = (x_1'*x_2-x0_1'*x0_2);
k11 = x_1'*t_1-x0_1'*t0_1;
k22 = x_2'*t_2-x0_2'*t0_2;
k12 = (x_1'*t_2-x0_1'*t0_2)+(x_2'*t_1-x0_2'*t0_1);
%-------------------------------------------------------------------------------------------%
% Stress
%-------------------------------------------------------------------------------------------%
n11 = Cn11*e11+Cn12*e22+Cn13*e12;
n22 = Cn21*e11+Cn22*e22+Cn23*e12;
n12 = Cn31*e11+Cn32*e22+Cn33*e12;
m11 = Cm11*k11+Cm12*k22+Cm13*k12;
m22 = Cm21*k11+Cm22*k22+Cm23*k12;
m12 = Cm31*k11+Cm32*k22+Cm33*k12;
%-------------------------------------------------------------------------------------------%
% Strain variation
%-------------------------------------------------------------------------------------------%
de11_pos1 = N1_1*x_1; de11_pos2 = N2_1*x_1;
de11_pos3 = N3_1*x_1; de11_pos4 = N4_1*x_1;
de11 = [de11_pos1;de11_pos2;de11_pos3;de11_pos4];
de22_pos1 = N1_2*x_2; de22_pos2 = N2_2*x_2;
de22_pos3 = N3_2*x_2; de22_pos4 = N4_2*x_2;
de22 = [de22_pos1;de22_pos2;de22_pos3;de22_pos4];
de12_pos1 = N1_1*x_2+N1_2*x_1;
de12_pos2 = N2_1*x_2+N2_2*x_1;
de12_pos3 = N3_1*x_2+N3_2*x_1;
de12_pos4 = N4_1*x_2+N4_2*x_1;
de12 = [de12_pos1;de12_pos2;de12_pos3;de12_pos4];
dk11_pos1 = N1_1*t_1; dk11_pos2 = N2_1*t_1;
dk11_pos3 = N3_1*t_1; dk11_pos4 = N4_1*t_1;
dk11_pos5 = N1_1*Rbar1'*x_1; dk11_pos6 = N2_1*Rbar2'*x_1;
dk11_pos7 = N3_1*Rbar3'*x_1; dk11_pos8 = N4_1*Rbar4'*x_1;
dk11 = [dk11_pos1;dk11_pos2;dk11_pos3;dk11_pos4;
    dk11_pos5;dk11_pos6;dk11_pos7;dk11_pos8];
dk22_pos1 = N1_2*t_2; dk22_pos2 = N2_2*t_2;
dk22_pos3 = N3_2*t_2; dk22_pos4 = N4_2*t_2;
dk22_pos5 = N1_2*Rbar1'*x_2; dk22_pos6 = N2_2*Rbar2'*x_2;
dk22_pos7 = N3_2*Rbar3'*x_2; dk22_pos8 = N4_2*Rbar4'*x_2;
dk22 = [dk22_pos1;dk22_pos2;dk22_pos3;dk22_pos4;
    dk22_pos5;dk22_pos6;dk22_pos7;dk22_pos8];
dk12_pos1 = N1_1*t_2+N1_2*t_1;
dk12_pos2 = N2_1*t_2+N2_2*t_1;
dk12_pos3 = N3_1*t_2+N3_2*t_1;
dk12_pos4 = N4_1*t_2+N4_2*t_1;
dk12_pos5 = N1_2*Rbar1'*x_1+N1_1*Rbar1'*x_2;
dk12_pos6 = N2_2*Rbar2'*x_1+N2_1*Rbar2'*x_2;
dk12_pos7 = N3_2*Rbar3'*x_1+N3_1*Rbar3'*x_2;
dk12_pos8 = N4_2*Rbar4'*x_1+N4_1*Rbar4'*x_2;
dk12 = [dk12_pos1;dk12_pos2;dk12_pos3;dk12_pos4;
    dk12_pos5;dk12_pos6;dk12_pos7;dk12_pos8];
```

```
%-------------------------------------------------------------------------------------%
% Stress increment
%-------------------------------------------------------------------------------------%
De11_pos1 = de11_pos1'; De11_pos2 = de11_pos2';
De11_pos3 = de11_pos3'; De11_pos4 = de11_pos4';
De11 = [De11_pos1 De11_pos2 De11_pos3 De11_pos4];
De22_pos1 = de22_pos1'; De22_pos2 = de22_pos2';
De22_pos3 = de22_pos3'; De22_pos4 = de22_pos4';
De22 = [De22_pos1 De22_pos2 De22_pos3 De22_pos4];
De12_pos1 = de12_pos1'; De12_pos2 = de12_pos2';
De12_pos3 = de12_pos3'; De12_pos4 = de12_pos4';
De12 = [De12_pos1 De12_pos2 De12_pos3 De12_pos4];
Dn11 = Cn11*De11+Cn12*De22+Cn13*De12;
Dn22 = Cn21*De11+Cn22*De22+Cn23*De12;
Dn12 = Cn31*De11+Cn32*De22+Cn33*De12;
Dk11_pos1 = dk11_pos1'; Dk11_pos2 = dk11_pos2';
Dk11_pos3 = dk11_pos3'; Dk11_pos4 = dk11_pos4';
Dk11_pos5 = dk11_pos5'; Dk11_pos6 = dk11_pos6';
Dk11_pos7 = dk11_pos7'; Dk11_pos8 = dk11_pos8';
Dk11 = [Dk11_pos1 Dk11_pos2 Dk11_pos3 Dk11_pos4 ...
    Dk11_pos5 Dk11_pos6 Dk11_pos7 Dk11_pos8];
Dk22_pos1 = dk22_pos1'; Dk22_pos2 = dk22_pos2';
Dk22_pos3 = dk22_pos3'; Dk22_pos4 = dk22_pos4';
Dk22_pos5 = dk22_pos5'; Dk22_pos6 = dk22_pos6';
Dk22_pos7 = dk22_pos7'; Dk22_pos8 = dk22_pos8';
Dk22 = [Dk22_pos1 Dk22_pos2 Dk22_pos3 Dk22_pos4 ...
    Dk22_pos5 Dk22_pos6 Dk22_pos7 Dk22_pos8];
Dk12_pos1 = dk12_pos1'; Dk12_pos2 = dk12_pos2';
Dk12_pos3 = dk12_pos3'; Dk12_pos4 = dk12_pos4';
Dk12_pos5 = dk12_pos5'; Dk12_pos6 = dk12_pos6';
Dk12_pos7 = dk12_pos7'; Dk12_pos8 = dk12_pos8';
Dk12 = [Dk12_pos1 Dk12_pos2 Dk12_pos3 Dk12_pos4 ...
    Dk12_pos5 Dk12_pos6 Dk12_pos7 Dk12_pos8];
Dm11 = Cm11*Dk11+Cm12*Dk22+Cm13*Dk12;
Dm22 = Cm21*Dk11+Cm22*Dk22+Cm23*Dk12;
Dm12 = Cm31*Dk11+Cm32*Dk22+Cm33*Dk12;
%-------------------------------------------------------------------------------------%
% Strain increment variation
%-------------------------------------------------------------------------------------%
Dde11_pos11 = N1_1*N1_1*I; Dde11_pos12 = N1_1*N2_1*I;
Dde11_pos13 = N1_1*N3_1*I; Dde11_pos14 = N1_1*N4_1*I;
Dde11_pos21 = N2_1*N1_1*I; Dde11_pos22 = N2_1*N2_1*I;
Dde11_pos23 = N2_1*N3_1*I; Dde11_pos24 = N2_1*N4_1*I;
Dde11_pos31 = N3_1*N1_1*I; Dde11_pos32 = N3_1*N2_1*I;
Dde11_pos33 = N3_1*N3_1*I; Dde11_pos34 = N3_1*N4_1*I;
Dde11_pos41 = N4_1*N1_1*I; Dde11_pos42 = N4_1*N2_1*I;
Dde11_pos43 = N4_1*N3_1*I; Dde11_pos44 = N4_1*N4_1*I;
Dde11 = [Dde11_pos11 Dde11_pos12 Dde11_pos13 Dde11_pos14
         Dde11_pos21 Dde11_pos22 Dde11_pos23 Dde11_pos24
         Dde11_pos31 Dde11_pos32 Dde11_pos33 Dde11_pos34
         Dde11_pos41 Dde11_pos42 Dde11_pos43 Dde11_pos44];
Dde22_pos11 = N1_2*N1_2*I; Dde22_pos12 = N1_2*N2_2*I;
Dde22_pos13 = N1_2*N3_2*I; Dde22_pos14 = N1_2*N4_2*I;
Dde22_pos21 = N2_2*N1_2*I; Dde22_pos22 = N2_2*N2_2*I;
Dde22_pos23 = N2_2*N3_2*I; Dde22_pos24 = N2_2*N4_2*I;
Dde22_pos31 = N3_2*N1_2*I; Dde22_pos32 = N3_2*N2_2*I;
Dde22_pos33 = N3_2*N3_2*I; Dde22_pos34 = N3_2*N4_2*I;
Dde22_pos41 = N4_2*N1_2*I; Dde22_pos42 = N4_2*N2_2*I;
Dde22_pos43 = N4_2*N3_2*I; Dde22_pos44 = N4_2*N4_2*I;
Dde22 = [Dde22_pos11 Dde22_pos12 Dde22_pos13 Dde22_pos14
         Dde22_pos21 Dde22_pos22 Dde22_pos23 Dde22_pos24
         Dde22_pos31 Dde22_pos32 Dde22_pos33 Dde22_pos34
```

```
        Dde22_pos41 Dde22_pos42 Dde22_pos43 Dde22_pos44];
Dde12_pos11 = N1_1*N1_2*I+N1_2*N1_1*I;
Dde12_pos12 = N1_1*N2_2*I+N1_2*N2_1*I;
Dde12_pos13 = N1_1*N3_2*I+N1_2*N3_1*I;
Dde12_pos14 = N1_1*N4_2*I+N1_2*N4_1*I;
Dde12_pos21 = N2_1*N1_2*I+N2_2*N1_1*I;
Dde12_pos22 = N2_1*N2_2*I+N2_2*N2_1*I;
Dde12_pos23 = N2_1*N3_2*I+N2_2*N3_1*I;
Dde12_pos24 = N2_1*N4_2*I+N2_2*N4_1*I;
Dde12_pos31 = N3_1*N1_2*I+N3_2*N1_1*I;
Dde12_pos32 = N3_1*N2_2*I+N3_2*N2_1*I;
Dde12_pos33 = N3_1*N3_2*I+N3_2*N3_1*I;
Dde12_pos34 = N3_1*N4_2*I+N3_2*N4_1*I;
Dde12_pos41 = N4_1*N1_2*I+N4_2*N1_1*I;
Dde12_pos42 = N4_1*N2_2*I+N4_2*N2_1*I;
Dde12_pos43 = N4_1*N3_2*I+N4_2*N3_1*I;
Dde12_pos44 = N4_1*N4_2*I+N4_2*N4_1*I;
Dde12 = [Dde12_pos11 Dde12_pos12 Dde12_pos13 Dde12_pos14
         Dde12_pos21 Dde12_pos22 Dde12_pos23 Dde12_pos24
         Dde12_pos31 Dde12_pos32 Dde12_pos33 Dde12_pos34
         Dde12_pos41 Dde12_pos42 Dde12_pos43 Dde12_pos44];
Ddk11_pos15 = N1_1*N1_1*Rbar1; Ddk11_pos16 = N1_1*N2_1*Rbar2;
Ddk11_pos17 = N1_1*N3_1*Rbar3; Ddk11_pos18 = N1_1*N4_1*Rbar4;
Ddk11_pos25 = N2_1*N1_1*Rbar1; Ddk11_pos26 = N2_1*N2_1*Rbar2;
Ddk11_pos27 = N2_1*N3_1*Rbar3; Ddk11_pos28 = N2_1*N4_1*Rbar4;
Ddk11_pos35 = N3_1*N1_1*Rbar1; Ddk11_pos36 = N3_1*N2_1*Rbar2;
Ddk11_pos37 = N3_1*N3_1*Rbar3; Ddk11_pos38 = N3_1*N4_1*Rbar4;
Ddk11_pos45 = N4_1*N1_1*Rbar1; Ddk11_pos46 = N4_1*N2_1*Rbar2;
Ddk11_pos47 = N4_1*N3_1*Rbar3; Ddk11_pos48 = N4_1*N4_1*Rbar4;
Ddk11_3x2 = [Ddk11_pos15 Ddk11_pos16 Ddk11_pos17 Ddk11_pos18
             Ddk11_pos25 Ddk11_pos26 Ddk11_pos27 Ddk11_pos28
             Ddk11_pos35 Ddk11_pos36 Ddk11_pos37 Ddk11_pos38
             Ddk11_pos45 Ddk11_pos46 Ddk11_pos47 Ddk11_pos48];
Ddk11_pos51 = N1_1*N1_1*Rbar1'; Ddk11_pos52 = N1_1*N2_1*Rbar1';
Ddk11_pos53 = N1_1*N3_1*Rbar1'; Ddk11_pos54 = N1_1*N4_1*Rbar1';
Ddk11_pos61 = N2_1*N1_1*Rbar2'; Ddk11_pos62 = N2_1*N2_1*Rbar2';
Ddk11_pos63 = N2_1*N3_1*Rbar2'; Ddk11_pos64 = N2_1*N4_1*Rbar2';
Ddk11_pos71 = N3_1*N1_1*Rbar3'; Ddk11_pos72 = N3_1*N2_1*Rbar3';
Ddk11_pos73 = N3_1*N3_1*Rbar3'; Ddk11_pos74 = N3_1*N4_1*Rbar3';
Ddk11_pos81 = N4_1*N1_1*Rbar4'; Ddk11_pos82 = N4_1*N2_1*Rbar4';
Ddk11_pos83 = N4_1*N3_1*Rbar4'; Ddk11_pos84 = N4_1*N4_1*Rbar4';
Ddk11_2x3 = [Ddk11_pos51 Ddk11_pos52 Ddk11_pos53 Ddk11_pos54
             Ddk11_pos61 Ddk11_pos62 Ddk11_pos63 Ddk11_pos64
             Ddk11_pos71 Ddk11_pos72 Ddk11_pos73 Ddk11_pos74
             Ddk11_pos81 Ddk11_pos82 Ddk11_pos83 Ddk11_pos84];
Ddk11_pos55 = -N1_1*(t1'*x_1)*(Rbar1'*Rbar1);
Ddk11_pos66 = -N2_1*(t2'*x_1)*(Rbar2'*Rbar2);
Ddk11_pos77 = -N3_1*(t3'*x_1)*(Rbar3'*Rbar3);
Ddk11_pos88 = -N4_1*(t4'*x_1)*(Rbar4'*Rbar4);
Ddk11_2x2 = [Ddk11_pos55 O2x2 O2x2 O2x2
             O2x2 Ddk11_pos66 O2x2 O2x2
             O2x2 O2x2 Ddk11_pos77 O2x2
             O2x2 O2x2 O2x2 Ddk11_pos88];
Ddk22_pos15 = N1_2*N1_2*Rbar1; Ddk22_pos16 = N1_2*N2_2*Rbar2;
Ddk22_pos17 = N1_2*N3_2*Rbar3; Ddk22_pos18 = N1_2*N4_2*Rbar4;
Ddk22_pos25 = N2_2*N1_2*Rbar1; Ddk22_pos26 = N2_2*N2_2*Rbar2;
Ddk22_pos27 = N2_2*N3_2*Rbar3; Ddk22_pos28 = N2_2*N4_2*Rbar4;
Ddk22_pos35 = N3_2*N1_2*Rbar1; Ddk22_pos36 = N3_2*N2_2*Rbar2;
Ddk22_pos37 = N3_2*N3_2*Rbar3; Ddk22_pos38 = N3_2*N4_2*Rbar4;
Ddk22_pos45 = N4_2*N1_2*Rbar1; Ddk22_pos46 = N4_2*N2_2*Rbar2;
Ddk22_pos47 = N4_2*N3_2*Rbar3; Ddk22_pos48 = N4_2*N4_2*Rbar4;
Ddk22_3x2 = [Ddk22_pos15 Ddk22_pos16 Ddk22_pos17 Ddk22_pos18
```

```
             Ddk22_pos25 Ddk22_pos26 Ddk22_pos27 Ddk22_pos28
             Ddk22_pos35 Ddk22_pos36 Ddk22_pos37 Ddk22_pos38
             Ddk22_pos45 Ddk22_pos46 Ddk22_pos47 Ddk22_pos48];
Ddk22_pos51 = N1_2*N1_2*Rbar1'; Ddk22_pos52 = N1_2*N2_2*Rbar1';
Ddk22_pos53 = N1_2*N3_2*Rbar1'; Ddk22_pos54 = N1_2*N4_2*Rbar1';
Ddk22_pos61 = N2_2*N1_2*Rbar2'; Ddk22_pos62 = N2_2*N2_2*Rbar2';
Ddk22_pos63 = N2_2*N3_2*Rbar2'; Ddk22_pos64 = N2_2*N4_2*Rbar2';
Ddk22_pos71 = N3_2*N1_2*Rbar3'; Ddk22_pos72 = N3_2*N2_2*Rbar3';
Ddk22_pos73 = N3_2*N3_2*Rbar3'; Ddk22_pos74 = N3_2*N4_2*Rbar3';
Ddk22_pos81 = N4_2*N1_2*Rbar4'; Ddk22_pos82 = N4_2*N2_2*Rbar4';
Ddk22_pos83 = N4_2*N3_2*Rbar4'; Ddk22_pos84 = N4_2*N4_2*Rbar4';
Ddk22_2x3 = [Ddk22_pos51 Ddk22_pos52 Ddk22_pos53 Ddk22_pos54
             Ddk22_pos61 Ddk22_pos62 Ddk22_pos63 Ddk22_pos64
             Ddk22_pos71 Ddk22_pos72 Ddk22_pos73 Ddk22_pos74
             Ddk22_pos81 Ddk22_pos82 Ddk22_pos83 Ddk22_pos84];
Ddk22_pos55 = -N1_2*(t1'*x_2)*(Rbar1'*Rbar1);
Ddk22_pos66 = -N2_2*(t2'*x_2)*(Rbar2'*Rbar2);
Ddk22_pos77 = -N3_2*(t3'*x_2)*(Rbar3'*Rbar3);
Ddk22_pos88 = -N4_2*(t4'*x_2)*(Rbar4'*Rbar4);
Ddk22_2x2 = [Ddk22_pos55 O2x2 O2x2 O2x2
             O2x2 Ddk22_pos66 O2x2 O2x2
             O2x2 O2x2 Ddk22_pos77 O2x2
             O2x2 O2x2 O2x2 Ddk22_pos88];
Ddk12_pos15 = N1_1*N1_2*Rbar1+N1_2*N1_1*Rbar1;
Ddk12_pos16 = N1_1*N2_2*Rbar2+N1_2*N2_1*Rbar2;
Ddk12_pos17 = N1_1*N3_2*Rbar3+N1_2*N3_1*Rbar3;
Ddk12_pos18 = N1_1*N4_2*Rbar4+N1_2*N4_1*Rbar4;
Ddk12_pos25 = N2_1*N1_2*Rbar1+N2_2*N1_1*Rbar1;
Ddk12_pos26 = N2_1*N2_2*Rbar2+N2_2*N2_1*Rbar2;
Ddk12_pos27 = N2_1*N3_2*Rbar3+N2_2*N3_1*Rbar3;
Ddk12_pos28 = N2_1*N4_2*Rbar4+N2_2*N4_1*Rbar4;
Ddk12_pos35 = N3_1*N1_2*Rbar1+N3_2*N1_1*Rbar1;
Ddk12_pos36 = N3_1*N2_2*Rbar2+N3_2*N2_1*Rbar2;
Ddk12_pos37 = N3_1*N3_2*Rbar3+N3_2*N3_1*Rbar3;
Ddk12_pos38 = N3_1*N4_2*Rbar4+N3_2*N4_1*Rbar4;
Ddk12_pos45 = N4_1*N1_2*Rbar1+N4_2*N1_1*Rbar1;
Ddk12_pos46 = N4_1*N2_2*Rbar2+N4_2*N2_1*Rbar2;
Ddk12_pos47 = N4_1*N3_2*Rbar3+N4_2*N3_1*Rbar3;
Ddk12_pos48 = N4_1*N4_2*Rbar4+N4_2*N4_1*Rbar4;
Ddk12_3x2 = [Ddk12_pos15 Ddk12_pos16 Ddk12_pos17 Ddk12_pos18
             Ddk12_pos25 Ddk12_pos26 Ddk12_pos27 Ddk12_pos28
             Ddk12_pos35 Ddk12_pos36 Ddk12_pos37 Ddk12_pos38
             Ddk12_pos45 Ddk12_pos46 Ddk12_pos47 Ddk12_pos48];
Ddk12_pos51 = N1_1*N1_2*Rbar1'+N1_2*N1_1*Rbar1';
Ddk12_pos52 = N1_1*N2_2*Rbar1'+N1_2*N2_1*Rbar1';
Ddk12_pos53 = N1_1*N3_2*Rbar1'+N1_2*N3_1*Rbar1';
Ddk12_pos54 = N1_1*N4_2*Rbar1'+N1_2*N4_1*Rbar1';
Ddk12_pos61 = N2_1*N1_2*Rbar2'+N2_2*N1_1*Rbar2';
Ddk12_pos62 = N2_1*N2_2*Rbar2'+N2_2*N2_1*Rbar2';
Ddk12_pos63 = N2_1*N3_2*Rbar2'+N2_2*N3_1*Rbar2';
Ddk12_pos64 = N2_1*N4_2*Rbar2'+N2_2*N4_1*Rbar2';
Ddk12_pos71 = N3_1*N1_2*Rbar3'+N3_2*N1_1*Rbar3';
Ddk12_pos72 = N3_1*N2_2*Rbar3'+N3_2*N2_1*Rbar3';
Ddk12_pos73 = N3_1*N3_2*Rbar3'+N3_2*N3_1*Rbar3';
Ddk12_pos74 = N3_1*N4_2*Rbar3'+N3_2*N4_1*Rbar3';
Ddk12_pos81 = N4_1*N1_2*Rbar4'+N4_2*N1_1*Rbar4';
Ddk12_pos82 = N4_1*N2_2*Rbar4'+N4_2*N2_1*Rbar4';
Ddk12_pos83 = N4_1*N3_2*Rbar4'+N4_2*N3_1*Rbar4';
Ddk12_pos84 = N4_1*N4_2*Rbar4'+N4_2*N4_1*Rbar4';
Ddk12_2x3 = [Ddk12_pos51 Ddk12_pos52 Ddk12_pos53 Ddk12_pos54
             Ddk12_pos61 Ddk12_pos62 Ddk12_pos63 Ddk12_pos64
             Ddk12_pos71 Ddk12_pos72 Ddk12_pos73 Ddk12_pos74
```

```
                Ddk12_pos81 Ddk12_pos82 Ddk12_pos83 Ddk12_pos84];
    Ddk12_pos55 = -N1_1*(t1'*x_2)*(Rbar1'*Rbar1)-N1_2*(t1'*x_1)*(Rbar1'*Rbar1);
    Ddk12_pos66 = -N2_1*(t2'*x_2)*(Rbar2'*Rbar2)-N2_2*(t2'*x_1)*(Rbar2'*Rbar2);
    Ddk12_pos77 = -N3_1*(t3'*x_2)*(Rbar3'*Rbar3)-N3_2*(t3'*x_1)*(Rbar3'*Rbar3);
    Ddk12_pos88 = -N4_1*(t4'*x_2)*(Rbar4'*Rbar4)-N4_2*(t4'*x_1)*(Rbar4'*Rbar4);
    Ddk12_2x2 = [Ddk12_pos55 O2x2 O2x2 O2x2
                 O2x2 Ddk12_pos66 O2x2 O2x2
                 O2x2 O2x2 Ddk12_pos77 O2x2
                 O2x2 O2x2 O2x2 Ddk12_pos88];
    %-------------------------------------------------------------------------------------------%
    % Geometric contribution
    %-------------------------------------------------------------------------------------------%
    DKgMB_12x12 = (Dde11*n11+Dde22*n22+Dde12*n12)*j0bar*Weight;
    DKgMB_12x8 = (Ddk11_3x2*m11+Ddk22_3x2*m22+Ddk12_3x2*m12)*j0bar*Weight;
    DKgMB_8x12 = (Ddk11_2x3*m11+Ddk22_2x3*m22+Ddk12_2x3*m12)*j0bar*Weight;
    DKgMB_8x8 = (Ddk11_2x2*m11+Ddk22_2x2*m22+Ddk12_2x2*m12)*j0bar*Weight;
    DKgMB = [DKgMB_12x12 DKgMB_12x8
             DKgMB_8x12 DKgMB_8x8];
    KgMB = KgMB+DKgMB;
    %-------------------------------------------------------------------------------------------%
    % Material contribution
    %-------------------------------------------------------------------------------------------%
    DKmMB_12x12 = (de11*Dn11+de22*Dn22+de12*Dn12)*j0bar*Weight;
    DKmMB_20x20 = (dk11*Dm11+dk22*Dm22+dk12*Dm12)*j0bar*Weight;
    DKmMB = [DKmMB_12x12 O12x8
             O8x12 O8x8]+DKmMB_20x20;
    KmMB = KmMB+DKmMB;
end
KMB = KgMB+KmMB;
%-------------------------------------------------------------------------------------------%
% Shear contribution
%-------------------------------------------------------------------------------------------%
for GPNumber = 1:TotGPS
    [GP,Weight] = GaussPoint(GPNumber,TotGPS);
    N1 = ShapeFunction(1,GP); N2 = ShapeFunction(2,GP);
    N3 = ShapeFunction(3,GP); N4 = ShapeFunction(4,GP);
    N1_1 = DerShapeFunction(1,1,GP); N1_2 = DerShapeFunction(1,2,GP);
    N2_1 = DerShapeFunction(2,1,GP); N2_2 = DerShapeFunction(2,2,GP);
    N3_1 = DerShapeFunction(3,1,GP); N3_2 = DerShapeFunction(3,2,GP);
    N4_1 = DerShapeFunction(4,1,GP); N4_2 = DerShapeFunction(4,2,GP);
    x_1 = x1*N1_1+x2*N2_1+x3*N3_1+x4*N4_1;
    x_2 = x1*N1_2+x2*N2_2+x3*N3_2+x4*N4_2;
    x0_1 = x01*N1_1+x02*N2_1+x03*N3_1+x04*N4_1;
    x0_2 = x01*N1_2+x02*N2_2+x03*N3_2+x04*N4_2;
    t = tGPS(:,GPNumber);
    t0 = t0GPS(:,GPNumber);
    j0bar = norm(cross(x0_1,x0_2),2);
    %-------------------------------------------------------------------------------------------%
    % Elastic covariant tensor
    %-------------------------------------------------------------------------------------------%
    [~,~,Cq] = ElasticCovariantTensor(x0_1,x0_2,t0,Global);
    Cq11 = Cq(1,1); Cq12 = Cq(1,2);
    Cq21 = Cq(2,1); Cq22 = Cq(2,2);
    %-------------------------------------------------------------------------------------------%
    % Strain
    %-------------------------------------------------------------------------------------------%
    y1 = x_1'*t-x0_1'*t0;
    y2 = x_2'*t-x0_2'*t0;
    %-------------------------------------------------------------------------------------------%
    % Stress
    %-------------------------------------------------------------------------------------------%
    q1 = Cq11*y1+Cq12*y2;
```

```
q2 = Cq21*y1+Cq22*y2;
%-------------------------------------------------------------------------------------%
% Strain variation
%-------------------------------------------------------------------------------------%
dy1_pos1 = N1_1*t; dy1_pos2 = N2_1*t;
dy1_pos3 = N3_1*t; dy1_pos4 = N4_1*t;
dy1_pos5 = N1*Rbar1'*x_1; dy1_pos6 = N2*Rbar2'*x_1;
dy1_pos7 = N3*Rbar3'*x_1; dy1_pos8 = N4*Rbar4'*x_1;
dy1 = [dy1_pos1;dy1_pos2;dy1_pos3;dy1_pos4;dy1_pos5;dy1_pos6;dy1_pos7;dy1_pos8];
dy2_pos1 = N1_2*t; dy2_pos2 = N2_2*t;
dy2_pos3 = N3_2*t; dy2_pos4 = N4_2*t;
dy2_pos5 = N1*Rbar1'*x_2; dy2_pos6 = N2*Rbar2'*x_2;
dy2_pos7 = N3*Rbar3'*x_2; dy2_pos8 = N4*Rbar4'*x_2;
dy2 = [dy2_pos1;dy2_pos2;dy2_pos3;dy2_pos4;dy2_pos5;dy2_pos6;dy2_pos7;dy2_pos8];
%-------------------------------------------------------------------------------------%
% Stress increment
%-------------------------------------------------------------------------------------%
Dy1_pos1 = dy1_pos1'; Dy1_pos2 = dy1_pos2';
Dy1_pos3 = dy1_pos3'; Dy1_pos4 = dy1_pos4';
Dy1_pos5 = dy1_pos5'; Dy1_pos6 = dy1_pos6';
Dy1_pos7 = dy1_pos7'; Dy1_pos8 = dy1_pos8';
Dy1 = [Dy1_pos1 Dy1_pos2 Dy1_pos3 Dy1_pos4 Dy1_pos5 Dy1_pos6 Dy1_pos7 Dy1_pos8];
Dy2_pos1 = dy2_pos1'; Dy2_pos2 = dy2_pos2';
Dy2_pos3 = dy2_pos3'; Dy2_pos4 = dy2_pos4';
Dy2_pos5 = dy2_pos5'; Dy2_pos6 = dy2_pos6';
Dy2_pos7 = dy2_pos7'; Dy2_pos8 = dy2_pos8';
Dy2 = [Dy2_pos1 Dy2_pos2 Dy2_pos3 Dy2_pos4 Dy2_pos5 Dy2_pos6 Dy2_pos7 Dy2_pos8];
Dq1 = Cq11*Dy1+Cq12*Dy2;
Dq2 = Cq21*Dy1+Cq22*Dy2;
%-------------------------------------------------------------------------------------%
% Strain increment variation
%-------------------------------------------------------------------------------------%
Ddy1_pos15 = N1_1*N1*Rbar1; Ddy1_pos16 = N1_1*N2*Rbar2;
Ddy1_pos17 = N1_1*N3*Rbar3; Ddy1_pos18 = N1_1*N4*Rbar4;
Ddy1_pos25 = N2_1*N1*Rbar1; Ddy1_pos26 = N2_1*N2*Rbar2;
Ddy1_pos27 = N2_1*N3*Rbar3; Ddy1_pos28 = N2_1*N4*Rbar4;
Ddy1_pos35 = N3_1*N1*Rbar1; Ddy1_pos36 = N3_1*N2*Rbar2;
Ddy1_pos37 = N3_1*N3*Rbar3; Ddy1_pos38 = N3_1*N4*Rbar4;
Ddy1_pos45 = N4_1*N1*Rbar1; Ddy1_pos46 = N4_1*N2*Rbar2;
Ddy1_pos47 = N4_1*N3*Rbar3; Ddy1_pos48 = N4_1*N4*Rbar4;
Ddy1_3x2 = [Ddy1_pos15 Ddy1_pos16 Ddy1_pos17 Ddy1_pos18
            Ddy1_pos25 Ddy1_pos26 Ddy1_pos27 Ddy1_pos28
            Ddy1_pos35 Ddy1_pos36 Ddy1_pos37 Ddy1_pos38
            Ddy1_pos45 Ddy1_pos46 Ddy1_pos47 Ddy1_pos48];
Ddy1_pos51 = N1*N1_1*Rbar1'; Ddy1_pos52 = N1*N2_1*Rbar1';
Ddy1_pos53 = N1*N3_1*Rbar1'; Ddy1_pos54 = N1*N4_1*Rbar1';
Ddy1_pos61 = N2*N1_1*Rbar2'; Ddy1_pos62 = N2*N2_1*Rbar2';
Ddy1_pos63 = N2*N3_1*Rbar2'; Ddy1_pos64 = N2*N4_1*Rbar2';
Ddy1_pos71 = N3*N1_1*Rbar3'; Ddy1_pos72 = N3*N2_1*Rbar3';
Ddy1_pos73 = N3*N3_1*Rbar3'; Ddy1_pos74 = N3*N4_1*Rbar3';
Ddy1_pos81 = N4*N1_1*Rbar4'; Ddy1_pos82 = N4*N2_1*Rbar4';
Ddy1_pos83 = N4*N3_1*Rbar4'; Ddy1_pos84 = N4*N4_1*Rbar4';
Ddy1_2x3 = [Ddy1_pos51 Ddy1_pos52 Ddy1_pos53 Ddy1_pos54
            Ddy1_pos61 Ddy1_pos62 Ddy1_pos63 Ddy1_pos64
            Ddy1_pos71 Ddy1_pos72 Ddy1_pos73 Ddy1_pos74
            Ddy1_pos81 Ddy1_pos82 Ddy1_pos83 Ddy1_pos84];
Ddy1_pos55 = -N1*(t1'*x_1)*(Rbar1'*Rbar1);
Ddy1_pos66 = -N2*(t2'*x_1)*(Rbar2'*Rbar2);
Ddy1_pos77 = -N3*(t3'*x_1)*(Rbar3'*Rbar3);
Ddy1_pos88 = -N4*(t4'*x_1)*(Rbar4'*Rbar4);
Ddy1_2x2 = [Ddy1_pos55 O2x2 O2x2 O2x2
            O2x2 Ddy1_pos66 O2x2 O2x2
```

```
                O2x2 O2x2 Ddy1_pos77 O2x2
                O2x2 O2x2 O2x2 Ddy1_pos88];
    Ddy2_pos15 = N1_2*N1*Rbar1; Ddy2_pos16 = N1_2*N2*Rbar2;
    Ddy2_pos17 = N1_2*N3*Rbar3; Ddy2_pos18 = N1_2*N4*Rbar4;
    Ddy2_pos25 = N2_2*N1*Rbar1; Ddy2_pos26 = N2_2*N2*Rbar2;
    Ddy2_pos27 = N2_2*N3*Rbar3; Ddy2_pos28 = N2_2*N4*Rbar4;
    Ddy2_pos35 = N3_2*N1*Rbar1; Ddy2_pos36 = N3_2*N2*Rbar2;
    Ddy2_pos37 = N3_2*N3*Rbar3; Ddy2_pos38 = N3_2*N4*Rbar4;
    Ddy2_pos45 = N4_2*N1*Rbar1; Ddy2_pos46 = N4_2*N2*Rbar2;
    Ddy2_pos47 = N4_2*N3*Rbar3; Ddy2_pos48 = N4_2*N4*Rbar4;
    Ddy2_3x2 = [Ddy2_pos15 Ddy2_pos16 Ddy2_pos17 Ddy2_pos18
                Ddy2_pos25 Ddy2_pos26 Ddy2_pos27 Ddy2_pos28
                Ddy2_pos35 Ddy2_pos36 Ddy2_pos37 Ddy2_pos38
                Ddy2_pos45 Ddy2_pos46 Ddy2_pos47 Ddy2_pos48];
    Ddy2_pos51 = N1*N1_2*Rbar1'; Ddy2_pos52 = N1*N2_2*Rbar1';
    Ddy2_pos53 = N1*N3_2*Rbar1'; Ddy2_pos54 = N1*N4_2*Rbar1';
    Ddy2_pos61 = N2*N1_2*Rbar2'; Ddy2_pos62 = N2*N2_2*Rbar2';
    Ddy2_pos63 = N2*N3_2*Rbar2'; Ddy2_pos64 = N2*N4_2*Rbar2';
    Ddy2_pos71 = N3*N1_2*Rbar3'; Ddy2_pos72 = N3*N2_2*Rbar3';
    Ddy2_pos73 = N3*N3_2*Rbar3'; Ddy2_pos74 = N3*N4_2*Rbar3';
    Ddy2_pos81 = N4*N1_2*Rbar4'; Ddy2_pos82 = N4*N2_2*Rbar4';
    Ddy2_pos83 = N4*N3_2*Rbar4'; Ddy2_pos84 = N4*N4_2*Rbar4';
    Ddy2_2x3 = [Ddy2_pos51 Ddy2_pos52 Ddy2_pos53 Ddy2_pos54
                Ddy2_pos61 Ddy2_pos62 Ddy2_pos63 Ddy2_pos64
                Ddy2_pos71 Ddy2_pos72 Ddy2_pos73 Ddy2_pos74
                Ddy2_pos81 Ddy2_pos82 Ddy2_pos83 Ddy2_pos84];
    Ddy2_pos55 = -N1*(t1'*x_2)*(Rbar1'*Rbar1);
    Ddy2_pos66 = -N2*(t2'*x_2)*(Rbar2'*Rbar2);
    Ddy2_pos77 = -N3*(t3'*x_2)*(Rbar3'*Rbar3);
    Ddy2_pos88 = -N4*(t4'*x_2)*(Rbar4'*Rbar4);
    Ddy2_2x2 = [Ddy2_pos55 O2x2 O2x2 O2x2
                O2x2 Ddy2_pos66 O2x2 O2x2
                O2x2 O2x2 Ddy2_pos77 O2x2
                O2x2 O2x2 O2x2 Ddy2_pos88];
    %-------------------------------------------------------------------------------------%
    % Geometric contribution
    %-------------------------------------------------------------------------------------%
    DKgS_12x8 = (Ddy1_3x2*q1+Ddy2_3x2*q2)*j0bar*Weight;
    DKgS_8x12 = (Ddy1_2x3*q1+Ddy2_2x3*q2)*j0bar*Weight;
    DKgS_8x8 = (Ddy1_2x2*q1+Ddy2_2x2*q2)*j0bar*Weight;
    DKgS = [O12x12 DKgS_12x8
            DKgS_8x12 DKgS_8x8];
    KgS = KgS+DKgS;
    %-------------------------------------------------------------------------------------%
    % Material contribution
    %-------------------------------------------------------------------------------------%
    DKmS = (dy1*Dq1+dy2*Dq2)*j0bar*Weight;
    KmS = KmS+DKmS;
end
KS = KgS+KmS;
%-----------------------------------------------------------------------------------------%
K = KMB+KS;
%-----------------------------------------------------------------------------------------%
end
```

InternalForces

```
function fInt = InternalForces(Global,ElementNumber)
%-----------------------------------------------------------------------------------------%
x0Nodes = Global.Configuration.x0Nodes;
xNodes = Global.Configuration.xNodes;
```

```
RNodes = Global.Configuration.RNodes;
Connectivity = Global.Mesh.Connectivity;
Node1 = Connectivity(ElementNumber,1); Node2 = Connectivity(ElementNumber,2);
Node3 = Connectivity(ElementNumber,3); Node4 = Connectivity(ElementNumber,4);
x01 = x0Nodes(Node1,:)'; x02 = x0Nodes(Node2,:)';
x03 = x0Nodes(Node3,:)'; x04 = x0Nodes(Node4,:)';
x1 = xNodes(Node1,:)'; x2 = xNodes(Node2,:)';
x3 = xNodes(Node3,:)'; x4 = xNodes(Node4,:)';
R1 = RNodes(3*(Node1-1)+1:3*Node1,:); R2 = RNodes(3*(Node2-1)+1:3*Node2,:);
R3 = RNodes(3*(Node3-1)+1:3*Node3,:); R4 = RNodes(3*(Node4-1)+1:3*Node4,:);
Rbar1 = R1(:,1:2); Rbar2 = R2(:,1:2); Rbar3 = R3(:,1:2); Rbar4 = R4(:,1:2);
t0GPMB = Global.Configuration.t0GaussPointsMembraneAndBending ...
    (3*(ElementNumber-1)+1:3*ElementNumber,:);
t0GPS = Global.Configuration.t0GaussPointsShear(3*(ElementNumber-1)+1:3*ElementNumber,:);
t0_1GPMB = Global.Configuration.t0_1GaussPointsMembraneAndBending ...
    (3*(ElementNumber-1)+1:3*ElementNumber,:);
t0_2GPMB = Global.Configuration.t0_2GaussPointsMembraneAndBending ...
    (3*(ElementNumber-1)+1:3*ElementNumber,:);
tGPS = Global.Configuration.tGaussPointsShear(3*(ElementNumber-1)+1:3*ElementNumber,:);
t_1GPMB = Global.Configuration.t_1GaussPointsMembraneAndBending ...
    (3*(ElementNumber-1)+1:3*ElementNumber,:);
t_2GPMB = Global.Configuration.t_2GaussPointsMembraneAndBending ...
    (3*(ElementNumber-1)+1:3*ElementNumber,:);
TotGPMB = Global.Element.TotGaussPointsMembraneAndBending;
TotGPS = Global.Element.TotGaussPointsShear;
O8x1 = zeros(8,1);
fIntMB = zeros(20,1); fIntS = zeros(20,1);
%-------------------------------------------------------------------------------------%
% Membrane and bending contribution
%-------------------------------------------------------------------------------------%
for GPNumber = 1:TotGPMB
    [GP,Weight] = GaussPoint(GPNumber,TotGPMB);
    N1_1 = DerShapeFunction(1,1,GP); N1_2 = DerShapeFunction(1,2,GP);
    N2_1 = DerShapeFunction(2,1,GP); N2_2 = DerShapeFunction(2,2,GP);
    N3_1 = DerShapeFunction(3,1,GP); N3_2 = DerShapeFunction(3,2,GP);
    N4_1 = DerShapeFunction(4,1,GP); N4_2 = DerShapeFunction(4,2,GP);
    x_1 = x1*N1_1+x2*N2_1+x3*N3_1+x4*N4_1;
    x_2 = x1*N1_2+x2*N2_2+x3*N3_2+x4*N4_2;
    x0_1 = x01*N1_1+x02*N2_1+x03*N3_1+x04*N4_1;
    x0_2 = x01*N1_2+x02*N2_2+x03*N3_2+x04*N4_2;
    t_1 = t_1GPMB(:,GPNumber);
    t_2 = t_2GPMB(:,GPNumber);
    t0 = t0GPMB(:,GPNumber);
    t0_1 = t0_1GPMB(:,GPNumber);
    t0_2 = t0_2GPMB(:,GPNumber);
    j0bar = norm(cross(x0_1,x0_2),2);
    %---------------------------------------------------------------------------------%
    % Elastic covariant tensor
    %---------------------------------------------------------------------------------%
    [Cn,Cm,~] = ElasticCovariantTensor(x0_1,x0_2,t0,Global);
    Cn11 = Cn(1,1); Cn12 = Cn(1,2); Cn13 = Cn(1,3);
    Cn21 = Cn(2,1); Cn22 = Cn(2,2); Cn23 = Cn(2,3);
    Cn31 = Cn(3,1); Cn32 = Cn(3,2); Cn33 = Cn(3,3);
    Cm11 = Cm(1,1); Cm12 = Cm(1,2); Cm13 = Cm(1,3);
    Cm21 = Cm(2,1); Cm22 = Cm(2,2); Cm23 = Cm(2,3);
    Cm31 = Cm(3,1); Cm32 = Cm(3,2); Cm33 = Cm(3,3);
    %---------------------------------------------------------------------------------%
    % Strain
    %---------------------------------------------------------------------------------%
    e11 = 0.5*(x_1'*x_1-x0_1'*x0_1);
    e22 = 0.5*(x_2'*x_2-x0_2'*x0_2);
    e12 = (x_1'*x_2-x0_1'*x0_2);
```

```
    k11 = x_1'*t_1-x0_1'*t0_1;
    k22 = x_2'*t_2-x0_2'*t0_2;
    k12 = (x_1'*t_2-x0_1'*t0_2)+(x_2'*t_1-x0_2'*t0_1);
    %-------------------------------------------------------------------------------------%
    % Stress
    %-------------------------------------------------------------------------------------%
    n11 = Cn11*e11+Cn12*e22+Cn13*e12;
    n22 = Cn21*e11+Cn22*e22+Cn23*e12;
    n12 = Cn31*e11+Cn32*e22+Cn33*e12;
    m11 = Cm11*k11+Cm12*k22+Cm13*k12;
    m22 = Cm21*k11+Cm22*k22+Cm23*k12;
    m12 = Cm31*k11+Cm32*k22+Cm33*k12;
    %-------------------------------------------------------------------------------------%
    % Strain variation
    %-------------------------------------------------------------------------------------%
    de11_pos1 = N1_1*x_1; de11_pos2 = N2_1*x_1;
    de11_pos3 = N3_1*x_1; de11_pos4 = N4_1*x_1;
    de11 = [de11_pos1;de11_pos2;de11_pos3;de11_pos4;O8x1];
    de22_pos1 = N1_2*x_2; de22_pos2 = N2_2*x_2;
    de22_pos3 = N3_2*x_2; de22_pos4 = N4_2*x_2;
    de22 = [de22_pos1;de22_pos2;de22_pos3;de22_pos4;O8x1];
    de12_pos1 = N1_1*x_2+N1_2*x_1;
    de12_pos2 = N2_1*x_2+N2_2*x_1;
    de12_pos3 = N3_1*x_2+N3_2*x_1;
    de12_pos4 = N4_1*x_2+N4_2*x_1;
    de12 = [de12_pos1;de12_pos2;de12_pos3;de12_pos4;O8x1];
    dk11_pos1 = N1_1*t_1; dk11_pos2 = N2_1*t_1;
    dk11_pos3 = N3_1*t_1; dk11_pos4 = N4_1*t_1;
    dk11_pos5 = N1_1*Rbar1'*x_1; dk11_pos6 = N2_1*Rbar2'*x_1;
    dk11_pos7 = N3_1*Rbar3'*x_1; dk11_pos8 = N4_1*Rbar4'*x_1;
    dk11 = [dk11_pos1;dk11_pos2;dk11_pos3;dk11_pos4;
        dk11_pos5;dk11_pos6;dk11_pos7;dk11_pos8];
    dk22_pos1 = N1_2*t_2; dk22_pos2 = N2_2*t_2;
    dk22_pos3 = N3_2*t_2; dk22_pos4 = N4_2*t_2;
    dk22_pos5 = N1_2*Rbar1'*x_2; dk22_pos6 = N2_2*Rbar2'*x_2;
    dk22_pos7 = N3_2*Rbar3'*x_2; dk22_pos8 = N4_2*Rbar4'*x_2;
    dk22 = [dk22_pos1;dk22_pos2;dk22_pos3;dk22_pos4;
        dk22_pos5;dk22_pos6;dk22_pos7;dk22_pos8];
    dk12_pos1 = N1_1*t_2+N1_2*t_1;
    dk12_pos2 = N2_1*t_2+N2_2*t_1;
    dk12_pos3 = N3_1*t_2+N3_2*t_1;
    dk12_pos4 = N4_1*t_2+N4_2*t_1;
    dk12_pos5 = N1_2*Rbar1'*x_1+N1_1*Rbar1'*x_2;
    dk12_pos6 = N2_2*Rbar2'*x_1+N2_1*Rbar2'*x_2;
    dk12_pos7 = N3_2*Rbar3'*x_1+N3_1*Rbar3'*x_2;
    dk12_pos8 = N4_2*Rbar4'*x_1+N4_1*Rbar4'*x_2;
    dk12 = [dk12_pos1;dk12_pos2;dk12_pos3;dk12_pos4;
        dk12_pos5;dk12_pos6;dk12_pos7;dk12_pos8];
    %-------------------------------------------------------------------------------------%
    DfIntMB = (de11*n11+de22*n22+de12*n12+dk11*m11+dk22*m22+dk12*m12)*j0bar*Weight;
    fIntMB = fIntMB+DfIntMB;
end
%-----------------------------------------------------------------------------------------%
% Shear contribution
%-----------------------------------------------------------------------------------------%
for GPNumber = 1:TotGPS
    [GP,Weight] = GaussPoint(GPNumber,TotGPS);
    N1 = ShapeFunction(1,GP); N2 = ShapeFunction(2,GP);
    N3 = ShapeFunction(3,GP); N4 = ShapeFunction(4,GP);
    N1_1 = DerShapeFunction(1,1,GP); N1_2 = DerShapeFunction(1,2,GP);
    N2_1 = DerShapeFunction(2,1,GP); N2_2 = DerShapeFunction(2,2,GP);
    N3_1 = DerShapeFunction(3,1,GP); N3_2 = DerShapeFunction(3,2,GP);
```

```
    N4_1 = DerShapeFunction(4,1,GP); N4_2 = DerShapeFunction(4,2,GP);
    x_1 = x1*N1_1+x2*N2_1+x3*N3_1+x4*N4_1;
    x_2 = x1*N1_2+x2*N2_2+x3*N3_2+x4*N4_2;
    x0_1 = x01*N1_1+x02*N2_1+x03*N3_1+x04*N4_1;
    x0_2 = x01*N1_2+x02*N2_2+x03*N3_2+x04*N4_2;
    t = tGPS(:,GPNumber);
    t0 = t0GPS(:,GPNumber);
    j0bar = norm(cross(x0_1,x0_2),2);
    %-------------------------------------------------------------------------------------------%
    % Elastic covariant tensor
    %-------------------------------------------------------------------------------------------%
    [~,~,Cq] = ElasticCovariantTensor(x0_1,x0_2,t0,Global);
    Cq11 = Cq(1,1); Cq12 = Cq(1,2);
    Cq21 = Cq(2,1); Cq22 = Cq(2,2);
    %-------------------------------------------------------------------------------------------%
    % Strain
    %-------------------------------------------------------------------------------------------%
    y1 = x_1'*t-x0_1'*t0;
    y2 = x_2'*t-x0_2'*t0;
    %-------------------------------------------------------------------------------------------%
    % Stress
    %-------------------------------------------------------------------------------------------%
    q1 = Cq11*y1+Cq12*y2;
    q2 = Cq21*y1+Cq22*y2;
    %-------------------------------------------------------------------------------------------%
    % Strain variation
    %-------------------------------------------------------------------------------------------%
    dy1_pos1 = N1_1*t; dy1_pos2 = N2_1*t;
    dy1_pos3 = N3_1*t; dy1_pos4 = N4_1*t;
    dy1_pos5 = N1*Rbar1'*x_1; dy1_pos6 = N2*Rbar2'*x_1;
    dy1_pos7 = N3*Rbar3'*x_1; dy1_pos8 = N4*Rbar4'*x_1;
    dy1 = [dy1_pos1;dy1_pos2;dy1_pos3;dy1_pos4;dy1_pos5;dy1_pos6;dy1_pos7;dy1_pos8];
    dy2_pos1 = N1_2*t; dy2_pos2 = N2_2*t;
    dy2_pos3 = N3_2*t; dy2_pos4 = N4_2*t;
    dy2_pos5 = N1*Rbar1'*x_2; dy2_pos6 = N2*Rbar2'*x_2;
    dy2_pos7 = N3*Rbar3'*x_2; dy2_pos8 = N4*Rbar4'*x_2;
    dy2 = [dy2_pos1;dy2_pos2;dy2_pos3;dy2_pos4;dy2_pos5;dy2_pos6;dy2_pos7;dy2_pos8];
    %-------------------------------------------------------------------------------------------%
    DfIntS = (dy1*q1+dy2*q2)*j0bar*Weight;
    fIntS = fIntS+DfIntS;
end
%-----------------------------------------------------------------------------------------------%
fInt = fIntMB+fIntS;
%-----------------------------------------------------------------------------------------------%
end
```

AssembleStiffnessMatrix

```
function KGlobal = AssembleStiffnessMatrix(KGlobal,KLocal,Global,ElementNumber)
%-----------------------------------------------------------------------------------------------%
TotDofSurface = Global.Mesh.TotDofSurface;
Connectivity = Global.Mesh.Connectivity(ElementNumber,:);
for i = 1:4
    I = Connectivity(i);
    for j = 1:4
        J = Connectivity(j);
        KGlobal(3*I-2:3*I,3*J-2:3*J) = KGlobal(3*I-2:3*I,3*J-2:3*J) ...
            +KLocal(3*i-2:3*i,3*j-2:3*j);
        KGlobal(3*I-2:3*I,TotDofSurface+2*J-1:TotDofSurface+2*J) = ...
            KGlobal(3*I-2:3*I,TotDofSurface+2*J-1:TotDofSurface+2*J) ...
            +KLocal(3*i-2:3*i,12+2*j-1:12+2*j);
```

```
        KGlobal(TotDofSurface+2*I-1:TotDofSurface+2*I,3*J-2:3*J) = ...
            KGlobal(TotDofSurface+2*I-1:TotDofSurface+2*I,3*J-2:3*J) ...
            +KLocal(12+2*i-1:12+2*i,3*j-2:3*j);
        KGlobal(TotDofSurface+2*I-1:TotDofSurface+2*I, ...
            TotDofSurface+2*J-1:TotDofSurface+2*J) = ...
            KGlobal(TotDofSurface+2*I-1:TotDofSurface+2*I, ...
            TotDofSurface+2*J-1:TotDofSurface+2*J)+...
            KLocal(12+2*i-1:12+2*i,12+2*j-1:12+2*j);
    end
end
%-------------------------------------------------------------------------------------------%
end
```

AssembleInternalForces

```
function fIntGlobal = AssembleInternalForces(fIntGlobal,fIntLocal,Global,ElementNumber)
%-------------------------------------------------------------------------------------------%
TotDofSurface = Global.Mesh.TotDofSurface;
Connectivity = Global.Mesh.Connectivity(ElementNumber,:);
for i = 1:4
    I = Connectivity(i);
    fIntGlobal(3*I-2:3*I) = fIntGlobal(3*I-2:3*I)+fIntLocal(3*i-2:3*i);
    fIntGlobal(TotDofSurface+2*I-1:TotDofSurface+2*I) = ...
    fIntGlobal(TotDofSurface+2*I-1:TotDofSurface+2*I)+fIntLocal(12+2*i-1:12+2*i);
end
%-------------------------------------------------------------------------------------------%
end
```

UpdateSurface

```
function UpdateSurface(Global,DxNodes)
%-------------------------------------------------------------------------------------------%
TotNodes = Global.Mesh.TotNodes;
xNodes = Global.Configuration.xNodes;
for NodeNumber = 1:TotNodes
    Dx = DxNodes(3*NodeNumber-2:3*NodeNumber,1)';
    x = xNodes(NodeNumber,:);
    xUpdated = x+Dx;
    Global.Configuration.xNodes(NodeNumber,:) = xUpdated;
end
%-------------------------------------------------------------------------------------------%
end
```

UpdateDirector

```
function UpdateDirector(Global,DTNodes)
%-------------------------------------------------------------------------------------------%
tNodes = Global.Configuration.tNodes;
tGaussPointsMembraneAndBending = Global.Configuration.tGaussPointsMembraneAndBending;
t_1GaussPointsMembraneAndBending = Global.Configuration.t_1GaussPointsMembraneAndBending;
t_2GaussPointsMembraneAndBending = Global.Configuration.t_2GaussPointsMembraneAndBending;
tGaussPointsShear = Global.Configuration.tGaussPointsShear;
RNodes = Global.Configuration.RNodes;
TotDofSurface = Global.Mesh.TotDofSurface;
TotNodes = Global.Mesh.TotNodes;
TotElements = Global.Mesh.TotElements;
TotGPMB = Global.Element.TotGaussPointsMembraneAndBending;
TotGPS = Global.Element.TotGaussPointsShear;
Connectivity = Global.Mesh.Connectivity;
```

```
%-------------------------------------------------------------------------------------------%
% Update nodes
%-------------------------------------------------------------------------------------------%
DtNodes = zeros(TotDofSurface,1);
for NodeNumber = 1:TotNodes
    DT = DTNodes(2*NodeNumber-1:2*NodeNumber,1);
    t = tNodes(NodeNumber,:)';
    R = RNodes(3*NodeNumber-2:3*NodeNumber,:);
    Rbar = R(:,1:2);
    Dt = Rbar*DT;
    DtNodes(3*NodeNumber-2:3*NodeNumber) = Dt;
    [tUpdated,DR] = ExpMap(t,Dt);
    RUpdated = DR*R;
    Global.Configuration.tNodes(NodeNumber,:) = tUpdated';
    Global.Configuration.RNodes(3*NodeNumber-2:3*NodeNumber,:) = RUpdated;
end
%-------------------------------------------------------------------------------------------%
% Update Gauss points
%-------------------------------------------------------------------------------------------%
for ElementNumber = 1:TotElements
    Node1 = Connectivity(ElementNumber,1); Node2 = Connectivity(ElementNumber,2);
    Node3 = Connectivity(ElementNumber,3); Node4 = Connectivity(ElementNumber,4);
    Dt1 = DtNodes(3*Node1-2:3*Node1); Dt2 = DtNodes(3*Node2-2:3*Node2);
    Dt3 = DtNodes(3*Node3-2:3*Node3); Dt4 = DtNodes(3*Node4-2:3*Node4);
    tGPMB = tGaussPointsMembraneAndBending(3*(ElementNumber-1)+1:3*ElementNumber,:);
    t_1GPMB = t_1GaussPointsMembraneAndBending(3*(ElementNumber-1)+1:3*ElementNumber,:);
    t_2GPMB = t_2GaussPointsMembraneAndBending(3*(ElementNumber-1)+1:3*ElementNumber,:);
    tGPS = tGaussPointsShear(3*(ElementNumber-1)+1:3*ElementNumber,:);
    % Update membrane and bending Gauss Points
    for GPNumber = 1:TotGPMB
        [GP,~] = GaussPoint(GPNumber,TotGPMB);
        N1 = ShapeFunction(1,GP); N2 = ShapeFunction(2,GP);
        N3 = ShapeFunction(3,GP); N4 = ShapeFunction(4,GP);
        N1_1 = DerShapeFunction(1,1,GP); N1_2 = DerShapeFunction(1,2,GP);
        N2_1 = DerShapeFunction(2,1,GP); N2_2 = DerShapeFunction(2,2,GP);
        N3_1 = DerShapeFunction(3,1,GP); N3_2 = DerShapeFunction(3,2,GP);
        N4_1 = DerShapeFunction(4,1,GP); N4_2 = DerShapeFunction(4,2,GP);
        tGP = tGPMB(:,GPNumber);
        t_1GP = t_1GPMB(:,GPNumber);
        t_2GP = t_2GPMB(:,GPNumber);
        DtGP = Dt1*N1+Dt2*N2+Dt3*N3+Dt4*N4;
        Dt_1GP = Dt1*N1_1+Dt2*N2_1+Dt3*N3_1+Dt4*N4_1;
        Dt_2GP = Dt1*N1_2+Dt2*N2_2+Dt3*N3_2+Dt4*N4_2;
        tGPUpdated = ExpMap(tGP,DtGP);
        t_1GPUpdated = DerExpMap(tGP,DtGP,t_1GP,Dt_1GP);
        t_2GPUpdated = DerExpMap(tGP,DtGP,t_2GP,Dt_2GP);
        Global.Configuration.tGaussPointsMembraneAndBending...
            (3*ElementNumber-2:3*ElementNumber,GPNumber) = tGPUpdated;
        Global.Configuration.t_1GaussPointsMembraneAndBending...
            (3*ElementNumber-2:3*ElementNumber,GPNumber) = t_1GPUpdated;
        Global.Configuration.t_2GaussPointsMembraneAndBending...
            (3*ElementNumber-2:3*ElementNumber,GPNumber) = t_2GPUpdated;
    end
    % Update shear Gauss Points
    for GPNumber = 1:TotGPS
        [GP,~] = GaussPoint(GPNumber,TotGPS);
        N1 = ShapeFunction(1,GP); N2 = ShapeFunction(2,GP);
        N3 = ShapeFunction(3,GP); N4 = ShapeFunction(4,GP);
        tGP = tGPS(:,GPNumber);
        DtGP = Dt1*N1+Dt2*N2+Dt3*N3+Dt4*N4;
        tGPUpdated = ExpMap(tGP,DtGP);
        Global.Configuration.tGaussPointsShear(3*ElementNumber-2:3*ElementNumber,:) = ...
```

```
            tGPUpdated;
    end
end
%-------------------------------------------------------------------------------------------%
end
```

ExpMap

```
function [tUpdated,DR] = ExpMap(t,Dt)
%-------------------------------------------------------------------------------------------%
I = eye(3);
Dtheta = cross(t,Dt);
Dtheta1 = Dtheta(1); Dtheta2 = Dtheta(2); Dtheta3 = Dtheta(3);
DthetaTens = [0 -Dtheta3 Dtheta2
              Dtheta3 0 -Dtheta1
              -Dtheta2 Dtheta1 0];
normDt = norm(Dt,2);
if normDt ~= 0
    tUpdated = cos(normDt)*t+sin(normDt)/normDt*Dt;
    DR = cos(normDt)*I+sin(normDt)/normDt*DthetaTens ...
        +(1-cos(normDt))/normDt^2*(Dtheta*Dtheta');
else
    tUpdated = t;
    DR = I;
end
%-------------------------------------------------------------------------------------------%
end
```

DerExpMap

```
function DertUpdated = DerExpMap(t,Dt,Dert,DerDt)
%-------------------------------------------------------------------------------------------%
I = eye(3);
normDt = norm(Dt,2);
if normDt ~= 0
   DertUpdated = cos(normDt)*Dert+(sin(normDt)/normDt*(I-(t*Dt'))+...
       1/normDt^2*(cos(normDt)-sin(normDt)/normDt)*(Dt*Dt'))*DerDt;
else
DertUpdated = Dert;
end
%-------------------------------------------------------------------------------------------%
end
```

ElasticCovariantTensor

```
function [Cn,Cm,Cq] = ElasticCovariantTensor(x0_1,x0_2,t0,Global)
%-------------------------------------------------------------------------------------------%
E = Global.Elastic.ElasticModule; nu = Global.Elastic.PoissonCoefficient;
h = Global.Geometry.Height; kappa = Global.Geometry.ShearFactor;
a0CovariantTensor = [x0_1 x0_2 t0]; a0ContravariantTensor = (a0CovariantTensor^-1)';
a01Contr = a0ContravariantTensor(:,1); a02Contr = a0ContravariantTensor(:,2);
a0Contr11 = a01Contr'*a01Contr;
a0Contr12 = a01Contr'*a02Contr;
a0Contr22 = a02Contr'*a02Contr;
Cn = [a0Contr11*a0Contr11 ...
    nu*a0Contr11*a0Contr22+(1-nu)*a0Contr12*a0Contr12 ...
    a0Contr11*a0Contr12;
    nu*a0Contr11*a0Contr22+(1-nu)*a0Contr12*a0Contr12 ...
    a0Contr22*a0Contr22 ...
```

```
    a0Contr22*a0Contr12;
    a0Contr11*a0Contr12 ...
    a0Contr22*a0Contr12 ...
    0.5*(1-nu)*a0Contr11*a0Contr22+0.5*(1+nu)*a0Contr12*a0Contr12]...
    *E*h/(1-nu^2);
Cm = Cn*h^2/12;
Cq = [a0Contr11 a0Contr12
      a0Contr12 a0Contr22]*kappa*E*h/(1-nu^2)*0.5*(1-nu);
  %-------------------------------------------------------------------------------------------%
end
```

PlotConfigurations

```
function PlotConfigurations(Global)
%-------------------------------------------------------------------------------------------%
Dx = zeros(Global.Mesh.TotNodes,Global.Solver.LoadSteps);
Dy = zeros(Global.Mesh.TotNodes,Global.Solver.LoadSteps);
Dz = zeros(Global.Mesh.TotNodes,Global.Solver.LoadSteps);
Ds = zeros(Global.Mesh.TotNodes,Global.Solver.LoadSteps);
for Step = 1:Global.Solver.LoadSteps
    Dx(:,Step+1) = Global.Plot.SavexNodesx(:,Step+1)-Global.Configuration.x0Nodes(:,1);
    Dy(:,Step+1) = Global.Plot.SavexNodesy(:,Step+1)-Global.Configuration.x0Nodes(:,2);
    Dz(:,Step+1) = Global.Plot.SavexNodesz(:,Step+1)-Global.Configuration.x0Nodes(:,3);
    for NodeNumber = 1:Global.Mesh.TotNodes
        Ds(NodeNumber,Step+1) = sqrt(Dx(NodeNumber,Step+1)^2 ...
            +Dy(NodeNumber,Step+1)^2+ ...
            Dz(NodeNumber,Step+1)^2);
    end
end
DsMax = max(max(Ds));
m = (64-1)/(DsMax);
q = (1*DsMax-64*0)/(DsMax-0);
mTick = (1-0)/(DsMax-0);
qTick =  (0*DsMax-1*0)/(DsMax-0);
cPlot = zeros(1,4,3);
for Step = 1:Global.Solver.LoadSteps+1
    c = zeros(Global.Mesh.TotNodes,1);
    for NodeNumber = 1:Global.Mesh.TotNodes
        displacement = Ds(NodeNumber,Step);
        c(NodeNumber,1) = m*displacement+q;
    end
    DsTicks = zeros(1,11);
    for i = 0:10
        j = i/10;
        DsTicks(1,i+1) = (j-qTick)/mTick;
    end
    %-------------------------------------------------------------------------------------------%
    f = figure('visible','off');
    title(['Deformed configuration, step = ' num2str(Step-1)...
        ', force = ' num2str(Global.Plot.ForcePlot(Step,1))''']);%'']);
    xlabel('x'); ylabel('y'); zlabel('z');
    view([25 25]); daspect([1 1 1]);
    ColorBar = colorbar('v');
    ColorBar.Limits = [0 1];
    ColorMap = ColorBar.Colormap;
    ColorBar.TickLabels = compose('%6.2e',DsTicks);
    title(ColorBar,'dz');
    hold on
    for ElementNumber = 1:Global.Mesh.TotElements
        Node1 = Global.Mesh.Connectivity(ElementNumber,1);
        Node2 = Global.Mesh.Connectivity(ElementNumber,2);
```

```
        Node3 = Global.Mesh.Connectivity(ElementNumber,3);
        Node4 = Global.Mesh.Connectivity(ElementNumber,4);
        x1 = Global.Plot.SavexNodesx(Node1,Step);
        y1 = Global.Plot.SavexNodesy(Node1,Step);
        z1 = Global.Plot.SavexNodesz(Node1,Step);
        x2 = Global.Plot.SavexNodesx(Node2,Step);
        y2 = Global.Plot.SavexNodesy(Node2,Step);
        z2 = Global.Plot.SavexNodesz(Node2,Step);
        x3 = Global.Plot.SavexNodesx(Node3,Step);
        y3 = Global.Plot.SavexNodesy(Node3,Step);
        z3 = Global.Plot.SavexNodesz(Node3,Step);
        x4 = Global.Plot.SavexNodesx(Node4,Step);
        y4 = Global.Plot.SavexNodesy(Node4,Step);
        z4 = Global.Plot.SavexNodesz(Node4,Step);
        Indexc1 = round(c(Node1,1)); Indexc2 = round(c(Node2,1));
        Indexc3 = round(c(Node3,1)); Indexc4 = round(c(Node4,1));
        c1 = ColorMap(Indexc1,:); c2 = ColorMap(Indexc2,:);
        c3 = ColorMap(Indexc3,:); c4 = ColorMap(Indexc4,:);
        cPlot(1,1,1:3) = c1; cPlot(1,2,1:3) = c2; cPlot(1,3,1:3) = c3; cPlot(1,4,1:3) = c4;
        xPlot = [x1;x2;x3;x4]; yPlot = [y1;y2;y3;y4]; zPlot = [z1;z2;z3;z4];
        patch(xPlot,yPlot,zPlot,cPlot);
    end
    hold off
    %-------------------------------------------------------------------------------------%
    Global.Plot.Movie(Step) = getframe(f);
    %-------------------------------------------------------------------------------------%
end
%-------------------------------------------------------------------------------------%
end
```

4.2.3 Script

Per ogni applicazione è stato scritto uno script che genera i dati di input per risolvere il problema inizializzando le classi. Esso è richiamato dalla function `Generate`. Il codice gira attraverso lo script `main`, riportato di seguito.

```
% Fem Solver for nonlinear geometrically exact shell element
%-------------------------------------------------------------------------------------%
Global = Generate('InputFile');
NonLinearSolver(Global);
%-------------------------------------------------------------------------------------%
PlotConfigurations(Global);
```

Lo script precedente consente di eseguire l'applicazione scelta semplicemente sostituendo ad `InputFile` la stringa con la quale essa è stata rinominata.

Capitolo 5

Applicazioni

In questo capitolo sono presentati i risultati del codice in riferimento ad alcune applicazioni presenti in [12]. In particolare, seguendo l'algoritmo descritto nella sezione 3.2.3.1, sono studiate le applicazioni relative ad una mensola incastrata con un momento flettente agente all'estremo libero (*Roll-up of a clamped beam*), ad una una semisfera caratterizzata da un foro in corrispondenza del polo (*Pinched hemispherical shell with an 18 degrees hole*, per la quale viene inoltre proposta un'applicazione di eversione, *Eversion of the hemisphere*) e ad una superficie cilindrica (*Snap-through of a hinged cylindrical panel*). Per l'ultima applicazione, inoltre, è implementato un solutore per un'analisi in displacement control ed un solutore per un'analisi in arc-length control, in modo da seguire l'intero percorso di carico che si perderebbe con il semplice algoritmo in force control. Le applicazioni sono richiamate con i classici nomi presenti in letteratura.

5.1 Roll-up of a clamped beam

Si studia il comportamento di una trave incastrata ad un estremo e con un momento flettente applicato all'altro.

Considerando i parametri elastici e costitutivi presentati in [12], il codice non converge; si sceglie dunque di riprodurre l'applicazione seguendo i dati presenti in [9], i quali definiscono lo shell con un modulo elastico pari a $E = 1.000$, un coefficiente di Poisson nullo $\nu = 0$ ed uno spessore pari a $h = 1$. La trave presenta una larghezza pari ad 1 ed una lunghezza pari a 12. Si sceglie di modellare la trave con 25 elementi equispaziati in direzione longitudinale e 2 elementi equispaziati in direzione trasversale. Si fa riferimento alla figura 5.1.

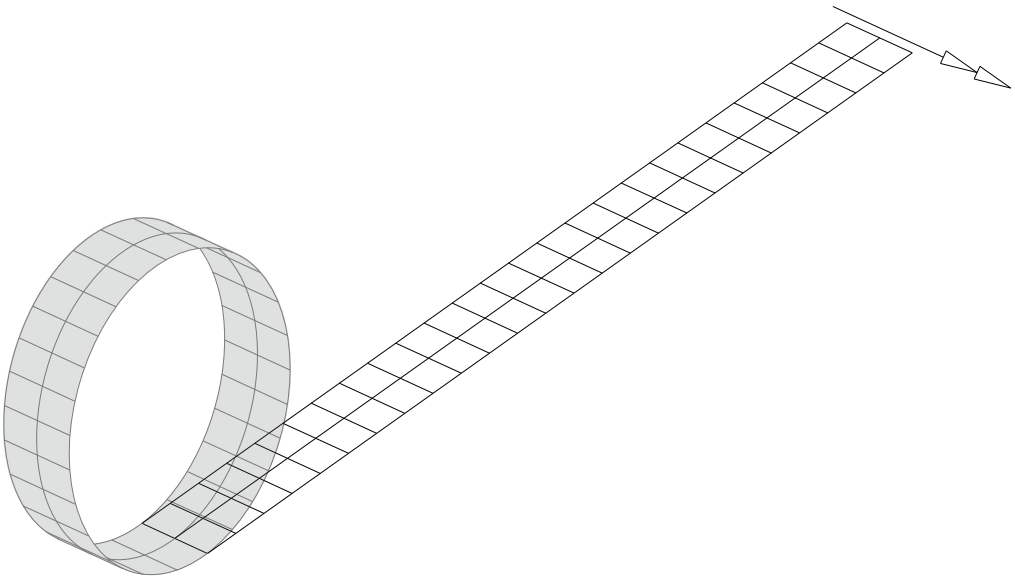


Figura 5.1: *In figura la configurazione iniziale e, evidenziata in grigetto, la configurazione deformata dell'applicazione. Immagine fedele a quanto raffigurato in [9].*

Seguendo quanto definito in [9], il momento che determina il completo arrotolamento della trave può essere ricavato analiticamente, imponendo una configurazione deformata a forma di circonferenza, a partire dal tensore materiale di deformazione secondo la legge elastica (2.22). Così facendo, si può verificare che si ottiene un momento totale all'estremo caricato pari a:

$$M = \frac{2EI\pi}{L}\Big(1 - \frac{2}{3}\frac{h^2\pi^2}{L^2}\Big)$$

avendo indicato con b la larghezza della trave e con $I = \frac{bh^3}{12}$ il *momento d'inerzia* della sezione.

L'analisi è svolta considerando 10 step di carico uniformi. In figura 5.2 è riportata la configurazione iniziale (si indica con `force` il valore del momento M agente). La mappa colori identifica il modulo dello spostamento in riferimento alla configurazione iniziale. La superficie media dei tre nodi incastrati presenta le seguenti coordinate (indicando per righe i tre nodi incastrati e per colonne, rispettivamente, le coordinate x, y e z).

```
0.0         0.0         0.0
0.0         0.5         0.0
0.0         1.0         0.0
```

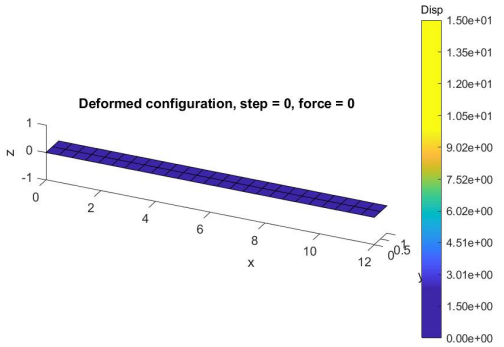
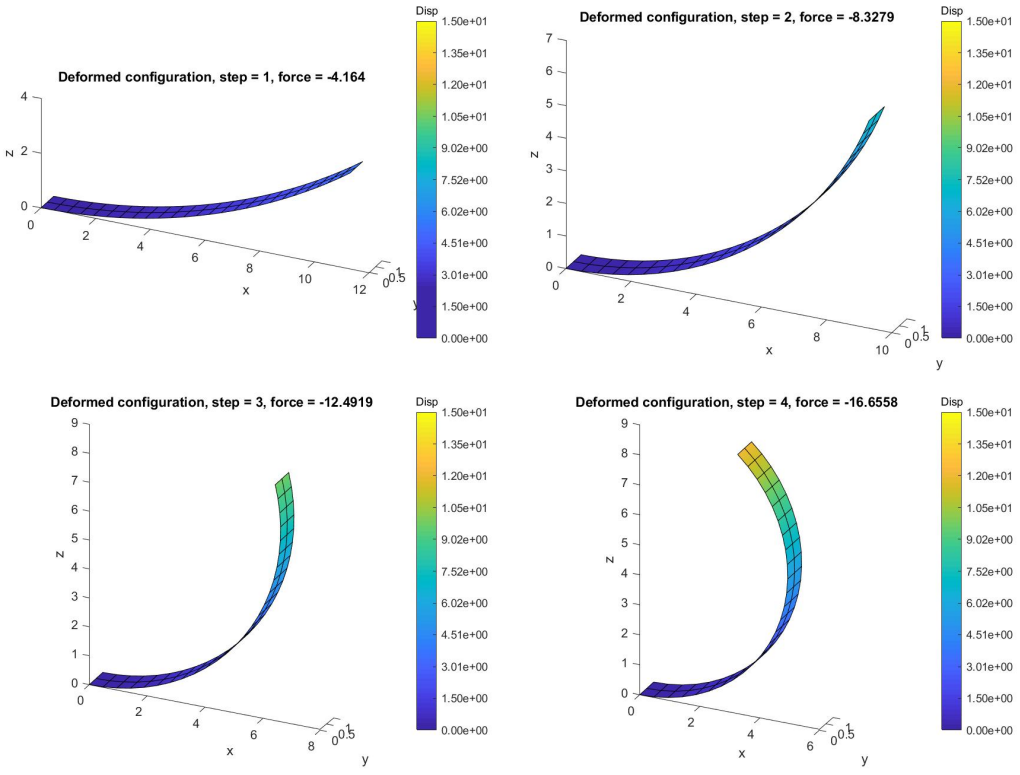
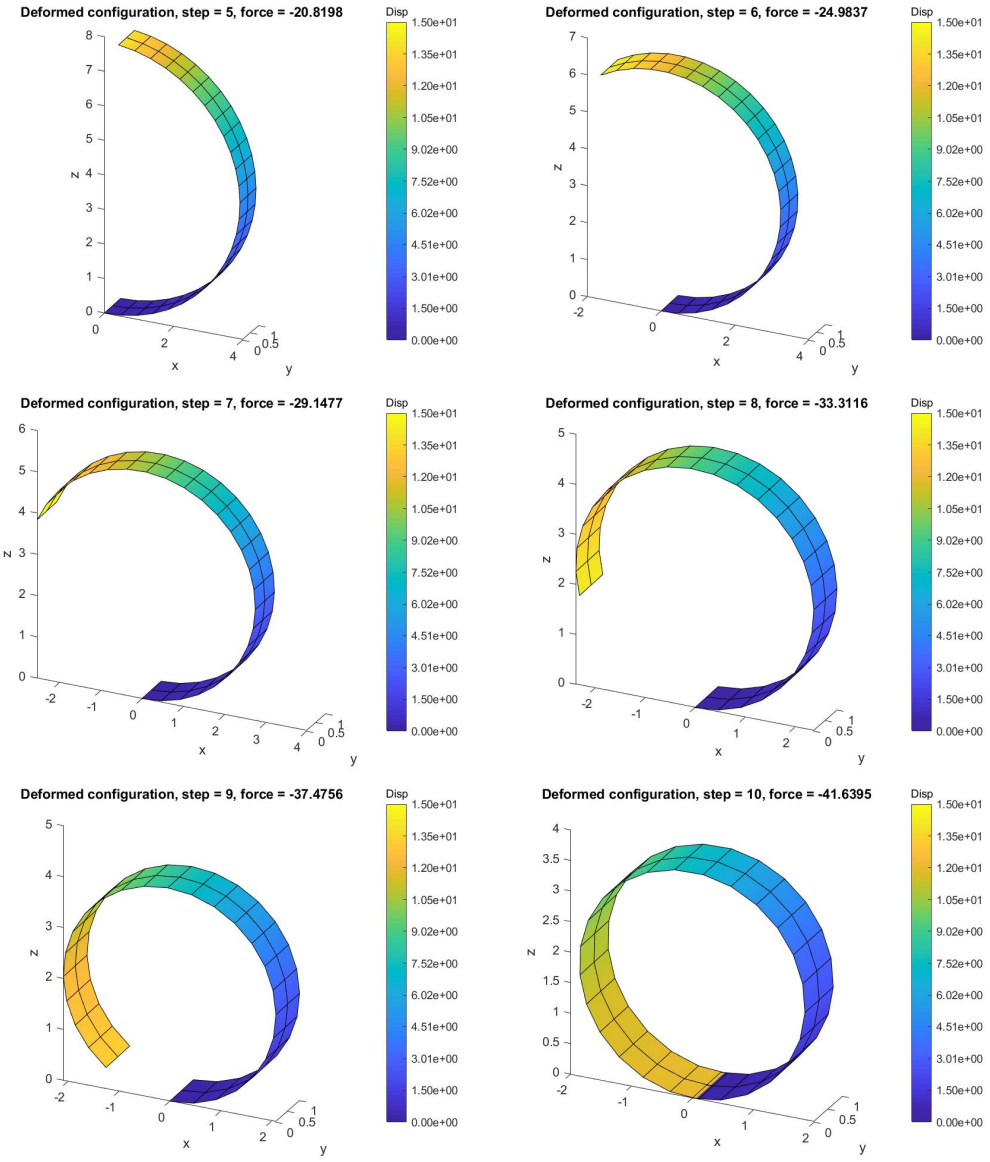


Figura 5.2

Nelle figure successive si riporta la configurazione deformata ad ogni step di carico.





Si nota che con il momento calcolato analiticamente si ha il completo arrotolamento della mesh. Le coordinate dei nodi appartenenti all'estremo libero alla convergenza risultano (indicando per righe i tre nodi all'estremo libero e per colonne, rispettivamente, le coordinate x, y e z).

```
0.104767318282422     -6.88534904759e-10     0.00340831133424287
0.104767318265396      0.499999999311465     0.00340831139908114
0.104767318248369      0.999999999311465     0.00340831146392191
```

5.1.1 Script di input

Si riporta lo script relativo al file di input.

```
% Input
%-------------------------------------------------------------------------------------------%
Lengthx = 12;
Lengthy = 1;
ElementsAlongx = 25;
ElementsAlongy = 2;
%-------------------------------------------------------------------------------------------%
TotNodesElement = 4;
TotGaussPointsMembraneAndBending = 4;
TotGaussPointsShear = 1;
%-------------------------------------------------------------------------------------------%
ElasticModule = 1000;
PoissonCoefficient = 0.0;
%-------------------------------------------------------------------------------------------%
Height = 1;
ShearFactor = 5/6;
%-------------------------------------------------------------------------------------------%
TotDofSurfaceNode = 3;
TotDofDirectorNode = 2;
%-------------------------------------------------------------------------------------------%
TotSpaceDimensions = 3;
ExternalMoment = -2*pi*ElasticModule*(Lengthy*Height^3/12)/Lengthx ...
    *(1-2*Height^2*pi^2/3/Lengthx^2);
ExternalForce = ExternalMoment;
%-------------------------------------------------------------------------------------------%
Tollerance = 10^-6;
MaxIterations = 100;
LoadSteps = 10;
%-------------------------------------------------------------------------------------------%
% Generate Problem
%-------------------------------------------------------------------------------------------%
NodesAlongx = ElementsAlongx+1;
NodesAlongy = ElementsAlongy+1;
Dx = Lengthx/ElementsAlongx;
Dy = Lengthy/ElementsAlongy;
ExternalMomentDistributed = ExternalMoment/(NodesAlongy-1);
I = eye(3); e3 = [0 0 1]';
%-------------------------------------------------------------------------------------------%
TotNodes = NodesAlongx*NodesAlongy;
TotElements = ElementsAlongx*ElementsAlongy;
TotDofNode = TotDofSurfaceNode+TotDofDirectorNode;
TotDofSurface = TotDofSurfaceNode*TotNodes;
TotDofDirector = TotDofDirectorNode*TotNodes;
TotDof = TotDofSurface+TotDofDirector;
TotDofSurfaceElement = TotDofSurfaceNode*TotNodesElement;
TotDofDirectorElement = TotDofDirectorNode*TotNodesElement;
TotDofElement = TotDofSurfaceElement+TotDofDirectorElement;
%-------------------------------------------------------------------------------------------%
x0Nodes = zeros(TotNodes,3);
t0Nodes = zeros(TotNodes,3);
R0Nodes = zeros(3*TotNodes,3);
Counter = 0;
for i = 1:NodesAlongx
    for j = 1:NodesAlongy
        Counter = Counter+1;
        x0Nodes(Counter,:) = [(i-1)*Dx (j-1)*Dy 0];
        t0Normalized = [0 0 1];
        t0Nodes(Counter,:) = t0Normalized;
        crosse3t0Normalized = cross(e3,t0Normalized');
        crosse3t0NormalizedTens = [0 -crosse3t0Normalized(3) crosse3t0Normalized(2);...
            crosse3t0Normalized(3) 0 -crosse3t0Normalized(1);...
            -crosse3t0Normalized(2) crosse3t0Normalized(1) 0];
```

```
        R0 = (e3'*t0Normalized')*I+crosse3t0NormalizedTens ...
            +(crosse3t0Normalized*crosse3t0Normalized')/...
            (1+(e3'*t0Normalized'));
        R0Nodes(3*(Counter-1)+1:3*Counter,:) = R0;
    end
end
%-------------------------------------------------------------------------------------------%
Connectivity = zeros(TotElements,4);
Counter = 0;
for i = 1:ElementsAlongx
    FirstNode1 = i*NodesAlongy+2;
    FirstNode2 = (i-1)*NodesAlongy+2;
    FirstNode3 = (i-1)*NodesAlongy+1;
    FirstNode4 = i*NodesAlongy+1;
    for j = 1:ElementsAlongy
        Counter = Counter+1;
        Node1 = FirstNode1+j-1;
        Node2 = FirstNode2+j-1;
        Node3 = FirstNode3+j-1;
        Node4 = FirstNode4+j-1;
        Connectivity(Counter,:) = [Node1 Node2 Node3 Node4];
    end
end
%-------------------------------------------------------------------------------------------%
t0GaussPointsMembraneAndBending = zeros(3*TotElements,TotGaussPointsMembraneAndBending);
t0_1GaussPointsMembraneAndBending = zeros(TotElements*3,TotGaussPointsMembraneAndBending);
t0_2GaussPointsMembraneAndBending = zeros(TotElements*3,TotGaussPointsMembraneAndBending);
for ElementNumber = 1:TotElements
    Node1 = Connectivity(ElementNumber,1); Node2 = Connectivity(ElementNumber,2);
    Node3 = Connectivity(ElementNumber,3); Node4 = Connectivity(ElementNumber,4);
    t01 = t0Nodes(Node1,:)'; t02 = t0Nodes(Node2,:)';
    t03 = t0Nodes(Node3,:)'; t04 = t0Nodes(Node4,:)';
    for GPNumber = 1:TotGaussPointsMembraneAndBending
        [GP,~] = GaussPoint(GPNumber,TotGaussPointsMembraneAndBending);
        N1 = ShapeFunction(1,GP); N2 = ShapeFunction(2,GP);
        N3 = ShapeFunction(3,GP); N4 = ShapeFunction(4,GP);
        N1_1 = DerShapeFunction(1,1,GP); N1_2 = DerShapeFunction(1,2,GP);
        N2_1 = DerShapeFunction(2,1,GP); N2_2 = DerShapeFunction(2,2,GP);
        N3_1 = DerShapeFunction(3,1,GP); N3_2 = DerShapeFunction(3,2,GP);
        N4_1 = DerShapeFunction(4,1,GP); N4_2 = DerShapeFunction(4,2,GP);
        t0 = t01*N1+t02*N2+t03*N3+t04*N4;
        t0_1 = t01*N1_1+t02*N2_1+t03*N3_1+t04*N4_1;
        t0_2 = t01*N1_2+t02*N2_2+t03*N3_2+t04*N4_2;
        t0Normalized = t0/norm(t0,2);
        t0_1Corrected = (t0_1-(t0Normalized'*t0_1)*t0Normalized)/norm(t0,2);
        t0_2Corrected = (t0_2-(t0Normalized'*t0_2)*t0Normalized)/norm(t0,2);
        t0GaussPointsMembraneAndBending ...
            (3*(ElementNumber-1)+1:3*ElementNumber,GPNumber) = t0Normalized;
        t0_1GaussPointsMembraneAndBending ...
            (3*(ElementNumber-1)+1:3*ElementNumber,GPNumber) = t0_1Corrected;
        t0_2GaussPointsMembraneAndBending ...
            (3*(ElementNumber-1)+1:3*ElementNumber,GPNumber) = t0_2Corrected;
    end
end
t0GaussPointsShear = zeros(3*TotElements,TotGaussPointsShear);
for ElementNumber = 1:TotElements
    Node1 = Connectivity(ElementNumber,1); Node2 = Connectivity(ElementNumber,2);
    Node3 = Connectivity(ElementNumber,3); Node4 = Connectivity(ElementNumber,4);
    t01 = t0Nodes(Node1,:)'; t02 = t0Nodes(Node2,:)';
    t03 = t0Nodes(Node3,:)'; t04 = t0Nodes(Node4,:)';
    for GPNumber = 1:TotGaussPointsShear
        [GP,~] = GaussPoint(GPNumber,TotGaussPointsShear);
```

```
        N1 = ShapeFunction(1,GP); N2 = ShapeFunction(2,GP);
        N3 = ShapeFunction(3,GP); N4 = ShapeFunction(4,GP);
        t0 = t01*N1+t02*N2+t03*N3+t04*N4;
        t0Normalized = t0/norm(t0,2);
        t0GaussPointsShear(3*(ElementNumber-1)+1:3*ElementNumber,GPNumber) = t0Normalized;
    end
end
%-------------------------------------------------------------------------------------------%
xNodes = x0Nodes;
tNodes = t0Nodes;
tGaussPointsMembraneAndBending = t0GaussPointsMembraneAndBending;
t_1GaussPointsMembraneAndBending = t0_1GaussPointsMembraneAndBending;
t_2GaussPointsMembraneAndBending = t0_2GaussPointsMembraneAndBending;
tGaussPointsShear = t0GaussPointsShear;
RNodes = R0Nodes;
%-------------------------------------------------------------------------------------------%
% Input boundary conditions
%-------------------------------------------------------------------------------------------%
BoundaryConditions = zeros(TotDof,1);
for i = 1:NodesAlongy
    BoundaryConditions(3*(i-1)+1,1) = 1;
    BoundaryConditions(3*(i-1)+2,1) = 1;
    BoundaryConditions(3*(i-1)+3,1) = 1;
    BoundaryConditions(TotDofSurface+2*(i-1)+1,1) = 1;
    BoundaryConditions(TotDofSurface+2*(i-1)+2,1) = 1;
end
SizeFreeDof = 0;
for i = 1:TotDof
    if BoundaryConditions(i,1) == 1
        SizeFreeDof = SizeFreeDof+1;
    end
end
FreeDof = zeros(SizeFreeDof,1);
Counter = 0;
for i = 1:TotDof
    if BoundaryConditions(i,1) == 0
        Counter = Counter+1;
        FreeDof(Counter,1) = i;
    end
end
%-------------------------------------------------------------------------------------------%
% Input external loads
%-------------------------------------------------------------------------------------------%
ExternalForces = zeros(TotDof,1);
ExternalForces(TotDofSurface+2*(TotNodes-NodesAlongy+1-1)+1,1) =  ...
    ExternalMomentDistributed/2;
for i = TotNodes-NodesAlongy+2:TotNodes-1
    ExternalForces(TotDofSurface+2*(i-1)+1,1) = ExternalMomentDistributed;
end
ExternalForces(TotDofSurface+2*(TotNodes-1)+1,1) = ExternalMomentDistributed/2;
%-------------------------------------------------------------------------------------------%
Figures = gobjects(LoadSteps+1,1);
Movie = struct('cdata',cell(LoadSteps+1,1),'colormap',cell(LoadSteps+1,1));
%-------------------------------------------------------------------------------------------%
SavexNodesx = zeros(TotNodes,LoadSteps+1);
SavexNodesx(:,1) = x0Nodes(:,1);
SavexNodesy = zeros(TotNodes,LoadSteps+1);
SavexNodesy(:,1) = x0Nodes(:,2);
SavexNodesz = zeros(TotNodes,LoadSteps+1);
SavexNodesz(:,1) = x0Nodes(:,3);
%-------------------------------------------------------------------------------------------%
DisplacementPlot = zeros(LoadSteps+1,1);
```

```
ForcePlot = zeros(LoadSteps+1,1);
NodePlot = TotNodes;
```

5.2 Pinched hemispherical shell with an 18 degrees hole

Si studia ora il comportamento di una mesh semisferica caratterizzata da un foro in sommità dell'ampiezza di 18°. In accordo ai dati riportati in [12], il raggio della sfera è pari a 10, il modulo elastico e il coefficiente di Poisson risultano, rispettivamente, $E = 6,825 \times 10^7$ e $\nu = 0,3$, mentre lo spessore degli elementi è scelto pari a $h = 0,04$.

Il sistema è caratterizzato da quattro forze esterne poste in quattro punti equidistanti sulla circonferenza di base, due verso l'interno e due verso l'esterno, rispettivamente, in posizioni diametralmente opposte. Il modulo di ogni forza è pari a $F = 200$. La traslazione dei quattro nodi sui quali è applicata la forza è vincolata verticalmente e nelle direzioni opposte a quella della forza applicata.

L'analisi è svolta considerando 256 elementi per ottante. Si fa riferimento alla figura 5.3 la quale riporta la mesh, iniziale e deformata, in corrispondenza di una mesh con 16 elementi per ottante.

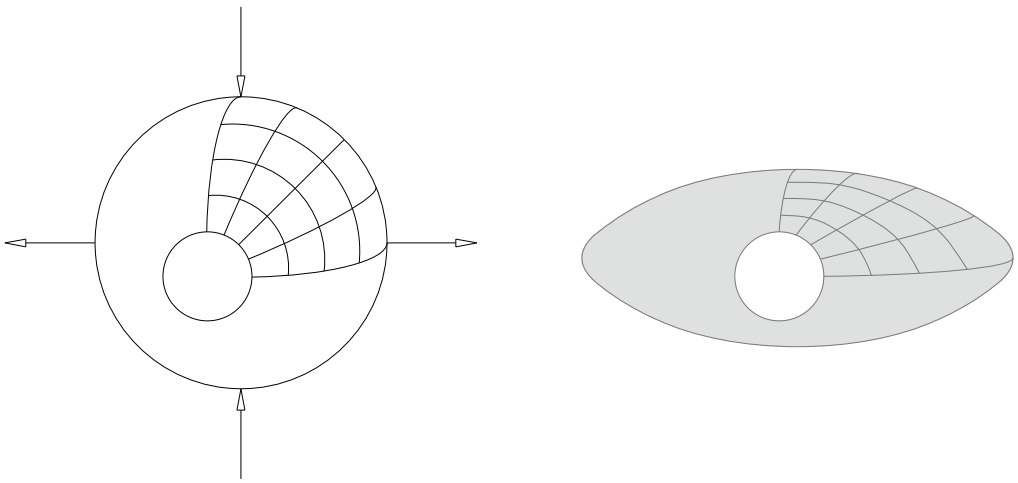


Figura 5.3: *In figura la configurazione iniziale e, evidenziata in grigetto, la configurazione deformata dell'applicazione. Immagine fedele a quanto raffigurato in [12].*

L'analisi è svolta considerando 10 step di carico uniformi. In figura 5.4 è riportata la configurazione iniziale (si indica con `force` il valore della singola

forza F agente). La mappa colori identifica il modulo dello spostamento in riferimento alla configurazione iniziale.

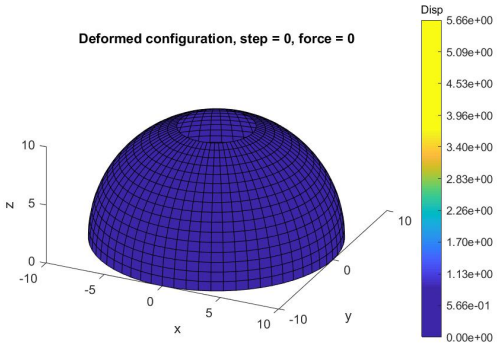
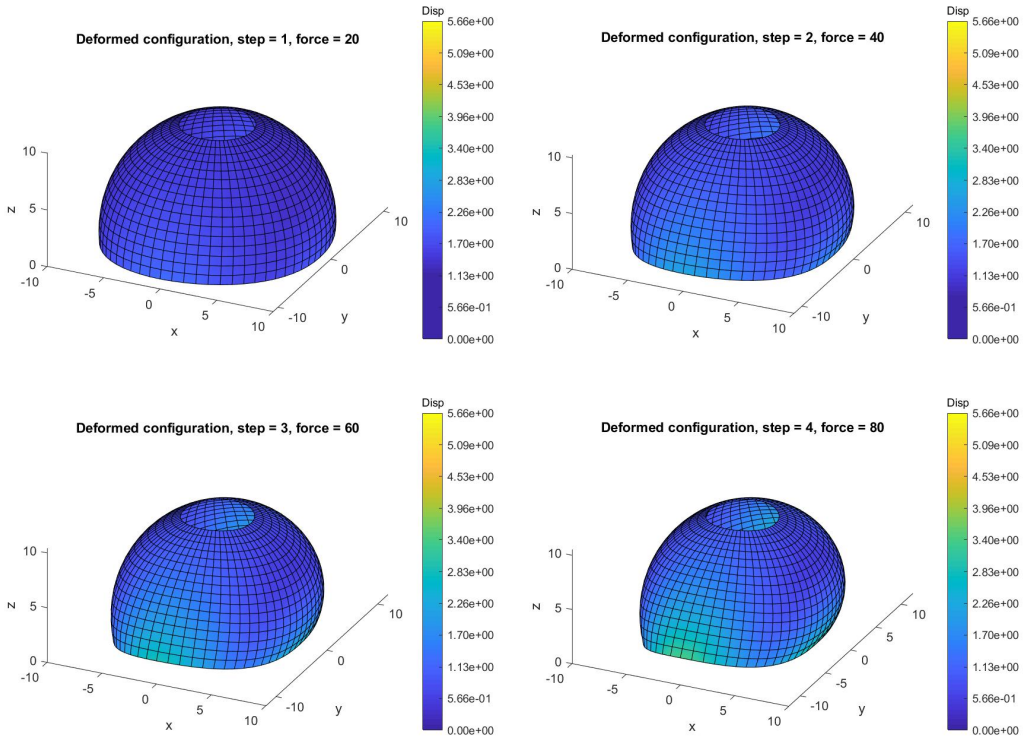
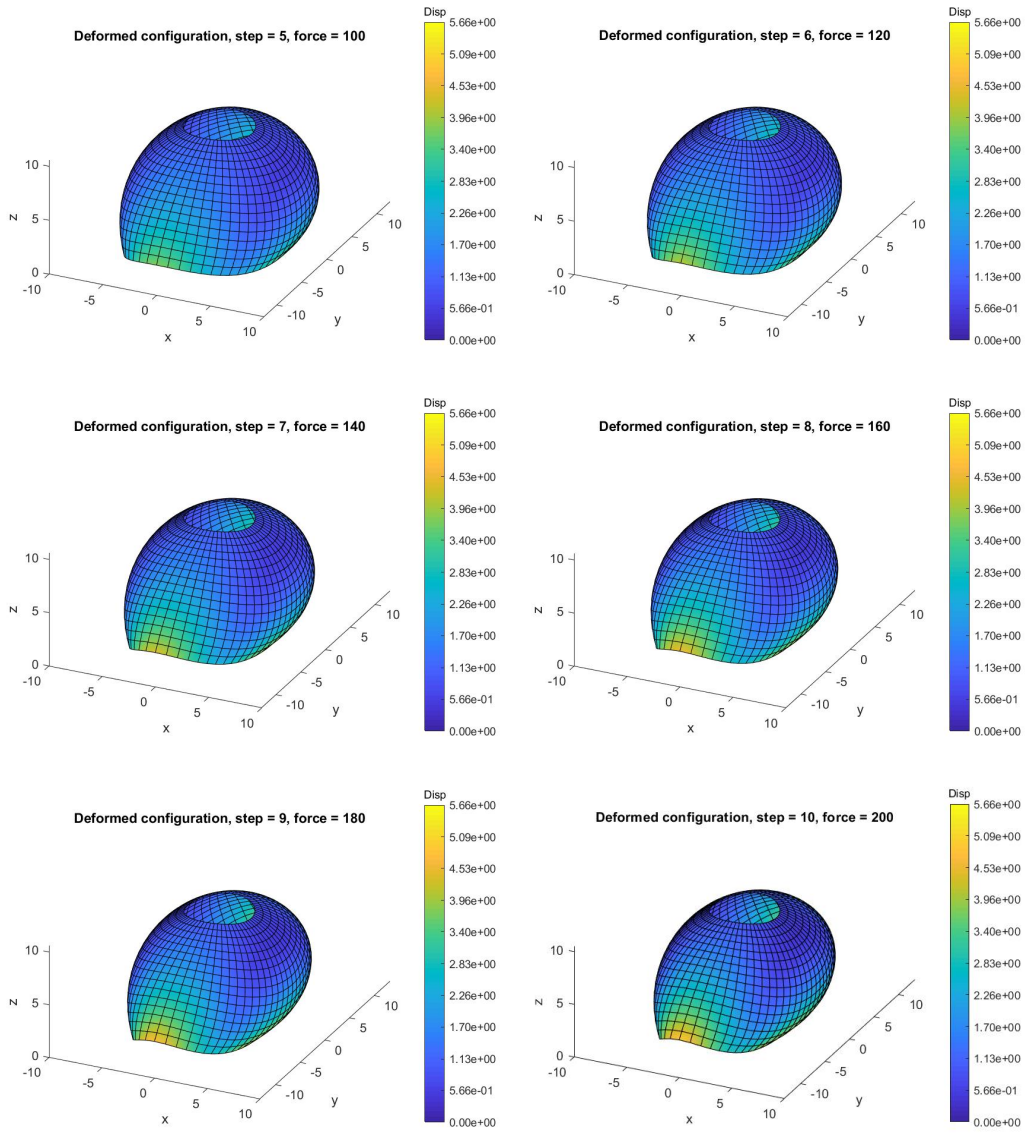


Figura 5.4

Nelle figure successive si riporta la configurazione deformata ad ogni step di carico.





In figura 5.5 è riportato un confronto con i risultati riportati in [12]; sono indicati gli spostamenti dei nodi caricati dalle forze verso l'interno (`pressed`) e verso l'esterno (`streched`), avendo indicato con un marker i punti corrispondenti ad ogni step di carico. In legenda, con la didascalia `Simo et al.`, si sono indicati i risultati riportati in [12].

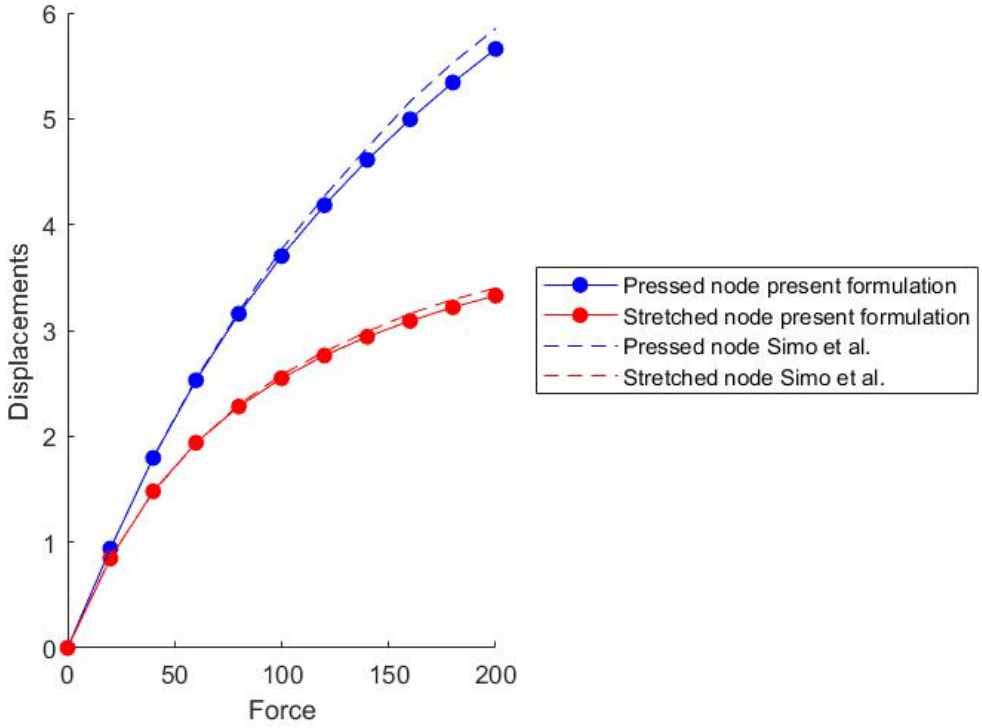


Figura 5.5: *Per un confronto con i dati nella presente formulazione, gli spostamenti riportati in [12] sono stati moltiplicati per* 2 *in modo da tenere in considerazione le condizioni di simmetria.*

5.2.1 Script di input

Si riporta lo script relativo al file di input.

```
% Input
%-------------------------------------------------------------------------------------------%
ElementsForParallel = 16;
ElementsForMeridian = 16;
alpha = 18*pi/180;
R = 10;
%-------------------------------------------------------------------------------------------%
TotNodesElement = 4;
TotGaussPointsMembraneAndBending = 4;
TotGaussPointsShear = 1;
%-------------------------------------------------------------------------------------------%
ElasticModule = 6.825*10^7;
PoissonCoefficient = 0.3;
%-------------------------------------------------------------------------------------------%
Height = 0.04;
ShearFactor = 5/6;
%-------------------------------------------------------------------------------------------%
TotDofSurfaceNode = 3;
```

```
TotDofDirectorNode = 2;
%-------------------------------------------------------------------------------------------%
TotSpaceDimensions = 3;
ExternalForce = 200;
%-------------------------------------------------------------------------------------------%
Tollerance = 10^-6;
MaxIterations = 100;
LoadSteps = 10;
%-------------------------------------------------------------------------------------------%
% Generate Problem
%-------------------------------------------------------------------------------------------%
NodesForParallel = ElementsForParallel;
NodesForMeridian = ElementsForMeridian+1;
Dtheta = pi/2/ElementsForParallel;
Dl = R*(pi/2-alpha)/ElementsForMeridian;
I = eye(3); e3 = [0 0 1]';
%-------------------------------------------------------------------------------------------%
TotNodes = 4*NodesForParallel*NodesForMeridian;
TotElements = 4*ElementsForParallel*ElementsForMeridian;
TotDofNode = TotDofSurfaceNode+TotDofDirectorNode;
TotDofSurface = TotDofSurfaceNode*TotNodes;
TotDofDirector = TotDofDirectorNode*TotNodes;
TotDof = TotDofSurface+TotDofDirector;
TotDofSurfaceElement = TotDofSurfaceNode*TotNodesElement;
TotDofDirectorElement = TotDofDirectorNode*TotNodesElement;
TotDofElement = TotDofSurfaceElement+TotDofDirectorElement;
%-------------------------------------------------------------------------------------------%
x0Nodes = zeros(TotNodes,3);
t0Nodes = zeros(TotNodes,3);
R0Nodes = zeros(3*TotNodes,3);
Counter = 0;
for i = 1:NodesForMeridian
    phi = (i-1)*Dl/R;
    for j = 1:4*NodesForParallel
        Counter = Counter+1;
        theta = (j-1)*Dtheta;
        x0 = [R*cos(phi)*cos(theta) R*cos(phi)*sin(theta) R*sin(phi)];
        x0Nodes(Counter,:) = x0;
        x0_theta = [-R*cos(phi)*sin(theta) R*cos(phi)*cos(theta) 0];
        x0_phi = [-R*sin(phi)*cos(theta) -R*sin(phi)*sin(theta) R*cos(phi)];
        t0 = cross(x0_theta,x0_phi);
        t0Normalized = t0/norm(t0,2);
        t0Nodes(Counter,:) = t0Normalized;
        crosse3t0Normalized = cross(e3,t0Normalized');
        crosse3t0NormalizedTens = [0 -crosse3t0Normalized(3) crosse3t0Normalized(2);...
            crosse3t0Normalized(3) 0 -crosse3t0Normalized(1);...
            -crosse3t0Normalized(2) crosse3t0Normalized(1) 0];
        R0 = (e3'*t0Normalized')*I+crosse3t0NormalizedTens+ ...
            (crosse3t0Normalized*crosse3t0Normalized')/(1+(e3'*t0Normalized'));
        R0Nodes(3*(Counter-1)+1:3*Counter,:) = R0;
    end
end
%-------------------------------------------------------------------------------------------%
Connectivity = zeros(TotElements,4);
Counter = 0;
for i = 1:ElementsForMeridian
    FirstNodeMeridian1 = i*4*ElementsForParallel+2;
    FirstNodeMeridian2 = i*4*ElementsForParallel+1;
    FirstNodeMeridian3 = (i-1)*4*ElementsForParallel+1;
    FirstNodeMeridian4 = (i-1)*4*ElementsForParallel+2;
    for j = 1:4*ElementsForParallel-1
        Counter = Counter+1;
```

```
        Node1 = FirstNodeMeridian1+j-1;
        Node2 = FirstNodeMeridian2+j-1;
        Node3 = FirstNodeMeridian3+j-1;
        Node4 = FirstNodeMeridian4+j-1;
        Connectivity(Counter,:) = [Node1 Node2 Node3 Node4];
    end
    Counter = Counter+1;
    Node2 = Node1;
    Node3 = Node4;
    Node1 = FirstNodeMeridian2;
    Node4 = FirstNodeMeridian3;
    Connectivity(Counter,:) = [Node1 Node2 Node3 Node4];
end
%-------------------------------------------------------------------------------------------%
t0GaussPointsMembraneAndBending = zeros(3*TotElements,TotGaussPointsMembraneAndBending);
t0_1GaussPointsMembraneAndBending = zeros(TotElements*3,TotGaussPointsMembraneAndBending);
t0_2GaussPointsMembraneAndBending = zeros(TotElements*3,TotGaussPointsMembraneAndBending);
for ElementNumber = 1:TotElements
    Node1 = Connectivity(ElementNumber,1); Node2 = Connectivity(ElementNumber,2);
    Node3 = Connectivity(ElementNumber,3); Node4 = Connectivity(ElementNumber,4);
    t01 = t0Nodes(Node1,:)'; t02 = t0Nodes(Node2,:)';
    t03 = t0Nodes(Node3,:)'; t04 = t0Nodes(Node4,:)';
    for GPNumber = 1:TotGaussPointsMembraneAndBending
        [GP,~] = GaussPoint(GPNumber,TotGaussPointsMembraneAndBending);
        N1 = ShapeFunction(1,GP); N2 = ShapeFunction(2,GP);
        N3 = ShapeFunction(3,GP); N4 = ShapeFunction(4,GP);
        N1_1 = DerShapeFunction(1,1,GP); N1_2 = DerShapeFunction(1,2,GP);
        N2_1 = DerShapeFunction(2,1,GP); N2_2 = DerShapeFunction(2,2,GP);
        N3_1 = DerShapeFunction(3,1,GP); N3_2 = DerShapeFunction(3,2,GP);
        N4_1 = DerShapeFunction(4,1,GP); N4_2 = DerShapeFunction(4,2,GP);
        t0 = t01*N1+t02*N2+t03*N3+t04*N4;
        t0_1 = t01*N1_1+t02*N2_1+t03*N3_1+t04*N4_1;
        t0_2 = t01*N1_2+t02*N2_2+t03*N3_2+t04*N4_2;
        t0Normalized = t0/norm(t0,2);
        t0_1Corrected = (t0_1-(t0Normalized'*t0_1)*t0Normalized)/norm(t0,2);
        t0_2Corrected = (t0_2-(t0Normalized'*t0_2)*t0Normalized)/norm(t0,2);
        t0GaussPointsMembraneAndBending ...
            (3*(ElementNumber-1)+1:3*ElementNumber,GPNumber) = t0Normalized;
        t0_1GaussPointsMembraneAndBending ...
            (3*(ElementNumber-1)+1:3*ElementNumber,GPNumber) = t0_1Corrected;
        t0_2GaussPointsMembraneAndBending ...
            (3*(ElementNumber-1)+1:3*ElementNumber,GPNumber) = t0_2Corrected;
    end
end
t0GaussPointsShear = zeros(3*TotElements,TotGaussPointsShear);
for ElementNumber = 1:TotElements
    Node1 = Connectivity(ElementNumber,1); Node2 = Connectivity(ElementNumber,2);
    Node3 = Connectivity(ElementNumber,3); Node4 = Connectivity(ElementNumber,4);
    t01 = t0Nodes(Node1,:)'; t02 = t0Nodes(Node2,:)';
    t03 = t0Nodes(Node3,:)'; t04 = t0Nodes(Node4,:)';
    for GPNumber = 1:TotGaussPointsShear
        [GP,~] = GaussPoint(GPNumber,TotGaussPointsShear);
        N1 = ShapeFunction(1,GP); N2 = ShapeFunction(2,GP);
        N3 = ShapeFunction(3,GP); N4 = ShapeFunction(4,GP);
        t0 = t01*N1+t02*N2+t03*N3+t04*N4;
        t0Normalized = t0/norm(t0,2);
        t0GaussPointsShear(3*(ElementNumber-1)+1:3*ElementNumber,GPNumber) = t0Normalized;
    end
end
%-------------------------------------------------------------------------------------------%
xNodes = x0Nodes;
tNodes = t0Nodes;
```

```
tGaussPointsMembraneAndBending = t0GaussPointsMembraneAndBending;
t_1GaussPointsMembraneAndBending = t0_1GaussPointsMembraneAndBending;
t_2GaussPointsMembraneAndBending = t0_2GaussPointsMembraneAndBending;
tGaussPointsShear = t0GaussPointsShear;
RNodes = RONodes;
%-------------------------------------------------------------------------------------%
NodesAlongMeridian1 = zeros(NodesForMeridian,1);
NodesAlongMeridian2 = zeros(NodesForMeridian,1);
NodesAlongMeridian3 = zeros(NodesForMeridian,1);
NodesAlongMeridian4 = zeros(NodesForMeridian,1);
FirstNodeMeridian1 = 1;
FirstNodeMeridian2 = FirstNodeMeridian1+NodesForParallel;
FirstNodeMeridian3 = FirstNodeMeridian2+NodesForParallel;
FirstNodeMeridian4 = FirstNodeMeridian3+NodesForParallel;
PressedNode = FirstNodeMeridian1;
StretchedNode = FirstNodeMeridian2;
for i = 1:NodesForMeridian
    Node1 = FirstNodeMeridian1+(i-1)*4*NodesForParallel;
    Node2 = FirstNodeMeridian2+(i-1)*4*NodesForParallel;
    Node3 = FirstNodeMeridian3+(i-1)*4*NodesForParallel;
    Node4 = FirstNodeMeridian4+(i-1)*4*NodesForParallel;
    NodesAlongMeridian1(i,1) = Node1;
    NodesAlongMeridian2(i,1) = Node2;
    NodesAlongMeridian3(i,1) = Node3;
    NodesAlongMeridian4(i,1) = Node4;
end
BoundaryConditions = zeros(TotDof,1);
for i = 1:NodesForMeridian
    Node1 = NodesAlongMeridian1(i,1);
    Node2 = NodesAlongMeridian2(i,1);
    Node3 = NodesAlongMeridian3(i,1);
    Node4 = NodesAlongMeridian4(i,1);
    BoundaryConditions(3*Node1-2+1,1) = 1;
    BoundaryConditions(3*Node2-2,1) = 1;
    BoundaryConditions(3*Node3-2+1,1) = 1;
    BoundaryConditions(3*Node4-2,1) = 1;
end
BoundaryConditions(3*FirstNodeMeridian1) = 1;
BoundaryConditions(3*FirstNodeMeridian3) = 1;
SizeFreeDof = 0;
for i = 1:TotDof
    if BoundaryConditions(i,1) == 1
        SizeFreeDof = SizeFreeDof+1;
    end
end
FreeDof = zeros(SizeFreeDof,1);
Counter = 0;
for i = 1:TotDof
    if BoundaryConditions(i,1) == 0
        Counter = Counter+1;
        FreeDof(Counter,1) = i;
    end
end
%-------------------------------------------------------------------------------------%
ExternalForces = zeros(TotDof,1);
ExternalForces(3*FirstNodeMeridian1-2,1) = -ExternalForce;
ExternalForces(3*FirstNodeMeridian2-2+1,1) = ExternalForce;
ExternalForces(3*FirstNodeMeridian3-2,1) = ExternalForce;
ExternalForces(3*FirstNodeMeridian4-2+1,1) = -ExternalForce;
%-------------------------------------------------------------------------------------%
Figures = gobjects(LoadSteps+1,1);
Movie = struct('cdata',cell(LoadSteps+1,1),'colormap',cell(LoadSteps+1,1));
```

```
%-------------------------------------------------------------------------------------------%
SavexNodesx = zeros(TotNodes,LoadSteps+1);
SavexNodesx(:,1) = x0Nodes(:,1);
SavexNodesy = zeros(TotNodes,LoadSteps+1);
SavexNodesy(:,1) = x0Nodes(:,2);
SavexNodesz = zeros(TotNodes,LoadSteps+1);
SavexNodesz(:,1) = x0Nodes(:,3);
%-------------------------------------------------------------------------------------------%
DisplacementPlot = zeros(LoadSteps+1,1);
ForcePlot = zeros(LoadSteps+1,1);
NodePlot = [PressedNode;StretchedNode];
```

5.2.2 Eversion of the hemisphere

Cambiando i parametri geometrici e costitutivi e le condizioni al contorno, in riferimento alla figura 5.6 (la quale riporta la mesh considerando 16 elementi per ottante), si riportano i risultati derivanti dall'applicazione di un carico verticale verso il basso uniformemente distribuito sui nodi in corrsipondenza del polo.

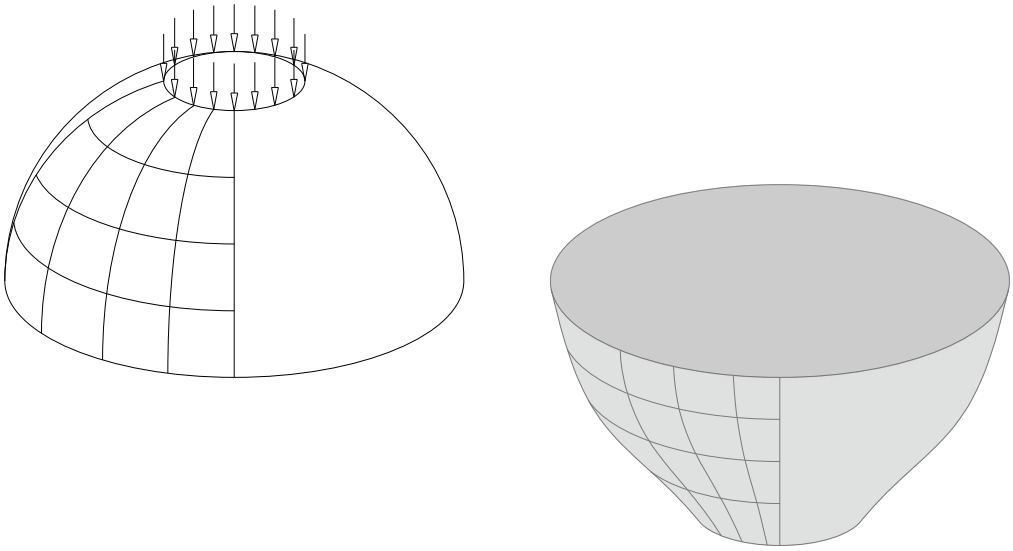


Figura 5.6: *In figura la configurazione iniziale e, evidenziata in grigetto, la configurazione deformata dell'applicazione.*

Si considera un modulo elastico pari a $E = 1.000$, un coefficiente di Poisson pari a $\nu = 0,3$ ed uno spessore dello shell pari a $h = 0,1$. Il raggio della sfera è pari a 1 (mentre il foro in sommità mantiene sempre l'ampiezza di 18°).

I nodi in corrispondeza della circonferenza di base sono vincolati lungo le tre direzioni. La risultante delle forze dirette verso il basso è pari a 12. Si sono considerati 256 elementi per ottante.

L'analisi è svolta considerando 10 step di carico uniformi. In figura 5.7 è riportata la configurazione iniziale (si indica con `force` il valore della singola forza F agente). La mappa colori identifica il modulo dello spostamento in riferimento alla configurazione iniziale.

In figura 5.8 è riportato il diagramma relativo allo spostamento dei nodi caricati rispetto al valore della risultante delle forze agenti (quantità normalizzate), rispettivamente alla convergenza per ogni step di carico, confrontato con delle applicazioni simili relative ad una semisfera completa (senza il foro), riportate in [8] e [12].

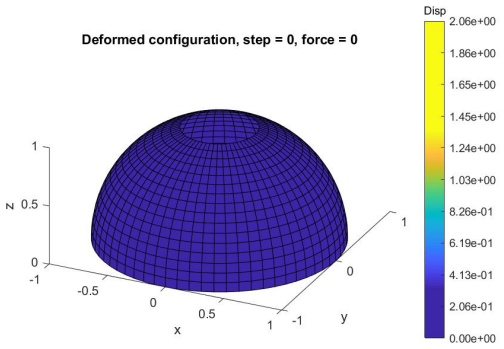
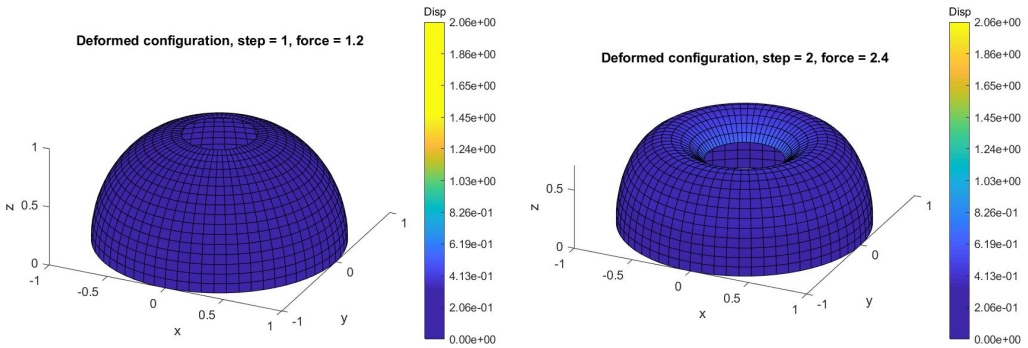
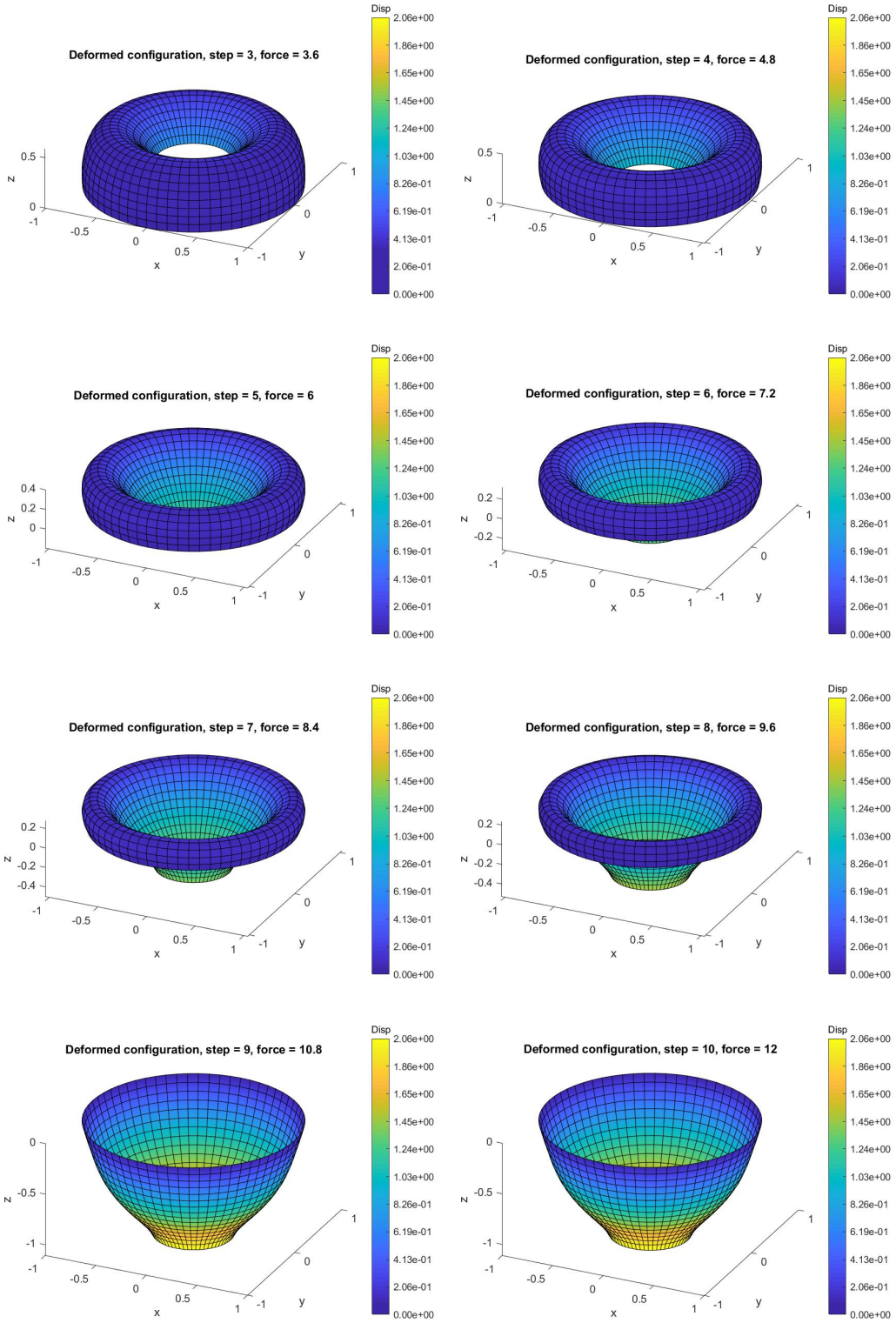


Figura 5.7

Nelle figure successive si riporta la configurazione deformata ad ogni step di carico.




Deformed configuration, step = 3, force = 3.6
Deformed configuration, step = 4, force = 4.8
Deformed configuration, step = 5, force = 6
Deformed configuration, step = 6, force = 7.2
Deformed configuration, step = 7, force = 8.4
Deformed configuration, step = 8, force = 9.6
Deformed configuration, step = 9, force = 10.8
Deformed configuration, step = 10, force = 12
Disp
2.06e+00
1.86e+00
1.65e+00
1.45e+00
1.24e+00
1.03e+00
8.26e-01
6.19e-01
4.13e-01
2.06e-01
0.00e+00
x
y
z

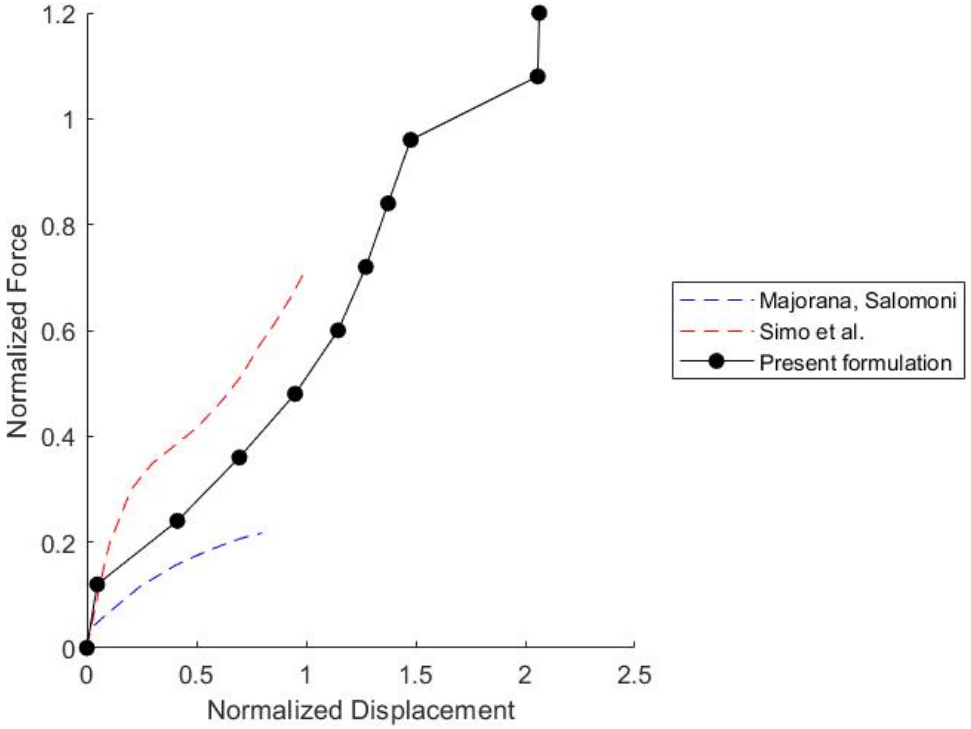


Figura 5.8: *In ascissa lo spostamento $\Delta\varphi$ normalizzato: $\Delta\varphi/R$, rispettivamente, dei nodi in corrispondenza della circonferenza sul foro per la presente formulazione, del nodo in corrispodenza del polo per le pubblicazioni [8], [12]. In ordinata la risultante delle forze esterne F normalizzata: $F/(Eh^2)$.*

5.2.2.1 Script di input

Si riporta lo script relativo al file di input.

```
% Input
%-------------------------------------------------------------------------------------------%
ElementsForParallel = 16;
ElementsForMeridian = 16;
alpha = 18*pi/180;
R = 1;
%-------------------------------------------------------------------------------------------%
TotNodesElement = 4;
TotGaussPointsMembraneAndBending = 4;
TotGaussPointsShear = 1;
%-------------------------------------------------------------------------------------------%
ElasticModule = 1000;
PoissonCoefficient = 0.3;
%-------------------------------------------------------------------------------------------%
Height = 0.1;
ShearFactor = 5/6;
%-------------------------------------------------------------------------------------------%
```

```
TotDofSurfaceNode = 3;
TotDofDirectorNode = 2;
%-------------------------------------------------------------------------------------------%
TotSpaceDimensions = 3;
ExternalForce = 12;
%-------------------------------------------------------------------------------------------%
Tollerance = 10^-6;
MaxIterations = 20;
LoadSteps = 10;
%-------------------------------------------------------------------------------------------%
% Generate Problem
%-------------------------------------------------------------------------------------------%
NodesForParallel = ElementsForParallel;
NodesForMeridian = ElementsForMeridian+1;
Dtheta = pi/2/ElementsForParallel;
Dl = R*(pi/2-alpha)/ElementsForMeridian;
I = eye(3); e3 = [0 0 1]';
%-------------------------------------------------------------------------------------------%
TotNodes = 4*NodesForParallel*NodesForMeridian;
TotElements = 4*ElementsForParallel*ElementsForMeridian;
TotDofNode = TotDofSurfaceNode+TotDofDirectorNode;
TotDofSurface = TotDofSurfaceNode*TotNodes;
TotDofDirector = TotDofDirectorNode*TotNodes;
TotDof = TotDofSurface+TotDofDirector;
TotDofSurfaceElement = TotDofSurfaceNode*TotNodesElement;
TotDofDirectorElement = TotDofDirectorNode*TotNodesElement;
TotDofElement = TotDofSurfaceElement+TotDofDirectorElement;
%-------------------------------------------------------------------------------------------%
x0Nodes = zeros(TotNodes,3);
t0Nodes = zeros(TotNodes,3);
R0Nodes = zeros(3*TotNodes,3);
Counter = 0;
for i = 1:NodesForMeridian
    phi = (i-1)*Dl/R;
    for j = 1:4*NodesForParallel
        Counter = Counter+1;
        theta = (j-1)*Dtheta;
        x0 = [R*cos(phi)*cos(theta) R*cos(phi)*sin(theta) R*sin(phi)];
        x0Nodes(Counter,:) = x0;
        x0_theta = [-R*cos(phi)*sin(theta) R*cos(phi)*cos(theta) 0];
        x0_phi = [-R*sin(phi)*cos(theta) -R*sin(phi)*sin(theta) R*cos(phi)];
        t0 = cross(x0_theta,x0_phi);
        t0Normalized = t0/norm(t0,2);
        t0Nodes(Counter,:) = t0Normalized;
        crosse3t0Normalized = cross(e3,t0Normalized');
        crosse3t0NormalizedTens = [0 -crosse3t0Normalized(3) crosse3t0Normalized(2);...
            crosse3t0Normalized(3) 0 -crosse3t0Normalized(1);...
            -crosse3t0Normalized(2) crosse3t0Normalized(1) 0];
        R0 = (e3'*t0Normalized')*I+crosse3t0NormalizedTens ...
            +(crosse3t0Normalized*crosse3t0Normalized')/(1+(e3'*t0Normalized'));
        R0Nodes(3*(Counter-1)+1:3*Counter,:) = R0;
    end
end
%-------------------------------------------------------------------------------------------%
Connectivity = zeros(TotElements,4);
Counter = 0;
for i = 1:ElementsForMeridian
    FirstNode1 = i*4*ElementsForParallel+2;
    FirstNode2 = i*4*ElementsForParallel+1;
    FirstNode3 = (i-1)*4*ElementsForParallel+1;
    FirstNode4 = (i-1)*4*ElementsForParallel+2;
    for j = 1:4*ElementsForParallel-1
```

```
        Counter = Counter+1;
        Node1 = FirstNode1+j-1;
        Node2 = FirstNode2+j-1;
        Node3 = FirstNode3+j-1;
        Node4 = FirstNode4+j-1;
        Connectivity(Counter,:) = [Node1 Node2 Node3 Node4];
    end
    Counter = Counter+1;
    Node2 = Node1;
    Node3 = Node4;
    Node1 = FirstNode2;
    Node4 = FirstNode3;
    Connectivity(Counter,:) = [Node1 Node2 Node3 Node4];
end
%-------------------------------------------------------------------------------------------%
t0GaussPointsMembraneAndBending = zeros(3*TotElements,TotGaussPointsMembraneAndBending);
t0_1GaussPointsMembraneAndBending = zeros(TotElements*3,TotGaussPointsMembraneAndBending);
t0_2GaussPointsMembraneAndBending = zeros(TotElements*3,TotGaussPointsMembraneAndBending);
for ElementNumber = 1:TotElements
    Node1 = Connectivity(ElementNumber,1); Node2 = Connectivity(ElementNumber,2);
    Node3 = Connectivity(ElementNumber,3); Node4 = Connectivity(ElementNumber,4);
    t01 = t0Nodes(Node1,:)'; t02 = t0Nodes(Node2,:)';
    t03 = t0Nodes(Node3,:)'; t04 = t0Nodes(Node4,:)';
    for GPNumber = 1:TotGaussPointsMembraneAndBending
        [GP,~] = GaussPoint(GPNumber,TotGaussPointsMembraneAndBending);
        N1 = ShapeFunction(1,GP); N2 = ShapeFunction(2,GP);
        N3 = ShapeFunction(3,GP); N4 = ShapeFunction(4,GP);
        N1_1 = DerShapeFunction(1,1,GP); N1_2 = DerShapeFunction(1,2,GP);
        N2_1 = DerShapeFunction(2,1,GP); N2_2 = DerShapeFunction(2,2,GP);
        N3_1 = DerShapeFunction(3,1,GP); N3_2 = DerShapeFunction(3,2,GP);
        N4_1 = DerShapeFunction(4,1,GP); N4_2 = DerShapeFunction(4,2,GP);
        t0 = t01*N1+t02*N2+t03*N3+t04*N4;
        t0_1 = t01*N1_1+t02*N2_1+t03*N3_1+t04*N4_1;
        t0_2 = t01*N1_2+t02*N2_2+t03*N3_2+t04*N4_2;
        t0Normalized = t0/norm(t0,2);
        t0_1Corrected = (t0_1-(t0Normalized'*t0_1)*t0Normalized)/norm(t0,2);
        t0_2Corrected = (t0_2-(t0Normalized'*t0_2)*t0Normalized)/norm(t0,2);
        t0GaussPointsMembraneAndBending ...
            (3*(ElementNumber-1)+1:3*ElementNumber,GPNumber) = t0Normalized;
        t0_1GaussPointsMembraneAndBending ...
            (3*(ElementNumber-1)+1:3*ElementNumber,GPNumber) = t0_1Corrected;
        t0_2GaussPointsMembraneAndBending ...
            (3*(ElementNumber-1)+1:3*ElementNumber,GPNumber) = t0_2Corrected;
    end
end
t0GaussPointsShear = zeros(3*TotElements,TotGaussPointsShear);
for ElementNumber = 1:TotElements
    Node1 = Connectivity(ElementNumber,1); Node2 = Connectivity(ElementNumber,2);
    Node3 = Connectivity(ElementNumber,3); Node4 = Connectivity(ElementNumber,4);
    t01 = t0Nodes(Node1,:)'; t02 = t0Nodes(Node2,:)';
    t03 = t0Nodes(Node3,:)'; t04 = t0Nodes(Node4,:)';
    for GPNumber = 1:TotGaussPointsShear
        [GP,~] = GaussPoint(GPNumber,TotGaussPointsShear);
        N1 = ShapeFunction(1,GP); N2 = ShapeFunction(2,GP);
        N3 = ShapeFunction(3,GP); N4 = ShapeFunction(4,GP);
        t0 = t01*N1+t02*N2+t03*N3+t04*N4;
        t0Normalized = t0/norm(t0,2);
        t0GaussPointsShear(3*(ElementNumber-1)+1:3*ElementNumber,GPNumber) = t0Normalized;
    end
end
%-------------------------------------------------------------------------------------------%
xNodes = x0Nodes;
```

```
tNodes = t0Nodes;
tGaussPointsMembraneAndBending = t0GaussPointsMembraneAndBending;
t_1GaussPointsMembraneAndBending = t0_1GaussPointsMembraneAndBending;
t_2GaussPointsMembraneAndBending = t0_2GaussPointsMembraneAndBending;
tGaussPointsShear = t0GaussPointsShear;
RNodes = R0Nodes;
%-------------------------------------------------------------------------------------------%
% Input boundary conditions
%-------------------------------------------------------------------------------------------%
BoundaryConditions = zeros(TotDof,1);
BoundaryConditions(1:3*4*NodesForParallel) = ones(3*4*NodesForParallel,1);
% BoundaryConditions ...
SizeFreeDof = 0;
for i = 1:TotDof
    if BoundaryConditions(i,1) == 1
        SizeFreeDof = SizeFreeDof+1;
    end
end
FreeDof = zeros(SizeFreeDof,1);
Counter = 0;
for i = 1:TotDof
    if BoundaryConditions(i,1) == 0
        Counter = Counter+1;
        FreeDof(Counter,1) = i;
    end
end
%-------------------------------------------------------------------------------------------%
% Input external loads
%-------------------------------------------------------------------------------------------%
ExternalForces = zeros(TotDof,1);
FirstNode = 4*NodesForParallel*(NodesForMeridian-1);
ExternalPressure = ExternalForce/4/NodesForParallel;
for i = 1:4*NodesForParallel
    Node = FirstNode+i;
    ExternalForces(3*Node,1) = -ExternalPressure;
end
NodePlot = Node;
%-------------------------------------------------------------------------------------------%
Figures = gobjects(LoadSteps+1,1);
Movie = struct('cdata',cell(LoadSteps+1,1),'colormap',cell(LoadSteps+1,1));
%-------------------------------------------------------------------------------------------%
SavexNodesx = zeros(TotNodes,LoadSteps+1);
SavexNodesx(:,1) = x0Nodes(:,1);
SavexNodesy = zeros(TotNodes,LoadSteps+1);
SavexNodesy(:,1) = x0Nodes(:,2);
SavexNodesz = zeros(TotNodes,LoadSteps+1);
SavexNodesz(:,1) = x0Nodes(:,3);
%-------------------------------------------------------------------------------------------%
DisplacementPlot = zeros(LoadSteps+1,1);
ForcePlot = zeros(LoadSteps+1,1);
```

5.3 Snap-through of a hinged cylindrical panel

La terza geometria proposta è in riferimento ad una superficie cilindrica, di ampiezza $0,2$ rad, raggio 2.540 e lunghezza 508. I parametri geometrici e costitutivi dello shell definiscono un modulo elastico $E = 3.102,75$, il coefficiente di Poisson $\nu = 0,3$ e lo spessore dello shell $h = 6,35$.

Il sistema è caratterizzato da una forza esterna spingente verso il basso sul nodo di simmetria pari a $F = 800$. Sono state bloccate le traslazioni lungo le tre direzioni nei nodi in corrispondenza delle due generatrici esterne.

Si nota che la struttura è definita da quattro superfici cilindriche in modo simmetrico.

L'analisi è stata svolta considerando 16 elementi per superficie. Si fa riferimento alla figura 5.9, la quale riporta la mesh iniziale e quella deformata.

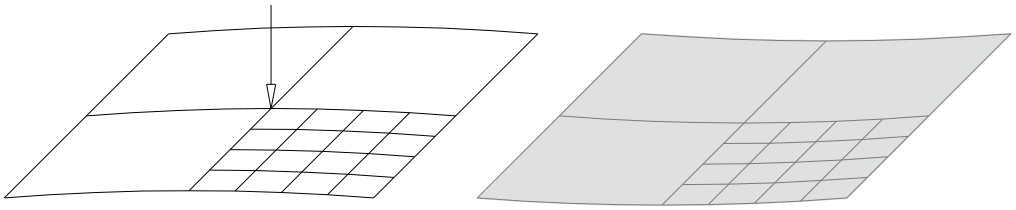


Figura 5.9: *In figura la configurazione iniziale e, evidenziata in grigetto, la configurazione deformata dell'applicazione. Immagine fedele a quanto raffigurato in [12].*

In questo caso, l'analisi è svolta considerando 30 step di carico uniformi. In figura 5.10 è riportata la configurazione iniziale (si indica con `force` il valore della singola forza F agente).

La mappa colori identifica il modulo dello spostamento in riferimento alla configurazione iniziale.

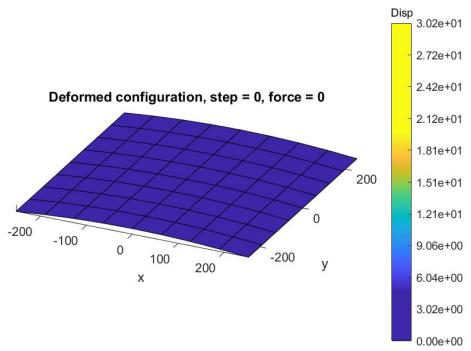
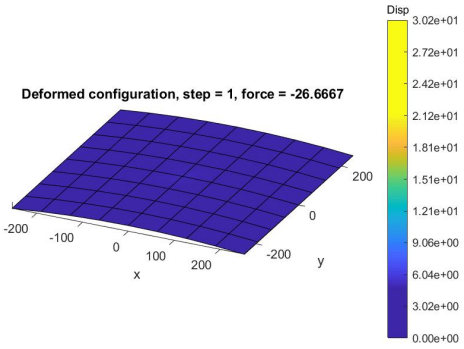
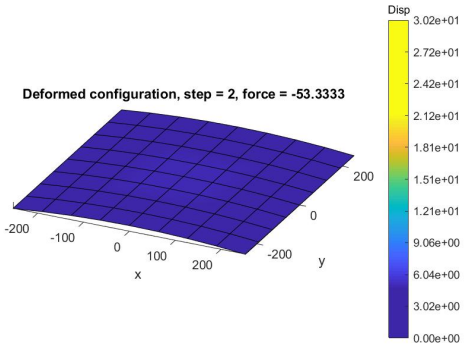
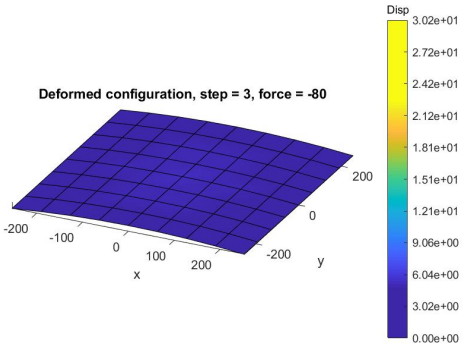


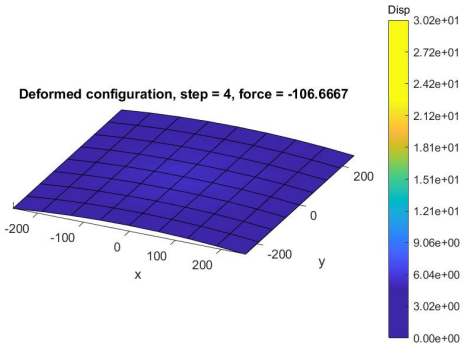
Figura 5.10

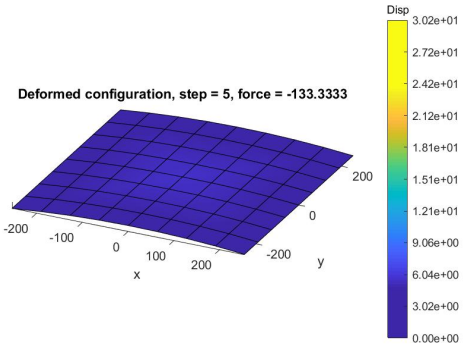
Nelle figure successive si riporta la configurazione deformata ad ogni step di carico.

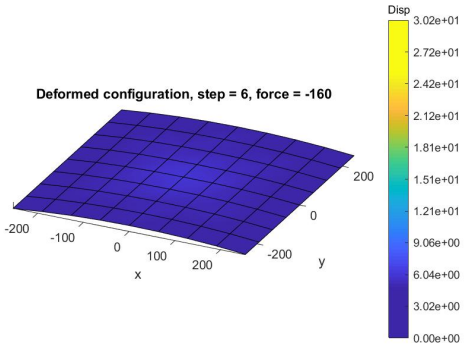

Deformed configuration, step = 1, force = -26.6667
Disp
3.02e+01
2.72e+01
2.42e+01
2.12e+01
1.81e+01
1.51e+01
1.21e+01
9.06e+00
6.04e+00
3.02e+00
0.00e+00
-200
-100
0
100
200
x
y

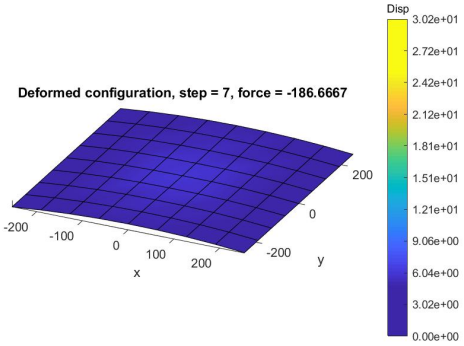

Deformed configuration, step = 2, force = -53.3333
Disp
3.02e+01
2.72e+01
2.42e+01
2.12e+01
1.81e+01
1.51e+01
1.21e+01
9.06e+00
6.04e+00
3.02e+00
0.00e+00
-200
-100
0
100
200
x
y

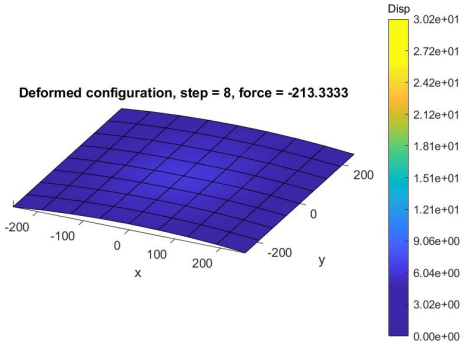

Deformed configuration, step = 3, force = -80
Disp
3.02e+01
2.72e+01
2.42e+01
2.12e+01
1.81e+01
1.51e+01
1.21e+01
9.06e+00
6.04e+00
3.02e+00
0.00e+00
-200
-100
0
100
200
x
y

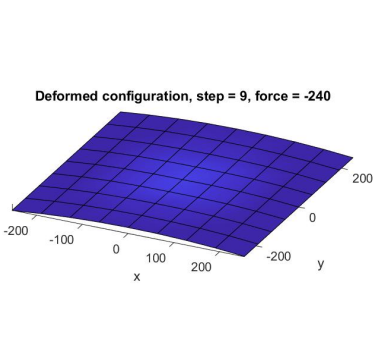

Deformed configuration, step = 4, force = -106.6667
Disp
3.02e+01
2.72e+01
2.42e+01
2.12e+01
1.81e+01
1.51e+01
1.21e+01
9.06e+00
6.04e+00
3.02e+00
0.00e+00
-200
-100
0
100
200
x
y

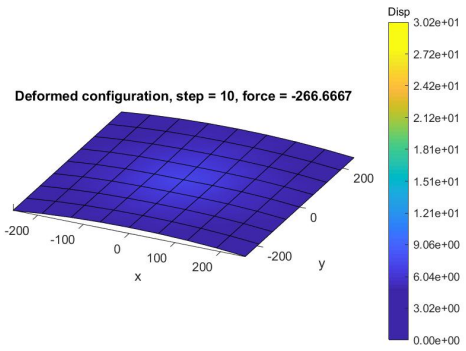

Deformed configuration, step = 5, force = -133.3333
Disp
3.02e+01
2.72e+01
2.42e+01
2.12e+01
1.81e+01
1.51e+01
1.21e+01
9.06e+00
6.04e+00
3.02e+00
0.00e+00
-200
-100
0
100
200
x
y

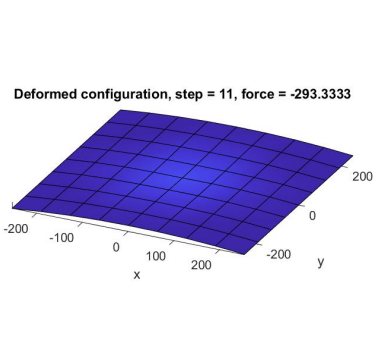

Deformed configuration, step = 6, force = -160
Disp
3.02e+01
2.72e+01
2.42e+01
2.12e+01
1.81e+01
1.51e+01
1.21e+01
9.06e+00
6.04e+00
3.02e+00
0.00e+00
-200
-100
0
100
200
x
y

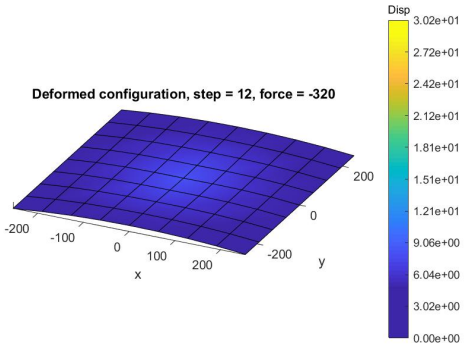

Deformed configuration, step = 7, force = -186.6667
Disp
3.02e+01
2.72e+01
2.42e+01
2.12e+01
1.81e+01
1.51e+01
1.21e+01
9.06e+00
6.04e+00
3.02e+00
0.00e+00
-200
-100
0
100
200
x
y

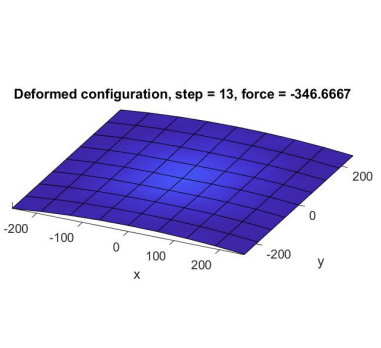

Deformed configuration, step = 8, force = -213.3333
Disp
3.02e+01
2.72e+01
2.42e+01
2.12e+01
1.81e+01
1.51e+01
1.21e+01
9.06e+00
6.04e+00
3.02e+00
0.00e+00
-200
-100
0
100
200
x
y

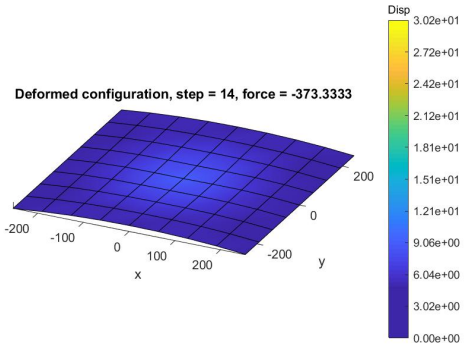

Deformed configuration, step = 9, force = -240
Disp
3.02e+01
2.72e+01
2.42e+01
2.12e+01
1.81e+01
1.51e+01
1.21e+01
9.06e+00
6.04e+00
3.02e+00
0.00e+00
-200
-100
0
100
200
x
y

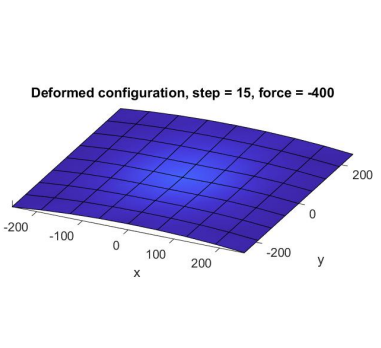

Deformed configuration, step = 10, force = -266.6667
Disp
3.02e+01
2.72e+01
2.42e+01
2.12e+01
1.81e+01
1.51e+01
1.21e+01
9.06e+00
6.04e+00
3.02e+00
0.00e+00
-200
-100
0
100
200
x
y

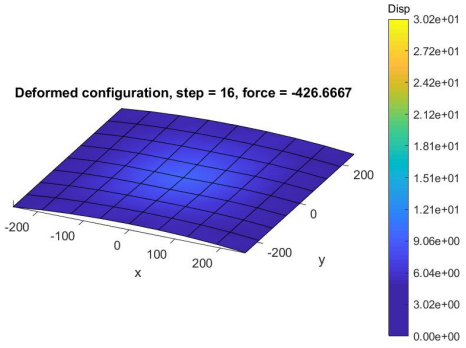

Deformed configuration, step = 11, force = -293.3333
Disp
3.02e+01
2.72e+01
2.42e+01
2.12e+01
1.81e+01
1.51e+01
1.21e+01
9.06e+00
6.04e+00
3.02e+00
0.00e+00
-200
-100
0
100
200
x
y

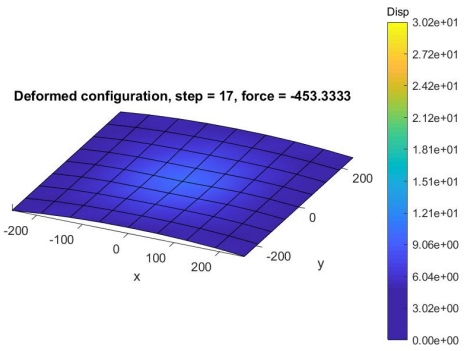

Deformed configuration, step = 12, force = -320
Disp
3.02e+01
2.72e+01
2.42e+01
2.12e+01
1.81e+01
1.51e+01
1.21e+01
9.06e+00
6.04e+00
3.02e+00
0.00e+00
-200
-100
0
100
200
x
y

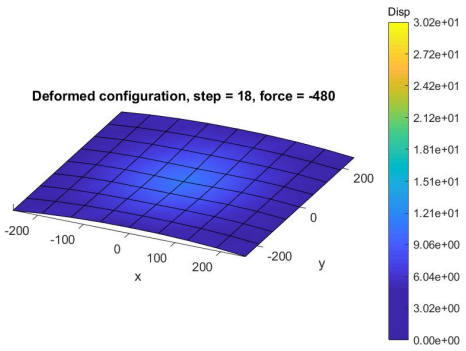

Deformed configuration, step = 13, force = -346.6667
Disp
3.02e+01
2.72e+01
2.42e+01
2.12e+01
1.81e+01
1.51e+01
1.21e+01
9.06e+00
6.04e+00
3.02e+00
0.00e+00
-200
-100
0
100
200
x
y

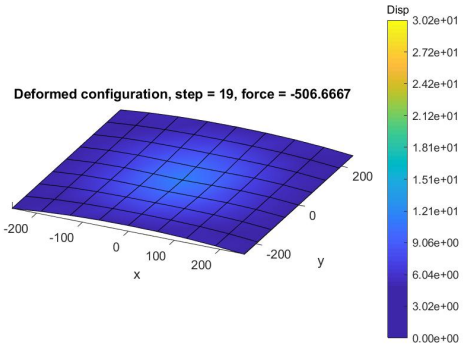

Deformed configuration, step = 14, force = -373.3333
Disp
3.02e+01
2.72e+01
2.42e+01
2.12e+01
1.81e+01
1.51e+01
1.21e+01
9.06e+00
6.04e+00
3.02e+00
0.00e+00
-200
-100
0
100
200
x
y

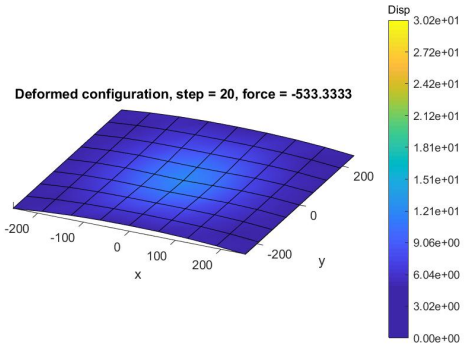

Deformed configuration, step = 15, force = -400
Disp
3.02e+01
2.72e+01
2.42e+01
2.12e+01
1.81e+01
1.51e+01
1.21e+01
9.06e+00
6.04e+00
3.02e+00
0.00e+00
-200
-100
0
100
200
x
y

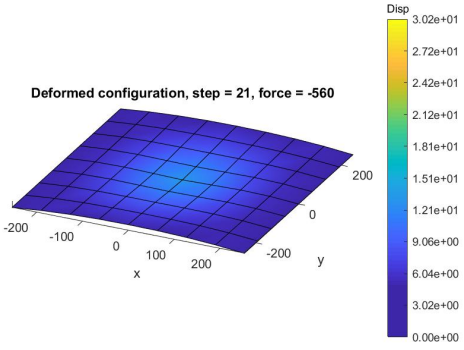

Deformed configuration, step = 16, force = -426.6667
Disp
3.02e+01
2.72e+01
2.42e+01
2.12e+01
1.81e+01
1.51e+01
1.21e+01
9.06e+00
6.04e+00
3.02e+00
0.00e+00
-200
-100
0
100
200
x
y

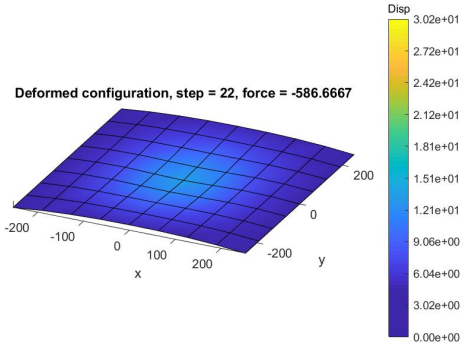

Deformed configuration, step = 17, force = -453.3333
Disp
3.02e+01
2.72e+01
2.42e+01
2.12e+01
1.81e+01
1.51e+01
1.21e+01
9.06e+00
6.04e+00
3.02e+00
0.00e+00
x
y
-200
-100
0
100
200

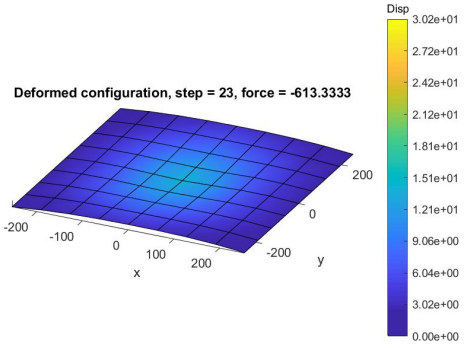

Deformed configuration, step = 18, force = -480
Disp
3.02e+01
2.72e+01
2.42e+01
2.12e+01
1.81e+01
1.51e+01
1.21e+01
9.06e+00
6.04e+00
3.02e+00
0.00e+00
x
y
-200
-100
0
100
200

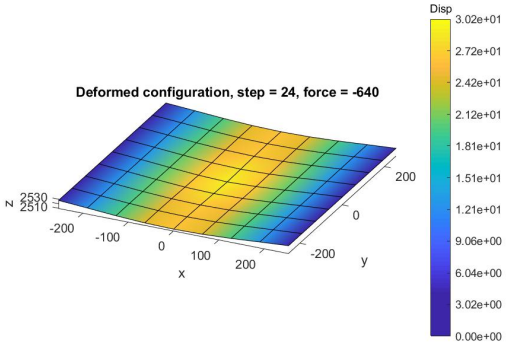

Deformed configuration, step = 19, force = -506.6667
Disp
3.02e+01
2.72e+01
2.42e+01
2.12e+01
1.81e+01
1.51e+01
1.21e+01
9.06e+00
6.04e+00
3.02e+00
0.00e+00
x
y
-200
-100
0
100
200

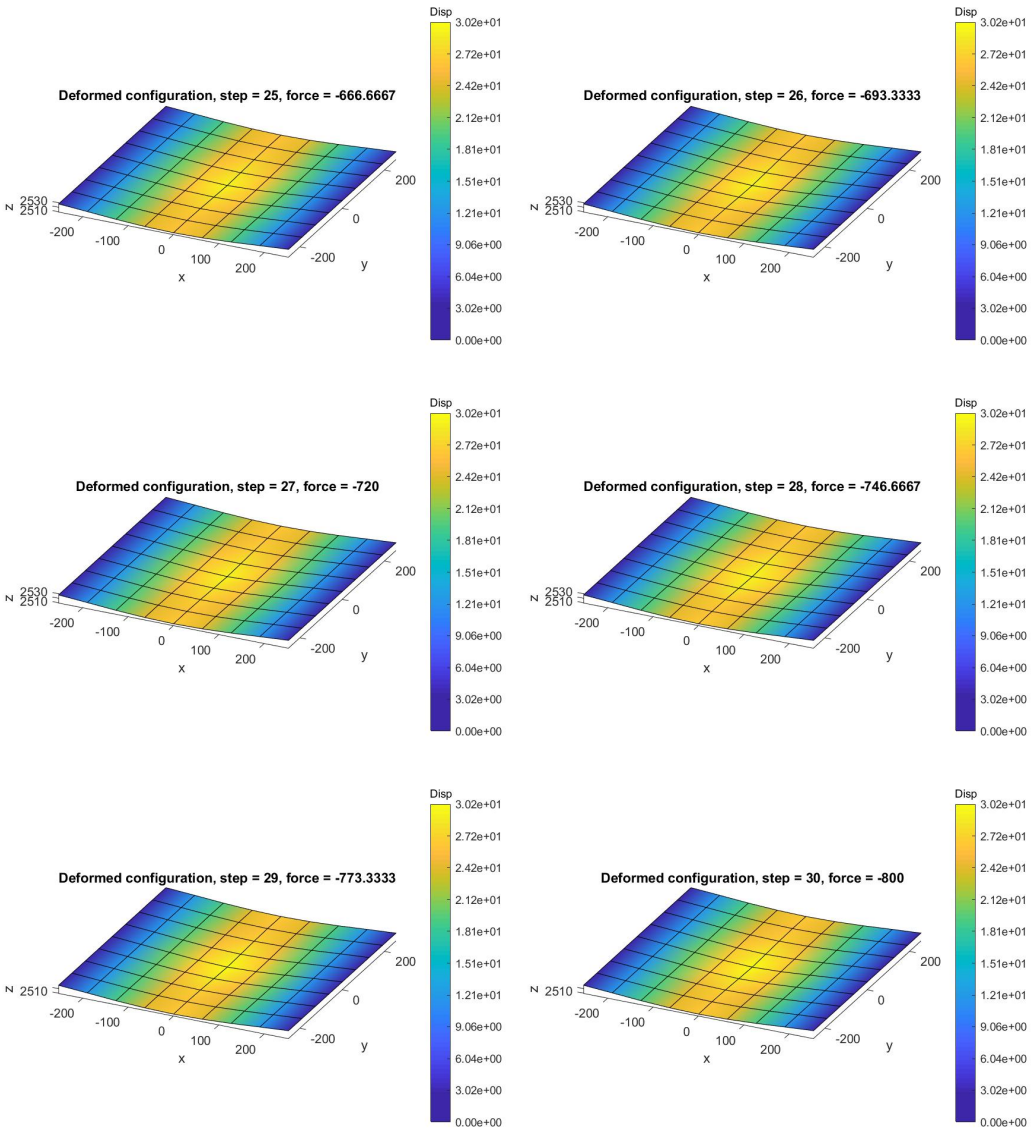

Deformed configuration, step = 20, force = -533.3333
Disp
3.02e+01
2.72e+01
2.42e+01
2.12e+01
1.81e+01
1.51e+01
1.21e+01
9.06e+00
6.04e+00
3.02e+00
0.00e+00
x
y
-200
-100
0
100
200


Deformed configuration, step = 21, force = -560
Disp
3.02e+01
2.72e+01
2.42e+01
2.12e+01
1.81e+01
1.51e+01
1.21e+01
9.06e+00
6.04e+00
3.02e+00
0.00e+00
x
y
-200
-100
0
100
200


Deformed configuration, step = 22, force = -586.6667
Disp
3.02e+01
2.72e+01
2.42e+01
2.12e+01
1.81e+01
1.51e+01
1.21e+01
9.06e+00
6.04e+00
3.02e+00
0.00e+00
x
y
-200
-100
0
100
200


Deformed configuration, step = 23, force = -613.3333
Disp
3.02e+01
2.72e+01
2.42e+01
2.12e+01
1.81e+01
1.51e+01
1.21e+01
9.06e+00
6.04e+00
3.02e+00
0.00e+00
x
y
-200
-100
0
100
200


Deformed configuration, step = 24, force = -640
Disp
3.02e+01
2.72e+01
2.42e+01
2.12e+01
1.81e+01
1.51e+01
1.21e+01
9.06e+00
6.04e+00
3.02e+00
0.00e+00
z
2530
2510
x
y
-200
-100
0
100
200



In figura 5.11 sono riportati i valori della forza esterna e dello spostamento verticale nel nodo centrale ad ogni step di carico, confrontati con l'intero percorso di carico riportato in [12].

Osservando le figure precedenti, si nota che fra lo step numero 23 e lo step numero 24 la configurazione subisce uno spostamento improvviso caratterizzato dal cambio di concavità della superficie cilindrica.

Osservando la figura 5.11, infatti, ci si accorge che, incrementando il carico a partire dallo step numero 23, non è possibile considerare tutto il percorso tensionale della struttura, definito dalla linea tratteggiata.

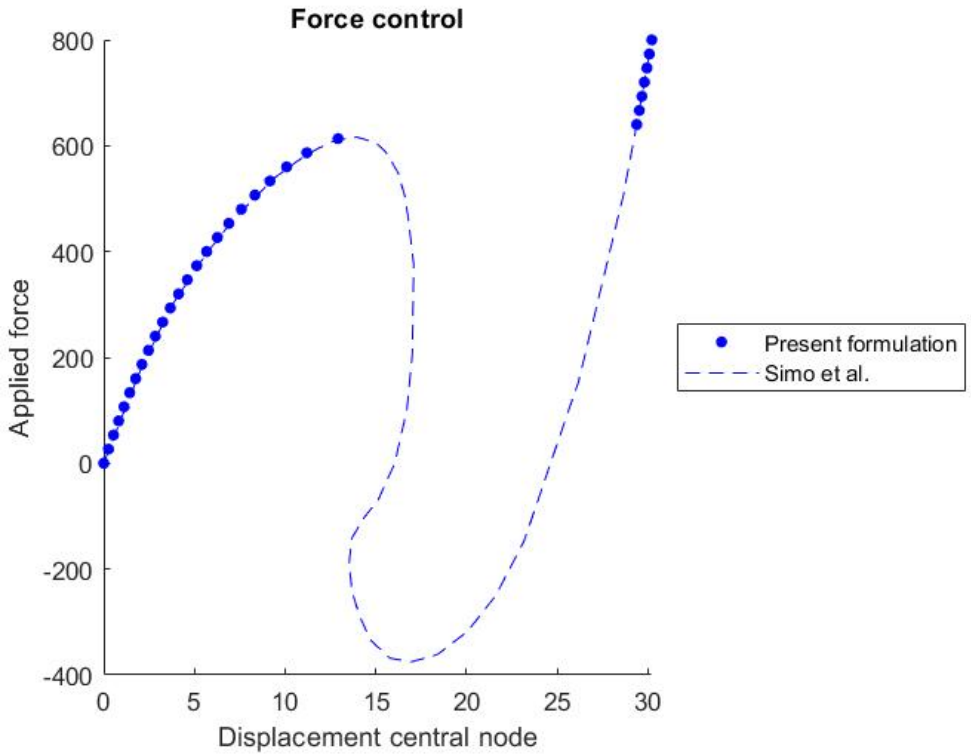


Figura 5.11

5.3.1 Script di input

Si riporta lo script relativo al file di input.

```
% Input
%-------------------------------------------------------------------------------------------%
ElementsTheta = 4;
ElementsLength = 4;
Theta = 0.1;
R = 2540;
Length = 2*254;
%-------------------------------------------------------------------------------------------%
TotNodesElement = 4;
TotGaussPointsMembraneAndBending = 4;
TotGaussPointsShear = 1;
%-------------------------------------------------------------------------------------------%
ElasticModule = 3102.75;
PoissonCoefficient = 0.3;
%-------------------------------------------------------------------------------------------%
Height = 6.35;
ShearFactor = 5/6;
%-------------------------------------------------------------------------------------------%
TotDofSurfaceNode = 3;
TotDofDirectorNode = 2;
%-------------------------------------------------------------------------------------------%
```

```
TotSpaceDimensions = 3;
ExternalForce = -800;
%-------------------------------------------------------------------------------------------%
Tollerance = 10^-6;
MaxIterations = 100;
LoadSteps = 30;
%-------------------------------------------------------------------------------------------%
% Generate Problem
%-------------------------------------------------------------------------------------------%
NodesTheta = 2*ElementsTheta+1;
NodesLength = 2*ElementsLength+1;
Dtheta = Theta/ElementsTheta;
DL = Length/2/ElementsLength;
CentralNode = NodesTheta*(NodesLength-1)/2+(NodesTheta-1)/2+1;
I = eye(3); e3 = [0 0 1]';
TotNodes = NodesTheta*NodesLength;
TotElements = 4*ElementsTheta*ElementsLength;
%-------------------------------------------------------------------------------------------%
TotDofNode = TotDofSurfaceNode+TotDofDirectorNode;
TotDofSurface = TotDofSurfaceNode*TotNodes;
TotDofDirector = TotDofDirectorNode*TotNodes;
TotDof = TotDofSurface+TotDofDirector;
TotDofSurfaceElement = TotDofSurfaceNode*TotNodesElement;
TotDofDirectorElement = TotDofDirectorNode*TotNodesElement;
TotDofElement = TotDofSurfaceElement+TotDofDirectorElement;
%-------------------------------------------------------------------------------------------%
x0Nodes = zeros(TotNodes,3);
t0Nodes = zeros(TotNodes,3);
R0Nodes = zeros(3*TotNodes,3);
Counter = 0;
for i = 1:NodesLength
    y = -Length/2+(i-1)*DL;
    for j = 1:NodesTheta
        Counter = Counter+1;
        theta = -Theta+(j-1)*Dtheta;
        x0 = [R*sin(theta) y R*cos(theta)];
        x0Nodes(Counter,:) = x0;
        x0_theta = [R*cos(theta) 0 -R*sin(theta)];
        x0_y = [0 1 0];
        t0 = cross(x0_theta,x0_y);
        t0Normalized = t0/norm(t0,2);
        t0Nodes(Counter,:) = t0Normalized;
        crosse3t0Normalized = cross(e3,t0Normalized');
        crosse3t0NormalizedTens = [0 -crosse3t0Normalized(3) crosse3t0Normalized(2);...
            crosse3t0Normalized(3) 0 -crosse3t0Normalized(1);...
            -crosse3t0Normalized(2) crosse3t0Normalized(1) 0];
        R0 = (e3'*t0Normalized')*I+crosse3t0NormalizedTens+ ...
            (crosse3t0Normalized*crosse3t0Normalized')/(1+(e3'*t0Normalized'));
        R0Nodes(3*(Counter-1)+1:3*Counter,:) = R0;
    end
end
%-------------------------------------------------------------------------------------------%
Connectivity = zeros(TotElements,4);
Counter = 0;
for i = 1:NodesLength
    FirstNode1 = i*NodesTheta+2;
    FirstNode2 = i*NodesTheta+1;
    FirstNode3 = (i-1)*NodesTheta+1;
    FirstNode4 = (i-1)*NodesTheta+2;
    for j = 1:2*ElementsTheta
        Counter = Counter+1;
        Node1 = FirstNode1+j-1;
```

```
        Node2 = FirstNode2+j-1;
        Node3 = FirstNode3+j-1;
        Node4 = FirstNode4+j-1;
        Connectivity(Counter,:) = [Node1 Node2 Node3 Node4];
    end
end
%-------------------------------------------------------------------------------------%
t0GaussPointsMembraneAndBending = zeros(3*TotElements,TotGaussPointsMembraneAndBending);
t0_1GaussPointsMembraneAndBending = zeros(TotElements*3,TotGaussPointsMembraneAndBending);
t0_2GaussPointsMembraneAndBending = zeros(TotElements*3,TotGaussPointsMembraneAndBending);
for ElementNumber = 1:TotElements
    Node1 = Connectivity(ElementNumber,1); Node2 = Connectivity(ElementNumber,2);
    Node3 = Connectivity(ElementNumber,3); Node4 = Connectivity(ElementNumber,4);
    t01 = t0Nodes(Node1,:)'; t02 = t0Nodes(Node2,:)';
    t03 = t0Nodes(Node3,:)'; t04 = t0Nodes(Node4,:)';
    for GPNumber = 1:TotGaussPointsMembraneAndBending
        [GP,~] = GaussPoint(GPNumber,TotGaussPointsMembraneAndBending);
        N1 = ShapeFunction(1,GP); N2 = ShapeFunction(2,GP);
        N3 = ShapeFunction(3,GP); N4 = ShapeFunction(4,GP);
        N1_1 = DerShapeFunction(1,1,GP); N1_2 = DerShapeFunction(1,2,GP);
        N2_1 = DerShapeFunction(2,1,GP); N2_2 = DerShapeFunction(2,2,GP);
        N3_1 = DerShapeFunction(3,1,GP); N3_2 = DerShapeFunction(3,2,GP);
        N4_1 = DerShapeFunction(4,1,GP); N4_2 = DerShapeFunction(4,2,GP);
        t0 = t01*N1+t02*N2+t03*N3+t04*N4;
        t0_1 = t01*N1_1+t02*N2_1+t03*N3_1+t04*N4_1;
        t0_2 = t01*N1_2+t02*N2_2+t03*N3_2+t04*N4_2;
        t0Normalized = t0/norm(t0,2);
        t0_1Corrected = (t0_1-(t0Normalized'*t0_1)*t0Normalized)/norm(t0,2);
        t0_2Corrected = (t0_2-(t0Normalized'*t0_2)*t0Normalized)/norm(t0,2);
        t0GaussPointsMembraneAndBending ...
            (3*(ElementNumber-1)+1:3*ElementNumber,GPNumber) = t0Normalized;
        t0_1GaussPointsMembraneAndBending ...
            (3*(ElementNumber-1)+1:3*ElementNumber,GPNumber) = t0_1Corrected;
        t0_2GaussPointsMembraneAndBending ...
            (3*(ElementNumber-1)+1:3*ElementNumber,GPNumber) = t0_2Corrected;
    end
end
t0GaussPointsShear = zeros(3*TotElements,TotGaussPointsShear);
for ElementNumber = 1:TotElements
    Node1 = Connectivity(ElementNumber,1); Node2 = Connectivity(ElementNumber,2);
    Node3 = Connectivity(ElementNumber,3); Node4 = Connectivity(ElementNumber,4);
    t01 = t0Nodes(Node1,:)'; t02 = t0Nodes(Node2,:)';
    t03 = t0Nodes(Node3,:)'; t04 = t0Nodes(Node4,:)';
    for GPNumber = 1:TotGaussPointsShear
        [GP,~] = GaussPoint(GPNumber,TotGaussPointsShear);
        N1 = ShapeFunction(1,GP); N2 = ShapeFunction(2,GP);
        N3 = ShapeFunction(3,GP); N4 = ShapeFunction(4,GP);
        t0 = t01*N1+t02*N2+t03*N3+t04*N4;
        t0Normalized = t0/norm(t0,2);
        t0GaussPointsShear(3*(ElementNumber-1)+1:3*ElementNumber,GPNumber) = t0Normalized;
    end
end
%-------------------------------------------------------------------------------------%
xNodes = x0Nodes;
tNodes = t0Nodes;
tGaussPointsMembraneAndBending = t0GaussPointsMembraneAndBending;
t_1GaussPointsMembraneAndBending = t0_1GaussPointsMembraneAndBending;
t_2GaussPointsMembraneAndBending = t0_2GaussPointsMembraneAndBending;
tGaussPointsShear = t0GaussPointsShear;
RNodes = R0Nodes;
%-------------------------------------------------------------------------------------%
NodesAlongLeftHinge = zeros(NodesLength,1);
```

```
NodesAlongRightHinge = zeros(NodesLength,1);
for i = 1:NodesLength
    NodesAlongLeftHinge(i,1) = (i-1)*NodesTheta+1;
    NodesAlongRightHinge(i,1) = (i-1)*NodesTheta+NodesTheta;
end
BoundaryConditions = zeros(TotDof,1);
for i = 1:NodesLength
    Node1 = NodesAlongLeftHinge(i,1);
    Node2 = NodesAlongRightHinge(i,1);
    BoundaryConditions(3*Node1-2,1) = 1;
    BoundaryConditions(3*Node1-1,1) = 1;
    BoundaryConditions(3*Node1,1) = 1;
    BoundaryConditions(3*Node2-2,1) = 1;
    BoundaryConditions(3*Node2-1,1) = 1;
    BoundaryConditions(3*Node2,1) = 1;
end
SizeFreeDof = 0;
for i = 1:TotDof
    if BoundaryConditions(i,1) == 1
        SizeFreeDof = SizeFreeDof+1;
    end
end
FreeDof = zeros(SizeFreeDof,1);
Counter = 0;
for i = 1:TotDof
    if BoundaryConditions(i,1) == 0
        Counter = Counter+1;
        FreeDof(Counter,1) = i;
    end
end
%-------------------------------------------------------------------------------------------%
ExternalForces = zeros(TotDof,1);
ExternalForces(3*CentralNode,1) = ExternalForce;
%-------------------------------------------------------------------------------------------%
Figures = gobjects(LoadSteps+1,1);
Movie = struct('cdata',cell(LoadSteps+1,1),'colormap',cell(LoadSteps+1,1));
%-------------------------------------------------------------------------------------------%
SavexNodesx = zeros(TotNodes,LoadSteps+1);
SavexNodesx(:,1) = x0Nodes(:,1);
SavexNodesy = zeros(TotNodes,LoadSteps+1);
SavexNodesy(:,1) = x0Nodes(:,2);
SavexNodesz = zeros(TotNodes,LoadSteps+1);
SavexNodesz(:,1) = x0Nodes(:,3);
%-------------------------------------------------------------------------------------------%
NodePlot = CentralNode;
ForcePlot = zeros(LoadSteps+1,1);
DisplacementPlot = zeros(LoadSteps+1,1);
```

5.3.2 Displacement control

Si presenta ora un'analisi definita in *controllo di spostamento* (*displacement control*), per la quale si impone che lo spostamento del nodo centrale ad ogni step sia pari ad un incremento noto $\Delta\varphi$, ipotizzato costante. A tal fine, essendo che in questo caso il valore del vettore delle forze esterne $[\mathbf{F}_{\text{ext}}]$ corrispondente allo spostamento ipotizzato risulta un'incognita, esso viene riscritto come il prodotto fra una quantità costante (e nota) $[\bar{\mathbf{F}}_{\text{ext}}]$ ed uno

scalare $\lambda \in \mathbb{R}$. In questo modo, avendo aggiunto una variabile al sistema vettoriale (3.56), è necessaria un'ulteriore equazione:

$$f(\mathbf{\Phi}) = \varphi_I^3 - \varphi_I^{3[s-1]} - \Delta\varphi = 0 \tag{5.1}$$

corrispondente alla congruenza dello spostamento per il nodo centrale I. Con l'apice $[s-1]$ si indicano le grandezze corrispondenti allo step di carico precedente $s-1$. Ad ogni step, il sistema da risolvere risulta, dunque, caratterizzato dalla (3.56), alla quale viene aggiunta la precedente:

$$\begin{cases} [\mathbf{R}(\mathbf{\Phi}, \lambda)] = [\mathbf{F}_{\text{int}}] - \lambda[\bar{\mathbf{F}}_{\text{ext}}] = [\mathbf{0}] \\ f(\mathbf{\Phi}) = \varphi_I^3 - \varphi_I^{3[s-1]} - \Delta\varphi = 0 \end{cases} \tag{5.2}$$

Si nota che, in questo caso, il vettore residuo $[\mathbf{R}]$ è in funzione anche del parametro λ.

Volendo risolvere il sistema (5.2) seguendo lo schema di Newton-Raphson, ad ogni step le due equazioni della (5.2) vengono sviluppate a partire da una configurazione $(\mathbf{\Phi}^{(k)}, \lambda^{(k)})$. Lo sviluppo della prima equazione al primo ordine risulta:

$$\begin{aligned} &[\mathbf{R}(\mathbf{\Phi}^{(k+1)}, \lambda^{(k+1)})] = [\mathbf{R}(\mathbf{\Phi}^{(k)} + \Delta\mathbf{\Phi}^{(k)}, \lambda^{(k)} + \Delta\lambda^{(k)})] \approx \\ &\quad \approx [\mathbf{R}(\mathbf{\Phi}^{(k)}, \lambda^{(k)})] + \frac{\partial[\mathbf{R}]}{\partial\mathbf{\Phi}}\Big|_{(\mathbf{\Phi}^{(k)}, \lambda^{(k)})}[\Delta\mathbf{\Phi}^{(k)}] + \frac{\partial[\mathbf{R}]}{\partial\lambda}\Big|_{(\mathbf{\Phi}^{(k)}, \lambda^{(k)})}\Delta\lambda^{(k)} = \\ &\quad = [\mathbf{R}(\mathbf{\Phi}^{(k)}, \lambda^{(k)})] + K^{(k)}[\Delta\mathbf{\Phi}^{(k)}] - \Delta\lambda^{(k)}[\bar{\mathbf{F}}_{\text{ext}}] = [\mathbf{0}] \end{aligned} \tag{5.3}$$

Lo sviluppo della seconda, invece:

$$\begin{aligned} &f(\mathbf{\Phi}^{(k+1)} + \Delta\mathbf{\Phi}^{(k+1)}) = f(\mathbf{\Phi}^{(k)} + \Delta\mathbf{\Phi}^{(k)}) \approx \\ &\quad \approx f(\mathbf{\Phi}^{(k)}) + \frac{\partial f}{\partial\mathbf{\Phi}} \cdot \Delta\mathbf{\Phi}^{(k)} = f(\mathbf{\Phi}^{(k)}) + [\mathbf{r}]^T[\Delta\mathbf{\Phi}^{(k)}] = 0 \end{aligned} \tag{5.4}$$

avendo introdotto il vettore $[\mathbf{r}]$, di dimensioni $5\mathrm{N_N} \times 1$, contenente tutti zeri eccetto il valore pari ad 1 nella posizione corrispondente al grado di libertà relativo all'abbassamento del nodo centrale, seguendo la numerazione della mesh. Lo sviluppo arrestato al primo ordine del sistema (5.2) risulta, dunque:

$$\begin{bmatrix} [\mathbf{R}^{(k+1)}] \\ f^{(k+1)} \end{bmatrix} = \begin{bmatrix} [\mathbf{R}^{(k)}] \\ f^{(k)} \end{bmatrix} + \begin{bmatrix} K^{(k)} & -[\bar{\mathbf{F}}_{\text{ext}}] \\ [\mathbf{r}]^T & 0 \end{bmatrix} \begin{bmatrix} [\Delta\mathbf{\Phi}^{(k)}] \\ \Delta\lambda^{(k)} \end{bmatrix} = \begin{bmatrix} [\mathbf{0}] \\ 0 \end{bmatrix} \tag{5.5}$$

Invertendo la relazione precedente, si ottengono gli incrementi $[\Delta\mathbf{\Phi}^{(k)}]$ e $\Delta\lambda^{(k)}$ da attribuire alla configurazione, secondo la procedura descritta della (3.3), e al parametro λ, secondo $\lambda^{(k+1)} = \lambda^{(k)} + \Delta\lambda^{(k)}$. Iterando fino a convergenza, si ottiene la configurazione corrispondente all'incremento di spostamento $\Delta\varphi$.

Il procedimento è riassunto nell'algoritmo seguente, il numero di incrementi di spostamento considerati è indicato dal numero DisplacementStep.

input: $[\bar{\mathbf{F}}_{\text{ext}}]$; Toll; MaxIter; DisplacementSteps; $\Delta\varphi$;
set: $\lambda^{(k)} = 0$;
for every DisplacementStep **do**
  set: $k = 0$; Err $= +\infty$;
  get: $\varphi_I^{3[s-1]}$;
  while Err $>$ Toll **and** $k <$ MaxIter **do**
    $k = k + 1$
    get: $K^{(k)}$; $[\mathbf{F}_{\text{int}}^{(k)}]$;
    $[\mathbf{R}^{(k)}] = [\mathbf{F}_{\text{int}}^{(k)}] - \lambda^{(k)}[\bar{\mathbf{F}}_{\text{ext}}]$;
    $f^{(k)} = \varphi_I^{3(k)} - \varphi_I^{3[s-1]} - \Delta\varphi$;
    Err $= \| \begin{bmatrix} [\mathbf{R}^{(k)}] & f^{(k)} \end{bmatrix} \|$;
    $$\begin{bmatrix} [\Delta\mathbf{\Phi}^{(k)}] \\ \Delta\lambda^{(k)} \end{bmatrix} = \begin{bmatrix} K^{(k)} & -[\bar{\mathbf{F}}_{\text{ext}}] \\ [\mathbf{r}]^T & 0 \end{bmatrix}^{-1} \begin{bmatrix} [\mathbf{R}^{(k)}] \\ f^{(k)} \end{bmatrix};$$
    $[\mathbf{\Phi}^{(k+1)}] = \text{update}([\mathbf{\Phi}^{(k)}], [\Delta\mathbf{\Phi}^{(k)}])$;;
    $\lambda^{(k+1)} = \lambda^{(k)} + \Delta\lambda^{(k)}$;
  end while
end for

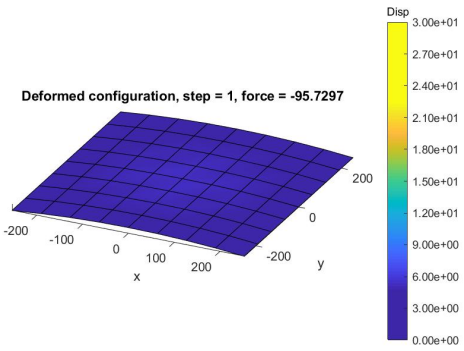
L'algoritmo è implementato nella function `NonLinearSolverDisplacementControl`, riportata di seguito.

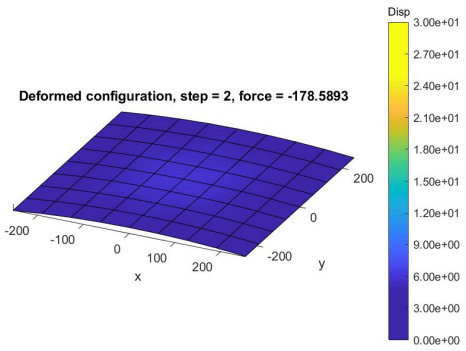
```
function NonLinearSolverDisplacementControl(Global)
%-------------------------------------------------------------------------------------------%
fExtTotal = Global.External.ExternalForces;
FreeDof = Global.External.FreeDof; FreeDofDC = [FreeDof;Global.Mesh.TotDof+1];
Du = zeros(Global.Mesh.TotDof+1,1);
%-------------------------------------------------------------------------------------------%
lambda = 0; DuDC = -1;
for Step = 1:Global.Solver.LoadSteps
Iteration = 0;
    xDC0 = Global.Configuration.xNodes(Global.Plot.NodePlot,3);
    fExt = lambda*fExtTotal;
    xDCNew = Global.Configuration.xNodes(Global.Plot.NodePlot,3); uDCNew = xDCNew-xDC0;
    %-------------------------------------------------------------------------------------------%
    Iteration = Iteration+1;
    fInt = zeros(Global.Mesh.TotDof,1); K = zeros(Global.Mesh.TotDof,Global.Mesh.TotDof);
    for ElementNumber = 1:Global.Mesh.TotElements
        KElement = StiffnessMatrix(Global,ElementNumber);
        fIntElement = InternalForces(Global,ElementNumber);
        K = AssembleStiffnessMatrix(K,KElement,Global,ElementNumber);
        fInt = AssembleInternalForces(fInt,fIntElement,Global,ElementNumber);
    end
    fIntDC = [fInt; uDCNew]; fExtDC = [fExt;DuDC];
    JnRow = zeros(1,Global.Mesh.TotDof); JnRow(1,3*Global.Plot.NodePlot) = 1;
    KDC = [K -fExtTotal;JnRow 0];
    ResDC = fIntDC-fExtDC; Err = norm(ResDC(FreeDofDC),2);
    Du(FreeDofDC,1) = -KDC(FreeDofDC,FreeDofDC)\ResDC(FreeDofDC);
```

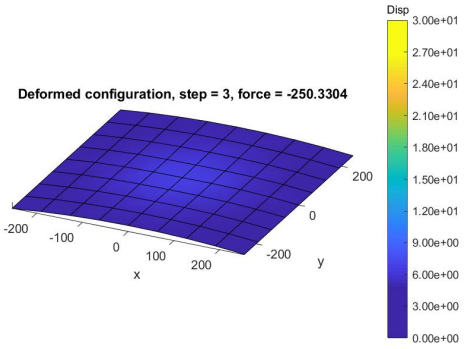
```
    DxNodes = Du(1:Global.Mesh.TotDofSurface);
    DTNodes = Du(Global.Mesh.TotDofSurface+1:Global.Mesh.TotDof,1);
    Dlambda = Du(Global.Mesh.TotDof+1);
    UpdateSurface(Global,DxNodes); UpdateDirector(Global,DTNodes);
    while Err > Global.Solver.Tollerance && Iteration < Global.Solver.MaxIterations
        Iteration = Iteration+1;
        fInt = zeros(Global.Mesh.TotDof,1);
        K = zeros(Global.Mesh.TotDof,Global.Mesh.TotDof);
        %-------------------------------------------------------------------------------%
        lambda = lambda+Dlambda;
        fExt = lambda*fExtTotal;
        %-------------------------------------------------------------------------------%
        for ElementNumber = 1:Global.Mesh.TotElements
            KElement = StiffnessMatrix(Global,ElementNumber);
            fIntElement = InternalForces(Global,ElementNumber);
            K = AssembleStiffnessMatrix(K,KElement,Global,ElementNumber);
            fInt = AssembleInternalForces(fInt,fIntElement,Global,ElementNumber);
        end
        xDCNew = Global.Configuration.xNodes(Global.Plot.NodePlot,3); uDCNew = xDCNew-xDC0;
        fIntDC = [fInt; uDCNew]; fExtDC = [fExt;DuDC];
        JnRow = zeros(1,Global.Mesh.TotDof); JnRow(1,3*Global.Plot.NodePlot) = 1;
        KDC = [K -fExtTotal;JnRow 0];
        ResDC = fIntDC-fExtDC; Err = norm(ResDC(FreeDofDC),2);
        Du(FreeDofDC,1) = -KDC(FreeDofDC,FreeDofDC)\ResDC(FreeDofDC);
        DxNodes = Du(1:Global.Mesh.TotDofSurface);
        DTNodes = Du(Global.Mesh.TotDofSurface+1:Global.Mesh.TotDof,1);
        Dlambda = Du(Global.Mesh.TotDof+1);
        UpdateSurface(Global,DxNodes); UpdateDirector(Global,DTNodes);
    end
    if Iteration >= 100
        error('Reached max number of iterations');
    end
    lambda = lambda+Dlambda;
    %-------------------------------------------------------------------------------%
    Global.Plot.SavexNodesx(:,Step+1) = Global.Configuration.xNodes(:,1);
    Global.Plot.SavexNodesy(:,Step+1) = Global.Configuration.xNodes(:,2);
    Global.Plot.SavexNodesz(:,Step+1) = Global.Configuration.xNodes(:,3);
    %-------------------------------------------------------------------------------%
    Global.Plot.ForcePlot(Step+1,1) = lambda*Global.External.ExternalForce;
    Global.Plot.DisplacementPlot(Step+1,1) = ...
        -(Global.Configuration.xNodes(Global.Plot.NodePlot,3) ...
        -Global.Configuration.x0Nodes(Global.Plot.NodePlot,3));
    %-------------------------------------------------------------------------------%
end
%-------------------------------------------------------------------------------%
end
```

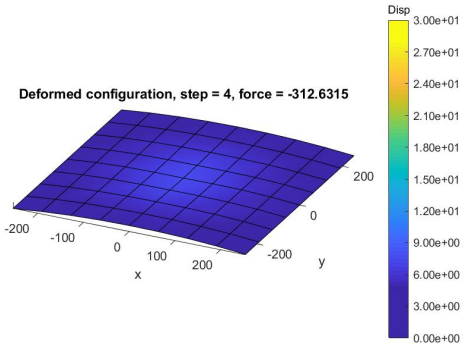
L'analisi è svolta considerando 30 step di incremento di spostamento uniformi, corripondenti a $\Delta\varphi = -1$.

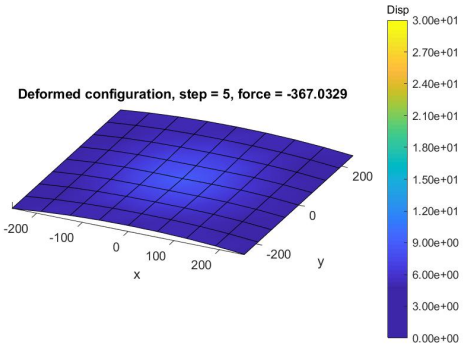
Nelle figure successive si riporta la configurazione deformata ad ogni step di carico.

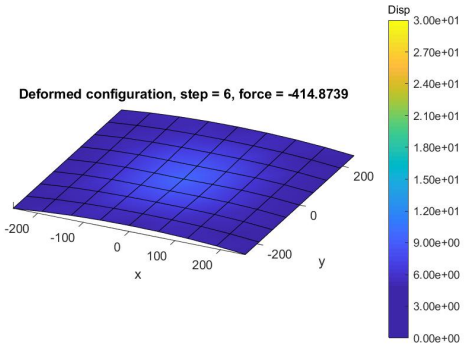

Deformed configuration, step = 1, force = -95.7297
Disp
3.00e+01
2.70e+01
2.40e+01
2.10e+01
1.80e+01
1.50e+01
1.20e+01
9.00e+00
6.00e+00
3.00e+00
0.00e+00
-200
-100
0
100
200
x
y

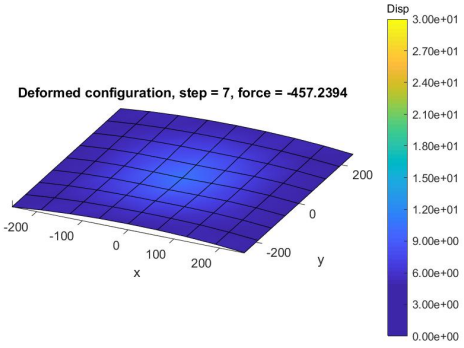

Deformed configuration, step = 2, force = -178.5893
Disp
3.00e+01
2.70e+01
2.40e+01
2.10e+01
1.80e+01
1.50e+01
1.20e+01
9.00e+00
6.00e+00
3.00e+00
0.00e+00
-200
-100
0
100
200
x
y

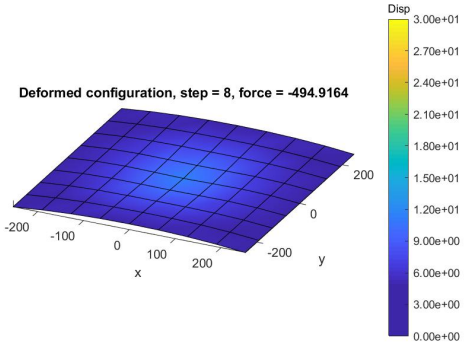

Deformed configuration, step = 3, force = -250.3304
Disp
3.00e+01
2.70e+01
2.40e+01
2.10e+01
1.80e+01
1.50e+01
1.20e+01
9.00e+00
6.00e+00
3.00e+00
0.00e+00
-200
-100
0
100
200
x
y

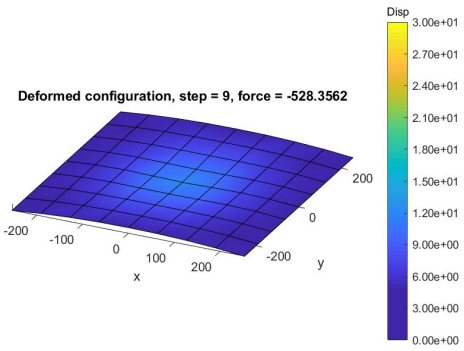

Deformed configuration, step = 4, force = -312.6315
Disp
3.00e+01
2.70e+01
2.40e+01
2.10e+01
1.80e+01
1.50e+01
1.20e+01
9.00e+00
6.00e+00
3.00e+00
0.00e+00
-200
-100
0
100
200
x
y

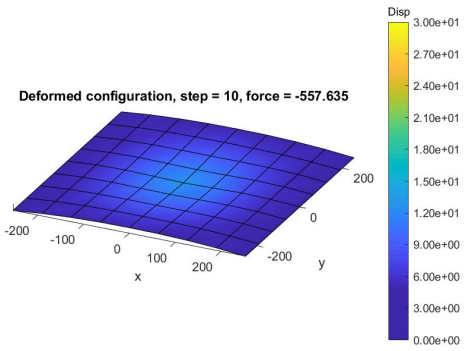

Deformed configuration, step = 5, force = -367.0329
Disp
3.00e+01
2.70e+01
2.40e+01
2.10e+01
1.80e+01
1.50e+01
1.20e+01
9.00e+00
6.00e+00
3.00e+00
0.00e+00
-200
-100
0
100
200
x
y

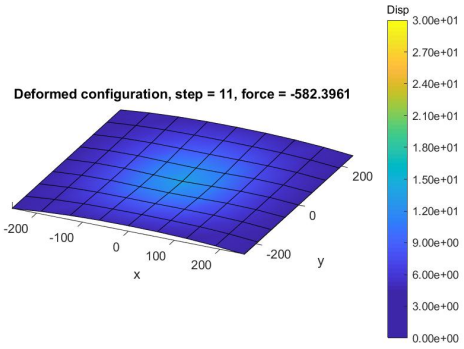

Deformed configuration, step = 6, force = -414.8739
Disp
3.00e+01
2.70e+01
2.40e+01
2.10e+01
1.80e+01
1.50e+01
1.20e+01
9.00e+00
6.00e+00
3.00e+00
0.00e+00
-200
-100
0
100
200
x
y

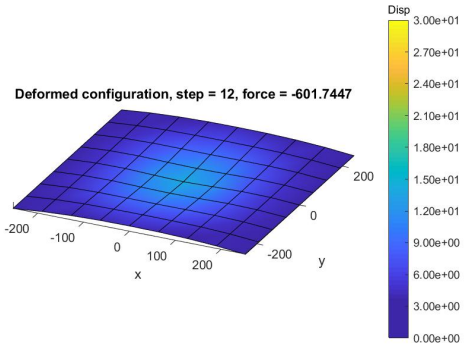

Deformed configuration, step = 7, force = -457.2394
Disp
3.00e+01
2.70e+01
2.40e+01
2.10e+01
1.80e+01
1.50e+01
1.20e+01
9.00e+00
6.00e+00
3.00e+00
0.00e+00
-200
-100
0
100
200
x
y

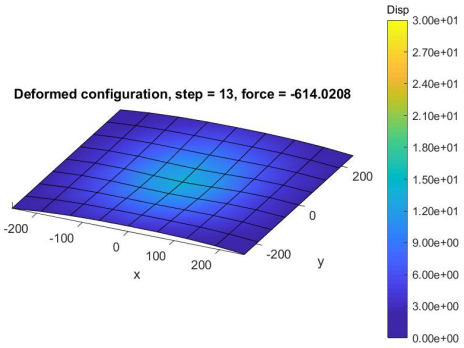

Deformed configuration, step = 8, force = -494.9164
Disp
3.00e+01
2.70e+01
2.40e+01
2.10e+01
1.80e+01
1.50e+01
1.20e+01
9.00e+00
6.00e+00
3.00e+00
0.00e+00
-200
-100
0
100
200
x
y

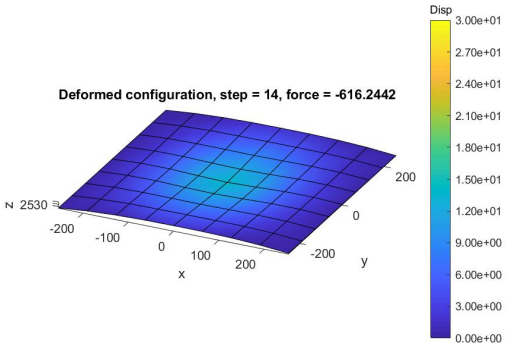

Deformed configuration, step = 9, force = -528.3562
Disp
3.00e+01
2.70e+01
2.40e+01
2.10e+01
1.80e+01
1.50e+01
1.20e+01
9.00e+00
6.00e+00
3.00e+00
0.00e+00
x
y
-200
-100
0
100
200

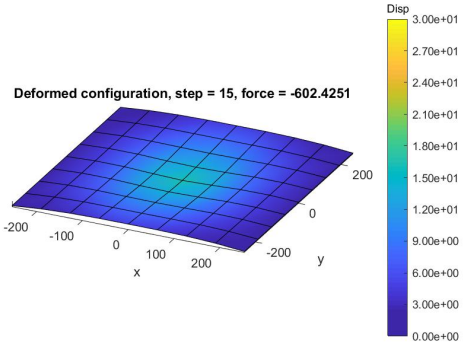

Deformed configuration, step = 10, force = -557.635
Disp
3.00e+01
2.70e+01
2.40e+01
2.10e+01
1.80e+01
1.50e+01
1.20e+01
9.00e+00
6.00e+00
3.00e+00
0.00e+00
x
y
-200
-100
0
100
200

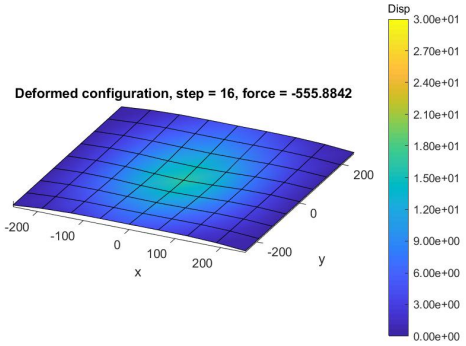

Deformed configuration, step = 11, force = -582.3961
Disp
3.00e+01
2.70e+01
2.40e+01
2.10e+01
1.80e+01
1.50e+01
1.20e+01
9.00e+00
6.00e+00
3.00e+00
0.00e+00
x
y
-200
-100
0
100
200

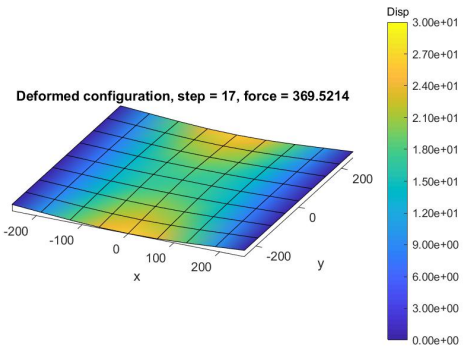

Deformed configuration, step = 12, force = -601.7447
Disp
3.00e+01
2.70e+01
2.40e+01
2.10e+01
1.80e+01
1.50e+01
1.20e+01
9.00e+00
6.00e+00
3.00e+00
0.00e+00
x
y
-200
-100
0
100
200

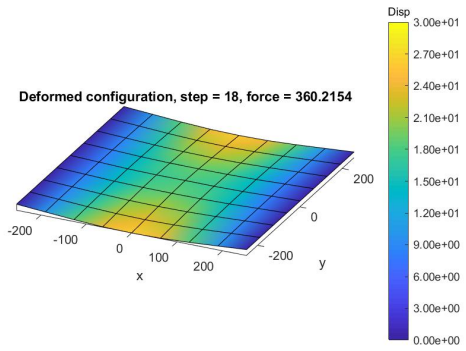

Deformed configuration, step = 13, force = -614.0208
Disp
3.00e+01
2.70e+01
2.40e+01
2.10e+01
1.80e+01
1.50e+01
1.20e+01
9.00e+00
6.00e+00
3.00e+00
0.00e+00
x
y
-200
-100
0
100
200

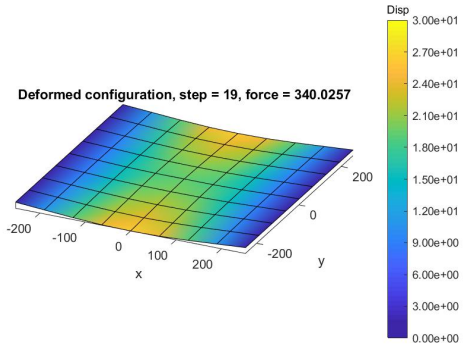

Deformed configuration, step = 14, force = -616.2442
Disp
3.00e+01
2.70e+01
2.40e+01
2.10e+01
1.80e+01
1.50e+01
1.20e+01
9.00e+00
6.00e+00
3.00e+00
0.00e+00
z
2530
x
y
-200
-100
0
100
200

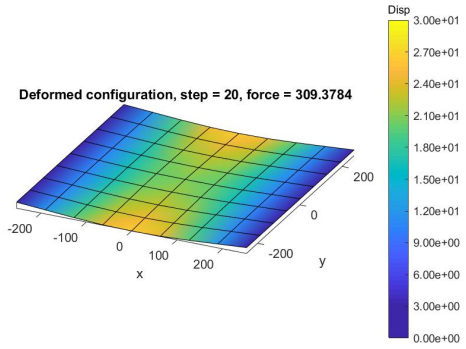

Deformed configuration, step = 15, force = -602.4251
Disp
3.00e+01
2.70e+01
2.40e+01
2.10e+01
1.80e+01
1.50e+01
1.20e+01
9.00e+00
6.00e+00
3.00e+00
0.00e+00
x
y
-200
-100
0
100
200

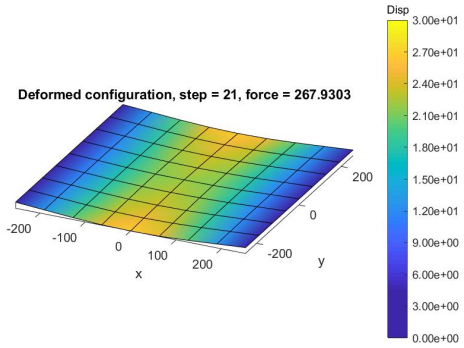

Deformed configuration, step = 16, force = -555.8842
Disp
3.00e+01
2.70e+01
2.40e+01
2.10e+01
1.80e+01
1.50e+01
1.20e+01
9.00e+00
6.00e+00
3.00e+00
0.00e+00
x
y
-200
-100
0
100
200

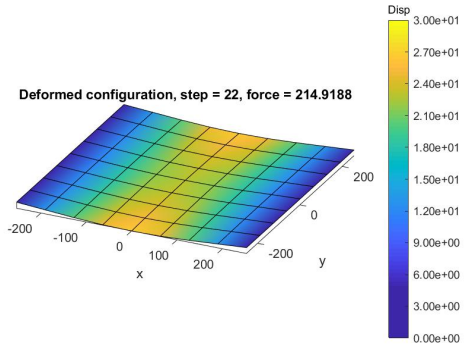

Deformed configuration, step = 17, force = 369.5214
Disp
3.00e+01
2.70e+01
2.40e+01
2.10e+01
1.80e+01
1.50e+01
1.20e+01
9.00e+00
6.00e+00
3.00e+00
0.00e+00
x
y
-200
-100
0
100
200

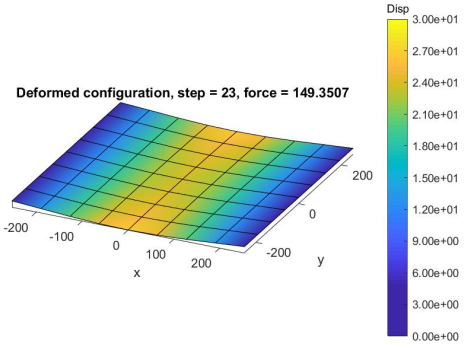

Deformed configuration, step = 18, force = 360.2154
Disp
3.00e+01
2.70e+01
2.40e+01
2.10e+01
1.80e+01
1.50e+01
1.20e+01
9.00e+00
6.00e+00
3.00e+00
0.00e+00
x
y
-200
-100
0
100
200

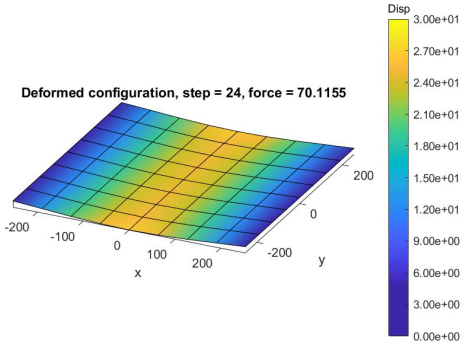

Deformed configuration, step = 19, force = 340.0257
Disp
3.00e+01
2.70e+01
2.40e+01
2.10e+01
1.80e+01
1.50e+01
1.20e+01
9.00e+00
6.00e+00
3.00e+00
0.00e+00
x
y
-200
-100
0
100
200

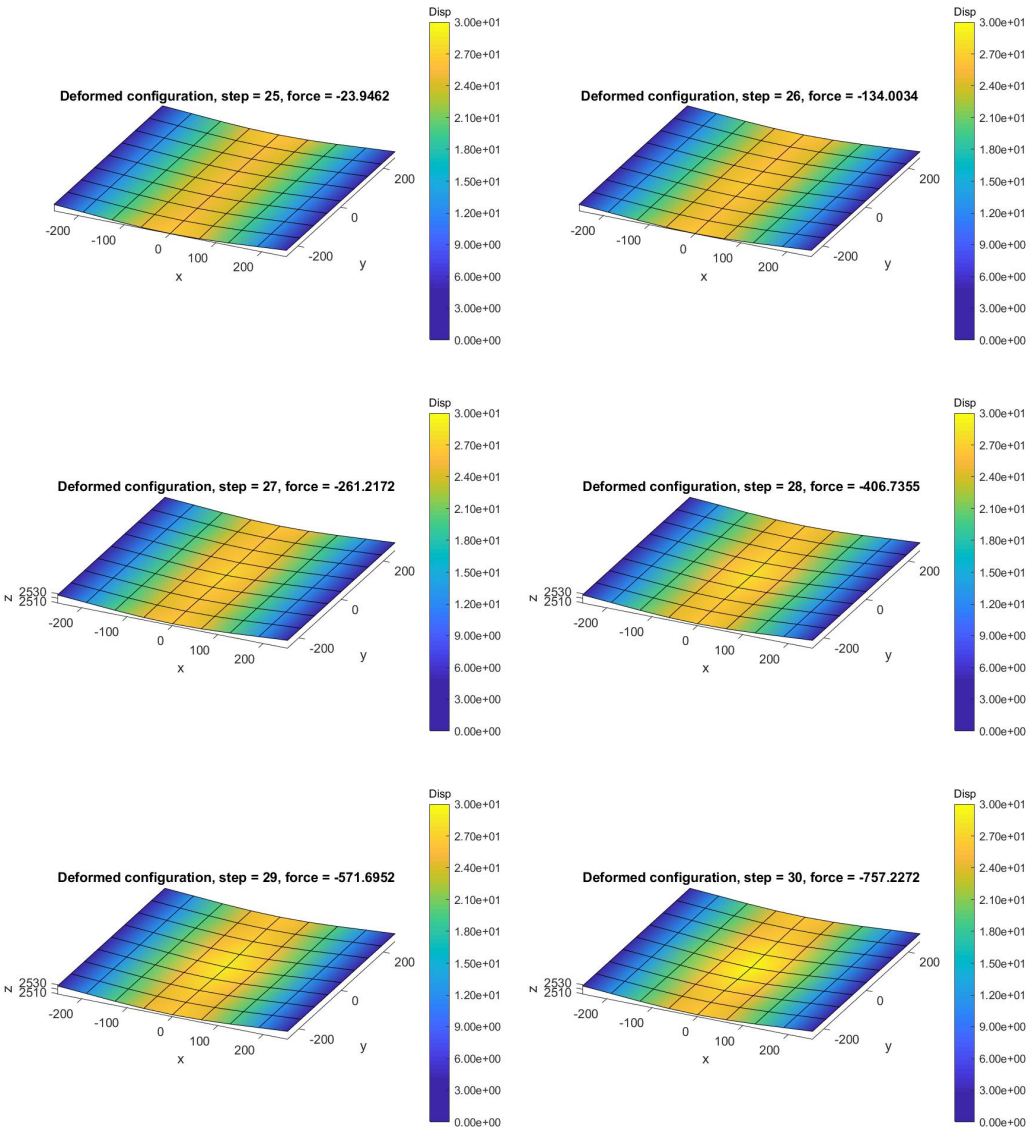

Deformed configuration, step = 20, force = 309.3784
Disp
3.00e+01
2.70e+01
2.40e+01
2.10e+01
1.80e+01
1.50e+01
1.20e+01
9.00e+00
6.00e+00
3.00e+00
0.00e+00
x
y
-200
-100
0
100
200


Deformed configuration, step = 21, force = 267.9303
Disp
3.00e+01
2.70e+01
2.40e+01
2.10e+01
1.80e+01
1.50e+01
1.20e+01
9.00e+00
6.00e+00
3.00e+00
0.00e+00
x
y
-200
-100
0
100
200


Deformed configuration, step = 22, force = 214.9188
Disp
3.00e+01
2.70e+01
2.40e+01
2.10e+01
1.80e+01
1.50e+01
1.20e+01
9.00e+00
6.00e+00
3.00e+00
0.00e+00
x
y
-200
-100
0
100
200


Deformed configuration, step = 23, force = 149.3507
Disp
3.00e+01
2.70e+01
2.40e+01
2.10e+01
1.80e+01
1.50e+01
1.20e+01
9.00e+00
6.00e+00
3.00e+00
0.00e+00
x
y
-200
-100
0
100
200


Deformed configuration, step = 24, force = 70.1155
Disp
3.00e+01
2.70e+01
2.40e+01
2.10e+01
1.80e+01
1.50e+01
1.20e+01
9.00e+00
6.00e+00
3.00e+00
0.00e+00
x
y
-200
-100
0
100
200



In figura 5.10 è riportata la configurazione iniziale. In figura 5.12 sono riportati i valori della forza esterna e dello spostamento verticale nel nodo centrale ad ogni step di carico, confrontati con l'intero percorso tensionale riportato in [12].

Si nota che, incrementando l'abbassamento del nodo centrale fra lo step numero 16 e lo step numero 17, la struttura subisce un cambio repentino di configurazione descritto dall'abbassamento delle due zone centrali opposte; analogamente, anche la forza esterna si riduce in modo discontinuo. Anche in questo caso, l'intero percorso tensionale della struttura non viene seguito.

Si riporta di seguito il `main` per richiamare l'algoritmo in displacement control.

```
% Fem Solver for nonlinear geometrically exact shell element
%-------------------------------------------------------------------------------------------%
Global = Generate('InputFile');
NonLinearSolverDisplacementControl(Global);
%-------------------------------------------------------------------------------------------%
PlotConfigurations(Global);
```

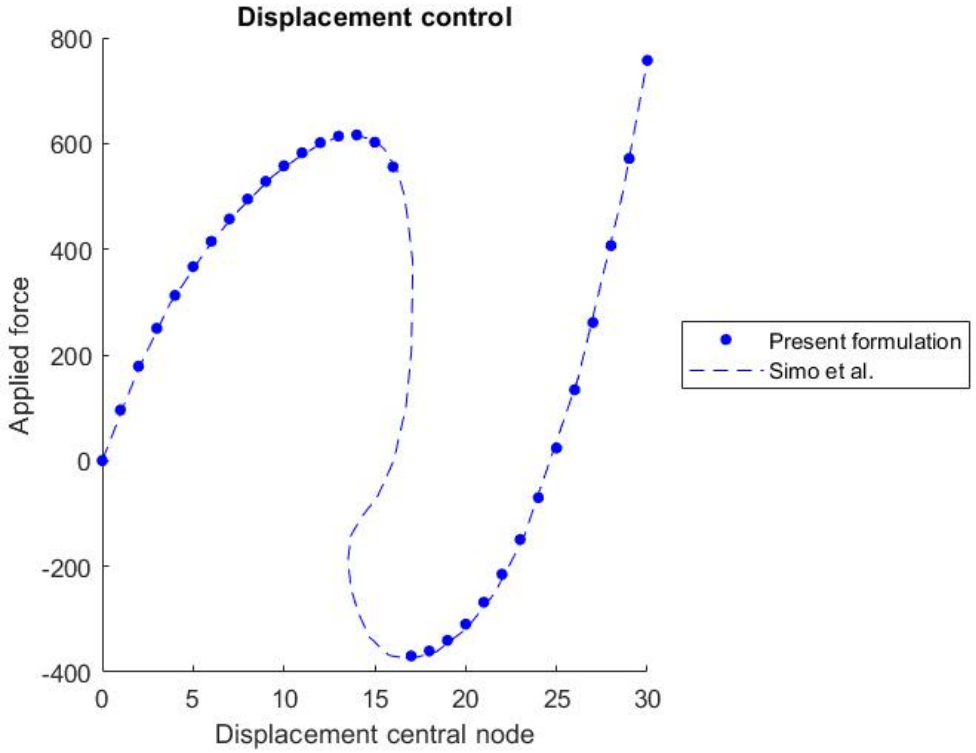


Figura 5.12

5.3.3 Arc-length control

Per seguire l'intero percorso tensionale, è necessario implementare un solutore con *controllo arc-length* (*arc-length control*), il quale mette a sistema l'equazione di equilibrio (3.56), con un'equazione di congruenza, analoga alla (5.1) in riferimento ad un algoritmo in displacement control. Per ogni step, definendo le quantità:

$$[D\mathbf{\Phi}^{(k)}] = \sum_{i=1}^{k} [\Delta\mathbf{\Phi}^{(i)}] \quad D\lambda^{(k)} = \sum_{i=1}^{k} \Delta\lambda^{(i)} \tag{5.6}$$

(corrispondenti alla somma di ogni incremento ad ogni iterazione delle precedenti quantità, in riferimento al singolo step di carico), si impone che la norma al quadrato del vettore $[D\mathbf{s}^{(k)}] = \left[[D\mathbf{\Phi}^{(k)}] \quad \psi D\lambda^{(k)}[\bar{\mathbf{F}}_{\text{ext}}]\right]^T$ sia costante ad ogni incremento e ad ogni iterazione e pari ad una costante scelta $R^2 \in \mathbb{R}$, avendo introdotto il *parametro di scala* $\psi \in \mathbb{R}$, scelto per la presente applicazione pari ad 1. Il punto ottenuto ad ogni step consiste nel'intersezione fra il percorso tensionale definito da $5 \times \mathrm{N_N} + 1$ variabili (corrispondenti ai $5 \times \mathrm{N_N}$ gradi di libertà della mesh ai quali viene aggiunta la variabile λ) ed un'ipersfera di raggio R. Il sistema da risolvere risulta, dunque:

$$\begin{cases} [\mathbf{R}(\mathbf{\Phi}, \lambda)] = [\mathbf{F}_{\text{int}}(\mathbf{\Phi})] - \lambda[\bar{\mathbf{F}}_{\text{ext}}] = [\mathbf{0}] \\ f(\mathbf{\Phi}, \lambda) = [D\mathbf{\Phi}]^T[D\mathbf{\Phi}] + \psi^2 D\lambda^2[\bar{\mathbf{F}}_{\text{ext}}]^T[\bar{\mathbf{F}}_{\text{ext}}] - R^2 = 0 \end{cases} \tag{5.7}$$

avendo definito la funzione f, la quale, a differenza dell'algoritmo in displacement control, risulta in funzione anche di λ. Il sistema precedente potrebbe essere risolto con un algoritmo di Newton-Raphson in modo analogo alla formulazione della sezione precedente; tuttavia, si è scelto di seguire la procedura di Chrisfield, seguendo quanto riportato in [1]. Si sviluppa al primo ordine, dunque, la prima equazione del sistema (5.7), in modo analogo alla (5.3):

$$\begin{aligned} &[\mathbf{R}(\mathbf{\Phi}^{(k+1)}, \lambda^{(k+1)})] = [\mathbf{R}(\mathbf{\Phi}^{(k)} + \Delta\mathbf{\Phi}^{(k)}, \lambda^{(k)} + \Delta\lambda^{(k)})] \approx \\ &\quad \approx [\mathbf{R}(\mathbf{\Phi}^{(k)}, \lambda^{(k)})] + \frac{\partial[\mathbf{R}]}{\partial\mathbf{\Phi}}\Big|_{(\mathbf{\Phi}^{(k)}, \lambda^{(k)})}[\Delta\mathbf{\Phi}^{(k)}] + \frac{\partial[\mathbf{R}]}{\partial\lambda}\Big|_{(\mathbf{\Phi}^{(k)}, \lambda^{(k)})}\Delta\lambda^{(k)} = \\ &\quad = [\mathbf{R}^{(k)})] + K^{(k)}[\Delta\mathbf{\Phi}^{(k)}] - \Delta\lambda^{(k)}[\bar{\mathbf{F}}_{\text{ext}}] = [\mathbf{0}] \end{aligned} \tag{5.8}$$

Dalla precedente, si può ottenere il vettore $[\Delta\mathbf{\Phi}^{(k)}]$, il quale risulta in funzione di $\Delta\lambda^{(k)}$:

$$[\Delta\mathbf{\Phi}^{(k)}] = -K^{(k)-1}[\mathbf{R}^{(k)})] + \Delta\lambda^{(k)}K^{(k)-1}[\bar{\mathbf{F}}_{\text{ext}}] = [\Delta\mathbf{\Phi}_R^{(k)}] + \Delta\lambda^{(k)}[\Delta\mathbf{\Phi}_F^{(k)}] \tag{5.9}$$

avendo definito $[\Delta\mathbf{\Phi}_R^{(k)}]$ e $[\Delta\mathbf{\Phi}_F^{(k)}]$. Sostituendo la (5.9) nella seconda del sistema (5.7) e svolgendo i calcoli, si ottiene un'equazione di secondo grado nell'incognita $\Delta\lambda^{(k)}$:

$$a_0 + a_1\Delta\lambda^{(k)} + a_2\Delta\lambda^{2(k)} = 0 \tag{5.10}$$

avendo introdotto i coefficienti:

$$\begin{aligned} a_0 &= ([D\mathbf{\Phi}]^{(k)} + [\Delta\mathbf{\Phi}_R^{(k)}])^T([D\mathbf{\Phi}]^{(k)} + [\Delta\mathbf{\Phi}_R^{(k)}]) + \psi^2 D\lambda^{(k)2}[\bar{\mathbf{F}}_{\text{ext}}]^T[\bar{\mathbf{F}}_{\text{ext}}] - R^2 \\ a_1 &= 2[\Delta\mathbf{\Phi}_F^{(k)}]^T([D\mathbf{\Phi}]^{(k)} + [\Delta\mathbf{\Phi}_R^{(k)}]) + \psi^2 D\lambda^{(k)}[\bar{\mathbf{F}}_{\text{ext}}]^T[\bar{\mathbf{F}}_{\text{ext}}] \\ a_2 &= [\Delta\mathbf{\Phi}_F^{(k)}]^T[\Delta\mathbf{\Phi}_F^{(k)}] + \psi^2[\bar{\mathbf{F}}_{\text{ext}}]^T[\bar{\mathbf{F}}_{\text{ext}}] \end{aligned} \tag{5.11}$$

Risolvendo la (5.10) si ottengono due valori di $\Delta\lambda_{1,2}^{(k)}$ e, di conseguenza, due valori di $[\Delta\mathbf{\Phi}_{1,2}^{(k)}]$ secondo la (5.9), che definiscono due vettori generalizzati $[\mathbf{s}_{1,2}^{(k)}]$. Questi, definiscono le due intersezioni fra la tangente alla curva del percorso di carico e l'ipersfera di raggio R: una soluzione si avvicina alla soluzione coerente all'evoluzione del percorso di carico, l'altra secondo il percorso inverso. Per scegliere la soluzione corretta, si considera quale soluzione massimizza il coseno dell'angolo fra il vettore $[\mathbf{s}^{(k)}]$ e il vettore all'iterazione precedente $[\mathbf{s}^{(k-1)}]$, indicato con $\cos([\mathbf{s}^{(k)}],[\mathbf{s}^{(k-1)}])$:

$$\cos([\mathbf{s}^{(k)}],[\mathbf{s}^{(k-1)}]) = \frac{[\mathbf{s}^{(k)}]^T[\mathbf{s}^{(k-1)}]}{R^2} \tag{5.12}$$

Ottenuta la soluzione corretta, si aggiornano le variabili:

$$\begin{gathered}[\mathbf{\Phi}^{(k+1)}] = \text{update}([\mathbf{\Phi}^{(k)}],[\Delta\mathbf{\Phi}^{(k+1)}]) \qquad [D\mathbf{\Phi}^{(k+1)}] = [D\mathbf{\Phi}^{(k)}] + [\Delta\mathbf{\Phi}^{(k)}] \\ \lambda^{(k+1)} = \lambda^{(k)} + \Delta\lambda^{(k)} \qquad D\lambda^{(k+1)} = D\lambda^{(k)} + \Delta\lambda^{(k)}\end{gathered} \tag{5.13}$$

dove con la procedure update si intende la procedura descritta dalla (3.3). Iterando fino a convergenza, si ha la soluzione. Si osserva che, seguendo questo procedimento, all'inizio di ogni step, non essendo noto il vettore $[\mathbf{s}^{(k-1)}]$, non è possibile scegliere la soluzione corretta secondo la (5.12). In questo caso, in riferimento al primo step dell'analisi, si procede scegliendo la soluzione $\Delta\lambda^{(k)}$ dello stesso segno di $\det(K^{(k)})$, per gli step successivi, invece, si assume come $[\mathbf{s}^{(k-1)}]$ l'ultimo vettore $[\mathbf{s}]$ calcolato all'iterazione precedente (si verfica facilmente che, alla prima iterazione, il coefficiente a_1 della (5.10) risulta pari a zero, di conseguenza si ha $\Delta\lambda_1^{(k)} = -\Delta\lambda_2^{(k)}$).

Il procedimento è riassunto nell'algoritmo seguente.

input: $[\bar{\mathbf{F}}_{\text{ext}}]$; Toll; MaxIter; Steps; R;
set: $\lambda^{(0)} = 0$;
get: $K^{(0)}$; $[\mathbf{F}_{\text{int}}^{(0)}]$;
$[\mathbf{R}^{(0)}] = [\mathbf{F}_{\text{int}}^{(0)}] - \lambda^{(0)}[\bar{\mathbf{F}}_{\text{ext}}]$;
$\text{Err} = \|[\mathbf{R}^{(0)}]\|$;
for every Step **do**
  set: $k = 0$; $\text{Err} = +\infty$;
  set: $[D\mathbf{\Phi}^{(k)}] = [\mathbf{0}]$; $D\lambda^{(k)} = 0$;
  while Err > Toll **and** $k <$ MaxIter **do**
    $k = k + 1$
    $[\Delta\mathbf{\Phi}_R^{(k)}] = -K^{(k)-1}\mathbf{R}^{(k)}$;
    $[\Delta\mathbf{\Phi}_F^{(k)}] = K^{(k)-1}\mathbf{F}_{\text{ext}}^{(k)}$;
    $a_0 = ([D\mathbf{\Phi}]^{(k)} + [\Delta\mathbf{\Phi}_R^{(k)}])^T([D\mathbf{\Phi}]^{(k)} +$
      $+ [\Delta\mathbf{\Phi}_R^{(k)}]) + \psi^2 D\lambda^{(k)2}[\bar{\mathbf{F}}_{\text{ext}}]^T[\bar{\mathbf{F}}_{\text{ext}}] - R^2$;

$a_1 = 2[\Delta\mathbf{\Phi}_F^{(k)}]^T([D\mathbf{\Phi}]^{(k)} + [\Delta\mathbf{\Phi}_R^{(k)}]) + \psi^2 D\lambda^{(k)}[\bar{\mathbf{F}}_{\text{ext}}]^T[\bar{\mathbf{F}}_{\text{ext}}]$;
$a_2 = [\Delta\mathbf{\Phi}_F^{(k)}]^T[\Delta\mathbf{\Phi}_F^{(k)}] + \psi^2[\bar{\mathbf{F}}_{\text{ext}}]^T[\bar{\mathbf{F}}_{\text{ext}}]$;
$\Delta\lambda_{1,2}^{(k)} = \dfrac{-a_1 \pm \sqrt{a_1^2 - 4a_0a_2}}{2a_2}$;
if Step $= 0$ And $k = 1$; **then**
if $\text{sgn}(\Delta\lambda_1^{(k)}) = \text{sgn}(\det(K^{(k)}))$; **then**
$\Delta\lambda^{(k)} = \Delta\lambda_1^{(k)}$;
else
$\Delta\lambda^{(k)} = \Delta\lambda_2^{(k)}$;
end if
$[\Delta\mathbf{\Phi}^{(k)}] = [\Delta\mathbf{\Phi}_R^{(k)}] + \Delta\lambda^{(k)}[\Delta\mathbf{\Phi}_F^{(k)}]$;
$[D\mathbf{\Phi}^{(k+1)}] = [D\mathbf{\Phi}^{(k)}] + [\Delta\mathbf{\Phi}^{(k)}]$;
$D\lambda^{(k+1)} = D\lambda^{(k)} + \Delta\lambda^{(k)}$;
$[D\mathbf{s}^{(k+1)}] = \left[[D\mathbf{\Phi}^{(k+1)}] \quad \psi D\lambda^{(k+1)}[\bar{\mathbf{F}}_{\text{ext}}]\right]^T$;
else
$[\Delta\mathbf{\Phi}_{1,2}^{(k)}] = [\Delta\mathbf{\Phi}_R^{(k)}] + \Delta\lambda_{1,2}^{(k)}[\Delta\mathbf{\Phi}_F^{(k)}]$;
$[D\mathbf{\Phi}_{1,2}^{(k+1)}] = [D\mathbf{\Phi}^{(k)}] + [\Delta\mathbf{\Phi}_{1,2}^{(k)}]$;
$D\lambda^{(k+1)} = D\lambda^{(k)} + \Delta\lambda_{1,2}^{(k)}$;
$[D\mathbf{s}^{(k+1)}] = \left[[D\mathbf{\Phi}_{1,2}^{(k+1)}] \quad \psi D\lambda_{1,2}^{(k+1)}[\bar{\mathbf{F}}_{\text{ext}}]\right]^T$;
$\cos\theta_{1,2} = \dfrac{[\mathbf{s}_{1,2}^{(k)}]^T[\mathbf{s}^{(k-1)}]}{R^2}$;
Choose the solution that maximise $\cos\theta$;
end if
$[\mathbf{\Phi}^{(k+1)}] = \text{update}([\mathbf{\Phi}^{(k)}], [\Delta\mathbf{\Phi}^{(k)}])$;;
$\lambda^{(k+1)} = \lambda^{(k)} + \Delta\lambda^{(k)}$;
get: $K^{(k+1)}$; $[\mathbf{F}_{\text{int}}^{(k+1)}]$;
$[\mathbf{R}^{(k+1)}] = [\mathbf{F}_{\text{int}}^{(k+1)}] - \lambda^{(k+1)}[\bar{\mathbf{F}}_{\text{ext}}]$;
Err $= \|[\mathbf{R}^{(k+1)}]\|$;
end while
end for

L'algoritmo è implementato nella function `NonLinearSolverArcLengthControl`, riportata qui di seguito.

```
function NonLinearSolverArcLengthControl(Global)
%-------------------------------------------------------------------------------------------%
FreeDof = Global.External.FreeDof;
uR = zeros(Global.Mesh.TotDof,1);
uF = zeros(Global.Mesh.TotDof,1);
ExternalForces = Global.External.ExternalForces;
psi = 1;
MaxIterations = 20;
%-------------------------------------------------------------------------------------------%
F = norm(ExternalForces,2);
```

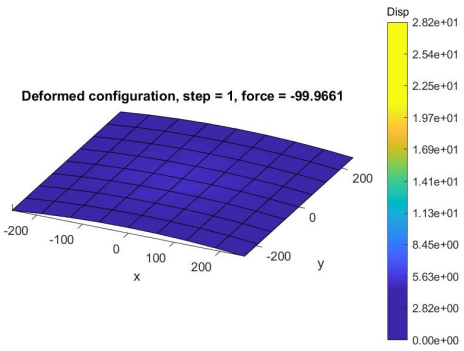
```
lambda = 0;
%-------------------------------------------------------------------------------------------%
fInt = zeros(Global.Mesh.TotDof,1); K = zeros(Global.Mesh.TotDof,Global.Mesh.TotDof);
for ElementNumber = 1:Global.Mesh.TotElements
    KElement = StiffnessMatrix(Global,ElementNumber);
    fIntElement = InternalForces(Global,ElementNumber);
    K = AssembleStiffnessMatrix(K,KElement,Global,ElementNumber);
    fInt = AssembleInternalForces(fInt,fIntElement,Global,ElementNumber);
end
fExt = zeros(Global.Mesh.TotDof,1);
Res = fInt-fExt;
Ds = zeros(Global.Mesh.TotDof+1,1);
%-------------------------------------------------------------------------------------------%
for Step = 1:Global.Solver.LoadSteps
%-------------------------------------------------------------------------------------------%
    Dx = zeros(Global.Mesh.TotDof,1);
    Dlambda = 0;
    Iteration = 0; SubStep = 0; R = 100;
    Err = 10^10;
    lambdaStep = lambda;
    ConfigurationStep = Configuration();
    ConfigurationStep.x0Nodes = Global.Configuration.x0Nodes;
    ConfigurationStep.t0Nodes = Global.Configuration.t0Nodes;
    ConfigurationStep.t0GaussPointsMembraneAndBending = ...
        Global.Configuration.t0GaussPointsMembraneAndBending;
    ConfigurationStep.t0GaussPointsShear = Global.Configuration.t0GaussPointsShear;
    ConfigurationStep.t0_1GaussPointsMembraneAndBending = ...
        Global.Configuration.t0_1GaussPointsMembraneAndBending;
    ConfigurationStep.t0_2GaussPointsMembraneAndBending = ...
        Global.Configuration.t0_2GaussPointsMembraneAndBending;
    ConfigurationStep.R0Nodes = Global.Configuration.R0Nodes;
    ConfigurationStep.xNodes = Global.Configuration.xNodes;
    ConfigurationStep.tNodes = Global.Configuration.tNodes;
    ConfigurationStep.tGaussPointsMembraneAndBending = ...
        Global.Configuration.tGaussPointsMembraneAndBending;
    ConfigurationStep.tGaussPointsShear = Global.Configuration.tGaussPointsShear;
    ConfigurationStep.t_1GaussPointsMembraneAndBending = ...
        Global.Configuration.t_1GaussPointsMembraneAndBending;
    ConfigurationStep.t_2GaussPointsMembraneAndBending = ...
        Global.Configuration.t_2GaussPointsMembraneAndBending;
    ConfigurationStep.RNodes = Global.Configuration.RNodes;
    KStep = K;
    ResStep = Res;
    DsStep = Ds;
    while Err > 10^-6 && Iteration < MaxIterations
        Iteration = Iteration+1;
        %---------------------------------------------------------------------------%
        uR(FreeDof,1) = -K(FreeDof,FreeDof)\Res(FreeDof,1);
        uF(FreeDof,1) = K(FreeDof,FreeDof)\ExternalForces(FreeDof,1);
        A0 = (Dx+uR)'*(Dx+uR)+psi^2*Dlambda^2*(ExternalForces'*ExternalForces)-R^2;
        A1 = 2*uF'*(Dx+uR)+2*psi^2*Dlambda*(ExternalForces'*ExternalForces);
        A2 = uF'*uF+psi^2*(ExternalForces'*ExternalForces);
        gamma1 = (-A1+sqrt(A1^2-4*A0*A2))/2/A2;
        gamma2 = (-A1-sqrt(A1^2-4*A0*A2))/2/A2;
        %---------------------------------------------------------------------------%
        if Step == 1 && Iteration == 1
            sgn = sign(det(K(FreeDof,FreeDof)));
            if sign(gamma1) == sgn
                gamma = gamma1;
            else
                gamma = gamma2;
            end
```

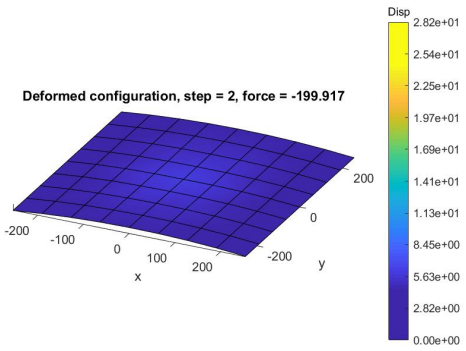
```
    u = uR+gamma*uF;
    Dx = Dx+u;
    Dlambda = Dlambda+gamma;
    Ds = [Dx;Dlambda*psi*F];
else
%-------------------------------------------------------------------------------%
    u1 = uR+gamma1*uF;
    Dx1 = Dx+u1;
    Dlambda1 = Dlambda+gamma1;
    Ds1 = [Dx1;Dlambda1*psi*F];
    cos1 = (Ds'*Ds1)/R^2;
    %-------------------------------------------------------------------------------%
    u2 = uR+gamma2*uF;
    Dx2 = Dx+u2;
    Dlambda2 = Dlambda+gamma2;
    Ds2 = [Dx2;Dlambda2*psi*F];
    cos2 = (Ds'*Ds2)/R^2;
    %-------------------------------------------------------------------------------%
    if cos1 >= cos2
        u = u1;
        gamma = gamma1;
        Dx = Dx1;
        Dlambda = Dlambda1;
        Ds = Ds1;
    else
        u = u2;
        gamma = gamma2;
        Dx = Dx2;
        Dlambda = Dlambda2;
        Ds = Ds2;
    end
end
%-------------------------------------------------------------------------------%
DxNodes = u(1:Global.Mesh.TotDofSurface);
DTNodes = u(Global.Mesh.TotDofSurface+1:Global.Mesh.TotDof,1);
UpdateSurface(Global,DxNodes); UpdateDirector(Global,DTNodes);
%-------------------------------------------------------------------------------%
lambda = lambda+gamma;
%-------------------------------------------------------------------------------%
fExt = lambda*ExternalForces;
fInt = zeros(Global.Mesh.TotDof,1);
K = zeros(Global.Mesh.TotDof,Global.Mesh.TotDof);
for ElementNumber = 1:Global.Mesh.TotElements
    KElement = StiffnessMatrix(Global,ElementNumber);
    fIntElement = InternalForces(Global,ElementNumber);
    K = AssembleStiffnessMatrix(K,KElement,Global,ElementNumber);
    fInt = AssembleInternalForces(fInt,fIntElement,Global,ElementNumber);
end
%-------------------------------------------------------------------------------%
Res = fInt-fExt;
Err = norm(Res(FreeDof,1),inf);
%-------------------------------------------------------------------------------%
if Iteration == MaxIterations
    if SubStep == 5
        error('Reached max number of iterations');
    end
    R = R/2;
    Dx = zeros(Global.Mesh.TotDof,1);
    Dlambda = 0;
    Iteration = 0;
    Err = 10^10;
    lambda = lambdaStep;
```

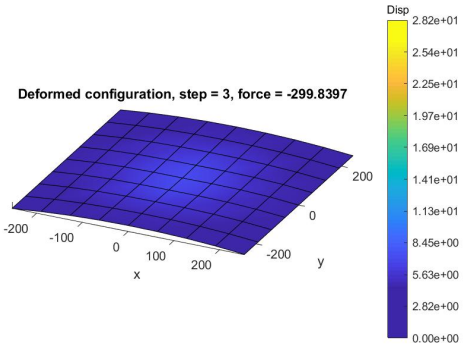
```
            ConfigurationStep.x0Nodes = Global.Configuration.x0Nodes;
            Global.Configuration.t0Nodes = ConfigurationStep.t0Nodes;
            Global.Configuration.t0GaussPointsMembraneAndBending = ...
                ConfigurationStep.t0GaussPointsMembraneAndBending;
            Global.Configuration.t0GaussPointsShear = ConfigurationStep.t0GaussPointsShear;
            Global.Configuration.t0_1GaussPointsMembraneAndBending = ...
                ConfigurationStep.t0_1GaussPointsMembraneAndBending;
            Global.Configuration.t0_2GaussPointsMembraneAndBending = ...
                ConfigurationStep.t0_2GaussPointsMembraneAndBending;
            Global.Configuration.R0Nodes = ConfigurationStep.R0Nodes;
            Global.Configuration.xNodes = ConfigurationStep.xNodes;
            Global.Configuration.tNodes = ConfigurationStep.tNodes;
            Global.Configuration.tGaussPointsMembraneAndBending = ...
                ConfigurationStep.tGaussPointsMembraneAndBending;
            Global.Configuration.tGaussPointsShear = ConfigurationStep.tGaussPointsShear;
            Global.Configuration.t_1GaussPointsMembraneAndBending = ...
                ConfigurationStep.t_1GaussPointsMembraneAndBending;
            Global.Configuration.t_2GaussPointsMembraneAndBending = ...
                ConfigurationStep.t_2GaussPointsMembraneAndBending;
            Global.Configuration.RNodes = ConfigurationStep.RNodes;
            K = KStep;
            Res = ResStep;
            Ds = DsStep;
        end
        %-------------------------------------------------------------------------------%
    end
    %-------------------------------------------------------------------------------%
    Global.Plot.SavexNodesx(:,Step+1) = Global.Configuration.xNodes(:,1);
    Global.Plot.SavexNodesy(:,Step+1) = Global.Configuration.xNodes(:,2);
    Global.Plot.SavexNodesz(:,Step+1) = Global.Configuration.xNodes(:,3);
    %-------------------------------------------------------------------------------%
    Global.Plot.ForcePlot(Step+1,1) = lambda*Global.External.ExternalForce;
    Global.Plot.DisplacementPlot(Step+1,1) = ...
        -(Global.Configuration.xNodes(Global.Plot.NodePlot,3) ...
        -Global.Configuration.x0Nodes(Global.Plot.NodePlot,3));
    %-------------------------------------------------------------------------------%
end
%-------------------------------------------------------------------------------%
end
```

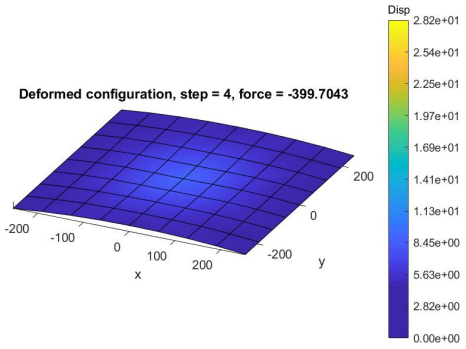
L'analisi è svolta considerando 30 step di incremento di uniformi, corripondenti ad un raggio pari a $R = 10$.

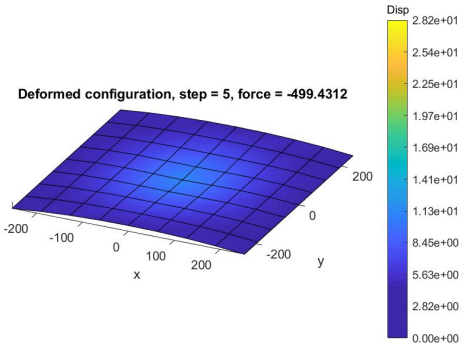
Sono inoltre previsti, nel caso non si raggiungesse la convergenza in 100 iterazioni, un numero massimo di 5 sottoincrementi, per i quali il raggio viene dimezzato ad ogni sottoincremento. Nelle figure successive si riporta la configurazione deformata ad ogni step di carico. Il numero di step considerato nel codice è indicato dal numero Steps.

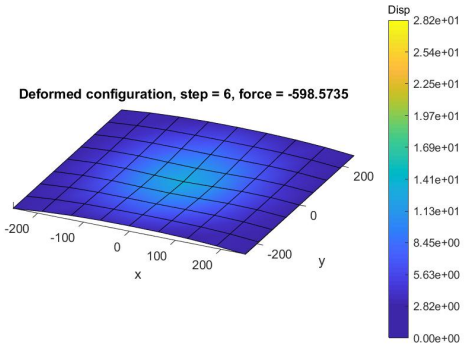

Deformed configuration, step = 1, force = -99.9661
Disp
2.82e+01
2.54e+01
2.25e+01
1.97e+01
1.69e+01
1.41e+01
1.13e+01
8.45e+00
5.63e+00
2.82e+00
0.00e+00
-200
-100
0
100
200
x
y

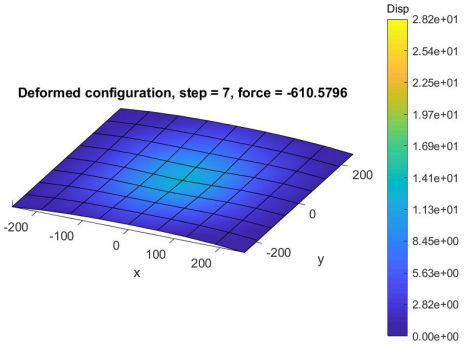

Deformed configuration, step = 2, force = -199.917
Disp
2.82e+01
2.54e+01
2.25e+01
1.97e+01
1.69e+01
1.41e+01
1.13e+01
8.45e+00
5.63e+00
2.82e+00
0.00e+00
-200
-100
0
100
200
x
y

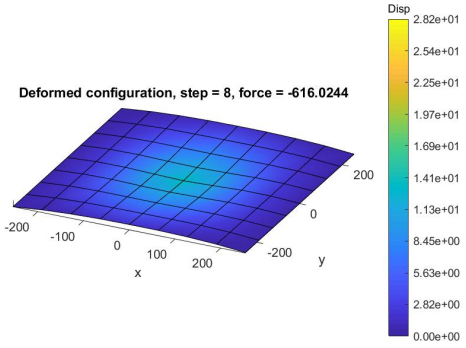

Deformed configuration, step = 3, force = -299.8397
Disp
2.82e+01
2.54e+01
2.25e+01
1.97e+01
1.69e+01
1.41e+01
1.13e+01
8.45e+00
5.63e+00
2.82e+00
0.00e+00
-200
-100
0
100
200
x
y

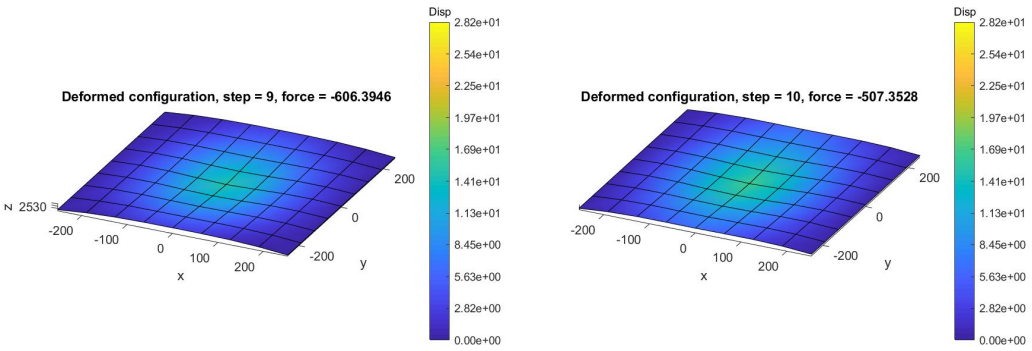

Deformed configuration, step = 4, force = -399.7043
Disp
2.82e+01
2.54e+01
2.25e+01
1.97e+01
1.69e+01
1.41e+01
1.13e+01
8.45e+00
5.63e+00
2.82e+00
0.00e+00
-200
-100
0
100
200
x
y

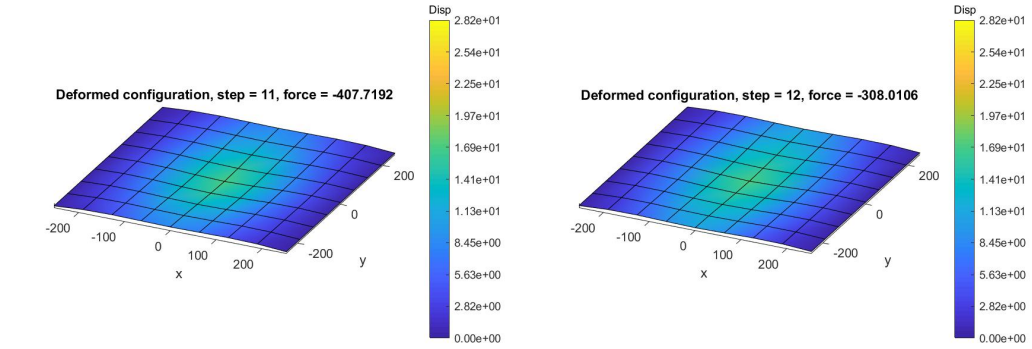

Deformed configuration, step = 5, force = -499.4312
Disp
2.82e+01
2.54e+01
2.25e+01
1.97e+01
1.69e+01
1.41e+01
1.13e+01
8.45e+00
5.63e+00
2.82e+00
0.00e+00
-200
-100
0
100
200
x
y

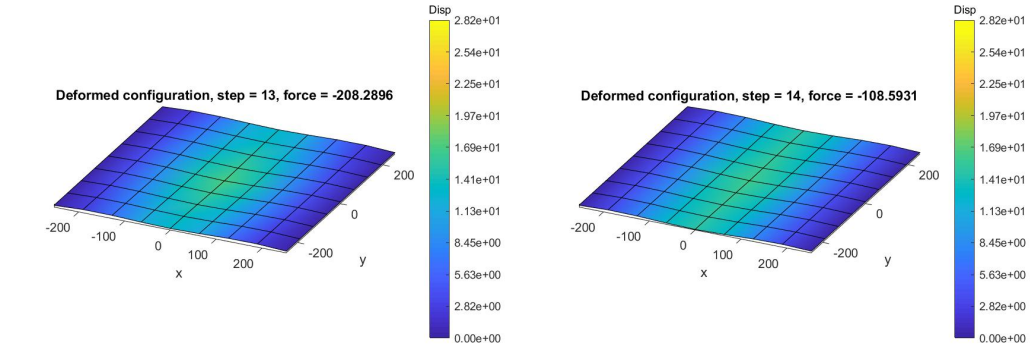

Deformed configuration, step = 6, force = -598.5735
Disp
2.82e+01
2.54e+01
2.25e+01
1.97e+01
1.69e+01
1.41e+01
1.13e+01
8.45e+00
5.63e+00
2.82e+00
0.00e+00
-200
-100
0
100
200
x
y

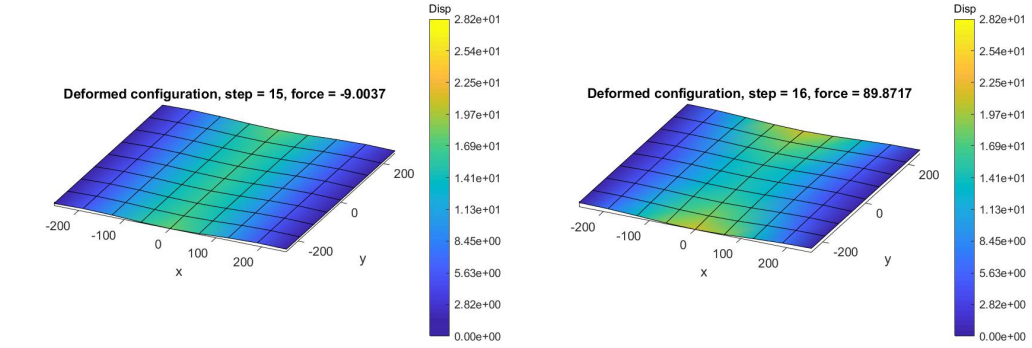

Deformed configuration, step = 7, force = -610.5796
Disp
2.82e+01
2.54e+01
2.25e+01
1.97e+01
1.69e+01
1.41e+01
1.13e+01
8.45e+00
5.63e+00
2.82e+00
0.00e+00
-200
-100
0
100
200
x
y

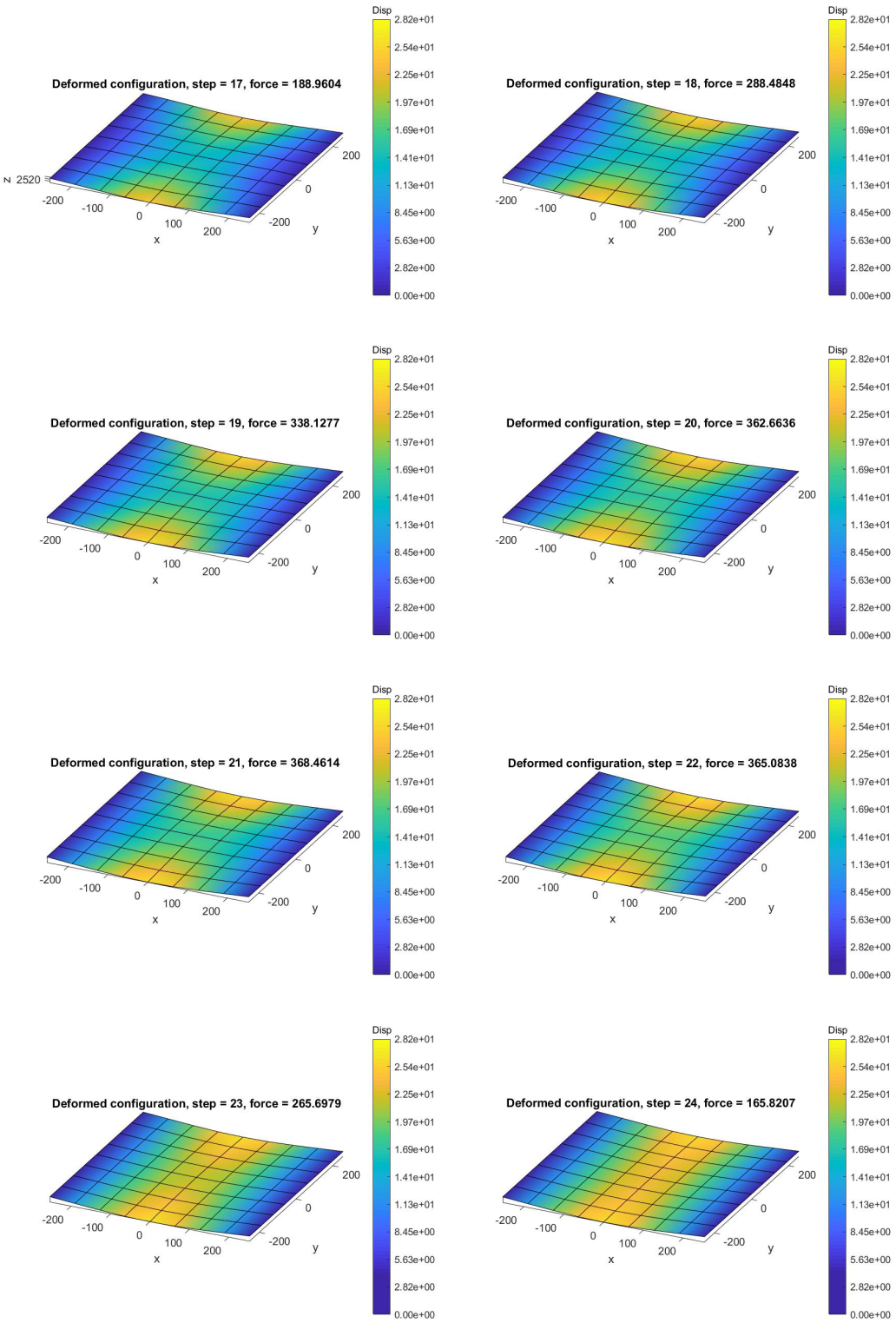

Deformed configuration, step = 8, force = -616.0244
Disp
2.82e+01
2.54e+01
2.25e+01
1.97e+01
1.69e+01
1.41e+01
1.13e+01
8.45e+00
5.63e+00
2.82e+00
0.00e+00
-200
-100
0
100
200
x
y

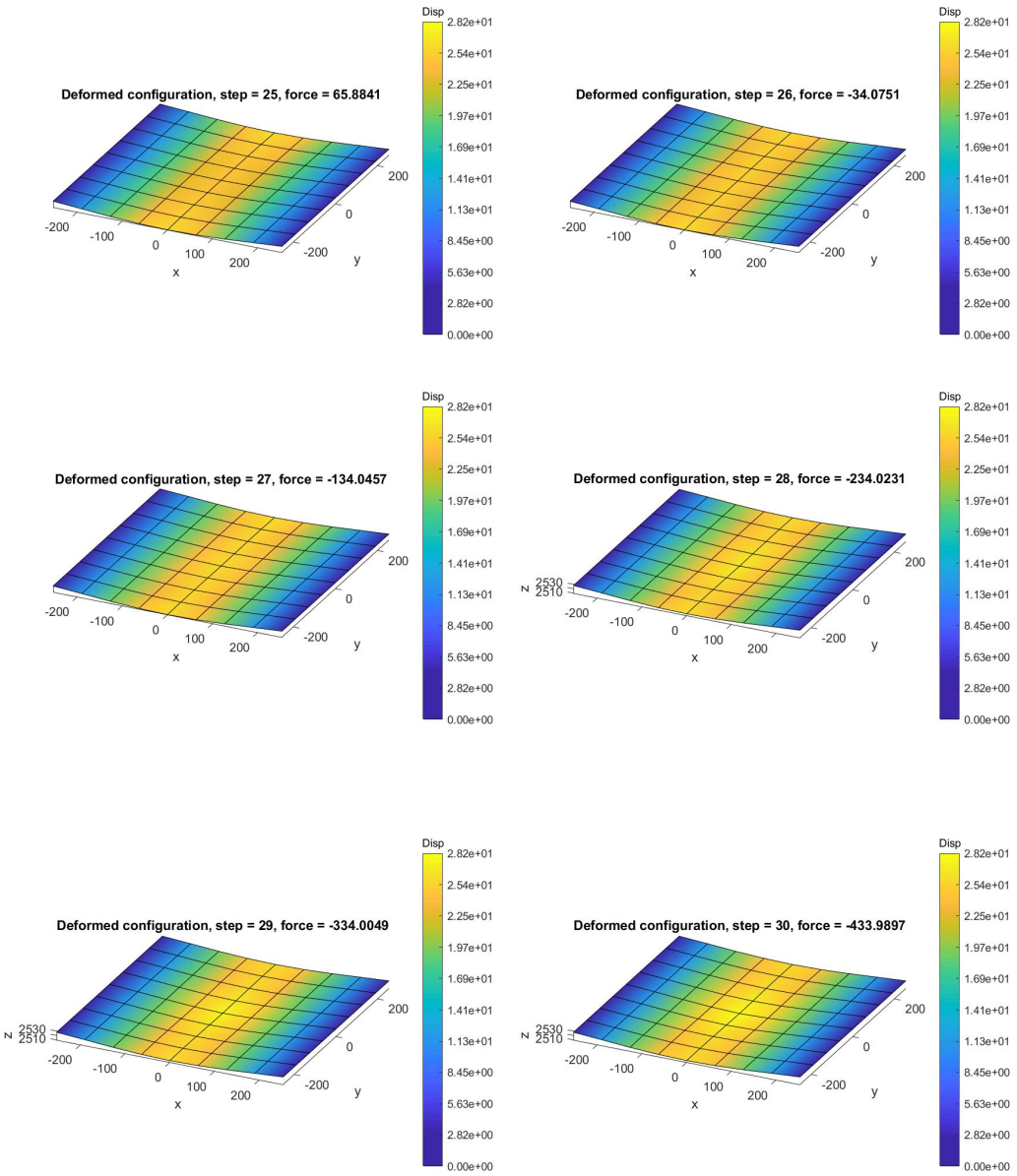

Deformed configuration, step = 9, force = -606.3946
Deformed configuration, step = 10, force = -507.3528
Disp
2.82e+01
2.54e+01
2.25e+01
1.97e+01
1.69e+01
1.41e+01
1.13e+01
8.45e+00
5.63e+00
2.82e+00
0.00e+00
z 2530
x
y


Deformed configuration, step = 11, force = -407.7192
Deformed configuration, step = 12, force = -308.0106
Disp
2.82e+01
2.54e+01
2.25e+01
1.97e+01
1.69e+01
1.41e+01
1.13e+01
8.45e+00
5.63e+00
2.82e+00
0.00e+00
x
y


Deformed configuration, step = 13, force = -208.2896
Deformed configuration, step = 14, force = -108.5931
Disp
2.82e+01
2.54e+01
2.25e+01
1.97e+01
1.69e+01
1.41e+01
1.13e+01
8.45e+00
5.63e+00
2.82e+00
0.00e+00
x
y


Deformed configuration, step = 15, force = -9.0037
Deformed configuration, step = 16, force = 89.8717
Disp
2.82e+01
2.54e+01
2.25e+01
1.97e+01
1.69e+01
1.41e+01
1.13e+01
8.45e+00
5.63e+00
2.82e+00
0.00e+00
x
y


Deformed configuration, step = 17, force = 188.9604
Deformed configuration, step = 18, force = 288.4848
Deformed configuration, step = 19, force = 338.1277
Deformed configuration, step = 20, force = 362.6636
Deformed configuration, step = 21, force = 368.4614
Deformed configuration, step = 22, force = 365.0838
Deformed configuration, step = 23, force = 265.6979
Deformed configuration, step = 24, force = 165.8207
Disp
2.82e+01
2.54e+01
2.25e+01
1.97e+01
1.69e+01
1.41e+01
1.13e+01
8.45e+00
5.63e+00
2.82e+00
0.00e+00
z 2520
x
y
-200
-100
0
100
200



In figura 5.10 è riportata la configurazione iniziale. In figura 5.11 sono riportati i valori della forza esterna e dello spostamento verticale nel nodo centrale ad ogni step di carico, confrontati con l'intero percorso di carico riportato in [12]. Si nota che, in questo caso, l'intero percorso tensionale della struttura viene seguito.

In figura 5.14, si può osservare un confronto fra i tre solutori. La configurazione di equilibrio definita da un'analisi in controllo di forza risulta dall'intersezione fra la curva tensionale e la retta corrispondente alla forza

esterna di riferimento al generico step. A partire dallo step s, infatti, la configurazione successiva corrispondente ad un ulteriore incremento della forza esterna si ottiene con un cambiamento improvviso della configurazione a forza costante. Questo fenomeno è definito *snap-through*.

Al contrario, in un'analisi in controllo di spostamento, l'equilibrio è definito dall'intersezione fra la curva tensionale e la retta verticale corrispondente allo spostamento imposto. Analogamente a quanto descritto nel capoverso precedente, a partire dallo step s, un successivo incremento di spostamento determina una configurazione di equilibrio attraverso un cambiamento improvviso a spostamento (del nodo scelto per il tracciamento della curva) costante. Questo fenomeno è definito *snap-back*.

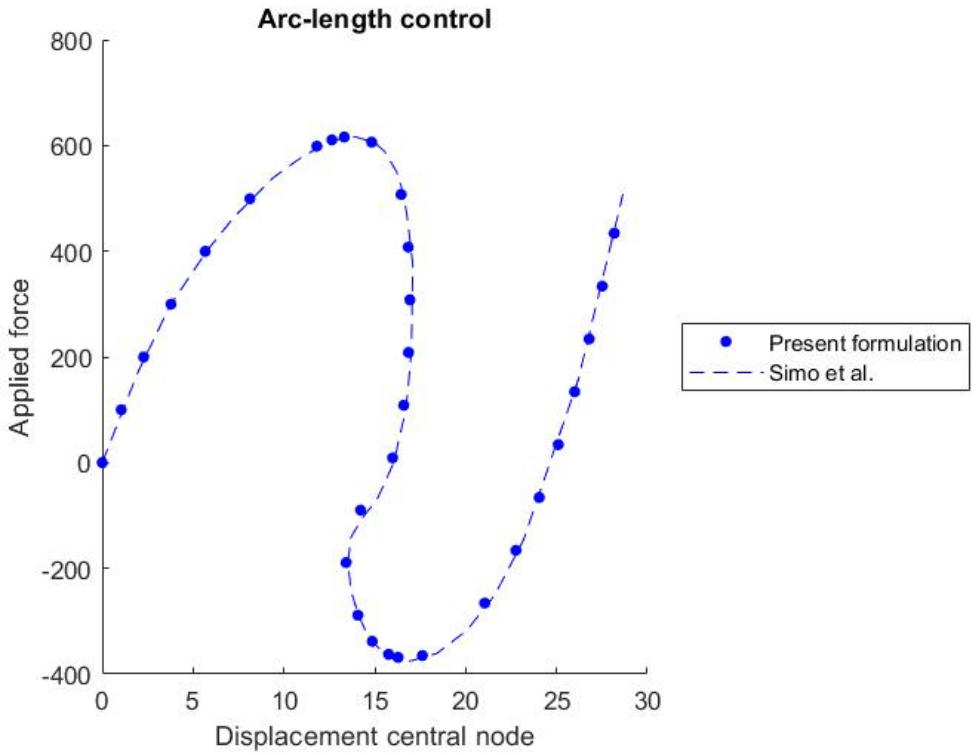


Figura 5.13

Il metodo efficace per risucire a seguire l'intero percorso tensionale è dato, dunque, da un'analisi in controllo arc-lenth, la quale determina l'intersezione fra la curva tensionale e l'iper-sfera definita dalla seconda equazione del sistema (5.7), la quale, considerando (senza perdita di generalità) una struttura definita da un singolo grado di libertà, si riduce ad una circonferenza nel piano forza-spostamento.

Si riporta di seguito il `main` per richiamare l'algoritmo in displacement control.

```
% Fem Solver for nonlinear geometrically exact shell element
%-------------------------------------------------------------------------------------------%
Global = Generate('InputFile');
NonLinearSolverArcLengthControl(Global);
%-------------------------------------------------------------------------------------------%
PlotConfigurations(Global);
```

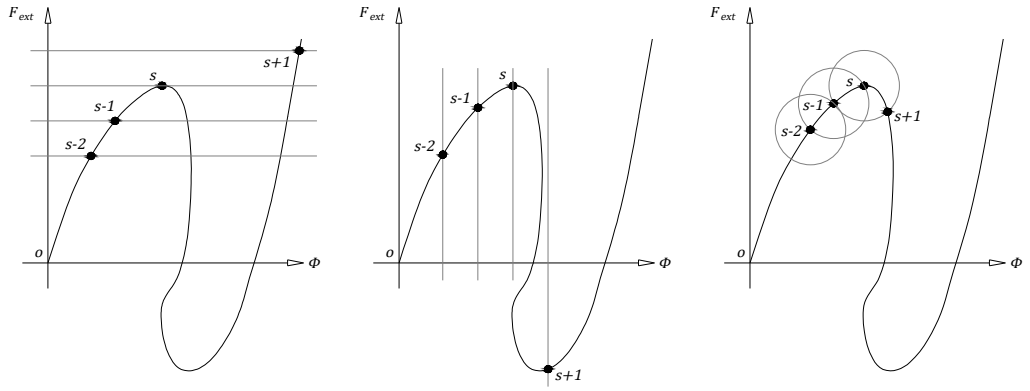


Figura 5.14

Conclusione

L'analisi di strutture a guscio rappresenta una tematica fondamentale dell'ingegneria strutturale, i moderni software commerciali di calcolo agli elementi finiti ne consentono uno studio sia in campo lineare, sia considerando la teoria a geometria esatta.

La conoscenza al dettaglio della struttura di un codice come quello sviluppato nella presente tesi consente all'ingegnere un approccio più consapevole ai moderni programmi di calcolo, in modo da poterne conoscere le limitatezze e di avere una visione più critica dei risultati ottenuti. Un ulteriore vantaggio consiste nella possibilità di interfacciarsi con essi mediando lo sviluppo di subroutine personalizzate, fino a sviluppare veri e propri programmi completi volti ad analisi al dettaglio di applicazioni particolari, talvolta difficilmente reperibili in ambito professionale.

Infine, il codice presentato potrebbe definire un punto di partenza per eventuali sviluppi ulteriori. Rimanendo nella serie di pubblicazioni presa di riferimento, [10]-[16], il primo intervento per completare le prime tre pubblicazioni potrebbe essere quello di aggiungere le analisi di *buckling* lineare e non lineare, e di completare il codice originale implementando la *mixed variational formulation* e l'*assumed strain method*.

Fatto questo, si potrebbe pensare di estendere il codice alle successive pubblicazioni della serie, considerando il grado di libertà relativo alle deformazioni lungo lo spessore dello shell [13] ed implementando una formulazione in modo da studiare superfici che presentano spigoli [16]. Oppure, considerando ulteriori legami costitutivi, ad esempio il legame elastoplastico [14], e risolvendo le equazioni in campo dinamico [15].

Si conclude questa tesi segnalando che, nel quarto e nel quinto capitolo, è riportato interamente il codice, in modo da consentire al lettore di interfacciarsi con le applicazioni proposte al capitolo 5 in prima persona.

Bibliografia

[1] Bonet, J., & Wood, R. (2008). DISCRETIZATION AND SOLUTION. In Nonlinear Continuum Mechanics for Finite Element Analysis (pp. 237-265). Cambridge: Cambridge University Press. https://doi.org/10.1017/CBO9780511755446.010

[2] Forte, S., Preziosi, L., Vianello, M. (2019). Corpi e deformazioni. In: Meccanica dei Continui. UNITEXT(), vol 114. Springer, Milano. https://doi.org/10.1007/978-88-470-3985-8_1

[3] Forte, S., Preziosi, L., Vianello, M. (2019). Leggi di bilancio, sforzi e disuguaglianza entropica. In: Meccanica dei Continui. UNITEXT(), vol 114. Springer, Milano. https://doi.org/10.1007/978-88-470-3985-8_3

[4] Forte S., Preziosi L., Vianello M. (2019). Appendice A: Vettori e tensori. In: Meccanica dei Continui. UNITEXT(), vol 114. Springer, Milano. https://doi.org/10.1007/978-88-470-3985-8_8

[5] Forte S., Preziosi L., Vianello M. (2019). Appendice B: Coordinate generali. In: Meccanica dei Continui. UNITEXT(), vol 114. Springer, Milano. https://doi.org/10.1007/978-88-470-3985-8_9

[6] Gambolati G., Ferronato M., (2015): Lezioni di metodi numerici per l'ingegneria. Progetto Libreria.

[7] Krenk, S. (2009). Non-linear Modeling and Analysis of Solids and Structures. Cambridge: Cambridge University Press. doi:10.1017/CBO9780511812163

[8] Majorana, C., Salomoni, V. (2008). Dynamic Nonlinear Material Behaviour of Thin Shells in Finite Displacements and Rotations. CMES-Computer Modeling in Engineering & Sciences, 33(1), 49–84. https://doi.org/10.3970/cmes.2008.033.049

[9] Müller Alexander & Bischoff Manfred. (2022). A Consistent Finite Element Formulation of the Geometrically Non-linear Reissner-Mindlin Shell Model. Archives of Computational Methods in Engineering. https://doi.org/10.1007/s11831-021-09702-7

[10] Simo J.C., Fox D.D. (1989): On a stress resultant geometrically exact shell model. Part I: Formulation and optimal parametrization. Computer Methods in Applied Mechanics and Engineering 72. https://doi.org/10.1016/0045-7825(89)90002-9

[11] Simo J.C., Fox D.D., Rifai M.S. (1989): On a stress resultant geometrically exact shell model. Part II: The linear theory; computational aspects. Computer Methods in Applied Mechanics and Engineering 73. https://doi.org/10.1016/0045-7825(89)90098-4

[12] Simo J.C., Fox D.D., Rifai M.S. (1990): On a stress resultant geometrically exact shell model. Part III: Computational aspects of the nonlinear theory. Computer Methods in Applied Mechanics and Engineering 79. https://doi.org/10.1016/0045-7825(90)90094-3

[13] Simo J.C., Fox D.D., Rifai M.S. (1990): On a stress resultant geometrically exact shell model. Part IV: Variable thickness shells with through-the-thickness stretching. Computer Methods in Applied Mechanics and Engineering 81. https://doi.org/10.1016/0045-7825(90)90143-A

[14] Simo J.C., Kennedy J.G. (1992): On a stress resultant geometrically exact shell model. Part V. Nonlinear plasticity: formulation and integration algorithms. Computer Methods in Applied Mechanics and Engineering 96. https://doi.org/10.1016/0045-7825(92)90129-8

[15] Simo, J.C., Rifai, M.S. and Fox, D.D. (1992), On a stress resultant geometrically exact shell model. Part VI: Conserving algorithms for nonlinear dynamics. Int. J. Numer. Meth. Engng., 34: 117-164. https://doi.org/10.1002/nme.1620340108.

[16] Simo J.C. (1993): On a stress resultant geometrically exact shell model. Part VII: Shell intersections with 56-DOF finite element formulations. Computer Methods in Applied Mechanics and Engineering 108. https://doi.org/10.1016/0045-7825(93)90008-L

[17] Zienkiewicz O, Taylor R, Fox D (2014): Chapter 10 - Background Mathematics and Linear Shell Theory. In: The Finite Element Method for Solid and Structural Mechanics, 7th Edition., Butterworth-Heinemann. https://doi.org/10.1016/B978-1-85617-634-7.00010-7

[18] Zienkiewicz O, Taylor R, Fox D (2014): Chapter 14 - A Nonlinear Geometrically Exact Shell Model. In: The Finite Element Method for Solid and Structural Mechanics, 7th Edition., Butterworth-Heinemann. https://doi.org/10.1016/B978-1-85617-634-7.00014-4