



UNIVERSITÀ DEGLI STUDI DELLA BASILICATA

SCUOLA DI INGEGNERIA

**CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA E
DELLE TECNOLOGIE DELL'INFORMAZIONE**

RELAZIONE PROGETTO MNCC

DOCUMENTAZIONE MATSOL

Docente:

Ch.mo Prof. Raffaele Fresa

Studenti:

Michael Pio Stolfi 68787

Rocco Samuele Tancredi 64366

Ivan Scarano 69156

ANNO ACCADEMICO 2023-2024

Indice

1. <u>Introduzione</u>	01
1.1. <u>Specifica del progetto</u>	
1.2. <u>Contesto del progetto</u>	
1.3. <u>Strumenti utilizzati</u>	
2. <u>Sviluppo</u>	05
2.1. <u>Costruzione del modello COMSOL</u>	
2.2. <u>Implementazione delle funzioni MATLAB</u>	
2.3. <u>Progettazione della Command Line Interface (CLI)</u>	
3. <u>Risultati</u>	20
3.1. <u>Presentazione dei risultati</u>	
4. <u>Documentazione delle funzioni</u>	22
4.1. <u>Riepilogo delle funzioni</u>	
4.2. <u>Funzioni raggruppate per compito</u>	
4.3. <u>Documentazione</u>	

Capitolo 1

Introduzione

L'obiettivo principale del progetto è stato quello di sviluppare un applicativo MATLAB che permettesse l'estrazione, l'elaborazione e il salvataggio di una serie di informazioni rilevanti da un qualsiasi modello COMSOL di interesse.

1.1 Specifica del progetto

La specifica di progetto prevedeva che l'applicativo mettesse a disposizione dell'utente tramite opportune elaborazioni, sia nel workspace base di MATLAB sia su un file su disco, i seguenti dati del modello COMSOL di interesse:

- 1) Le matrici di incidenza
- 2) Le funzioni di forma
- 3) Le matrici Jacobiane e di Trasformazione
- 4) Le proprietà dei materiali

Per quanto riguarda il primo punto, l'applicativo avrebbe dovuto estrarre/generare le seguenti matrici di incidenza:

- 1) Nodi-Elementi
- 2) Nodi-Facce
- 3) Nodi-Facce di Frontiera
- 4) Nodi-Lati
- 5) Facce-Elementi
- 6) Lati-Elementi
- 7) Lati-Facce
- 8) Lati-Facce di Frontiera

9) Domini-Elementi

10) Facce di Frontiera Dominio-Facce di Frontiera Elementi

Per quanto riguarda invece il secondo punto esso avrebbe dovuto mettere a disposizione le funzioni di forma della famiglia di Lagrange, utilizzate da COMSOL per tutti i possibili tipi di elementi di mesh ovvero: tetraedro, piramide, prisma, esaedro; sia per il primo che per il secondo ordine.

Per quanto riguarda il terzo punto esso avrebbe dovuto estrarre/generare le matrici Jacobiane e le matrici di Trasformazione che permettono la conversione dell'elemento di mesh locale nell'elemento di mesh globale.

Infine, per quanto riguarda il quarto punto esso avrebbe dovuto estrarre tutte le informazioni utili sui materiali presenti nel componente selezionato del modello.

1.2 Contesto del progetto

Lo sviluppo di un'applicazione in MATLAB, integrata con COMSOL tramite LiveLink, risulta essere particolarmente utile in contesti di ricerca e sviluppo di modelli ingegneristici complessi. Questo tipo di integrazione facilita l'accesso a informazioni chiave per la comprensione e l'analisi dei modelli fisici simulati.

Partendo col dire che uno dei vantaggi che tale progetto comporta è quello di automatizzare molte delle operazioni che altrimenti richiederebbero un intervento manuale, risparmiando tempo e riducendo il margine d'errore. COMSOL è uno strumento estremamente potente per la simulazione multi-fisica, ma la sua interfaccia nativa può risultare poco pratica per utenti che necessitano di estrarre specifici elementi del modello o che necessitano di processare informazioni in modo personalizzato e automatizzato. Questo è particolarmente utile quando i risultati di COMSOL devono essere rielaborati ulteriormente o integrati in

workflow più complessi. Il LiveLink tra COMSOL e MATLAB fornisce quindi un mezzo per adattare l'ambiente di simulazione alle esigenze particolari dell'utente, senza dover dipendere unicamente dalle funzionalità della sua interfaccia nativa. Sviluppare un'applicazione che permette di automatizzare l'estrazione e l'elaborazione dei dati significa non solo aumentare l'efficienza analitica, ma anche assicurare che ogni passo del processo sia facilmente replicabile.

Un altro innegabile vantaggio è la possibilità di trasferire i dati tra ambienti di simulazione. Il fatto che COMSOL sia un ambiente chiuso in cui la maggior parte delle operazioni avviene tramite un'interfaccia grafica rende difficile esportare e utilizzare i dati al di fuori del suo contesto. Il progetto in questione permette di superare queste barriere, consentendo di sfruttare i dati estratti tra differenti ambienti di simulazione, calcolo e/o analisi, rendendo il flusso di lavoro più fluido e flessibile.

1.3 Strumenti utilizzati

In questo progetto sono stati impiegati diversi strumenti. Di seguito, ne viene fornita una descrizione dei principali:

1) MATLAB

MATLAB è stato lo strumento principale per lo sviluppo dell'applicativo. Si tratta di un ambiente di programmazione avanzato e di un linguaggio di calcolo scientifico ampiamente utilizzato per l'analisi numerica, la simulazione e lo sviluppo di algoritmi. Nello specifico, MATLAB è stato utilizzato per:

- Implementare le routine di estrazione dei dati dai modelli COMSOL;
- Eseguire l'elaborazione e la visualizzazione dei dati estratti;
- Salvare i risultati sia nel workspace base di MATLAB che in file esterni per l'archiviazione e/o l'analisi successiva.

2) COMSOL Multiphysics

COMSOL Multiphysics è un software per la simulazione multi-fisica che consente di modellare e risolvere problemi in una vasta gamma di campi ingegneristici, tra cui meccanica strutturale, termica, fluidodinamica e elettromagnetismo. In questo progetto, COMSOL è stato utilizzato come piattaforma di simulazione per generare i modelli fisici da cui sono stati estratti i dati. I modelli COMSOL contengono informazioni di interesse come:

- Matrici di incidenza;
- Funzioni di forma degli elementi di mesh;
- Matrici Jacobiane e di trasformazione;
- Proprietà dei materiali utilizzati nei modelli.

3) COMSOL Server

COMSOL Server è il server che ha permesso l'interazione tra COMSOL e MATLAB ed è il cuore dell'interfaccia di comunicazione LiveLink. Questo strumento consente di interfacciare direttamente i modelli di COMSOL con l'ambiente MATLAB. Tramite il LiveLink, è stato possibile:

- Accedere alle strutture di dati interne dei modelli COMSOL e portarle in MATLAB;
- Eseguire simulazioni in modalità batch direttamente da MATLAB.

Capitolo 2

Sviluppo

Lo sviluppo del progetto si è articolato principalmente in tre fasi: la costruzione del modello COMSOL, l'implementazione delle funzioni MATLAB e infine la progettazione della Command Line Interface (CLI).

La prima fase, indispensabile per il proseguo del progetto, è consistita nella costruzione di un modello COMSOL che fosse il più generale possibile, in termini di numero di componenti, mesh, fisiche, studi e step contenuti. Questo era fondamentale per permettere poi la costruzione di un applicativo che fosse il più generale possibile e robusto a scenari articolati e/o alla presenza di mesh complesse.

La seconda fase, che è il cuore del progetto, è consistita nello sviluppo di un gran numero di funzioni MATLAB, che eseguono tutte le elaborazioni necessarie alla estrazione/generazione delle informazioni di interesse. Queste essendo state pensate per essere generiche ed efficienti hanno spesso delle implementazioni piuttosto complesse, e richiedono il passaggio di numerosi parametri per funzionare correttamente.

La terza e ultima fase, è consistita nello sviluppo del main che è la sede di tutta la logica della Command Line Interface (CLI). La CLI è l'applicativo a riga di comando che permette all'utente in maniera semplice ed intuitiva di interfacciarsi con le funzioni descritte precedentemente; l'idea è che l'utente risponda a una serie di domande sulla natura del dato da estrarre ed essa provveda a chiamare correttamente la funzione di interesse.

2.1. Costruzione del modello COMSOL

Come detto in precedenza la costruzione di un modello COMSOL era indispensabile poiché l'applicativo avrebbe avuto come input principale proprio tali modelli, e inoltre era importante che fosse il più generale possibile. Il modello costruito a scopo di sviluppo/test è stato intitolato *component_library_RF.mph*. Esso si compone di 4 componenti, alcuni componenti hanno fisiche multiple, altri componenti hanno mesh multiple, altri hanno anche domini multipli.

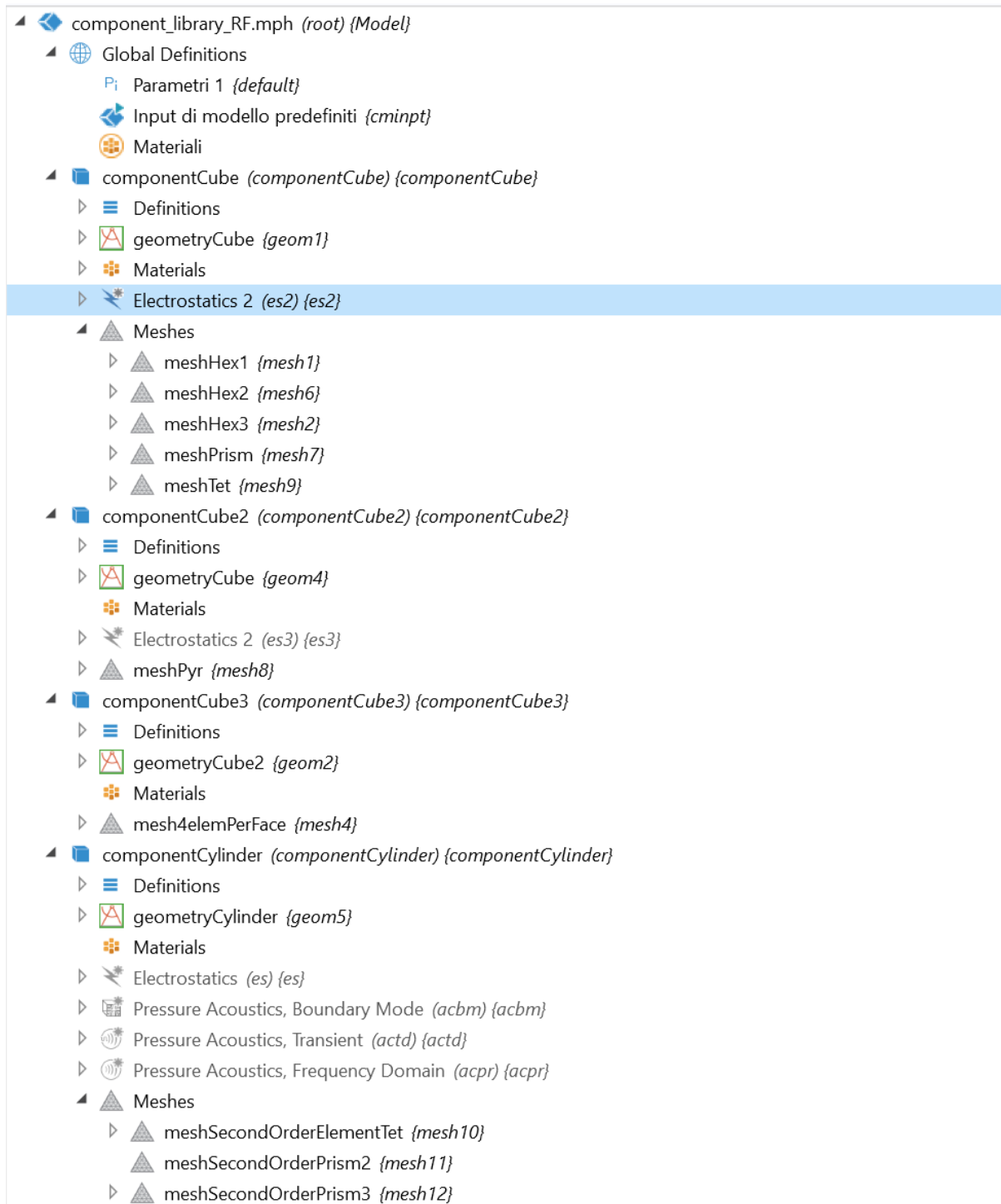


Figura 1: albero del modello con i suoi vari nodi(componenti)

Il componente “componentCube3” è il più banale, esso ha un singolo dominio esaedrico, una singola mesh composta da 8 esaedri del primo ordine, e nessuna fisica; esso è stato fondamentale nelle prime fasi di sviluppo per comprendere come avviene la comunicazione tra MATLAB e COMSOL e inoltre per iniziare l’implementazione delle funzioni di generazione delle matrici di incidenza per elementi di mesh del primo ordine.

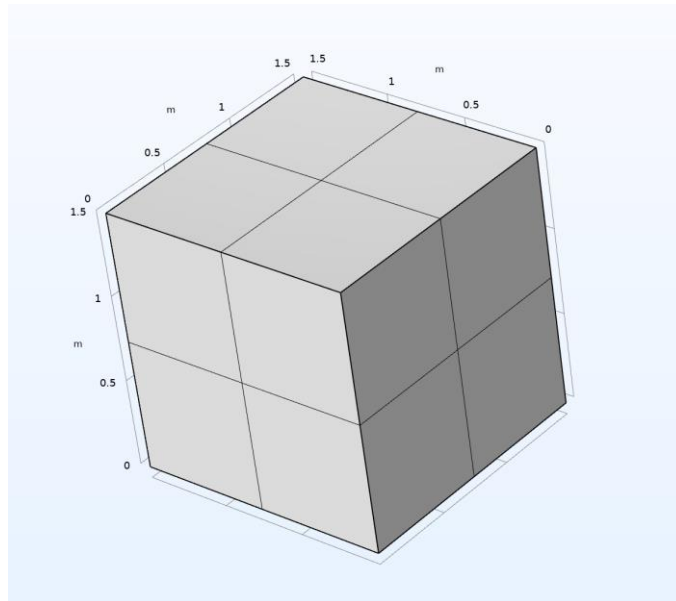


Figura 2: componente componentCube3

Il componente “componentCube” è una versione avanzata del precedente, esso ha due domini esaedrici, diverse mesh che coprono 3 dei 4 tipi di elementi di mesh presenti in COMSOL, e una fisica; esso è risultato importante durante la generalizzazione delle funzioni a più domini, e durante l’introduzione della gestione degli elementi del secondo ordine.

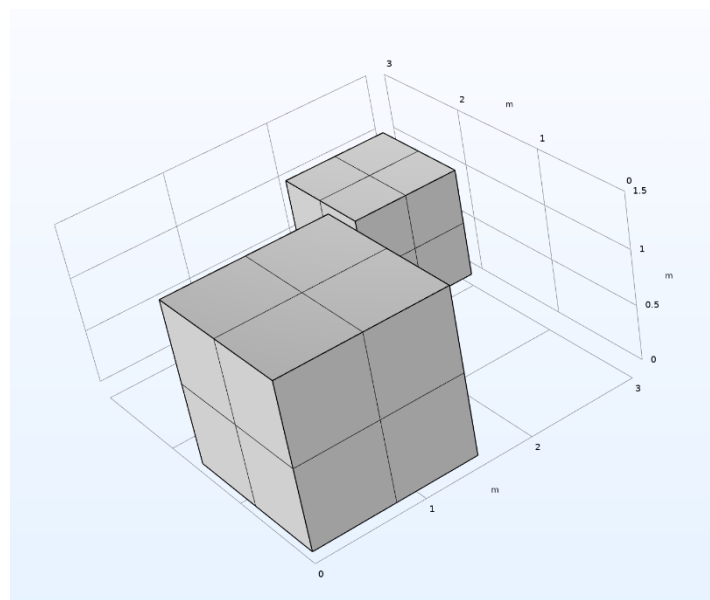


Figura 3: componente componentCube

Infine l’ultimo componente di interesse è il componente “componentCube2”, esso ha una geometria ben più complessa degli altri componenti, e ha un dominio unico che è il risultato dell’operazione di unione di 3 domini differenti; è risultato indispensabile durante l’introduzione nelle funzioni della gestione del tipo di elemento di mesh piramidale. L’elemento di mesh piramidale è l’unico elemento

di mesh utilizzato da COMSOL che non è possibile inserire volontariamente nella propria mesh, viene invece inserito da COMSOL come elemento di giunzione tra una mesh contenente prismi e una contenente tetraedri. La geometria particolare di questo componente serve appunto a scatenare la creazione di tali elementi di mesh.

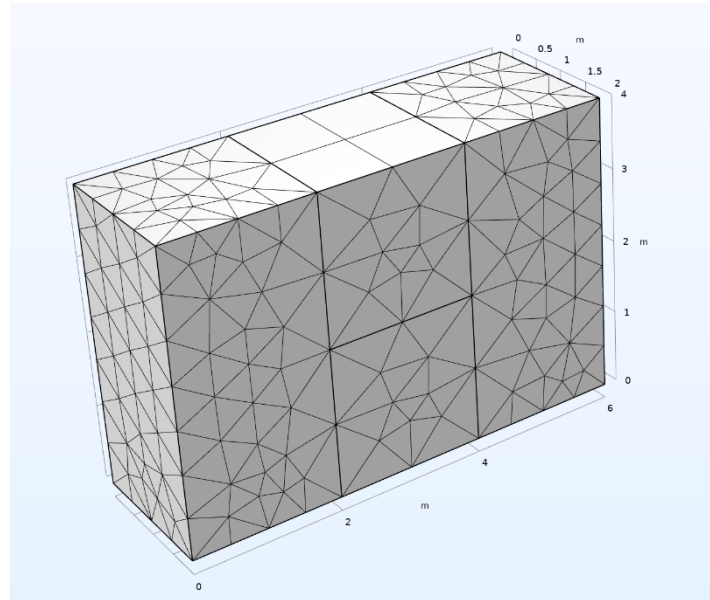


Figura 4: componente componentCube2

Nel modello sono stati inseriti inoltre diversi studi, alcuni con diversi step che vanno a coprire tutti gli studi presenti in COMSOL, questo è servito durante l'implementazione dei controlli sulla mesh e sullo studio selezionato dall'utente tra quelli resi disponibili dalla CLI.

2.1. Implementazione delle funzioni MATLAB

In questo sottoparagrafo verranno analizzate soltanto le principali funzioni MATLAB, seguendo un flusso di esecuzione standard, se si è interessati ai dettagli di una singola funzione fare riferimento al capitolo 4.

Iniziando dal caso in cui l'utente sia interessato a ottenere le matrici di incidenza, interrogando opportunamente la CLI essa scatenerà la chiamata della funzione *createIncidenceMatricesForCLI*. Quest'ultima è un punto di ingresso unico per il calcolo delle matrici di incidenza e si occuperà di chiamare diverse altre funzioni per ottenere il risultato desiderato, di conseguenza la sua intestazione è piuttosto articolata come è possibile osservare:

```
incidenceMatrices = createIncidenceMatricesForCLI (
    model,
    selectedComponentGeometryTag,
    geometryTagPos,
```

```
meshdata,  
meshdataTypeList,  
searchedString,  
elementsOrder,  
flagsStruct,  
fileName,  
flagFacesEqual)
```

Si rimanda sempre al capitolo 4 per una analisi puntuale dei parametri di ingresso della funzione.

Tale funzione si occupa anzitutto di estrarre la matrice delle COORDINATE NODALI tramite il seguente snippet di codice:

```
if elementsOrder == 2  
    nodes = double(meshdata.nodes(geometryTagPos).coords);  
else  
    nodes = meshdata.vertex;  
end  
% Trasposizione della matrice degli elementi  
transposedMatrixNodes = nodes';
```

È interessante notare come la funzione si comporti in maniera differente in relazione all'ordine degli elementi della mesh, questo è dovuto al fatto che la struttura dati *meshdata*, che viene passata alla funzione *createIncidenceMatricesForCLI* viene estrapolata con due chiamate differenti dal server COMSOL a seconda dell'ordine degli elementi della mesh, e quindi può essere composta in maniera differente. Infatti se gli elementi sono del primo ordine viene estrapolata grazie alla seguente istruzione:

```
[~,meshdata] = mphmeshstats(model, selectedMeshTag);
```

mentre se gli elementi sono del secondo ordine viene estrapolata grazie alla seguente istruzione:

```
meshdata = mphxmeshinfo(model);
```

questo è collegato al meccanismo con cui COMSOL gestisce le mesh. In COMSOL tutte le mesh al momento della creazione sono del primo ordine, quando poi viene aggiunta una fisica, aggiunto uno studio e calcolata una soluzione, in base al tipo di discretizzazione impostato nella sezione della fisica la mesh viene modificata per adattarsi alle necessità della fisica. La funzione *mphmeshstats* restituisce statistiche sulla mesh e informazioni sui dati della mesh prima di qualsiasi elaborazione. Mentre la funzione *mphxmeshinfo* restituisce le informazioni riguardo alla cosiddetta “mesh estesa” ovvero la mesh modificata per adattarsi alla fisica.

Tale funzione si occupa inoltre di estrarre la matrice NODI-ELEMENTI tramite il seguente snippet di codice:

```
meshdataTypePos = strcmp(meshdataTypeList, searchedString);
%N.B.: Come da documentazione gli elementi sono indicizzati da
%      0 quindi bisogna aggiungere 1
if elementsOrder == 2
    try
        if strcmp(searchedString, 'tet')
            elements = double(meshdata.
                               elements(geometryTagPos).
                               tet.nodes+1);
        elseif strcmp(searchedString, 'pyr')
            elements = double(meshdata.
                               elements(geometryTagPos).
                               pyr.nodes+1);
        elseif strcmp(searchedString, 'prism')
            elements = double(meshdata.
                               elements(geometryTagPos).
                               prism.nodes+1);
        elseif strcmp(searchedString, 'hex')
            elements = double(meshdata.
                               elements(geometryTagPos).
                               hex.nodes+1);
        end
    catch
        incidenceMatrices = struct();
        return;
    end
else
    elements = double(meshdata.elem{meshdataTypePos}+1);
end
transposedMatrixElements = elements';
```

Anche in questo caso la funzione si comporta in maniera differente in relazione all'ordine degli elementi della mesh. Inoltre come inserito nel commento alle matrici viene sommato elemento per elemento un 1 poiché i nodi che compongono gli elementi in COMSOL sono indicizzati su base 0.

Estrapolate le precedenti due matrici grazie alla API del LiveLink e grazie ad alcune leggere elaborazioni, la funzione si occupa di generare tutte le altre matrici, poiché nessun'altra matrice è estrapolabile. Tutte le altre matrici vengono create sulla base delle informazioni precedentemente recuperate e sulla base delle convenzioni di numerazione degli elementi mesh adottate da COMSOL. Per fare ciò, la funzione *createIncidenceMatricesForCLI* chiama opportunamente una serie di sei altre funzioni che, come è possibile osservare nell'immagine sottostante:

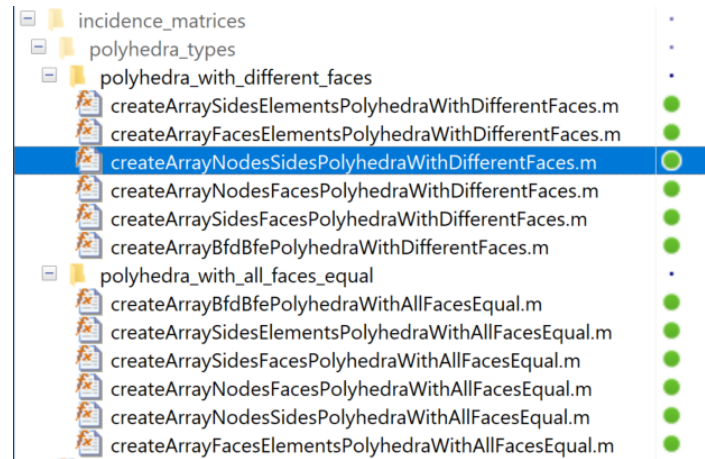


Figura 5: le due famiglie di funzioni di creazione delle matrici

sono divise in due famiglie, ci sono sei funzioni che si occupano della creazione delle altre nove matrici di incidenza nel caso in cui la mesh sia composta di elementi che sono poliedri con tutte le facce uguali, ovvero tetraedro ed esaedro; inoltre ci sono poi le funzioni equivalenti per mesh composte da poliedri aventi facce differenti, come il prisma e la piramide. Questa distinzione è dovuta alla gestione differente che avviene dei nodi che compongono facce e lati nei due casi. Analizzando a scopo di esempio il caso in cui l'utente stia cercando di ottenere informazioni sulle matrici di incidenza per un componente avente una mesh composta da esaedri, allora in questa situazione verrà chiamata anzitutto la funzione *createArrayNodesFacesPolyhedraWithAllFacesEqual* avente la seguente intestazione:

```
[arrayNodesFaces, arrayNodesBoundaryFaces]=
createArrayNodesFacesPolyhedraWithAllFacesEqual(
                                                    tableNodesElements,
                                                    elementType,
                                                    elementsOrder)
```

quest'ultima in base al numero di ordine degli elementi della mesh, e in base al tipo di elemento che compone la mesh costruirà la matrici NODI-FACCE seguendo le convenzioni di stile di COMSOL.

```
if elementsOrder == 2
    if strcmp(elementType, 'hex')
        faces = [
            nodes([1, 2, 3, 6, 9, 8, 7, 4, 5]);
            nodes([19, 20, 21, 24, 27, 26, 25, 22, 23]);
            nodes([7, 4, 1, 10, 19, 22, 25, 16, 13]);
            nodes([3, 6, 9, 18, 27, 24, 21, 12, 15]);
            nodes([9, 8, 7, 16, 25, 26, 27, 18, 17]);
            nodes([1, 2, 3, 12, 21, 20, 19, 10, 11]);
        ];
```

```

elseif strcmp(elementType, 'tet')
    faces = [
        nodes([6, 9, 10, 7, 1, 4]);
        nodes([6, 9, 10, 8, 3, 5]);
        nodes([6, 5, 3, 2, 1, 4]);
        nodes([10, 7, 1, 2, 3, 8]);
    ];
end
else

```

Successivamente nell'ordine verrà chiamata la funzione *createArrayNodesSidesPolyhedraWithAllFacesEqual* avente la seguente intestazione:

```

[arrayNodesSides] =
createArrayNodesSidesPolyhedraWithAllFacesEqual(
                                                    tableNodesFaces,
                                                    elementType,
                                                    elementsOrder)

```

quest'ultima in maniera del tutto simile alla precedente costruirà la matrice NODI-LATI seguendo sempre le convenzioni di stile di COMSOL.

In seguito la funzione *createArrayFacesElementsPolyhedraWithAllFacesEqual* viene chiamata, essa ha la seguente intestazione:

```

arrayFacesElements =
createArrayFacesElementsPolyhedraWithAllFacesEqual(
                                                    tableNodesElements,
                                                    tableNodesFaces,
                                                    elementType)

```

quest'ultima combinando in la matrice NODI-ELEMENTI con la matrice NODI-FACCE costruirà la matrice FACCE-ELEMENTI.

Poi la funzione *createArraySidesElementsPolyhedraWithAllFacesEqual* viene chiamata, essa ha la seguente intestazione:

```

arraySidesElements =
createArraySidesElementsPolyhedraWithAllFacesEqual(
                                                    tableNodesElements,
                                                    tableNodesSides,
                                                    elementType,
                                                    elementsOrder)

```

quest'ultima invece combinando in la matrice NODI-ELEMENTI con la matrice NODI-LATI costruirà la matrice LATI-ELEMENTI.

Poi la funzione *createArraySidesFacesPolyhedraWithAllFacesEqual* viene chiamata, essa ha la seguente intestazione:

```
arraySidesFaces =  
createArraySidesFacesPolyhedraWithAllFacesEqual(  
                                                    tableNodesFaces,  
                                                    tableNodesSides,  
                                                    elementType,  
                                                    elementsOrder)
```

quest'ultima invece combinando in la matrice NODI-FACCE con la matrice NODI-LATI costruirà la matrice LATI-FACCE.

La matrice DOMINI-ELEMENTI viene gestita direttamente nella funzione *createIncidenceMatricesForCLI* poiché essa nel caso di elementi del primo ordine è direttamente estrapolabile grazie alla API del LiveLink, mentre nel caso di elementi del secondo ordine il LiveLink attualmente non fornisce alcuna informazione.

```
if elementsOrder == 2  
    cprintf('Keywords', 'Unfortunately, the LiveLink  
                        does not \n');  
    cprintf('Keywords', 'currently provide information  
                        about the \n');  
    cprintf('Keywords', 'domain of membership for  
                        second-order elements, \n');  
    cprintf('Keywords', 'so this matrix cannot be  
                        generated. \n');  
else  
    arrayDomainsElements =  
                                meshdata.elementity{meshdata.typePos};  
end
```

Infine per la creazione dell'ultima matrice verrà chiamata la funzione *createArrayBfdBfePolyhedraWithAllFacesEqual*, la quale effettua una serie di calcoli piuttosto complessi e di conseguenza ha la seguente intestazione:

```
arrayBoundaryFacesDomainBoundaryFacesElement =  
createArrayBfdBfePolyhedraWithAllFacesEqual(  
                                                    model,  
                                                    tableNodesBoundaryFaces,  
                                                    tableNodalCoordinates,  
                                                    selectedComponentGeometryTag,  
                                                    numberOfBoundary,  
                                                    elementType,  
                                                    elementsOrder)
```


Quest'ultimo fornisce una serie di informazioni generali sull'applicativo, tra cui il numero di versione e una descrizione del suo possibile utilizzo.

MATSOL è pesantemente basato sul COMSOL LiveLink for MATLAB, non può funzionare in assenza di esso, infatti il secondo passo è proprio un tentativo di stabilire una connessione con il server COMSOL come è possibile osservare nell'immagine sottostante:

```
=====
Please wait while the connection to the Comsol Server is checked...
Connection successfully established!
=====
```

Figura 7: tentativo connessione con server COMSOL

Nel caso in cui il tentativo dia esito negativo la CLI avvisa l'utente dell'accaduto, e termina.

Il primo passo che richiede una interazione dell'utente è il terzo ovvero la selezione del modello. In quest'ultimo verrà aperta una finestra del File Explorer che permetterà all'utente di selezionare in maniera comoda il modello di interesse.

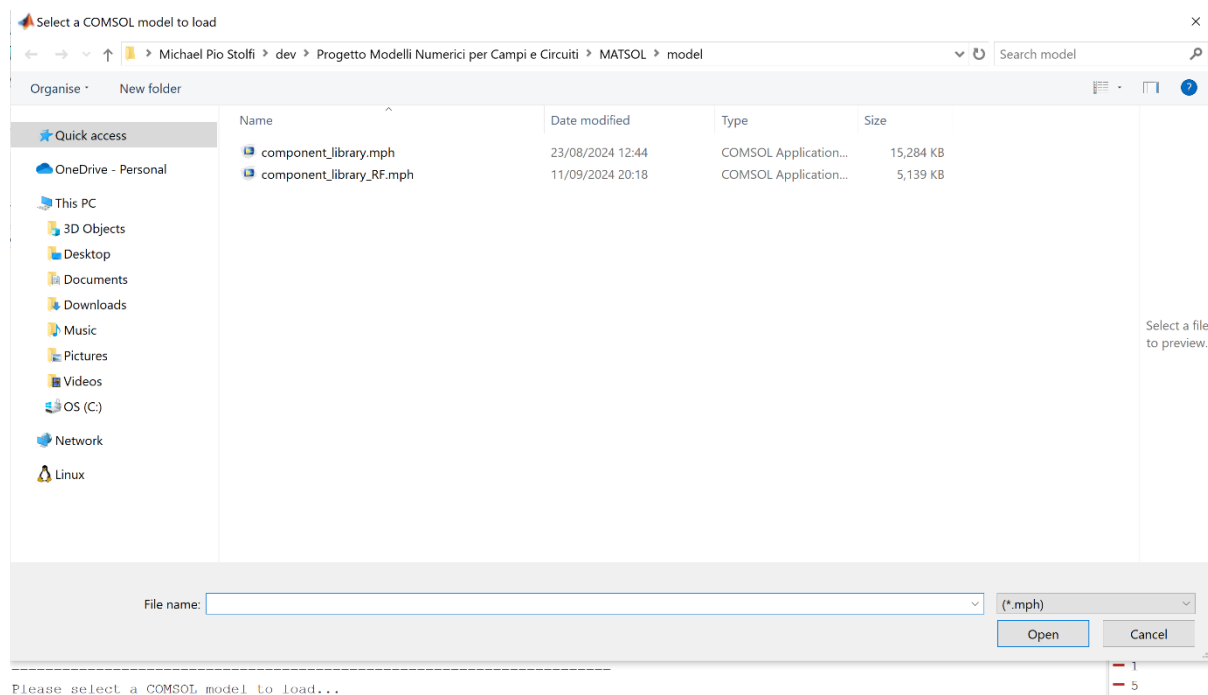


Figura 8: selezione modello

Nel caso in cui l'utente chiuda la finestra senza selezionare alcun modello, oppure selezioni un file con formato diverso dal “.mph”, oppure il modello selezionato sia corrotto allora la CLI avvisa l'utente dell'accaduto, e termina.

Il passo successivo è la selezione del componente, MATSOL scansiona il modello selezionato, preleva tutti i componenti e permette all'utente di selezionarne uno di interesse come illustrato nell'immagine sottostante:

```
=====
Please select the component of interest from those available:

1) componentCube
2) componentCube2
3) componentCube3
4) componentCylinder

Selection-> |
```

Figura 9: selezione del componente

Nel caso in cui il modello non possenga ancora alcun componente, la CLI avvisa l'utente dell'accaduto, e termina.

Il seguente passo è la selezione della mesh, MATSOL scansiona il componente selezionato, preleva tutte le mesh associate ad esso, e permette all'utente di selezionarne una di interesse come illustrato nell'immagine sottostante:

```
=====
Please select the mesh of interest from those available:

1) meshHex1
2) meshHex2
3) meshHex3
4) meshPrism
5) meshTet

Selection-> |
```

Figura 10: selezione della mesh

Nel caso in cui il componente non possenga ancora alcuna mesh, la CLI avvisa l'utente dell'accaduto, e termina.

Il seguente passo è la selezione dello studio, MATSOL scansiona il modello selezionato, preleva tutti gli studi presenti, e permette all'utente di selezionarne uno di interesse come illustrato nell'immagine sottostante:

```

=====
Please select the study of interest from those available:

1) std1
2) std2
3) std3
4) std4
5) std5
6) std6
7) std7
8) std8
9) std9
10) std10
11) std11

Selection->

```

Figura 11: selezione dello studio

Nel caso in cui il componente non possenga ancora alcuno studio, la CLI avvisa l'utente dell'accaduto, e termina.

Il seguente passo è la selezione dello step, MATSOL scansiona lo studio selezionato, preleva tutte gli step presenti, e permette all'utente di selezionarne uno di interesse come illustrato nell'immagine sottostante:

```

=====
Please select the step of interest from those available:

1) eig
2) timod

Selection-> |

```

Figura 12: selezione dello step

Nel caso in cui lo studio non possenga ancora alcuno step, la CLI avvisa l'utente dell'accaduto, e termina.

Successivamente avvengono una serie di controlli sui dati appena inseriti, se ad esempio nello step selezionato per il componente selezionato non si è inserita alcuna mesh, allora la cosa viene notificata all'utente e l'applicazione termina. Se invece nello step seleziona per il componente selezionato si è inserita una mesh differente da quella inserita precedentemente allora la cosa viene notificata all'utente e l'applicazione termina. Poi viene verificato se per il componente selezionato sono presenti le funzioni di forma, in caso della loro assenza viene suggerito all'utente di impostare una fisica per il componente di interesse e l'applicazione termina. Poi viene verificato se è presente una soluzione per il modello selezionato in caso della sua assenza ancora una volta la cosa viene notificata all'utente e l'applicazione termina. Se invece una soluzione attiva è

presente ma è allegata a uno studio differente da quello selezionato la cosa viene notificata all'utente e l'applicazione termina.

In seguito a tutte le verifiche precedenti avviene la valutazione del numero di ordine degli elementi della mesh selezionata, passo cruciale per tutte le valutazioni che avvengono nelle funzioni come visto nel sottoparagrafo precedente.

```
=====
Please wait while the mesh element order number is evaluated...
Evaluation completed!
=====
```

Figura 13: valutazione del numero d'ordine degli elementi della mesh

Dopo tutta la serie di inserimenti precedenti, in cui si sono delineati gli oggetti di interesse per l'utente, avviene la selezione del tipo di dato che si vuole far estrarre/generare da MATSOL. Quindi viene mostrato un menù di selezione dei servizi di estrazione/generazione attualmente disponibili in MATSOL come illustrato nell'immagine sottostante:

TODO: AGGIORNARE!

```
=====
Please select the data you want to extract/generate from the model:

    1) Incidence Matrices

Selection->
```

Figura 14: selezione del servizio di estrazione/generazione

Una volta selezionato il servizio di interesse, la CLI entrerà nella selezione interna al servizio, comportandosi in maniera differente da servizio a servizio.

Nel caso in cui si sia selezionato il servizio di estrazione/generazione delle matrici di incidenza allora verrà mostrato all'utente un ulteriore menù di selezione in cui potrà inserire a quale matrice di incidenza è interessato tra quelle presenti.

```
=====
Please select which matrix you want to extract/generate:

    1) NODES-FACES
    2) NODES-BOUNDARY_FACES
    3) NODES-SIDES
    4) FACES-ELEMENTS
    5) SIDES-ELEMENTS
    6) SIDES-FACES
    7) SIDES-BOUNDARY_FACES
    8) DOMAINS-ELEMENTS
    9) BOUNDARY_FACES_DOMAINS-BOUNDARY_FACES_ELEMENTS
   10) ALL

Selection-> |
```

Figura 15: selezione matrice

Si noti il fatto che sono esenti da selezione la matrice di COORDINATE NODALI e la matrice NODI-ELEMENTI, il motivo è legato al fatto che tali matrici, insieme alla matrice DOMINI-ELEMENTI, sono le uniche totalmente estratte tramite il LiveLink, che subiscono leggere elaborazioni e che sono fondamentali alla generazione di tutte le altre matrici. Conseguentemente tali matrici vengono sempre estratte e saranno presenti sempre nella struttura dati di output qualsiasi matrice sia selezionata dall'utente.

Si noti inoltre come vi sia la possibilità di generare tutte le matrici, questa scelta è sconsigliata nel caso di mesh con molti elementi poiché può portare a tempi di elaborazione complessivi molto lunghi.

Infine la CLI chiede all'utente un'ultima selezione che riguarda la scelta di salvare o non salvare anche su disco la struttura dati contenente le matrici precedentemente selezionate.

```
=====
Wants the matrix to be saved on disk too:
```

```
1) YES
```

```
2) NO
```

```
Selection-> |
```

Figura 16: scelta di salvataggio

Se viene scelto di salvare su disco allora, come detto precedentemente, il risultato del lavoro di MATSOL verrà salvato, oltre che nel workspace base di MATLAB, anche nella cartella predefinita di progetto *saved_matrices* in un file formato json leggibile dall'uomo. Tale scelta è altamente consigliata nel caso di elaborazioni lunghe e complesse poiché permette di effettuare i calcoli una volta per poi poterli sfruttare successivamente senza alcuna preoccupazione di preservare le variabili nel workspace.

Qualsiasi sia la matrice selezionata, la CLI di MATSOL analizza il tipo di elementi presenti nella mesh selezionata e avvia una o più esecuzioni della funzione *createIncidenceMatricesForCLI*, salvando poi opportunamente una o più strutture dati contenenti le suddette matrici.

Infine la CLI pulisce il workspace base di MATLAB, preservando solo le variabili di interesse, e termina.

Capitolo 3

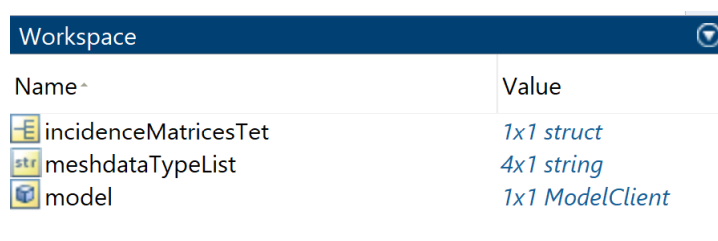
Risultati

In questo capitolo verranno presentati i risultati di una esecuzione di MATSOL. Una esecuzione di MATSOL restituisce due serie di risultati: i risultati fissi ovvero quei risultati che non cambiano mai e sono presenti a ogni esecuzione, e i risultati variabili ovvero quei risultati che cambiano in base a quale servizio di estrazione/generazione si è deciso di eseguire.

Indipendentemente dal servizio eseguito vi sarà sempre la possibilità di salvare i risultati dell'elaborazione su disco, questa feature è stata pensata nell'ottica di rendere MATSOL robusto a elaborazioni lunghe e di renderlo facilmente integrabile in altri progetti MATLAB.

3.1 Presentazione dei risultati

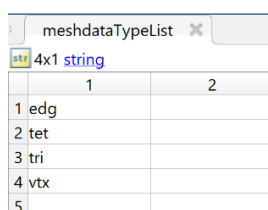
Tra i risultati fissi vi sono quei risultati che non cambiano mai e sono presenti a ogni esecuzione, questo perché possono risultare sempre utili a un utente che vuole interfacciarsi con un modello COMSOL.



Name	Value
incidenceMatricesTet	1x1 struct
meshdataTypeList	4x1 string
model	1x1 ModelClient

Figura 17: esempio di workspace base di MATLAB dopo una esecuzione di MATSOL

I risultati fissi attualmente presenti in output a una esecuzione di MATSOL vi sono *meshdataTypeList* e *model*. Il primo dei due ovvero *meshdataTypeList* è un array di stringhe contenenti i tag degli elementi che compongono la mesh attualmente in esame e che quindi sono estrapolabili dalla matrice *meshdata*. Vi sono tra essi tag che fanno riferimento a elementi monodimensionali come “vtx”, oppure bidimensionali come “edg” e “tri” oppure tridimensionali come “tet”.



	1	2
1	edg	
2	tet	
3	tri	
4	vtx	
5		

Figura 18: esempio di stringhe contenute nell'array *meshdataTypeList*

Il secondo risultato è invece *model* il quale è in sostanza il riferimento a un oggetto di tipo ModelClient molto utile se si vuole ad esempio continuare a navigare l'albero dei nodi che compone il modello in esame,²³ alla ricerca di altri elementi di interesse.

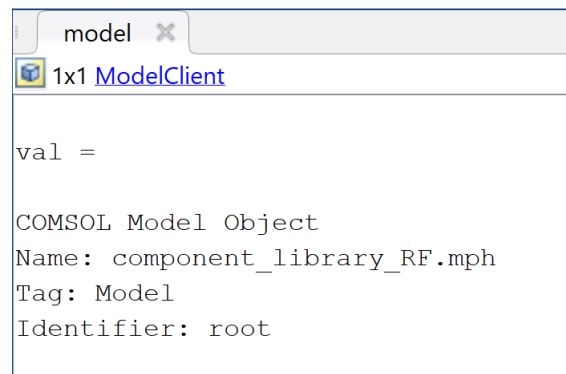


Figura 19: esempio di riferimento a un modello selezionato

Tra i risultati variabili invece ci sono quei risultati che cambiano in base a quale servizio di estrazione/generazione si è deciso di eseguire.

Se si è deciso di eseguire il servizio di estrazione/generazione delle matrici di incidenza allora nel workspace di MATLAB sarà presente una o più strutture dati contenenti tutti gli array delle matrici di incidenza richieste per i vari elementi di mesh che compongono la mesh selezionata. Il nome di tali strutture dati può essere uno tra i seguenti: “incidenceMatricesTet”, “incidenceMatricesPyr”, “incidenceMatricesPrism” e “incidenceMatricesHex”. Ovviamente la loro dimensione come detto cambia in base a quali matrici di incidenza si è chiesto di generare a MATSOL e si presentano come da immagine sottostante:

incidenceMatricesTet	
1x1 struct with 10 fields	
Field	Value
arrayNodalCoordinates	63x3 double
arrayNodesElements	24x10 double
arrayNodesFaces	60x6 double
arrayNodesBoundaryFaces	24x6 double
arrayNodesSides	98x2 double
arrayFacesElements	24x4 double
arraySidesElements	24x12 double
arraySidesFaces	60x6 double
arraySidesBoundaryFaces	24x6 double
arrayBoundaryFacesDomainBoundaryFacesElement	24x1 double

Figura 20: esempio di struct incidenceMatricesTet

Capitolo 4

Documentazione delle funzioni

In questo capitolo verrà fornita una panoramica documentativa delle varie funzioni che compongono MATSOL. Verranno trattate per compito, ovvero area di interesse, al momento vi sono tre aree di interesse: cli, incidence_matrices e utility.

Nella prima area vi sono tutte le funzioni che permettono il funzionamento della CLI, quindi sono principalmente funzioni di prelevamento e convalida dell'input dall'utente e funzioni di verifica e valutazione di alcuni parametri.

Nella seconda area vi sono tutte le funzioni che permettono tutte le elaborazioni necessarie alla creazione delle matrici di incidenza, che è uno dei servizi offerti da MATSOL. Sono principalmente funzioni di interrogazione del server COMSOL e elaborazione vettorizzata di matrici.

Infine nella terza area vi sono tutte le funzioni di utilità utilizzate in tutte le altre aree. Sono funzioni assortite che fanno calcoli matriciali e vettoriali, stampa su schermo, salvataggio e caricamento da file.

4.1. Riepilogo delle funzioni

[checkConnection](#)

[checkFaceInDomain](#)

[componentPicker](#)

[computePlaneEquationFromPoints](#)

[createArrayBfdBfePolyhedraWithAllFacesEqual](#)

[createArrayBfdBfePolyhedraWithDifferentFaces](#)

[createArrayFacesElementsPolyhedraWithAllFacesEqual](#)

[createArrayFacesElementsPolyhedraWithDifferentFaces](#)

[createArrayNodesFacesPolyhedraWithAllFacesEqual](#)

[createArrayNodesFacesPolyhedraWithDifferentFaces](#)

[createArrayNodesSidesPolyhedraWithAllFacesEqual](#)

[createArrayNodesSidesPolyhedraWithDifferentFaces](#)

[createArraySidesElementsPolyhedraWithAllFacesEqual](#)

[createArraySidesElementsPolyhedraWithDifferentFaces](#)

[createArraySidesFacesPolyhedraWithAllFacesEqual](#)

[createArraySidesFacesPolyhedraWithDifferentFaces](#)

[createIncidenceMatricesForCLI](#)

[evaluateOrderNumber](#)

[loadFromJson](#)

[meshPicker](#)

[modelPicker](#)

[printSplashScreen](#)

[projectToPlane](#)

[saveToJson](#)

[sortPolygonVertices](#)

[stepPicker](#)

[studyPicker](#)

[validateInput](#)

4.2. Funzioni raggruppate per compito

Funzioni della CLI

FUNZIONE	SCOPO
checkConnection	Verificare connessione server COMSOL
componentPicker	Selezione componente
createIncidenceMatricesForCLI	Creazione matrici di incidenza
evaluateOrderNumber	Valutare ordine elementi mesh
meshPicker	Selezionare mesh
modelPicker	Selezionare modello
printSplashScreen	Stampare splash screen
stepPicker	Selezionare step
studyPicker	Selezionare studio
validateInput	Validare input utente

Funzioni matrici di incidenza

FUNZIONE	SCOPO
createArrayBfdBfePolyhedraWithAllFacesEqual	Mat. FD-FE
createArrayFacesElementsPolyhedraWithAllFacesEqual	Mat. F-E
createArrayNodesFacesPolyhedraWithAllFacesEqual	Mat. N-F
createArrayNodesSidesPolyhedraWithAllFacesEqual	Mat N-S
createArraySidesElementsPolyhedraWithAllFacesEqual	Mat. S-E
createArraySidesFacesPolyhedraWithAllFacesEqual	Mat. S-F
createArrayBfdBfePolyhedraWithDifferentFaces	Mat. FD-FE
createArrayFacesElementsPolyhedraWithDifferentFaces	Mat. F-E
createArrayNodesFacesPolyhedraWithDifferentFaces	Mat. N-F
createArrayNodesSidesPolyhedraWithDifferentFaces	Mat N-S
createArraySidesElementsPolyhedraWithDifferentFaces	Mat. S-E
createArraySidesFacesPolyhedraWithDifferentFaces	Mat. S-F

Funzioni utilità

FUNZIONE	SCOPO
checkFaceInDomain	Verificare faccia mesh in dominio
computePlaneEquationFromPoints	Calcolare equazione piano dati tre punti
loadFromJson	Caricare da un file Json
projectToPlane	Proiettare i punti su un piano
saveToJson	Salvare su un file Json
sortPolygonVertices	Ordinare vertici poligono verso antiorar.

4.3. Documentazione

`checkConnection`

Descrizione

Questa funzione si occupa di effettuare un tentativo di connessione con un server COMSOL.

Sintassi

```
isConnected = checkConnection()
```

Parametri di Input

Nessun parametro di input

Parametri di Output

PARAMETRO	TIPO	DESCRIZIONE
<code>isConnected</code>	boolean	Valore booleano che indica se l'istanza MATLAB è collegata a una istanza del server COMSOL tramite il LiveLink

`checkFaceInDomain`

Descrizione

Questa funzione si occupa di verificare se una faccia di un elemento di una mesh è compresa in una faccia del dominio

Sintassi

```
in_domain = checkFaceInDomain(domain_face, mesh_face)
```

Parametri di Input

PARAMETRO	TIPO	DESCRIZIONE
<code>domain_face</code>	Nx3 matrix of double	Matrice Nx3 dove ogni riga è un vertice del dominio [x, y, z]
<code>mesh_face</code>	Nx3 matrix of double	Matrice Nx3, dove ogni riga è un vertice dell'elemento della mesh [x, y, z]

Parametri di Output

PARAMETRO	TIPO	DESCRIZIONE
<code>in_domain</code>	boolean	Valore booleano che indica se la faccia della mesh è all'interno della faccia del dominio

`componentPicker`

Descrizione

Questa funzione si occupa di permettere all'utente la scelta del componente di interesse, prelevando i componenti disponibili nel modello e mostrandoli a schermo prima della scelta.

Sintassi

```
selectedComponent = componentPicker(model)
```

Parametri di Input

PARAMETRO	TIPO	DESCRIZIONE
<code>model</code>	ModelClient Obj	Oggetto di tipo ModelClient risultato della chiamata <i>mphload</i> sul percorso di un modello COMSOL

Parametri di Output

PARAMETRO	TIPO	DESCRIZIONE
<code>selectedComponent</code>	ModelNodeClient Obj	Oggetto di tipo ModelNodeClient utile per l'estrazione di geometria e mesh

`computePlaneEquationFromPoints`

Descrizione

Questa funzione si occupa di calcolare l'equazione del piano passante per tre punti non allineati.

Sintassi

```
plane_eq = computePlaneEquationFromPoints(p1, p2, p3)
```

Parametri di Input

PARAMETRO	TIPO	DESCRIZIONE
<code>p1, p2, p3</code>	1x3 matrix of double	Punto dello spazio rappresentato come un vettore di coordinate [x, y, z]

Parametri di Output

PARAMETRO	TIPO	DESCRIZIONE
<code>plane_eq</code>	1x4 matrix of double	Coefficienti dell'equazione del piano passante per i tre punti specificati

Descrizione

Questa funzione si occupa di generare la matrice **FACCE_FRONTIERA_DOMINIO-FACCE_FRONTIERA_ELEMENTO** per elementi di mesh che sono poliedri con tutte le facce uguali (tetraedro, esaedro).

Sintassi

```
arrayBoundaryFacesDomainBoundaryFacesElement =  
createArrayBfdBfePolyhedraWithAllFacesEqual(  
    model,  
    tableNodesBoundaryFaces,  
    tableNodalCoordinates,  
    selectedComponentGeometryTag,  
    numberOfBoundary,  
    elementType,  
    elementsOrder)
```

Parametri di Input

PARAMETRO	TIPO	DESCRIZIONE
model	ModelClient Obj	Oggetto di tipo ModelClient risultato della chiamata <i>mphload</i> sul percorso di un modello COMSOL
tableNodesBoundaryFaces	NxM matrix of double	Matrice NODI-FACCE_FRONTIERA
tableNodalCoordinates	NxM matrix of double	Matrice COORDINATE NODALI
selectedComponentGeometryTag	string	Stringa contenente il tag della geometria del componente
numberOfBoundary	integer	Numero dei domini che compongono la geometria del componente
elementType	string	Stringa contenente il tipo di elemento che compone la mesh
elementsOrder	integer	Numero di ordine del tipo di elemento che compone la mesh

Parametri di Output

PARAMETRO	TIPO	DESCRIZIONE
arrayBoundaryFacesDomainBoundaryFacesElement	NxM matrix of double	Matrice NxM BFD-BFE

createArrayBfdBfePolyhedraWithDifferentFaces

Descrizione

Questa funzione si occupa di generare la matrice **FACCE_FRONTIERA_DOMINIO-FACCE_FRONTIERA_ELEMENTO** per elementi di mesh che sono poliedri con facce differenti tra loro (prisma, piramide).

Sintassi

```
arrayBoundaryFacesDomainBoundaryFacesElement =
createArrayBfdBfePolyhedraWithDifferentFaces (
    model,
    tableNodesBoundaryFaces,
    tableNodalCoordinates,
    selectedComponentGeometryTag,
    numberOfBoundary,
    elementsOrder)
```

Parametri di Input

PARAMETRO	TIPO	DESCRIZIONE
model	ModelClient Obj	Oggetto di tipo ModelClient risultato della chiamata <i>mphload</i> sul percorso di un modello COMSOL
tableNodesBoundaryFaces	NxM matrix of double	Matrice NODI-FACCE_FRONTIERA
tableNodalCoordinates	NxM matrix of double	Matrice COORDINATE NODALI

<code>selectedComponentGeometryTag</code>	string	Stringa contenente il tag della geometria del componente
<code>numberOfBoundary</code>	integer	Numero dei domini che compongono la geometria del componente
<code>elementsOrder</code>	integer	Numero di ordine del tipo di elemento che compone la mesh

Parametri di Output

PARAMETRO	TIPO	DESCRIZIONE
<code>arrayBoundaryFacesDomainBoundaryFacesElement</code>	NxM matrix of double	Matrice NxM BFD-BFE

`createArrayFacesElementsPolyhedraWithAllFacesEqual`

Descrizione

Questa funzione si occupa di creare la matrice FACCE-ELEMENTI per elementi di mesh che sono poliedri con tutte le facce uguali (tetraedro, esaedro).

Sintassi

```
arrayFacesElements =
createArrayFacesElementsPolyhedraWithAllFacesEqual(
    tableNodesElements,
    tableNodesFaces,
    elementType)
```

Parametri di Input

PARAMETRO	TIPO	DESCRIZIONE
<code>tableNodesElements</code>	NxM matrix of double	Matrice NODI-ELEMENTI
<code>tableNodesFaces</code>	NxM matrix of double	Matrice NODI-FACCE
<code>elementType</code>	string	Stringa contenente il tipo di elemento che compone la mesh

Parametri di Output

PARAMETRO	TIPO	DESCRIZIONE
arrayFacesElements	NxM matrix of double	Matrice FACCE-ELEMENTI

`createArrayFacesElementsPolyhedraWithDifferentFaces`

Descrizione

Questa funzione si occupa di creare la matrice FACCE-ELEMENTI per elementi di mesh che sono poliedri con facce differenti tra loro (prisma, piramide).

Sintassi

```
arrayFacesElements =  
createArrayFacesElementsPolyhedraWithDifferentFaces(  
    tableNodesElements,  
    tableNodesFaces,  
    elementsOrder)
```

Parametri di Input

PARAMETRO	TIPO	DESCRIZIONE
tableNodesElements	NxM matrix of double	Matrice NODI-ELEMENTI
tableNodesFaces	NxM matrix of double	Matrice NODI-FACCE
elementsOrder	integer	Numero di ordine del tipo di elemento che compone la mesh

Parametri di Output

PARAMETRO	TIPO	DESCRIZIONE
arrayFacesElements	NxM matrix of double	Matrice FACCE-ELEMENTI

`createArrayNodesFacesPolyhedraWithAllFacesEqual`

Descrizione

Questa funzione si occupa di creare la matrice NODI-FACCE (sia per tutte le facce, che per le sole facce di frontiera) per tutti gli elementi di mesh che sono poliedri con tutte le facce uguali (tetraedro, esaedro).

Sintassi

```
[arrayNodesFaces, arrayNodesBoundaryFaces] =  
createArrayNodesFacesPolyhedraWithAllFacesEqual(  
    tableNodesElements,  
    elementType,  
    elementsOrder)
```

Parametri di Input

PARAMETRO	TIPO	DESCRIZIONE
<code>tableNodesElements</code>	NxM matrix of double	Matrice NODI-ELEMENTI
<code>elementType</code>	string	Stringa contenente il tipo di elemento che compone la mesh
<code>elementsOrder</code>	integer	Numero di ordine del tipo di elemento che compone la mesh

Parametri di Output

PARAMETRO	TIPO	DESCRIZIONE
<code>arrayNodesFaces</code>	NxM matrix of double	Matrice NODI-FACCE
<code>arrayNodesBoundaryFaces</code>	NxM matrix of double	Matrice NODI-FACCE_FRONTIERA

`createArrayNodesFacesPolyhedraWithDifferentFaces`

Descrizione

Questa funzione si occupa di creare la matrice NODI-FACCE (sia per tutte le facce, che per le sole facce di frontiera) per tutti gli elementi di mesh che sono poliedri con facce differenti tra loro (prisma, piramide).

Sintassi

```
[arrayNodesFaces, arrayNodesBoundaryFaces] =  
createArrayNodesFacesPolyhedraWithDifferentFaces (  
                                                    tableNodesElements,  
                                                    elementType,  
                                                    elementsOrder)
```

Parametri di Input

PARAMETRO	TIPO	DESCRIZIONE
tableNodesElements	NxM matrix of double	Matrice NODI-ELEMENTI
elementType	string	Stringa contenente il tipo di elemento che compone la mesh
elementsOrder	integer	Numero di ordine del tipo di elemento che compone la mesh

Parametri di Output

PARAMETRO	TIPO	DESCRIZIONE
arrayNodesFaces	NxM matrix of double	Matrice NODI-FACCE
arrayNodesBoundaryFaces	NxM matrix of double	Matrice NODI-FACCE_FRONTIERA

```
createArrayNodesSidesPolyhedraWithAllFacesEqual
```

Descrizione

Questa funzione si occupa di creare la matrice NODI-LATI per tutte le facce, per elementi di mesh che sono poliedri con tutte le facce uguali (tetraedro, esaedro).

Sintassi

```
arrayNodesSides =  
createArrayNodesSidesPolyhedraWithAllFacesEqual (  
                                                    tableNodesFaces,  
                                                    elementType,  
                                                    elementsOrder)
```

Parametri di Input

PARAMETRO	TIPO	DESCRIZIONE
tableNodesFaces	NxM matrix of double	Matrice NODI-FACCE
elementType	string	Stringa contenente il tipo di elemento che compone la mesh
elementsOrder	integer	Numero di ordine del tipo di elemento che compone la mesh

Parametri di Output

PARAMETRO	TIPO	DESCRIZIONE
arrayNodesSides	NxM matrix of double	Matrice NODI-LATI

`createArrayNodesSidesPolyhedraWithDifferentFaces`

Descrizione

Questa funzione si occupa di creare la matrice NODI-LATI per tutte le facce, per elementi di mesh che sono poliedri con facce differenti tra loro (prisma, piramide).

Sintassi

```
arrayNodesSides =  
createArrayNodesSidesPolyhedraWithDifferentFaces (  
                                                    tableNodesFaces,  
                                                    elementsOrder)
```

Parametri di Input

PARAMETRO	TIPO	DESCRIZIONE
tableNodesFaces	NxM matrix of double	Matrice NODI-FACCE
elementsOrder	integer	Numero di ordine del tipo di elemento che compone la mesh

Parametri di Output

PARAMETRO	TIPO	DESCRIZIONE
arrayNodesSides	NxM matrix of double	Matrice NODI-LATI

`createArraySidesElementsPolyhedraWithAllFacesEqual`

Descrizione

Questa funzione si occupa di creare la matrice LATI-ELEMENTI, per elementi di mesh che sono poliedri con tutte le facce uguali (tetraedro, esaedro).

Sintassi

```
arraySidesElements =  
createArraySidesElementsPolyhedraWithAllFacesEqual(  
                                                    tableNodesElements,  
                                                    tableNodesSides,  
                                                    elementType,  
                                                    elementsOrder)
```

Parametri di Input

PARAMETRO	TIPO	DESCRIZIONE
tableNodesElements	NxM matrix of double	Matrice NODI-ELEMENTI
tableNodesSides	NxM matrix of double	Matrice NODI-LATI
elementType	string	Stringa contenente il tipo di elemento che compone la mesh
elementsOrder	integer	Numero di ordine del tipo di elemento che compone la mesh

Parametri di Output

PARAMETRO	TIPO	DESCRIZIONE
arraySidesElements	NxM matrix of double	Matrice LATI-ELEMENTI

`createArraySidesElementsPolyhedraWithDifferentFaces`

Descrizione

Questa funzione si occupa di creare la matrice LATI-ELEMENTI per elementi di mesh che sono poliedri con facce differenti tra loro (prisma, piramide).

Sintassi

```
arraySidesElements =  
createArraySidesElementsPolyhedraWithDifferentFaces (  
                                                    tableNodesElements,  
                                                    tableNodesSides,  
                                                    elementType,  
                                                    elementsOrder)
```

Parametri di Input

PARAMETRO	TIPO	DESCRIZIONE
tableNodesElements	NxM matrix of double	Matrice NODI-ELEMENTI
tableNodesSides	NxM matrix of double	Matrice NODI-LATI
elementType	string	Stringa contenente il tipo di elemento che compone la mesh
elementsOrder	integer	Numero di ordine del tipo di elemento che compone la mesh

Parametri di Output

PARAMETRO	TIPO	DESCRIZIONE
arraySidesElements	NxM matrix of double	Matrice LATI-ELEMENTI

```
createArraySidesFacesPolyhedraWithAllFacesEqual
```

Descrizione

Questa funzione si occupa di creare la matrice LATI-FACCE, per elementi di mesh che sono poliedri con tutte le facce uguali (tetraedro, esaedro).

Sintassi

```
arraySidesFaces =  
createArraySidesFacesPolyhedraWithAllFacesEqual (  
                                                    tableNodesFaces,  
                                                    tableNodesSides,  
                                                    elementType,  
                                                    elementsOrder)
```

Parametri di Input

PARAMETRO	TIPO	DESCRIZIONE
tableNodesFaces	NxM matrix of double	Matrice NODI-FACCE
tableNodesSides	NxM matrix of double	Matrice NODI-LATI
elementType	string	Stringa contenente il tipo di elemento che compone la mesh
elementsOrder	integer	Numero di ordine del tipo di elemento che compone la mesh

Parametri di Output

PARAMETRO	TIPO	DESCRIZIONE
arraySidesFaces	NxM matrix of double	Matrice LATI-FACCE

`createArraySidesFacesPolyhedraWithDifferentFaces`

Descrizione

Questa funzione si occupa di creare la matrice LATI-FACCE, per elementi di mesh che sono poliedri con facce differenti tra loro (prisma, piramide).

Sintassi

```
arraySidesFaces =  
createArraySidesFacesPolyhedraWithDifferentFaces (  
                                                    tableNodesFaces,  
                                                    tableNodesSides,  
                                                    elementsOrder)
```

Parametri di Input

PARAMETRO	TIPO	DESCRIZIONE
tableNodesFaces	NxM matrix of double	Matrice NODI-FACCE
tableNodesSides	NxM matrix of double	Matrice NODI-LATI
elementsOrder	integer	Numero di ordine del tipo di elemento che compone la mesh

Parametri di Output

PARAMETRO	TIPO	DESCRIZIONE
arraySidesFaces	NxM matrix of double	Matrice LATI-FACCE

Descrizione

Questa funzione si occupa di calcolare le matrici di incidenza selezionate per la mesh specificata del modello scelto.

Sintassi

```
incidenceMatrices = createIncidenceMatricesForCLI(  
    model,  
    selectedComponentGeometryTag,  
    geometryTagPos,  
    meshdata,  
    meshdataTypeList,  
    searchedString,  
    elementsOrder,  
    flagsStruct,  
    fileName,  
    flagFacesEqual)
```

Parametri di Input

PARAMETRO	TIPO	DESCRIZIONE
model	ModelClient Obj	Oggetto di tipo ModelClient risultato della chiamata <i>mphload</i> sul percorso di un modello COMSOL
selectedComponentGeometryTag	string	Stringa contenente il tag della geometria del componente
geometryTagPos	integer	Intero che indica la posizione del tag della geometria di interesse all'interno della lista dei tag di tutte le geometrie del modello
meshdata	struct	Struttura contenente le informazioni relative alla mesh
meshdataTypeList	array of string	Lista dei tipi di elemento di cui vi sono informazioni

		nella struttura meshdata
searchedString	string	Stringa contenente il tipo di elemento di mesh contenuto nella mesh selezionata
elementsOrder	integer	Numero di ordine del tipo di elemento che compone la mesh
flagsStruct	struct	Struttura contenente due valori booleani per ciascuna matrice, utile per pilotare la costruzione delle matrici e il loro salvataggio
fileName	char	Lista di caratteri contenente il percorso e il nome del file in cui salvare sul disco la struct
flagFacesEqual	boolean	Valore booleano che indica la presenza di elementi di mesh con facce tutte uguali, utile per permettere di chiamare la funzione di creazione della famiglia corretta

Parametri di Output

PARAMETRO	TIPO	DESCRIZIONE
incidenceMatrices	struct	Struttura dati contenente tutti gli array delle matrici di incidenza

Descrizione

Funzione che si occupa di valutare il numero di ordine degli elementi di mesh della mesh selezionata.

Sintassi

```
elementsOrder = evaluateOrderNumber(model)
```

Parametri di Input

PARAMETRO	TIPO	DESCRIZIONE
model	ModelClient Obj	Oggetto di tipo ModelClient risultato della chiamata <i>mphload</i> sul percorso di un modello COMSOL

Parametri di Output

PARAMETRO	TIPO	DESCRIZIONE
elementsOrder	integer	Numero di ordine del tipo di elemento che compone la mesh

Descrizione

Questa funzione si occupa di caricare nel workspace base di MATLAB la struttura dati contenente le matrici di incidenza salvata nel file Json.

Sintassi

```
loadFromJson()
```

Parametri di Input

Nessun parametro di input

Parametri di Output

Nessun parametro di output

Descrizione

Questa funzione si occupa di permettere all'utente la scelta della mesh di interesse, prelevando le mesh disponibili nel componente e mostrandoli a schermo prima della scelta.

Sintassi

```
[selectedMesh, selectedMeshTag] = meshPicker(  
                                         model,  
                                         selectedComponent)
```

Parametri di Input

PARAMETRO	TIPO	DESCRIZIONE
model	ModelClient Obj	Oggetto di tipo ModelClient risultato della chiamata <i>mphload</i> sul percorso di un modello COMSOL
selectedComponent	ModelNodeClient Obj	Oggetto di tipo ModelNodeClient utile per l'estrazione di geometria e mesh

Parametri di Output

PARAMETRO	TIPO	DESCRIZIONE
selectedMesh	MeshSequenceClient Obj	Oggetto di tipo MeshSequenceClient
selectedMeshTag	string	Stringa contenente il tag della mesh selezionata

Descrizione

Questa funzione si occupa di permettere all'utente la selezione, dal suo file system locale, del modello di interesse da caricare poi in MATLAB per le elaborazioni.

Sintassi

```
model = modelPicker()
```

Parametri di Input

Nessun parametro di input

Parametri di Output

PARAMETRO	TIPO	DESCRIZIONE
model	ModelClient Obj	Oggetto di tipo ModelClient risultato della chiamata <i>mphload</i> sul percorso di un modello COMSOL

```
printSplashScreen
```

Descrizione

Questa funzione si occupa di stampare a schermo nella command window di MATLAB lo splash screen di MATSOL.

Sintassi

```
printSplashScreen()
```

Parametri di Input

Nessun parametro di input

Parametri di Output

Nessun parametro di output

projectToPlane

Descrizione

Questa funzione si occupa di proiettare i punti dello spazio 3D su un piano definito dalla normale.

Sintassi

```
[points_2D, T] = projectToPlane(points, normal, T)
```

Parametri di Input

PARAMETRO	TIPO	DESCRIZIONE
<code>points</code>	Nx3 matrix of double	Matrice Nx3, contenente i punti dello spazio rappresentato come un vettore di coordinate [x, y, z] da proiettare
<code>normal</code>	1x3 array of double	vettore che definisce il piano di proiezione
<code>T</code>	4x4 matrix of double	Matrice di rotazione, se fornita, viene utilizzata per la trasformazione; altrimenti, viene calcolata

Parametri di Output

PARAMETRO	TIPO	DESCRIZIONE
<code>points_2D</code>	Nx2 matrix of double	Vettore di punti in 2D, proiettati sul piano specificato
<code>T</code>	4x4 matrix of double	Matrice di rotazione, se fornita, viene utilizzata per la trasformazione; altrimenti, viene calcolata

Descrizione

Questa funzione si occupa di salvare la struttura dati contenente le matrici di incidenza in un file Json.

Sintassi

```
saveToJson(incidenceMatrices, fileName)
```

Parametri di Input

PARAMETRO	TIPO	DESCRIZIONE
incidenceMatrices	struct	Struttura dati contenente tutti gli array delle matrici di incidenza
fileName	char	Lista di caratteri contenente il percorso e il nome del file in cui salvare sul disco la struct

Parametri di Output

Nessun parametro di output

sortPolygonVertices

Descrizione

Questa funzione si occupa di ordinare i vertici di un poligono in senso antiorario.

Sintassi

```
sorted_points = sortPolygonVertices(points)
```

Parametri di Input

PARAMETRO	TIPO	DESCRIZIONE
sorted_points	Nx3 matrix of double	Matrice contenente i punti nello spazio 3D [x,y,z] da ordinare

Parametri di Output

PARAMETRO	TIPO	DESCRIZIONE
<code>points</code>	Nx3 matrix of double	Matrice contenente i punti nello spazio 3D [x, y, z] ordinati in senso antiorario

`stepPicker`

Descrizione

Questa funzione si occupa di permettere all'utente la selezione dello step di interesse tra quelli disponibili per lo studio selezionato.

Sintassi

```
[selectedStep, selectedStepTag] = stepPicker(selectedStudy)
```

Parametri di Input

PARAMETRO	TIPO	DESCRIZIONE
<code>selectedStudy</code>	StudyClient Obj	Oggetto di tipo StudyClient utilizzato per l'estrazione degli step

Parametri di Output

PARAMETRO	TIPO	DESCRIZIONE
<code>selectedStep</code>	StudyFeatureClient Obj	Oggetto di tipo StudyFeatureClient
<code>selectedStepTag</code>	string	Stringa contenente il tag dello step selezionato

`studyPicker`

Descrizione

Questa funzione si occupa di permettere all'utente la selezione dello studio di interesse tra quelli disponibili per il modello selezionato.

Sintassi

```
[selectedStudy, selectedStudyTag] = studyPicker(model)
```

Parametri di Input

PARAMETRO	TIPO	DESCRIZIONE
model	ModelClient Obj	Oggetto di tipo ModelClient risultato della chiamata <i>mphload</i> sul percorso di un modello COMSOL

Parametri di Output

PARAMETRO	TIPO	DESCRIZIONE
selectedStudy	StudyClient Obj	Oggetto di tipo StudyClient
selectedStudyTag	string	Stringa contenente il tag dello studio selezionato

`validateInput`

Descrizione

Questa funzione si occupa di validare l'input inserito dall'utente.

Sintassi

```
choice = validateInput(maxNumberOfChoice)
```

Parametri di Input

PARAMETRO	TIPO	DESCRIZIONE
maxNumberOfChoice	integer	Intero che definisce il numero massimo che l'utente può inserire

Parametri di Output

PARAMETRO	TIPO	DESCRIZIONE
choice	integer	Intero che definisce la scelta dell'utente