POLITECNICO DI MILANO

Facoltà di Ingegneria Industriale
Corso di Laurea in Ingegneria Meccanica

Analisi strutturali in tempo reale controllate attraverso un dispositivo haptic

Relatore: Ing. Francesco Ferrise

Tesi di Laurea Magistrale di:

Luca MARSEGLIA Matricola 766263

Anno Accademico 2012-2013

"L'unico vero viaggio verso la scoperta
non consiste nella ricerca di nuovi paesaggi,
ma nell'avere nuovi occhi."

M. Proust

Ringraziamenti

Non mi sembra vero di aver finito. E' come se dal primo giorno fossero passati allo stesso tempo un secolo e un secondo. E' anche grazie alle tante persone che mi hanno aiutato e che mi sono state vicine se sono arrivato qui.

Comincio con i ringraziamenti "formali".
Ringrazio subito Francesco Ferrise, il mio mentore, ma soprattutto un amico. Grazie per la disponibilità, per i consigli, per l'amicizia e la fiducia che mi hai dedicato. Ringrazio il Prof. Cugini, sempre disponibile e aperto a nuove idee, e i Proff. Cocchetti, Cascini e Cornaggia, che hanno visto e provato in anteprima l'applicazione e hanno saputo darmi preziosi consigli con tanto entusiasmo.

Passo adesso ai ringraziamenti che mi stanno più a cuore.
Prima di tutto ringrazio la mia famiglia, i miei genitori e mia sorella. Grazie per avermi ascoltato e confortato nei tanti momenti difficili. Non credo che troverò mai le parole per esprimere completamente la mia gratitudine e il mio affetto per tutti voi.
Grazie anche a tutti i miei zii. Vi voglio bene, ognuno di voi per me è unico!
Un grazie speciale alla mia ragazza, Debora, che nonostante mille difficoltà mi è sempre stata vicina, fin dal mio primo giorno al Politecnico.
Grazie ai miei "colleghi": Giovanni, Sergio, Roberto, Stefano, Michela, Alessandro, Giulio, con i quali ho vissuto tanti momenti speciali, tante paure, tante soddisfazioni.
Grazie a tutti i miei amici, siete in troppi e non vi elenco, ma grazie a tutti di cuore!
Grazie alla Scala e a tutti i miei colleghi. Sono stati anni incredibili!
Ringrazio in generale tutti quelli che hanno creduto in me, perché mi hanno dato la forza di continuare. Vorrei ringraziare anche quelli che non hanno creduto in me, perché hanno solo fatto aumentare la mia determinazione e la mia testardaggine.

Indice

Sommario

L'obiettivo principale di questa tesi è di combinare dispositivi *haptic* e simulazioni agli elementi finiti (FEM) in tempo reale. Software e tecnologie disponibili già da tempo sono stati utilizzati in modo innovativo, con lo scopo di rendere più intuitivo l'approccio degli utenti alle varie fasi delle simulazioni. Il risultato finale è un'applicazione, chiamata "HaptikFem", rivolta a docenti e studenti e ideata per facilitare la comprensione e l'apprendimento di concetti a volte difficili da assimilare. Con essa, gli utenti sono ora in grado di deformare alcune strutture virtuali attraverso un sistema *haptic*, di vedere istantaneamente il risultato della corrispondente analisi FEM, e di "sentire" la risposta della struttura attraverso segnali tattili.

Dopo una fase di ricerca, è stato chiaro che non esisteva un unico software con il quale poter realizzare tutti gli obiettivi prefissi. L'applicazione finale è, infatti, il risultato di una combinazione tra due software: Comsol è utilizzato per la modellazione delle strutture e per la simulazione FEM, e consente di salvare i risultati in un formato compatibile con Matlab, il secondo software utilizzato, un programma per il calcolo numerico molto versatile e noto nelle scuole di ingegneria. Tutti i file dell'applicazione sono compatibili con Matlab e sono stati modificati per permettere alla simulazione di funzionare in tempo reale. Ciò consente agli utenti di visualizzare istantaneamente il risultato aggiornato della FEM a ogni variazione di carico. Per stabilire la connessione tra il sistema *haptic* e Matlab, è utilizzata la "Haptik Library", sviluppata dall'università di Siena, che permette di aggiungere il *feedback* tattile alle simulazioni FEM [17]. Gli utenti sono inoltre in grado di utilizzare un mouse o un trackpad per le simulazioni nel caso non fosse disponibile un dispositivo *haptic*. L'applicazione è dotata di una semplice e intuitiva interfaccia grafica, che permette una rapida scelta di modelli, vincoli e carichi, e la cui efficacia è stata sottoposta a una fase finale di test con venti utenti volontari.

Capitolo 1: Introduzione

L'utilizzo combinato della potenza degli odierni calcolatori e della versatilità dei nuovi software consente oggi agli sviluppatori di confrontarsi con realtà ancora inesplorate. Si tratta di una tendenza molto in voga negli ultimi anni, nei quali si sta tentando in diversi modi di colmare il divario tra utenti e nuove tecnologie, rendendole sempre più "*user-friendly*". Si pensi ad esempio al livello di realismo ottenuto dai film 3D o alle sensazioni restituite dai nuovi videogiochi. Una prospettiva interessante è costituita, ad esempio, dall'opportunità di fornire una percezione tattile a entità virtuali, avvicinandole per quanto più possibile al mondo reale. Gli strumenti che oggi rendono possibili queste possibilità sono chiamati "aptici" (in inglese "*haptic*"), termine che deriva dal greco "*aptio*", che significa "tocco". Un sistema aptico è, quindi, un dispositivo robotico studiato per interagire direttamente con l'utente, il quale riceve come risposta ai movimenti delle sensazioni tattili. Si tratta di apparecchi che possono avere un grado di complessità e un costo variabile, secondo l'utilizzo e le potenzialità: si va da semplici mouse dotati di rotelle in grado di bloccarsi quando l'utente urta con il puntatore un oggetto virtuale a veri e propri esoscheletri, oggi utilizzati anche in campo militare e medico [8] [9].

L'obiettivo principale di questa tesi è di utilizzare con scopo didattico i sistemi *haptic* in simulazioni ingegneristiche per renderle più comprensibili e coinvolgenti attraverso un *feedback* non più solo visivo, ma anche tattile.
In molti corsi d'ingegneria, agli studenti è richiesto di affrontare problemi legati allo studio di sforzi e deformazioni di componenti e strutture. Analisi di questo tipo costituiscono una parte importante di ogni progetto, poiché lo studio approfondito delle caratteristiche meccaniche e strutturali dei singoli elementi consente di realizzare prodotti conformi alle specifiche di resistenza e durata richieste.

Per facilitare la comprensione dei principi fisici alla base di tali analisi, i professori invitano spesso gli studenti a utilizzare particolari programmi, ideati per la risoluzione di questa tipologia di problemi, e a risolvere alcuni esercizi. Tali software sono comunemente chiamati "FEM", abbreviazione che deriva da "*Finite Elements Method*". L'analisi agli elementi finiti è, infatti, il metodo algoritmico che sta alla base della risoluzione delle strutture, e si basa su una tecnica numerica atta a cercare soluzioni approssimate di problemi descritti da equazioni differenziali. La caratteristica principale del metodo degli elementi finiti è la discretizzazione delle strutture attraverso la creazione di una griglia ("*mesh*") composta da primitive di forma codificata, come ad esempio triangoli e quadrilateri (per i domini 2D) o esaedri e tetraedri (per i domini 3D), chiamate "elementi finiti". Su ciascun elemento caratterizzato da questa forma elementare, la soluzione del problema è espressa dalla combinazione lineare di funzioni dette "funzioni di base" o "funzioni di forma" che, una volta risolte, permettono il calcolo di sforzi e deformazioni in ogni punto del modello.

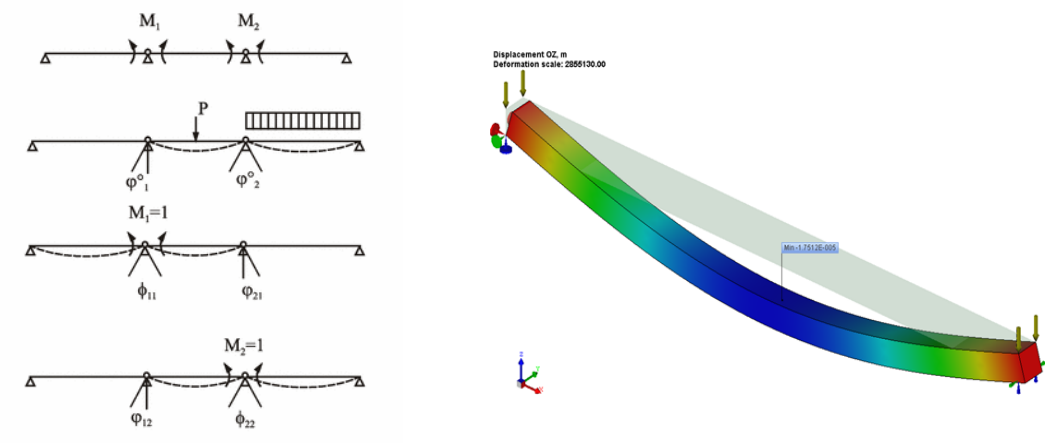


Figura 1.1: Metodo "classico" di risoluzione di una struttura (a sinistra) e esempio di una struttura simile risolta con un software FEM (a destra)

Infatti, mentre in classe si utilizzano strumenti di calcolo relativamente semplici per determinare gli sforzi e le deformazioni in alcuni punti di particolare interesse di una struttura, con questi software lo studente è in grado di ottenere velocemente i valori di sforzi e deformazioni per tutti i punti di strutture anche molto complesse. L'utilizzo dei software per le analisi FEM richiede tuttavia un buon grado di esperienza o l'assistenza continua da parte di un docente. Il risultato che lo studente può raggiungere dopo aver impostato una numerosa serie di parametri, è senz'altro molto più completo di quello ottenuto in classe "su carta", ma si riferisce comunque a una struttura in una condizione preimpostata. Se l'utente volesse cambiare la geometria, il materiale, il carico o i vincoli, dovrebbe rivedere alcune impostazioni e lanciare nuovamente la simulazione per visualizzare il nuovo risultato a schermo, senza possibilità di altra interazione se non quella visiva per immaginare la risposta della struttura a una sollecitazione esterna di diversa entità. Ciò che ci si è proposto di fare è di superare questi limiti, con lo scopo di aumentare il grado di coinvolgimento e apprendimento degli utenti.

Gli obiettivi di questo lavoro sono dunque:

- Realizzare una simulazione FEM in tempo reale in grado di rispondere in maniera pressoché immediata alle sollecitazioni imposte dall'utente.
- Utilizzare un dispositivo *haptic* per l'applicazione dei carichi, in modo che l'utente possa "sentire" una risposta tattile e realistica della struttura.
- Realizzare un'interfaccia "*user-friendly*", in grado di poter essere utilizzata anche dagli utenti meno esperti, che includa i modelli più utilizzati durante le lezioni.
- Valutare l'efficacia dell'applicazione realizzata, svolgendo opportuni test-case su un gruppo di utenti.

Figura 1.1: Concept dell'utilizzo di applicazioni FEM in real-time con l'utilizzo di dispositivi haptic durante una lezione di meccanica delle strutture

Un'applicazione che lavora in tempo reale ("*real-time*") deve essere in grado, come dice il nome stesso, di rispondere agli input in tempi infinitesimi, idealmente nulli [32]. Si parla di applicazioni in *tempo reale stretto* quando i tempi di risposta sono dell'ordine dei millisecondi. Tuttavia anche applicazioni con tempi più laschi, dell'ordine del secondo, possono essere considerate *real-time*. Per la risoluzione di problematiche di questo tipo, si utilizzano normalmente architetture hardware dedicate, e sistemi operativi e applicazioni appositamente concepiti. Normalmente queste tre componenti (hardware, software di base e software applicativo) sono strettamente legate, in modo da ottimizzare i tempi. Esempi pratici sono quelli dei sistemi di controllo di volo degli aerei o delle centrali nucleari: è impensabile che sistemi del genere non reagiscano a un evento in tempo reale, poiché le conseguenze potrebbero essere catastrofiche. Quando si parla di applicazioni *real-time* utilizzate per la visualizzazione su schermo, si parla solitamente di *frame-rate*, ossia della frequenza di aggiornamento dell'immagine. Ovviamente più il *rate* è elevato, più fluida sarà la visualizzazione, che di conseguenza sembrerà aggiornarsi istantaneamente.

Nelle applicazioni utilizzate per la realtà aumentata ad esempio, nelle quali si utilizzano specifici "visori" o occhiali per integrare nel mondo reale alcuni oggetti virtuali, la frequenza di aggiornamento deve essere sufficientemente elevata per poter garantire una perfetta sovrapposizione tra "mondo reale" e "mondo virtuale"
Nel caso dell'applicazione che ci si propone di realizzare si deve tenere presente che non si hanno a disposizione un hardware, un software di base o un software applicativo dedicati alla simulazione FEM in tempo reale. Sarà quindi necessario scendere a diversi compromessi, il primo dei quali riguarderà la frequenza di aggiornamento delle immagini. Tenendo conto della "pesantezza" di calcolo di una singola risoluzione agli elementi finiti, si riterrà soddisfacente raggiungere i 10 Hz. Altre problematiche dovranno essere affrontate al momento dell'utilizzo del sistema *haptic*, che richiede una frequenza di aggiornamento di 1 *kHz*.

Nei capitoli che seguono, si elencheranno i software in commercio per le analisi agli elementi finiti e se ne valuteranno le caratteristiche tenendo conto della velocità di calcolo e della compatibilità con altri programmi e librerie, poiché non avendo a disposizione dei software dedicati, si dovrà necessariamente ricorrere a una soluzione "ibrida". Si passerà poi alla realizzazione di un primo semplice modello, al suo salvataggio in un formato compatibile con Matlab, e alla successiva modifica dei file. Solo in seguito saranno descritte le procedure utilizzate per far comunicare il sistema *haptic* con i software utilizzati. Si passerà poi ai capitoli che si occupano della realizzazione di una semplice interfaccia grafica e della descrizione dei modelli implementati. Gli ultimi due capitoli riguardano l'analisi di una fase di test svolta con utenti e le conclusioni, con spunti per eventuali sviluppi futuri.

Capitolo 2: Stato dell'arte

Prima di procedere con il capitolo dedicato alla scelta dei software necessari per procedere nello sviluppo del lavoro, si è ritenuta doverosa una fase di ricerca di progetti simili a quello che ci si propone di realizzare, con lo scopo di valutare soluzioni già trovate da altri e di comprendere quali potrebbero essere le possibili strade da intraprendere.

- Il contributo maggiore a questa tesi è stato dato senz'altro dal lavoro di ricerca e tesi svolto da A. E. Uva, M. Fiorentino, G. Monno e S. Cristiano del DIMeG del Politecnico di Bari, che si sono occupati dello sviluppo di un'applicazione di realtà aumentata ("*Augmented Reality*", o "AR") in grado di effettuare un'analisi agli elementi finiti in tempo reale [1]. Il software da loro realizzato utilizza il tracciamento ottico per la parte di acquisizione e Comsol per la parte di simulazione FEM. Attraverso l'utilizzo di opportuni *marker,* il risultato della FEM viene sovrapposto, in tempo reale, alla visualizzazione in realtà aumentata. Se l'utente indossa gli occhiali per la AR e deforma la struttura reale, vedrà sovrapporsi su di essa il risultato della FEM.

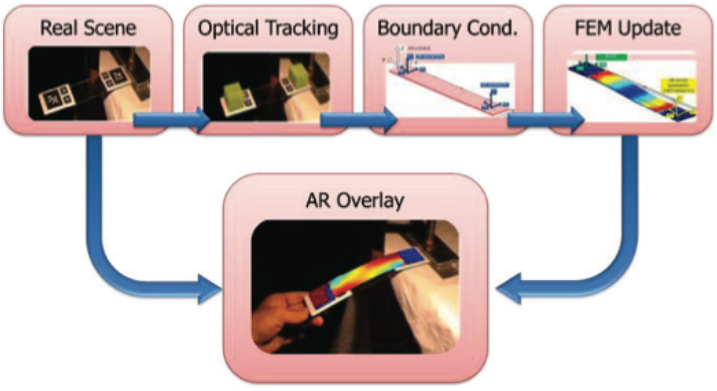


Figura 2.1: Schema logico del funzionamento dell'applicazione "touch and see"

- Nel Dipartimento di Prototipazione della State University of New York, i ricercatori Kevin T. McDonnel e Hong Quin hanno messo a punto un'applicazione in grado di applicare analisi FEM ricorsive a una struttura grossolana iniziale, allo scopo di affinare in modo sempre maggiore la mesh [2]. Il metodo proposto è totalmente nuovo, poiché prevede di scolpire e deformare in modo guidato un oggetto con l'ausilio del software e di un dispositivo *haptic*, fino al raggiungimento della configurazione ideale e al *rendering* dell'oggetto realizzato.

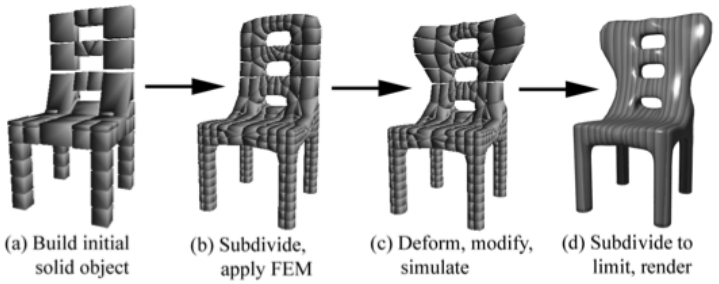


Figura 2.2: Step successivi del metodo proposto da McDonnel e Quin

- Alla Drexel University di Philadelphia, Teranoot Chantasopeephan, Jaydev P. Desai e Alan C. W. Lau hanno lavorato sulla possibilità di intraprendere alcune operazioni chirurgiche di alta precisione, come il taglio di tessuti molli, utilizzando un dispositivo *haptic* appositamente costruito [3]. Per giungere a un risultato soddisfacente sono stati condotti studi approfonditi sulla composizione dei tessuti molli, che sono stati infine descritti come oggetti dal comportamento "non lineare, non omogeneo e viscoelastico", dunque molto complesso. Dopo numerosi test i ricercatori stati in grado di simulare alla perfezione il comportamento di alcuni tessuti, avvalendosi dell'utilizzo di schematizzazioni FEM con elementi 3D tetraedrici.

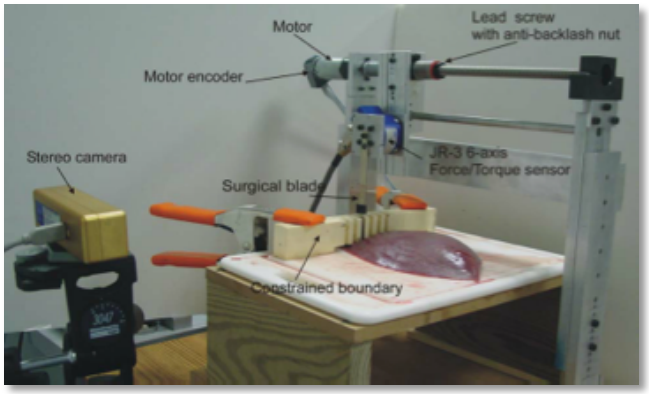
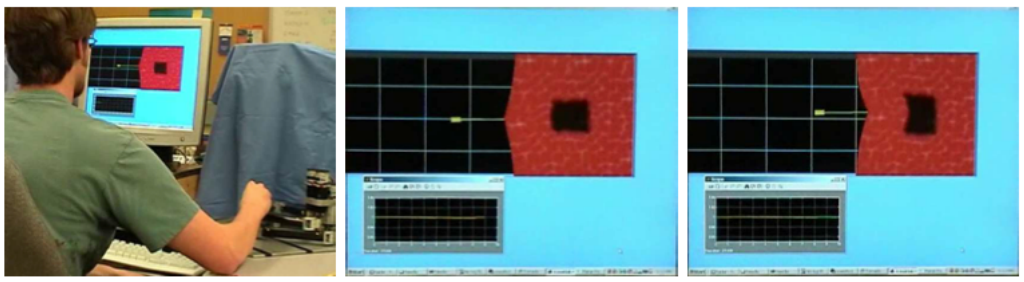


Figura 2.3: Apparato sperimentale allestito per misurare le forze durante il processo di taglio e rilevare gli spostamenti del bisturi con una stereo camera

- Nel Dipartimento di Ingegneria Informatica della University of British Columbia, Andrew H. Gosline, Septimiu E. Salcudean e Joseph Yan hanno messo a punto un sofware per la simulazione in tempo reale della deformazione di oggetti cavi con comportamento lineare elastico riempiti di fluido [4]. Il fluido è modellizzato utilizzando un'analisi fluido-statica, la cui grandezza fondamentale è la pressione. Il software tiene conto del legame tra forza e pressione, e l'utente può deformare l'involucro elastico utilizzando un dispositivo *haptic*.



(a) Haptic Simulation Setup (b) Pocket under tension (c) Pocket under compression

Figura 2.4: Simulazione della deformazione di un dominio elastico contenente un fluido con un dispositivo haptic

- Un altro lavoro d'interesse sulle potenzialità dell'utilizzo dei sistemi *haptic* in ambito biomedico è stato svolto da Jeffrey Berkley, George Turkiyyah, Daniel Berg, Mark Ganter, Suzanne Weghorst della Univesity of Washington, che hanno sviluppato un sistema per effettuare suture utilizzando un dispositivo *haptic* appositamente costruito [5]. Per giungere a un risultato finale con dei *feedback* realistici, sono stati condotti studi approfonditi su processi di sutura reali, allo scopo di misurare con accuratezza forze e movimenti.

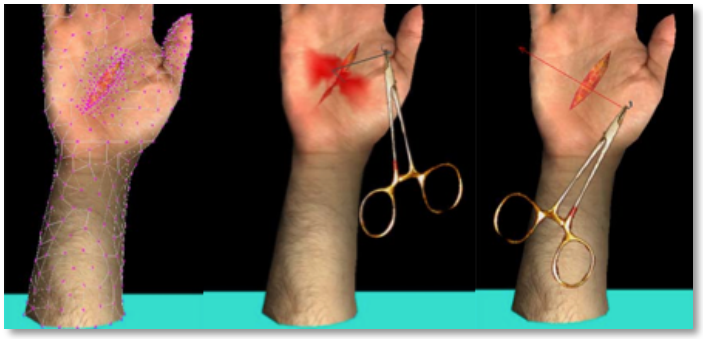


Figura 2.4: Sutura virtuale realizzata utilizzando un dispositivo haptic

Capitolo 3: Hardware e Software

Nei paragrafi seguenti saranno illustrate le caratteristiche dei principali sistemi *haptic* e dei software per le analisi FEM in commercio. A seguire saranno brevemente elencati i principali hardware e software che sono stati utilizzati nel lavoro e le motivazioni che hanno portato a tali scelte.

3.1 I sistemi *haptic*

Come già accennato, si dicono "*haptic*" tutti quei sistemi che permettono l'interazione con oggetti virtuali attraverso il senso del tatto. Fin dall'inizio del XX secolo il termine "aptico" è stato usato da molti psicologi e da neuroscienziati per indicare l'interazione tattile tra il paziente e oggetti reali. Oggi con questo termine ci si riferisce, in ambito ingegneristico, a particolari sistemi, elettronici e/o meccanici, che consentono agli utenti di "toccare" oggetti virtuali creati con specifici software, aggiungendo quindi una nuova sfera percettiva alle simulazioni.

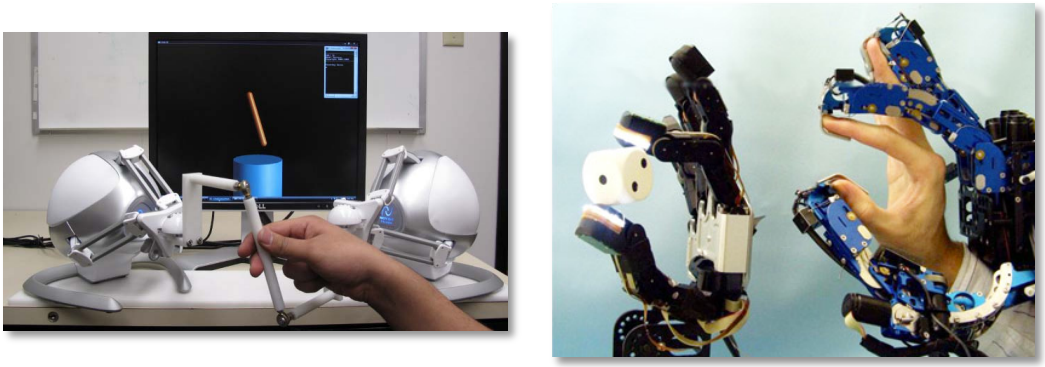


Figura 3.1: Due diverse tipologie di sistema haptic: a braccetti motorizzati (a sinistra) e a esoscheletro (a destra)

La maggior parte dei sistemi *haptic* è costituita da un'unità di controllo elettronica collegata direttamente al computer attraverso una porta Ethernet, seriale o USB e da uno o più braccetti meccanici articolati. Questi ultimi possono prevedere la presenza di un'impugnatura o di un *end-effector*, e devono essere guidati direttamente dall'utente. L'unità di controllo ha lo scopo di registrare in tempo reale la posizione nello spazio del puntatore, misurandone a ogni istante le esatte coordinate spaziali, sia in termini di traslazioni sia di rotazioni, e inviarle al software. Nella base del sistema sono solitamente alloggiati uno o più motori elettrici che servono a simulare la risposta del materiale che si sta cercando di toccare, dando l'illusione del contatto con un oggetto reale e non virtuale.

I sistemi *haptic* possono potenzialmente essere utilizzati in diversi ambiti:

- Medico: con lo scopo di guidare micro robot per interventi poco invasivi, per diagnosi remote, o come aiuto per utenti disabili.
- Intrattenimento: per videogames e simulatori, per consentire agli utenti di manipolare solidi, fluidi e personaggi virtuali come se fossero reali.
- Educazione: per dare agli studenti una migliore percezione dei fenomeni.
- Industriale: attraverso l'integrazione di sistemi *haptic* e CAD. In questo modo ingegneri e progettisti possono creare e manipolare liberamente i componenti.
- Arti grafiche: per applicazioni interattive in occasione di mostre, esibizioni musicali (ad esempio per controllare in remoto strumenti musicali), per "scolpire" virtualmente oggetti ecc.

Uno dei dispositivi disponibili nel laboratorio di Prototipazione Virtuale del Politecnico di Milano è il sistema *Phantom* distribuito da Sensable di Figura 1.1, che ha una struttura a forma di braccio meccanico, alla cui estremità è collegata, per mezzo di tre giunti, una penna [6]. Il primo giunto è attuato e sensorizzato e permette all'interfaccia di ruotare lungo l'asse verticale. Il secondo e il terzo giunto, anch'essi attuati e sensorizzati, conferiscono invece all'*end-effector* la possibilità di traslare lungo i restanti due assi, conferendo un orientamento nello spazio di 3 DOF (*Degree Of Freedom*). Le sensazioni che si possono ottenere con questo dispositivo sono pertanto quelle che si potrebbero avere nella realtà se si entrasse in contatto con gli oggetti reali utilizzando la punta di una penna.

Figura 3.2: Sistema haptic "Phantom" usato per applicazioni biomediche

3.2 Software e librerie per la gestione dei sistemi *haptic*

In seguito alla realizzazione e commercializzazione dei sistemi *haptic* sono stati sviluppati diversi software e librerie che consentono agli utenti di interfacciare tali sistemi con il proprio calcolatore. Tali risorse, essendo per la maggior parte *open-source*, sono spesso il frutto del lavoro di "comunità" di ricercatori e appassionati. Di seguito è riportato un breve elenco dei software e delle librerie più utilizzate.

3.2.1 OpenSceneGraph Haptic Library (OSGHaptics)

OpenSceneGraph è un insieme di strumenti grafici 3D a elevate prestazioni, ed è totalmente open source [13]. E' scritto completamente in C++, sfrutta API di OpenGL e lavora sulla maggior parte dei sistemi operativi, tra cui Windows, OSX, Linux, IRIX, Solaris, HP-Ux, AIX e FreeBSD. I formati di file supportati sono: COLLADA, LightWave (.lwo), Alias Wavefront (.obj), OpenFlight (.flt), TerraPage (.txp), Carbon Graphics GEO (.geo), 3D Studio MAX (.3ds), Peformer (.pfb), AutoCAd? (.dxf), Quake Character Models (.md2). Direct X (.x), Inventor Ascii 2.0 (.iv)/ VRML 1.0 (.wrl), Designer Workshop (.dw) e AC3D (.ac) oltre al format .osg ASCII. I formati immagine supportati sono: .rgb, .gif, .jpg, .png, .tiff, .pic, .bmp, .dds, .tga e QuickTime. A oggi lo sviluppo di OpenSceneGraph è fermo alla versione 2.8 del Febbraio 2008, anche se nuove librerie vengono aggiunte periodicamente. Tra queste è presente la libreria osgHaptics, che utilizza il *toolkit* OpenHaptics fornito da Sensable. Sfruttando questa risorsa, OpenSceneGraph riesce a interfacciarsi i dispositivi *haptic* della Sensable, come i "Phantom".

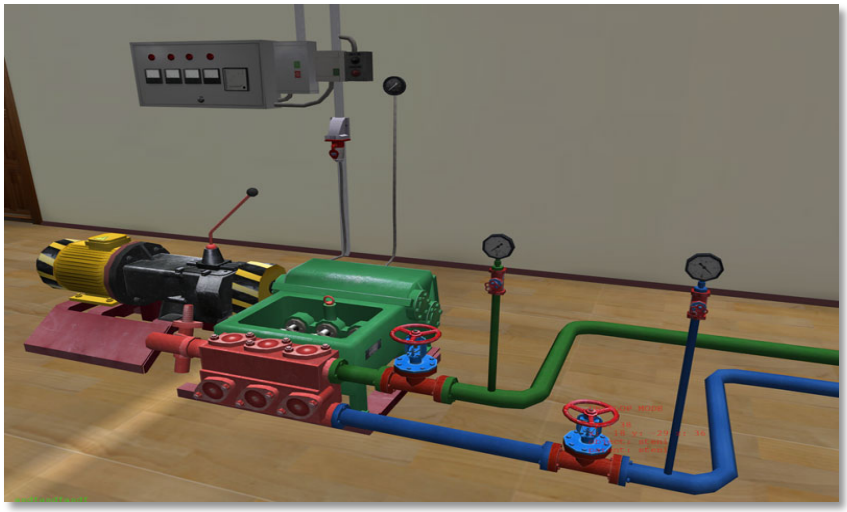


Figura 3.3: Simulazione di un gasdotto ottenuta con OpenSceneGraph

3.2.2 OpenHaptics

OpenHaptics è un *toolkit* realizzato da Sensable che è in grado di semplificare e velocizzare l'utilizzo dei sistemi *haptic* [12]. E' pensato per lavorare con i dispositivi Phantom, e oltre a supportare sistemi fino a 7 gradi di libertà, include la libreria Quickhaptics, che permette di creare in modo molto più veloce applicazioni e programmi ad-hoc sfruttando C++. Supporta Linux e Windows 7 ed è in grado di effettuare un monitoraggio continuo della profondità di penetrazione rispetto alla superficie di contatto, cosa che risulta utile per le simulazioni, in particolare in ambito medico. E' stato inoltre abilitato il supporto ai più comuni formati di modelli 3D, quali .3DS, .OBJ, .STL e .PLY.

3.2.3 CHAI3D

CHAI3D è un set di librerie *open-source* basato su C++ creato per la visualizzazione, l'interazione e la simulazione real-time su calcolatori equipaggiati con sistemi *haptic* [15]. E' in grado di supportare sistemi *haptic* a tre, sei e sette gradi di libertà ed è stato pensato soprattutto per l'ambito della ricerca e dell'educazione. Essendo *open-source*, il codice sorgente delle librerie è completamente accessibile e modificabile. Le librerie di CHAI3D sono supportate da Windows, Mac OSX e Linux. Il *rendering* sfrutta OpenGL e si ha la possibilità di usare *texture* 2D e 3D e di inserire le proprietà specifiche di ogni materiale utilizzato. I formati delle *mesh* supportate sono .3DS e .OBJ, quelli delle immagini .BMP e .TGA. Sono inoltre supportati a maggior parte dei sistemi *haptic* in commercio, tra cui i Phantom e in Novint Falcon. Sono disponibili le estensioni ODE (*Open Dynamic Engine*), GEM (per modelli deformabili) e BASS (libreria audio).

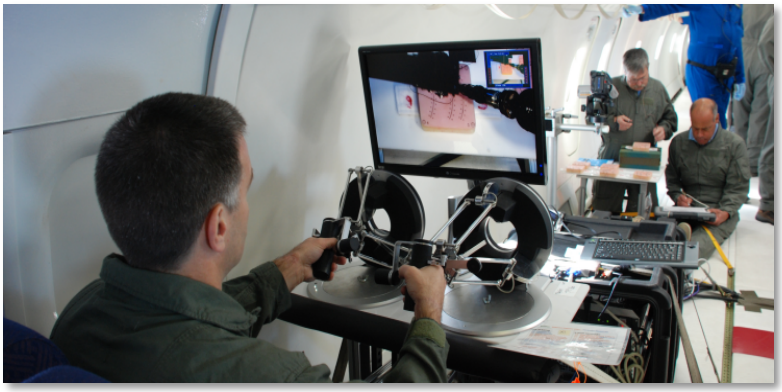


Figura 3.4: Esempio di applicazione haptic per ricerca in campo militare

3.2.4 H3DAPI

H3DAPI è una piattaforma *open-source* per lo sviluppo di software *haptic* basata su OpenGL e X3D [16]. E' stata creata con lo scopo di semplificare e velocizzare la fase di sviluppo di applicazioni e programmi, combinando i vantaggi offerti da X3D, C++ e Python. E' supportata da Windows, Linux e Mac OSX.

3.2.5 Haptik Library

L'Haptik Library è una libreria *open-source* realizzata dall'università di Siena, e fornisce un "livello di astrazione hardware" per l'accesso ai sistemi *haptic* [17]. In tal modo dispositivi differenti, provenienti da diversi produttori, possono interfacciarsi in modo immediato bypassando tutte le dipendenze da particolari configurazioni hardware, API e driver. Supporta sia OpenGL sia DirectX e consiste di una serie di *plugin* che possono essere estesi o personalizzati in modo semplice. La libreria può essere utilizzata da utenti che hanno familiarità con linguaggi di programmazione differenti, quali C++, Java, Simulink e Matlab. L'Haptik Library supporta anche i sistemi Falcon e può essere installata esclusivamente su Windows.

3.3 Software commerciali per le analisi FEM:

Nei prossimi paragrafi saranno brevemente illustrate le caratteristiche principali dei software commerciali per le analisi FEM più conosciuti e utilizzati. Questi software sono disponibili a pagamento poiché il loro codice è proprietario e registrato.

3.3.1 Simulia Abaqus

Abaqus è uno dei software multifisici a pagamento più famosi e diffusi per la simulazione realistica e di alta qualità [19]. Il programma è in grado di simulare la risposta magnetostatica, di eseguire analisi di viscoelasticità non-lineare, è in grado di simulare materiali porosi e la viscosità non newtoniana. E' possibile inoltre convertire le parti della *mesh* in entità geometriche, il che torna utile per creare *mesh* deformabili. Ogni analisi agli elementi finiti è suddivisa nelle tre fasi caratteristiche di tutti i software che si occupano di FEM: *pre-processing*, analisi agli elementi finiti e *post-processing*. In software è anche in grado di acquisire dati pre-processati da altri software.

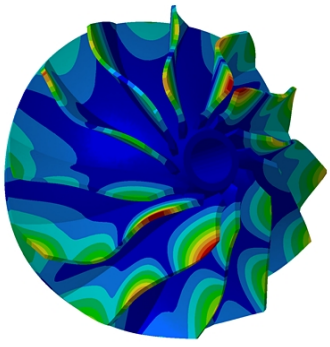


Figura 3.5: Esempio dello stato di sforzo di una girante calcolato con Abaqus

3.3.2 Comsol

Comsol è uno dei software per l'analisi agli elementi finiti più completi ed evoluti in commercio [21]. Consente non solo di modellare componenti e studiarne la distribuzione degli sforzi, ma può essere utile anche per simulare fenomeni di tipo elettrico, magnetico, gravitazionale, fluidodinamico ecc. E' inoltre in grado di interfacciarsi in modo diretto con software quali Microsoft Excel, Matlab, Solidworks e molti altri.

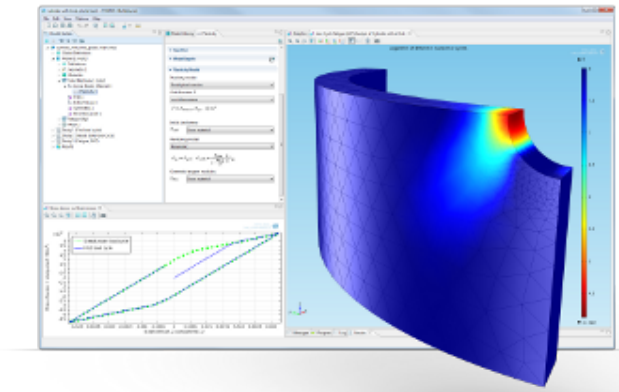


Figura 3.6: Esempio di struttura 3D ottenuta con Comsol

3.3.3 Altair

Altair è un altro famoso software di analisi agli elementi finiti, e possiede un'ampia gamma di interfacce di comunicazione con i sistemi CAD/CAE [26]. Consente la modellazione con elementi finiti 2D o 3D. Le particolarità più rilevanti sono di poter unire tra loro geometrie di modelli già dotati di mesh e di visualizzare in 3D gli strati dei materiali compositi sottoposti a sollecitazione.

3.3.4 Ansys

Ansys consente di modellizzare in modo semplice componenti con geometrie complesse attraverso un'interfaccia di disegno avanzata, molto simile a quella dei sistemi CAD [27]. Permette anche di unire componenti contenuti in file diversi, ed è in grado di individuare automaticamente il contatto tra due oggetti durante la fase di assemblaggio. Utilizza un linguaggio di programmazione che può essere modificato dall'utente, che può quindi accedere direttamente agli algoritmi di risoluzione e di costruzione della geometria. E' anche in grado di esportare i risultati in formato compatibile con Word o PowerPoint.

3.3.5 MSC Nastran

E' uno dei software di simulazione più diffusi, soprattutto perché è in grado di effettuare analisi multi-fisiche [28]. Consente inoltre di eseguire calcoli sulla durata dei singoli componenti in relazione alla sollecitazione che ciascuno di essi subisce.

3.3.6 Altri software commerciali

Si elencano di seguito altri software commerciali che hanno caratteristiche del tutto simili a quelle già viste per i software precedenti: ADINA, ALGOR, ANSA, AutoForm (utile per i processi di formatura di componenti metallici), CosmosWorks, FEFLOW (multi-fisico), Femap, LUSAS (software Britannico spesso utilizzato nei cosi di ingegneria), VisualFEA (software Coreano).

3.4 Software *open source* per le analisi FEM:

3.4.1 CalculiX

CalculiX è un software FEM 3D gratuito molto versatile [18]. Esso permette di creare gli elementi finiti e di sottoporli alle fasi di calcolo e *post-processing*. Il software permette di risolvere analisi lineari e non, e permette di ottenere soluzioni a problemi statici, dinamici e di tipo termico. E' inoltre compatibile con il formato di Abaqus, che può essere utilizzato per pre-processare i dati. Il Software è anche in grado di lavorare con i formati compatibili con Nastran, Ansys, Dolfyn, Duns, ISAAC e OpenFOAM ed è pensato per essere installato su piattaforme Unix (come Linux e Irix) e Windows.

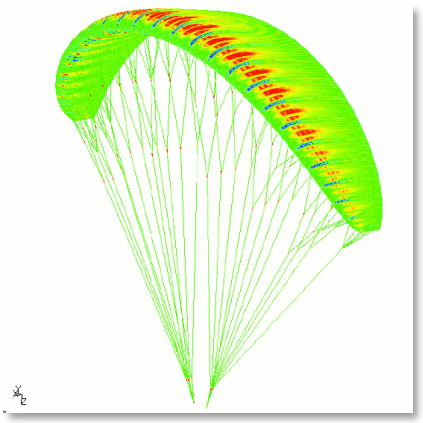


Figura 3.7: Sollecitazioni in una vela da parapendio calcolate con CalculiX

3.4.2 Code Aster

Si tratta di un codice *open-source* sviluppato per ingegneria civile e per problemi strutturali in genere [29]. E' un risolutore abbastanza semplice, che si limita ad applicare la teoria degli elementi finiti ai modelli impostati dall'utente. Secondo le esigenze può svolgere anche calcoli inerenti fatica, danneggiamento, contatto e frattura. Il codice sorgente è formato da sole 1.500.000 righe, scritte in Fortran e Python.

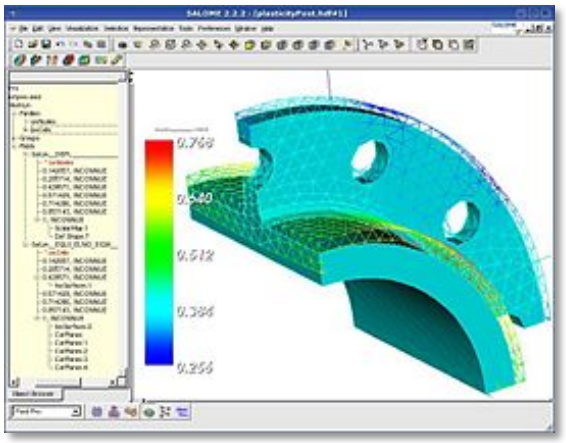


Figura 3.8: Interfaccia di Code Aster

3.4.3 FreeFem++

FreeFem++ è un linguaggio di programmazione *freeware* per la risoluzione equazioni differenziali con il metodo degli elementi finiti [30]. E' scritto in C++ ed è stato sviluppato dall'università Francese Pierre et Marie Curie. E' un linguaggio multi-piattaforma che gira su Linux, Solaris, OSX e Windows. Esiste anche una versione per la risoluzione di strutture 3D, chiamata "FreeFem3D".

3.4.4 Altri software *open source*

Anche in questo caso si elencano altri famosi software *freeware* e totalmente *open-source* che hanno caratteristiche simili a quelli elencati nei paragrafi precedenti: DUNE (scritto interamente in C++), Elmer (multi-fisico, basato su C++ e Fortran90), FEBio (sviluppato per problemi di biomeccanica), Deal.II (un pacchetto di codici per la risoluzione agli elementi finiti).

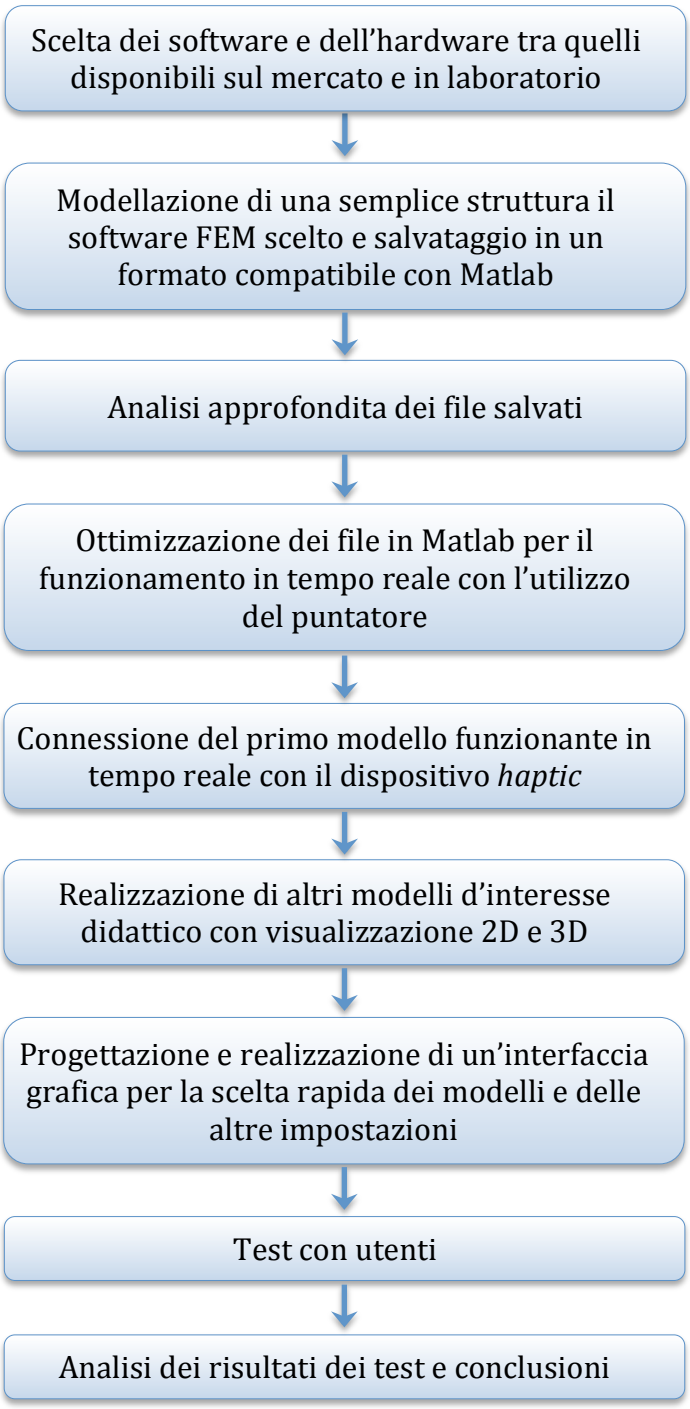
3.5 Tabella riassuntiva dei software

Vista l'ampia disponibilità di Software, si ritiene opportuno riassumere le principali potenzialità di quelli più importanti nella seguente tabella, che verrà utilizzata nei paragrafi seguenti come aiuto nella scelta dei software più vantaggiosi:

Software	Open source	Compatibilità	Multi fisico	Multi piattaforma	Real time
Abaqus	✗	Catia, Ansys, Nastran	✓	Win, Linux	✗
Comsol	✗	Autocad, Inventor, SolidEdge, SolidWorks, Matlab, Java, C, Excel	✓	Win, Linux, Mac OSX	✗
Altair HyperWorks	✗	Catia, SolidWorks, Acis, Abaqus, Ansys	✓	Win, Unix, Linux	✗
Ansys	✗	Tutti i file CAD in formato .IGES	✓	Win, Linux	✗
Nastran	✗	SolidWorks	✓	Win, Linux	✗
CalculiX	✓	Nastran, Abaqus, Ansys, Code Aster, openFOAM	✗	Win, Linux, Irix	✗
Code Aster	✓	Gmsh, SALOME	✗	Win, Linux, Mac OSX, Ubuntu	✗
FreeFem++	✓	Matlab (attraverso appositi convertitori *open source*)	✓	Win, Linux, Mac OSX	✗

Tabella 3.1: Caratteristiche salienti dei più famosi software per l'analisi agli elementi finiti

Di seguito è riportato uno schema che riassume i passi che sono stati affrontati per giungere agli obbiettivi che ci si è prefissi e che saranno descritti nei prossimi paragrafi:



3.6 Scelta dell'hardware

Come è già stato accennato nei paragrafi precedenti, il dispositivo *haptic* disponibile in laboratorio è un "*Phantom Desktop*", prodotto dalla Sensable, azienda leader nel settore [6] [11]. Il *Phantom* è un dispositivo *haptic* costituito sostanzialmente da bracci snodabili, ai quali è applicata una penna mediante la quale l'utente può interagire con una struttura virtuale e sentire una risposta reale, data dalla resistenza del materiale. In questo modo chi lo usa ha l'illusione di toccare o modellare un oggetto reale.

Figura 3.9: *Phantom Desktop* della Sensable

Il *Phantom* è dotato di 6 gradi di libertà in input e 3 in output, ha una risoluzione di 0.023 mm, riesce ad esercitare una forza massima di quasi 8 N e ha un'area di lavoro pari a 160x120 x120 mm (LxAxP).

Di seguito è riportata la tabella che richiama tutte le altre specifiche tecniche, così come indicate dal costruttore.

PHANTOM DESKTOP TECHNICAL SPECIFICATIONS

Force feedback workspace	~6.4 W x 4.8 H x 4.8 D in > 160 W x 120 H x 120 D mm
Footprint (Physical area the base of device occupies on desk)	5 5/8 W x 7 1/4 D in ~143 W x 184 D mm
Weight (device only)	6 lbs 5oz
Range of motion	Hand movement pivoting at wrist
Nominal position resolution	> 1100 dpi ~ 0.023 mm
Backdrive friction	< 0.23 oz (0.06 N)
Maximum exertable force at nominal (orthogonal arms) position	1.8 lbf (7.9 N)
Continuous exertable force (24 hrs)	0.4 lbf (1.75 N)
Stiffness	X axis > 10.8 lbs / in (1.86 N / mm) Y axis > 13.6 lbs / in (2.35 N / mm) Z axis > 8.6 lbs / in (1.48 N / mm)
Inertia (apparent mass at tip)	~0.101 lbm (45 g)
Force feedback	x, y, z
Position sensing [Stylus gimbal]	x, y, z (digital encoders) [Pitch, roll, yaw (± 3% linearity potentiometers)
Interface	Parallel port and FireWire® option*
Supported platforms	Intel or AMD-based PCs
OpenHaptics®SDK compatibility	Yes
Applications	Selected Types of Haptic Research, the FreeForm® Modeling™, and the FreeForm® Modeling Plus™ systems

Tabella 3.2: Specifiche tecniche del Phantom Desktop

3.7 Scelta dei software

Nel Paragrafo 3.5 sono stati elencati alcuni software e alcune librerie in commercio in grado di interfacciarsi con i sistemi *haptic,* e i molti software disponibili per l'analisi agli elementi finiti. Si ricorda a questo punto che gli scopi della presente tesi sono fondamentalmente quelli di stabilire se esista un modo per utilizzare i sistemi *haptic* nell'ambito di un'analisi FEM *real-time*, fornendo all'utente non più solo un input visivo, ma anche tattile, con la sensazione di "toccare" una struttura, e vederne sforzi e deformazioni conseguenti in tempo reale. Com'era lecito supporre fin da principio, tra i software e le librerie analizzate non esiste uno strumento in grado di condurre direttamente allo scopo prefissato, pertanto lo strumento finale dovrà essere in realtà un "ibrido", in cui più software e librerie cooperano per giungere a un unico risultato. Dopo attente riflessioni si è scelto di scartare il linguaggio C, così come quello Java, e di utilizzare un software di programmazione noto e molto più familiare a tutti gli studenti d'ingegneria meccanica: Matlab. Questo software, come si vedrà in un'analisi più approfondita nei prossimi paragrafi, è un potentissimo strumento di calcolo, che tuttavia non è normalmente utilizzato per applicazioni *real-time*, specialmente quando i calcoli sono particolarmente onerosi. Per questo motivo, dopo aver analizzato attentamente i software per l'analisi agli elementi finiti disponibili di Tabella 1.1, si è scelto di non integrare direttamente in Matlab gli algoritmi necessari alla risoluzione delle analisi FEM, che pure sono ampiamente disponibili, ma di "alleggerire" il sistema delegando questa parte del lavoro a un secondo software, che già nativamente è in grado di lavorare in simbiosi con Matlab attraverso uno specifico modulo del quale si parlerà estesamente in seguito: Comsol e in particolare l'interfaccia "*Comsol with Matlab*".

Risolto il grosso problema della scelta dei principali software da utilizzare, si giunge all'ultimo importante nodo: il dialogo tra Matlab, Comsol e un sistema *haptic*. Come si può notare dall'elenco stilato nel primo capitolo, l'unica libreria con caratteristiche

interessanti e in grado di interfacciarsi con Matlab è la "*Haptik Library*", pertanto la scelta in questo senso è stata di fatto obbligata.
Nei paragrafi che seguono, si approfondiranno il ruolo e le potenzialità di ciascuno degli strumenti di lavoro scelti.

3.7.1 Matlab

Matlab è un famosissimo strumento di calcolo e programmazione allo stesso tempo potente e versatile [31]. È utilizzato per i più disparati scopi, che vanno dalla semplice rappresentazione di grafici allo studio di frequenze proprie e modi di vibrare delle strutture. A differenza del linguaggio C e Java, quello utilizzato in Matlab, pur avendo molte caratteristiche in comune con entrambi, è il più noto in ambito didattico. Poiché ci si propone di realizzare un'applicazione facilmente comprensibile anche da utenti con conoscenze di programmazione non avanzate e da studenti, ciò costituisce certamente un punto di forza, ma non il più importante. L'elemento fondamentale che fa di Matlab il vero e proprio fulcro di questo lavoro, è che esso riesca da un lato a interfacciarsi alla perfezione con Comsol, consentendo, come si vedrà, un'incredibile semplificazione sia dal punto di vista computazionale sia di modellazione, attraverso l'utilizzo di opportune librerie e, dall'altro lato, Matlab può essere in grado di comunicare con gli hardware *haptic* per mezzo dell'Haptik Library. In definitiva, Matlab è quindi lo strumento attraverso il quale si può creare una connessione tra *haptic* e FEM, che è appunto lo scopo che ci si è prefissi. La versione utilizzata in questo lavoro è la R2012a.

3.7.2 Comsol

Comsol è uno dei software per la modellazione agli elementi finiti utilizzati in alcuni corsi di ingegneria meccanica [21]. A differenza di altri software molto popolari al Politecnico di Milano, come ad esempio Abaqus, possiede un'interfaccia nativa realizzata per la comunicazione diretta con Matlab, chiamata "*Comsol with Matlab*". L'utente può quindi realizzare un modello in Comsol e salvare il risultato in un file .m, ossia nel tipico formato di Matlab. In seguito, avviando l'interfaccia "*Comsol with Matlab*", sarà avviata automaticamente una particolare routine di caricamento di librerie, al termine della quale sarà possibile avviare da Matlab il file .m precedentemente salvato, e visualizzare lo stesso risultato visto in Comsol in una finestra grafica di Matlab.

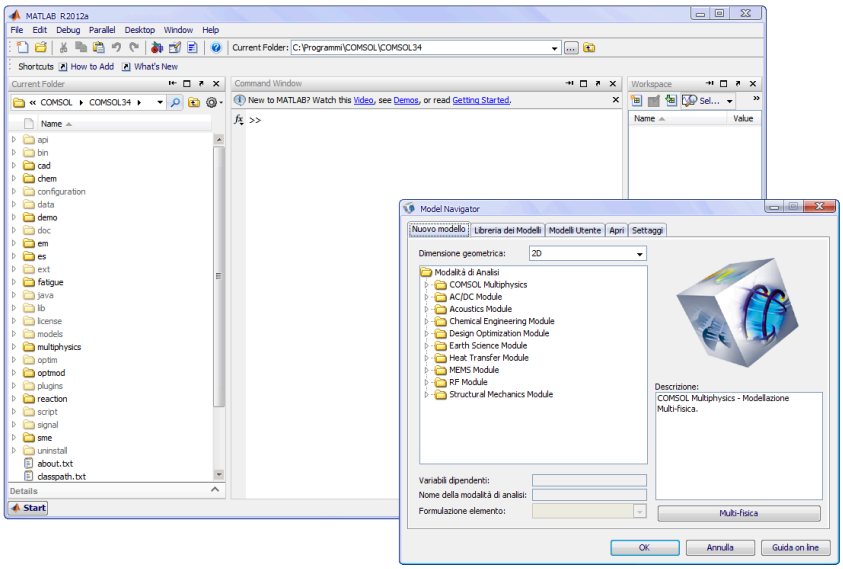
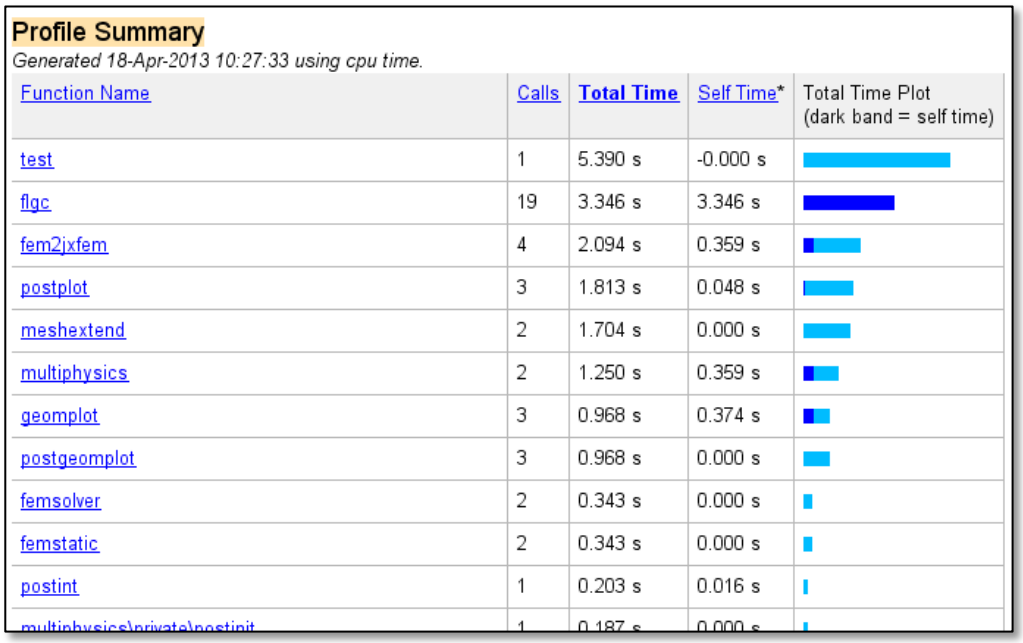


Figura 3.10: finestre aperte automaticamente dopo l'avvio di "Comsol with Matlab"

La parte di analisi agli elementi finiti potrebbe essere scritta senza eccessive difficoltà direttamente in Matlab, ma ciò comporterebbe svantaggi da molti punti di vista: l'utente non realizzerebbe il modello della struttura all'interno di un'interfaccia grafica, ma interamente sotto forma di codice di programmazione. Inoltre, in genere gli algoritmi per l'analisi FEM scritti in linguaggio Matlab sono complessi e lenti, e sfruttano funzioni salvate in molti *file* separati e dunque più difficili da comprendere e modificare. Infine, il risultato fornito dall'analisi FEM così ottenuta non sarebbe, "certificato". In rete si possono trovare molti algoritmi *open-source* costruiti da appassionati o da esperti del settore, ma è difficile capire quali siano davvero affidabili. Comsol, sotto questo punto di vista, poiché nasce come strumento ingegneristico, possiede un valore intrinseco per quanto riguarda l'affidabilità e l'attendibilità dei risultati. Sono state testate molte versioni di Comsol, a partire dalle più recenti, allo scopo di individuare quella che richiedesse il minor tempo di calcolo, fattore fondamentale per un processo in tempo reale. Per testare le prestazioni dei file .m generati con Comsol, si è deciso di utilizzare il "*Profiler*" di Matlab, un modulo interno sviluppato proprio per tale scopo. È con una certa sorpresa che si è scoperta una notevole lentezza delle versioni più recenti del programma. Analizzando le performance dei file .m generati con Comsol 4.3a (la più recente in commercio al momento), si nota che molte sotto-funzioni fanno un ricorso diretto a librerie Java, il che, molto probabilmente, rende il processo risolutivo più accurato ma anche molto lento (per le esigenze richieste).

Quella scelta per il proseguimento del lavoro è l'ormai datata versione 3.4 che, come si può notare dal *benchmark*, è in grado di risolvere lo stesso semplice modello di prova (chiamato "test" e costituito da una semplice barretta sottoposta a trazione) in un tempo considerevolmente più basso: circa 5,4 secondi contro 16,8 secondi ottenuti con la stessa struttura e lo stesso calcolatore utilizzando la versione 4.3.



Profile Summary

Generated 18-Apr-2013 10:27:33 using cpu time.

Function Name	Calls	**Total Time**	Self Time*	Total Time Plot (dark band = self time)
test	1	5.390 s	-0.000 s	
flgc	19	3.346 s	3.346 s	
fem2jxfem	4	2.094 s	0.359 s	
postplot	3	1.813 s	0.048 s	
meshextend	2	1.704 s	0.000 s	
multiphysics	2	1.250 s	0.359 s	
geomplot	3	0.968 s	0.374 s	
postgeomplot	3	0.968 s	0.000 s	
femsolver	2	0.343 s	0.000 s	
femstatic	2	0.343 s	0.000 s	
postint	1	0.203 s	0.016 s	
multiphysics\private\postinit	1	0.187 s	0.000 s	

Figura 3.11: Analisi delle prestazioni del file ottenuto con Comsol 3.4 utilizzando il "Profiler" di Matlab

Il tempo di risoluzione è stato in seguito ulteriormente migliorato con particolari tecniche di "pulizia" e "segmentazione" dell'algoritmo, descritte nel Capitolo 5.

3.7.3 Haptik Library

La scelta di utilizzare l'interfaccia "*Comsol with Matlab*" consente di dare una prima risposta alle esigenze di trovare un opportuno ambiente di programmazione e una tecnica di risoluzione FEM. A questo stadio rimane tuttavia ancora irrisolto il problema costituito dal dialogo tra il sistema *haptic* e tali software. Una prima idea, rivelatasi in seguito complessa e dispersiva, è stata quella di utilizzare due programmi in parallelo, uno dedicato esclusivamente alla parte *haptic*, uno alla parte di simulazione e visualizzazione, il tutto su uno o su due calcolatori distinti. Dopo aver svolto diversi approfondimenti e test, si è deciso di abbandonare questa strada, principalmente per le problematiche legate alla sincronizzazione tra il software dedicato al sistema *haptic* e quelli dedicati alla simulazione/visualizzazione. Si è deciso quindi di cercare una soluzione più semplice e pratica, allo scopo di creare un pacchetto completo in grado di lavorare su una sola macchina. La risposta al problema è stata la libreria *Haptik Library*. Come precisato nel capitolo precedente, si tratta di una raccolta di librerie e dll creati con l'intento di rendere compatibili la maggior parte dei sistemi *haptic* in commercio con il linguaggio C, ma anche con Matlab [17]. Caricando l'Haptik Library in Matlab si è infatti in grado di svolgere poche ma essenziali operazioni con l'hardware *haptic* a disposizione.

I passi successivi del lavoro, descritti nei prossimi capitoli, trattano della modellazione di un primo semplice modello in Comsol, del salvataggio dei file in un formato compatibile con Matlab e alla loro ottimizzazione per il funzionamento in tempo reale e con il dispositivo *haptic* scelto.

Capitolo 4: Modellazione ed esportazione

In questo capitolo saranno descritti tutti i passi che hanno portato dalla realizzazione di un semplice modello in Comsol e al successivo salvataggio in un formato compatibile con Matlab.

4.1 Modellazione con visualizzazione 2D in Comsol

Per rendere più semplice e intuitiva la comprensione della fase di modellazione e delle successive, sarà seguita passo-passo la modellazione di una semplice barretta a sezione quadrata, vincolata a terra con una cerniera e un carrello o con 2 cerniere, e soggetta ad uno spostamento imposto in uno dei suoi nodi. Questa semplice struttura 2D, le cui caratteristiche geometriche e gli schemi di vincolo sono riassunti nell'immagine seguente, è inclusa anche nell'applicazione finale (Modello 2).

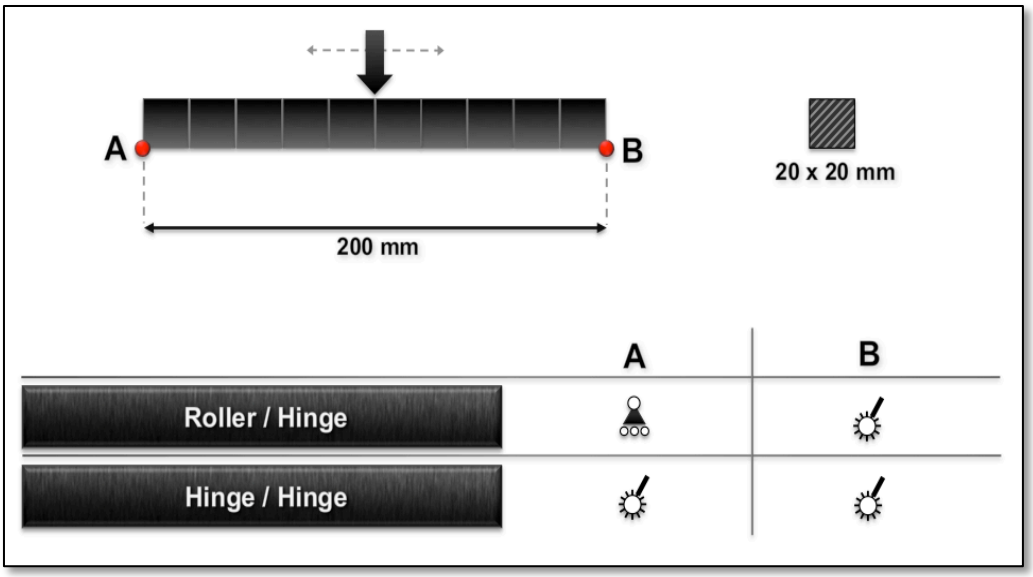


Figura 4.1: Schermata riassuntiva del modello 2 nell'applicazione finale

Una volta avviato il programma "*Comsol with Matlab*" si aprono Matlab, che per il momento può essere ridotto a icona, e la finestra iniziale di Comsol, che è quella che verrà utilizzata per la modellazione.

Figura 4.2: Icona di "Comsol with Matlab"

Accedendo al sottomenu "*Settings*", è possibile impostare diversi parametri, tra i quali il colore dello sfondo e il sistema di misura che verranno utilizzati nei passi successivi.

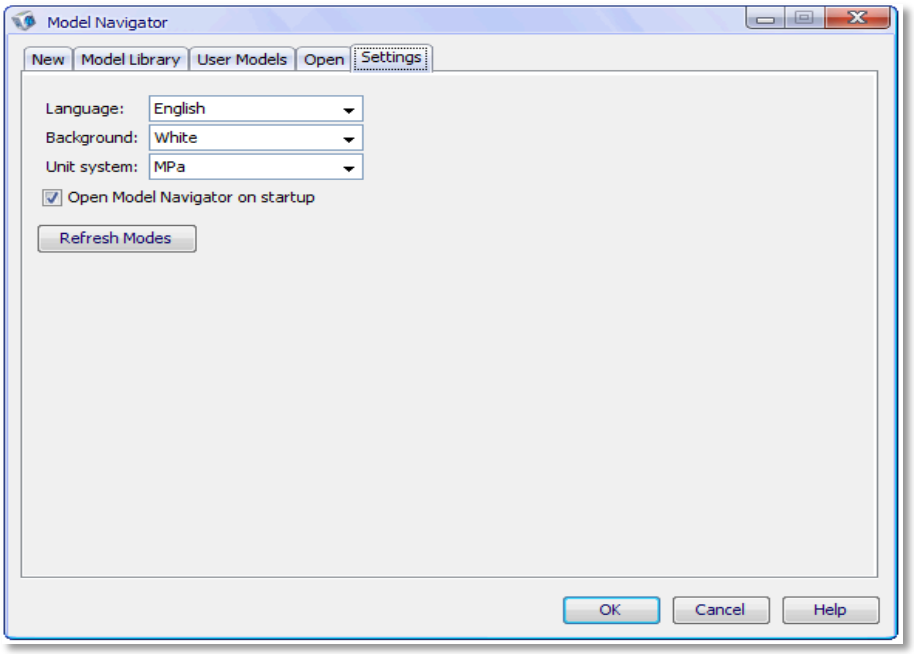


Figura 4.3: Schermata iniziale delle impostazioni di Comsol

Il sistema di misura da scegliere è MPa, che consente di avere come unità di lunghezza il millimetro, di forza il Newton e di peso la tonnellata. Gli sforzi e i moduli elastici saranno invece espressi in MPa. Si passa quindi alla scelta del tipo di modello da costruire. Tra i molti disponibili, quelli d'interesse sono "*plane stress*" e "*plane strain*", che si trovano entrambi all'interno del modulo di meccanica strutturale. In particolare, si eseguirà un'analisi di tipo "*plane stress*" quando una delle tre dimensioni dell'oggetto modellato è trascurabile rispetto alle altre, come ad esempio accade per le lamine di sottile spessore. L'analisi "*plane strain*", al contrario, sarà utilizzata per elementi piuttosto "tozzi", nei quali la terza dimensione non può essere trascurata. Nell'esempio in questione, come si può notare dalle dimensioni riportate in Figura 3.1, la barretta ha una sezione quadrata con lato di 20 mm. Pertanto, giacché lo spessore non è trascurabile rispetto alle altre dimensioni, si sceglie "*Plane Strain*" dal menu. Poiché i modelli 2D sono piuttosto "leggeri", l'analisi sarà svolta con elementi di tipo quadratico.

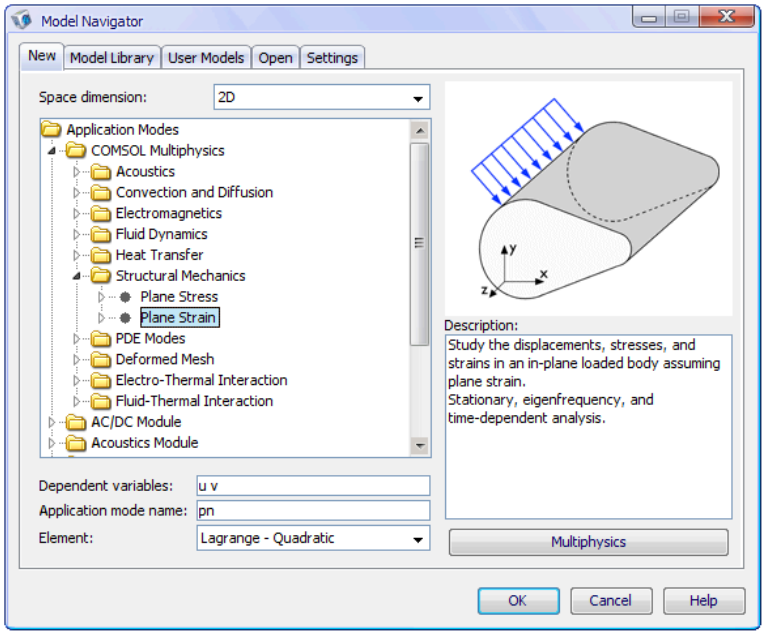


Figura 4.4: Schermata iniziale delle impostazioni di Comsol

Terminata la fase d'inizializzazione, sarà avviata la classica interfaccia grafica di Comsol. In Figura 3.1 si nota che la barretta è composta da 10 "cubetti", che creano una serie di domini interni con "nodi" che potranno essere utilizzati, come si vedrà in seguito, per l'applicazione di carichi e vincoli. Per facilitare la costruzione del modello, si accede al menu:

Options ➔ Constants...

Qui si dichiara per comodità una costante "L" pari a 20 mm descritta come "lato base".

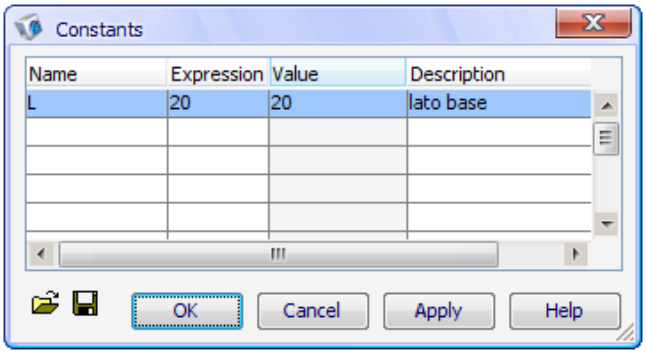


Figura 4.5: Dichiarazione delle costanti

A questo punto si può procedere con la costruzione vera e propria dell'oggetto. Si comincia disegnando il primo quadrato. Se si seleziona:

Draw ➔ Specify Objects ➔ Rectangle...

si apre un menu che consente di impostare le proprietà geometriche del rettangolo. In questo caso s'impongono "*Width*" e "*Height*" pari a "L".

La posizione del quadrato può essere impostata dall'angolo inferiore sinistro, imponendo che questo abbia coordinate (0,0). Alla voce "*Style*" si deve fare attenzione a selezionare "*Solid*".

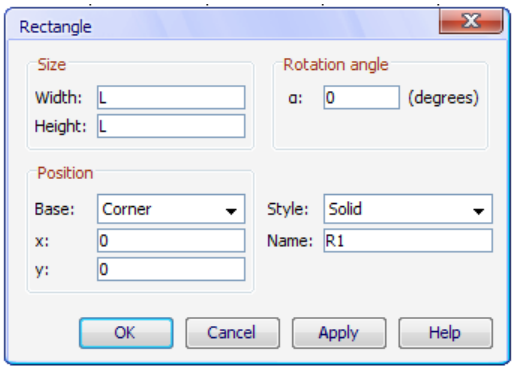


Figura 4.6: Impostazione delle proprietà geometriche

Dopo aver cliccato su "OK", sarà visualizzato il primo quadrato. Si procede quindi ripetutamente, disegnando allo stesso modo gli altri quadrati, uno accanto all'altro. Per farlo è necessario ovviamente variarne la posizione, imponendo ad esempio per il secondo quadrato x=L, per il terzo x=2*L e così via, lasciando sempre y=0. Dopo aver disegnato dieci quadrati il risultato sarà il seguente:

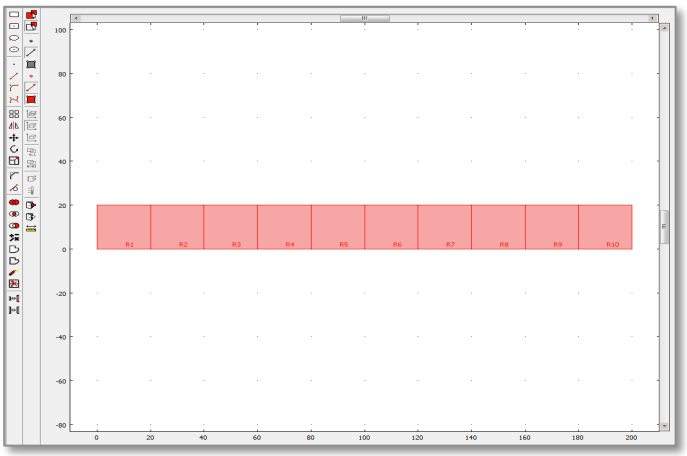


Figura 4.7: Disegno ultimato in Comsol

Si passa poi all'impostazione delle caratteristiche del materiale. Si accede quindi al menu "*Material*" dal seguente percorso:

Physics ➔ Subdomain Settings ➔ Material…

Si aprirà una finestra nella quale sarà richiesto di selezionare i sottodomini cui applicare le caratteristiche. Si selezionano qui tutte le entità geometriche create nella fase precedente (si coloreranno di rosa). Come parametri del materiale si possono impostare, ad esempio, quelli tipici dell'acciaio:

- E (modulo elastico) = **210000 MPa**
- ν (coefficiente di Poisson) = **0.3**
- ρ (densità) = **7870 [Kg/m^3]**

In questa fase è importante notare che, poiché il sistema di misura scelto imposta in automatico la tonnellata come unità di misura del peso e il millimetro per la lunghezza, di default la densità sarà in [t/mm^3]. Per questo motivo, è necessario "forzare" il programma a utilizzare chilogrammi e metri, indicando dopo il valore numerico la dicitura [Kg/m^3], al fine di ottenere le corrette sollecitazioni. L'ultimo parametro da impostare è lo spessore della barretta, in questo caso pari a 20 mm. S'impone quindi:

- thickness = **20 mm**

Se tutto è andato a buon fine, il menu dovrebbe apparire come nell'immagine seguente:

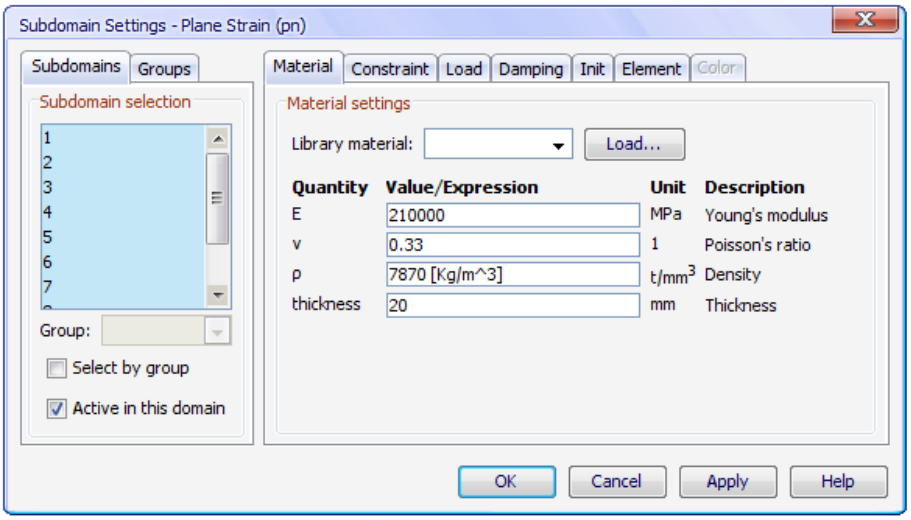


Figura 4.8: Impostazioni delle caratteristiche del materiale

Si passa ora alla definizione dei vincoli e del carico. Entrambe le entità saranno imposte in alcuni punti nodi della struttura. Poiché Comsol permette di disporre vincoli e condizioni di carico o spostamento imposto anche su lati e sottodomini, è pertanto necessario scegliere l'opzione corretta dal menù *Physics*:

Physics ➔ Point Settings...

Nella schermata che seguirà, sarà richiesto di selezionare i punti sui quali applicare i vincoli (*Constraints*). Si supponga di voler imporre il classico schema di vincolo "carrello-cerniera". Si seleziona prima di tutto il nodo 1 dal menu "*Point Selection*". Si noterà che il nodo 1 corrisponde all'estremo inferiore sinistro della barretta, che si colorerà di rosso. Se si vuole imporre in quel punto un carrello, basterà ora imporre:

- R_y = **0 mm**

Ciò consente di azzerare gli spostamenti del nodo 1 lungo la direzione y (verticale), lasciando liberi gli spostamenti lungo x e le rotazioni nel piano xy. Allo stesso modo, selezionando l'estremità inferiore destra della barretta, o equivalentemente selezionando il nodo 21 dal sottomenu "*Point Selection*", si potrà imporre un vincolo di tipo cerniera impostando:

- R_x = **0 mm**
- R_y = **0 mm**

In questo modo, infatti, si azzereranno le traslazioni nelle direzioni x e y del nodo 21, ma saranno comunque consentite le rotazioni nel piano xy.

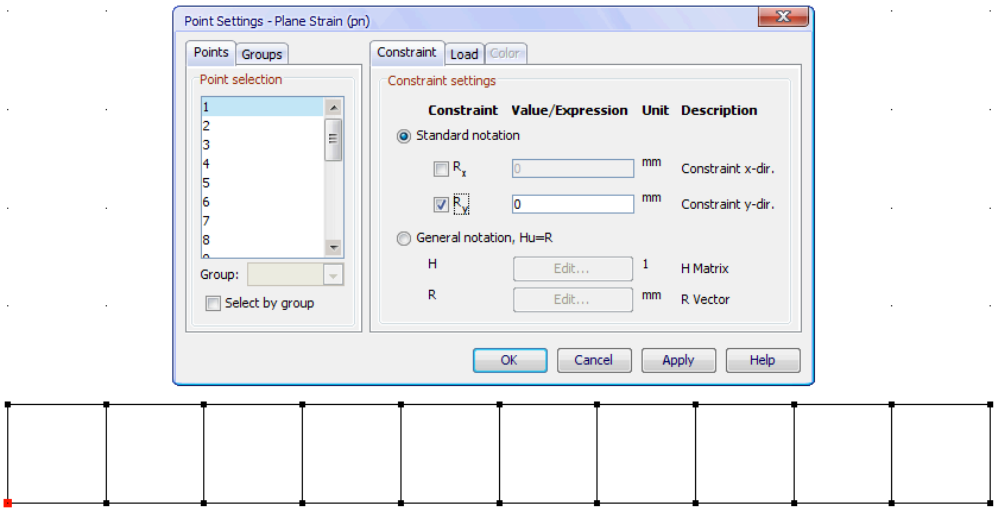


Figura 4.9: Definizione del vincolo carrello nodo 1

Si può quindi procedere, in modo del tutto analogo, alla definizione del carico. Si supponga di voler applicare una forza di 500 N in direzione y, orientata verso il basso, nel nodo intermedio superiore della barretta. Si accede al menu:

Physics ➔ Point Settings...

Qui si deve fare attenzione a selezionare il sottomenu "*Load*". Si seleziona quindi il nodo intermedio direttamente sulla struttura (o equivalentemente si seleziona il nodo 12 dal menu a tendina "*Point selection*") e si pone:

- F_x = **0 N**
- F_y = **-500 N**

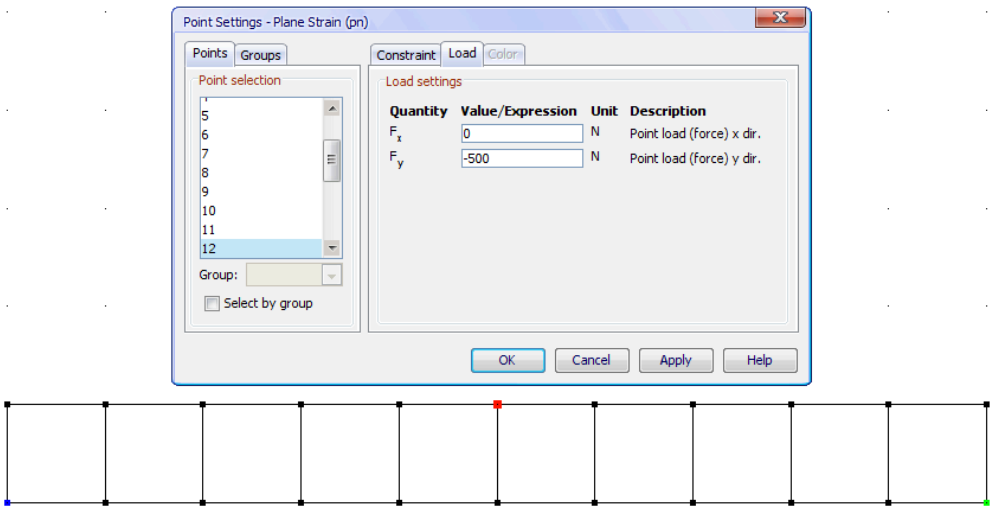


Figura 4.10: Definizione del carico nel nodo 12

Si continua ora con le fasi finali della modellazione in Comsol. Si accede al menu di impostazione della mesh dal seguente percorso:

Mesh ➔ Free Mesh Parameters…

Si aprirà una finestra nella quale sono richieste le impostazioni per la mesh da attribuire alla struttura. La mesh può avere dimensioni predefinite o essere definita dall'utente. In questo caso è stata impostata una mesh di tipo "*Normal*" dal sottomenu "*Global*", lasciando invariati tutti gli altri parametri.

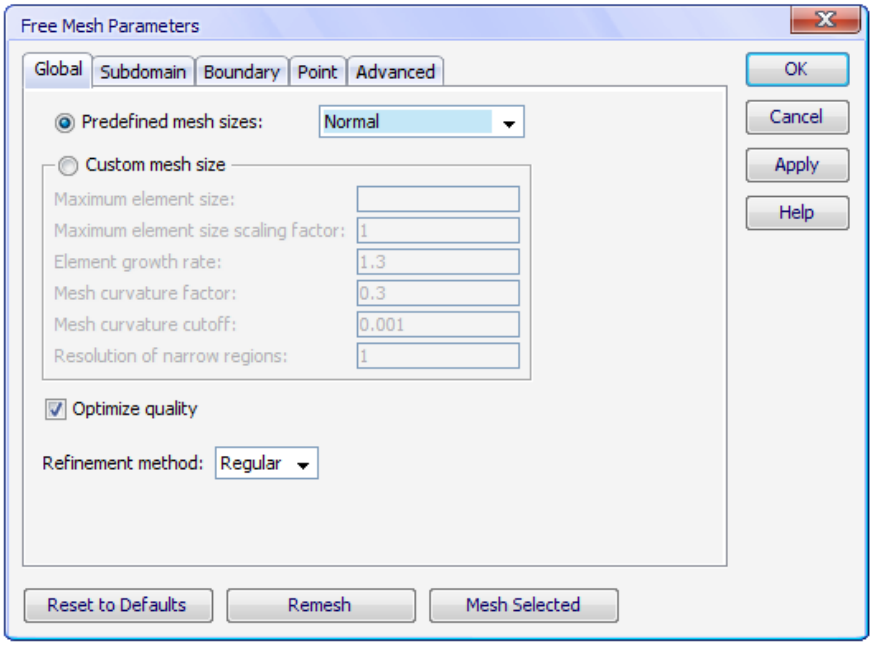


Figura 4.11: Definizione dei parametri della mesh

Dopo aver selezionato tutti i sottodomini, se si clicca su "*Mesh Selected*", si noterà l'immediata creazione della mesh all'interno della struttura, così come appare nell'immagine seguente:

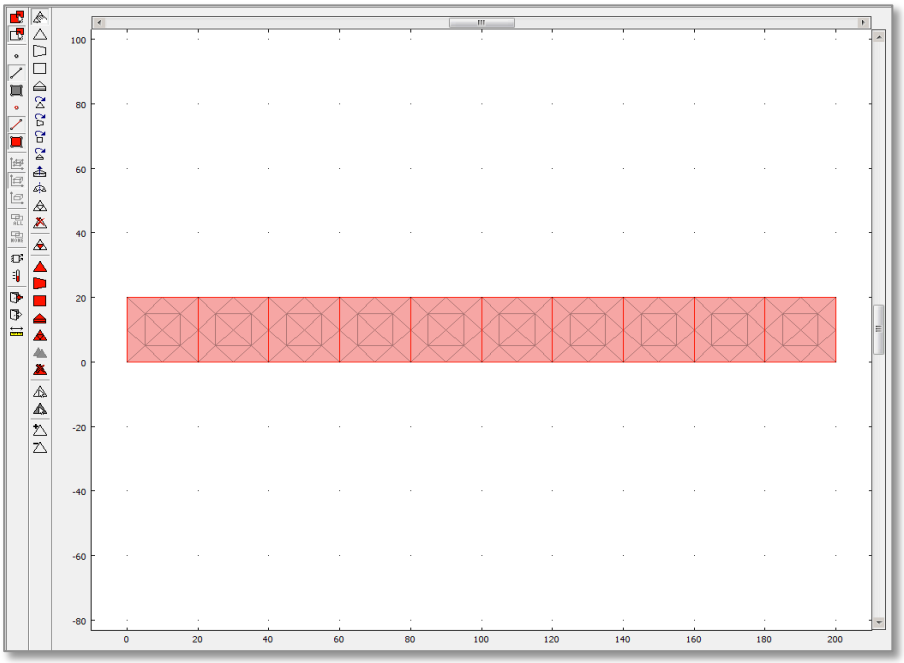


Figura 4.12: Applicazione della mesh alla struttura

Se si ritiene che la mesh sia troppo rada, è possibile tornare allo step precedente e selezionare una mesh di dimensioni "*Fine*", "*Finer*" o "*Extra Finer*" secondo le esigenze, tenendo conto che una mesh più fitta offre un risultato più accurato, ma richiede normalmente tempi più alti di risoluzione. Per molti modelli con visualizzazione 2D dell'applicazione finale è stata usata una mesh di tipo "*Normal*", in considerazione del fatto che questa impostazione consente di avere buona accuratezza e al contempo discrete performance fintanto che il numero degli elementi della mesh si mantiene intorno al centinaio.

L'ultima fase della modellazione prevede la definizione delle entità da visualizzare. Per regolare tali parametri si accede dunque al menu di *postprocessing*:

Postprocessing ➔ Plot parameters…

Nel sottomenu "*General*" si deve fare attenzione che sia presente la spunta su "*Deformed Shape*", in modo che sia visualizzata la configurazione deformata della struttura.

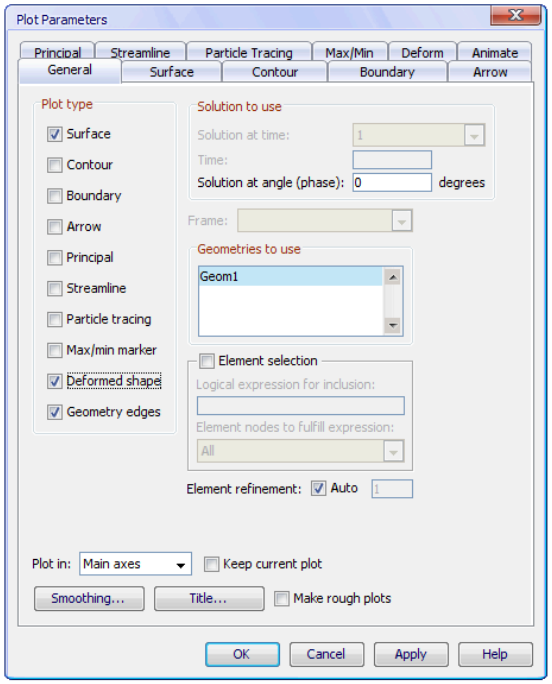


Figura 4.13: Impostazione dei parametri di postprocessing

Accedendo inoltre al sottomenu "*Deform*", è possibile regolare il fattore di scala per la rappresentazione della deformata. Se si desidera ad esempio raddoppiare la rappresentazione della deformazione, si può imporre "2" come fattore di scala. Ciò torna utile per amplificare e riuscire a distinguere visivamente deformazioni molto piccole.

In questo caso si imposta "auto" come fattore di scala. Nel sottomenu "*Surface*" si può invece stabilire quale quantità debba essere rappresentata sulla superficie della struttura. In questo caso si imposta:

- Predefined quantities: **von Mises stress**
- Expression: **mises_pn**
- Unit: **Mpa**

In tal modo, i colori generati sulla superficie daranno un'idea precisa della distribuzione degli sforzi di von Mises. Nella stessa schermata si imposta come "*Colormap*", la classica gamma denominata "*jet*" costituita da 1024 colori:

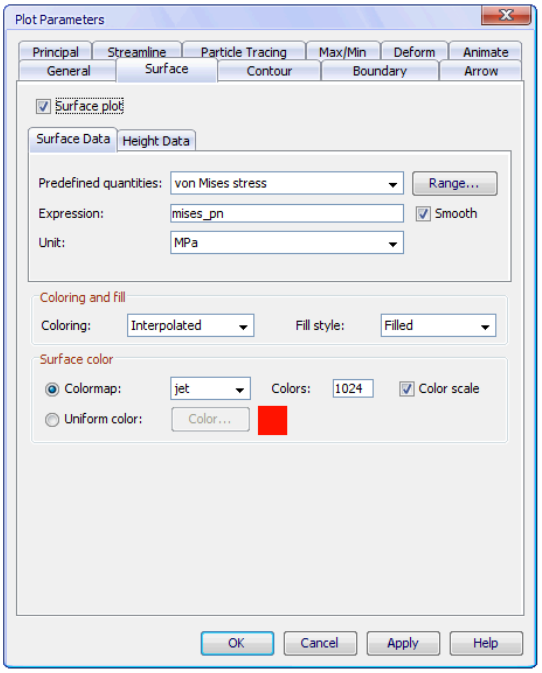


Figura 4.14: Impostazione dei dati per la superficie e della colormap

Si giunge quindi all'ultima fase, quella di avvio della simulazione, al termine della quale si potrà valutare la bontà del modello visualizzando il risultato. Per avviare la risoluzione della struttura si segue il percorso:

Solve ➔ Solve Problem

Immediatamente sarà avviata la risoluzione del modello e se tutto è andato a buon fine, sarà visualizzata la seguente soluzione:

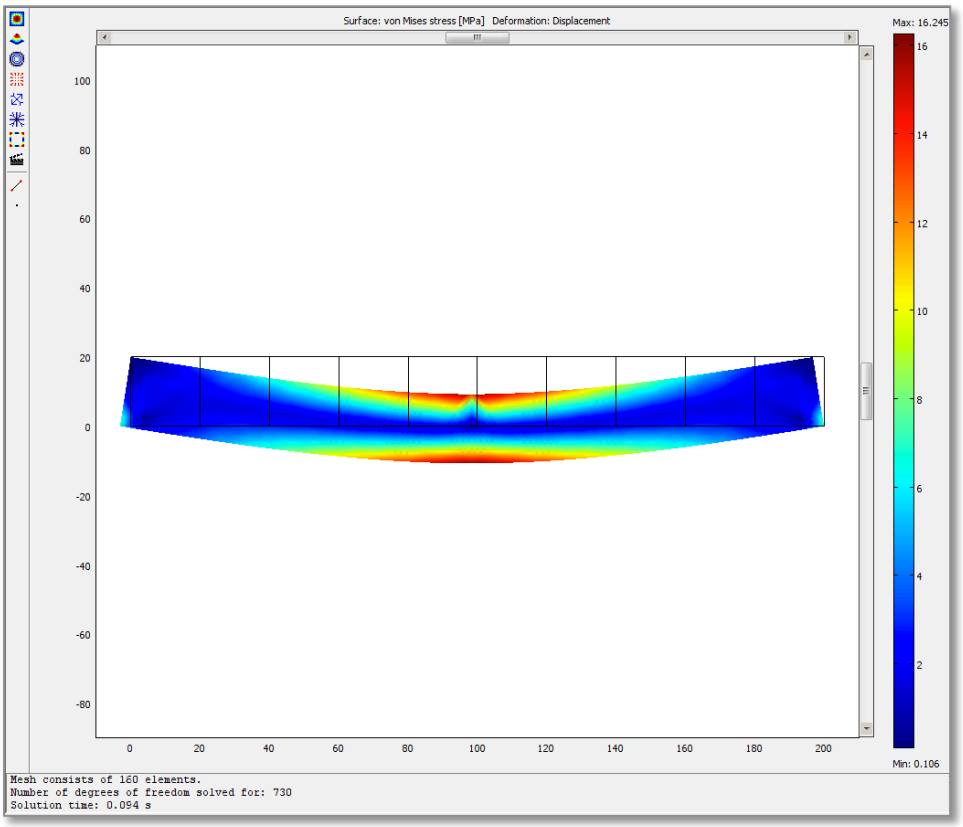


Figura 4.15: Visualizzazione del risultato (deformata e sforzi di von Mises)

Da una semplice valutazione visiva si può capire immediatamente se il risultato è quello atteso, o se è stato commesso qualche errore. La maggior parte degli errori normalmente riguarda le impostazioni riguardanti i carichi e i vincoli. In questo caso, come si nota da Figura 3.15, la soluzione fornita da Comsol sembra coerente con i vincoli e con il carico imposti. Si nota in particolare che il carrello (nell'angolo in basso a sinistra), si è spostato lungo l'asse x, trasferendo in direzione orizzontale parte della deformazione e delle sollecitazioni generate dal carico. Anche la cerniera posta nell'angolo inferiore destro della barretta si comporta come previsto.

Da questa schermata sono inoltre desumibili due altre importanti informazioni. Sopra la *colormap* è presente la seguente indicazione:

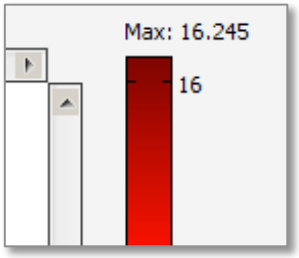


Figura 4.16: Valore massimo raggiunto dallo sforzo di von Mises

La dicitura "*Max: 16.245*", sta a indicare che con lo schema di vincolo e carico adottato il valore massimo raggiunto dallo sforzo di von Mises è di circa 16.2 MPa. Tale condizione, osservando Figura 3.15, è raggiunta nelle zone colorate in rosso scuro che, come si nota, in accordo con la teoria, si collocano all'intradosso, dove le fibre saranno sottoposte a compressione, e all'estradosso, dove le fibre sono invece sottoposte a trazione.

Una seconda importante informazione, che dà un'idea della velocità di risoluzione del software, è presente al di sotto dell'area grafica:

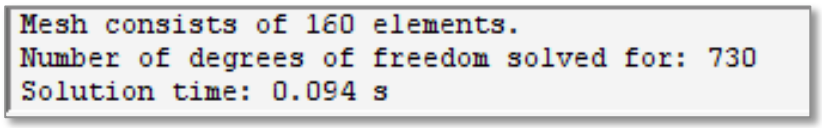


Figura 4.17: Tempo di risoluzione della struttura

Qui sono indicati il numero di elementi che costituiscono la mesh, il numero di gradi di libertà di cui la mesh gode e il dato più importante in previsione di una simulazione in *real-time*: il tempo di risoluzione. Senza ottimizzazione, una semplice struttura può essere risolta da Comsol in meno di un decimo di secondo. Ciò vuol dire che, allo stato attuale, senza ottimizzazione, si potrebbe in teoria già eseguire una simulazione a più di 10 fps.

4.2 Salvataggio del file

Dopo aver completato la parte di modellazione con Comsol e aver verificato la compatibilità del risultato ottenuto con quello atteso, è possibile salvare il modello in un file di formato .m, perfettamente compatibile con Matlab. Prima di procedere è opportuno osservare che quando un modello realizzato con Comsol viene salvato nel formato di Matlab, questo conterrà tutte le operazioni svolte dall'utente in ordine cronologico, compresi eventuali errori. Anche se nel capitolo seguente sarà descritta approfonditamente la fase di "pulizia" e editing del file, sarebbe opportuno realizzare la modellazione della struttura senza errori dall'inizio alla fine.

Ciò consente infatti di avere un file .m "pulito", ossia senza ambiguità e senza parti inutili. Per salvare il modello in un file .m si accede al seguente percorso:

File ➔ Save As…

Dopo aver selezionato l'opzione "Model M-file (*.m)" il file è stato salvato, in questo caso, con il nome "barretta". Nella stessa cartella è stato salvato per sicurezza lo stesso file anche nel formato di Comsol, così da poterlo riaprire ed eventualmente modificare in seguito, selezionando il formato "COMSOL Multiphysics Model File (*.mph; *.fl)".

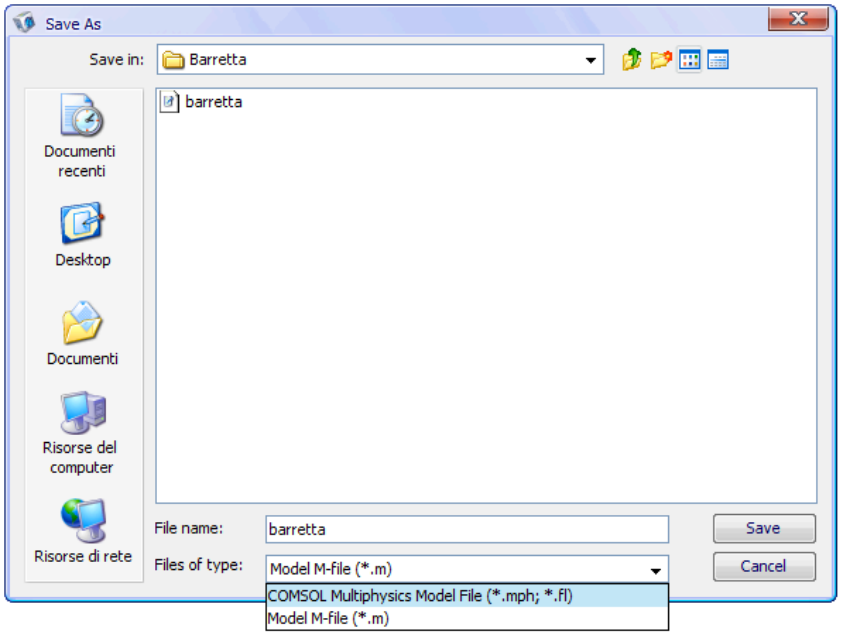


Figura 4.18: Salvataggio del modello nei formati di Matlab o Comsol

4.3 Apertura del file in Matlab

Si può ora accedere a Matlab, che al momento dell'apertura di "*Comsol with Matlab*" era stato ridotto a icona. Per prima cosa è necessario includere il file .m appena generato nella "*Current Folder*" di Matlab, ossia nella cartella di lavoro del software. Per farlo si seleziona la cartella contenente il file dal menu corrispondente:

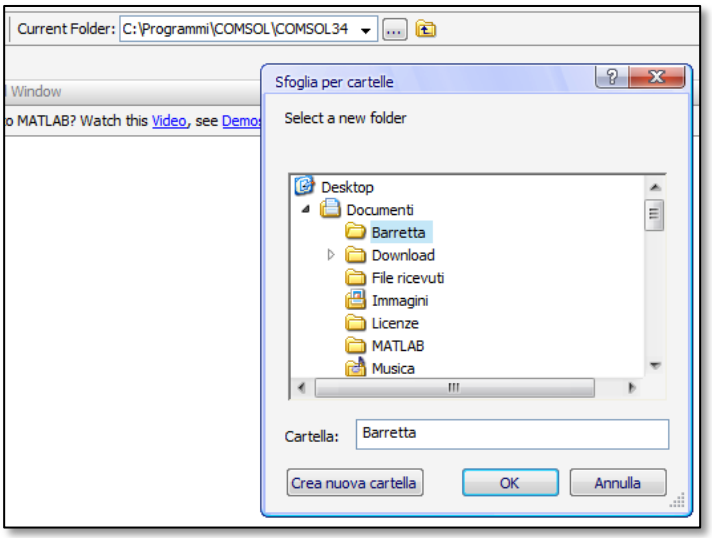


Figura 4.19: Selezione della corretta cartella di lavoro in Matlab

A questo punto nella finestra sinistra del software dovrebbero essere visibili i due file ("barretta.m" e "barretta.mph") salvati precedentemente. Il file nel formato di Comsol si può ignorare e si può aprire invece il file "barretta.m":

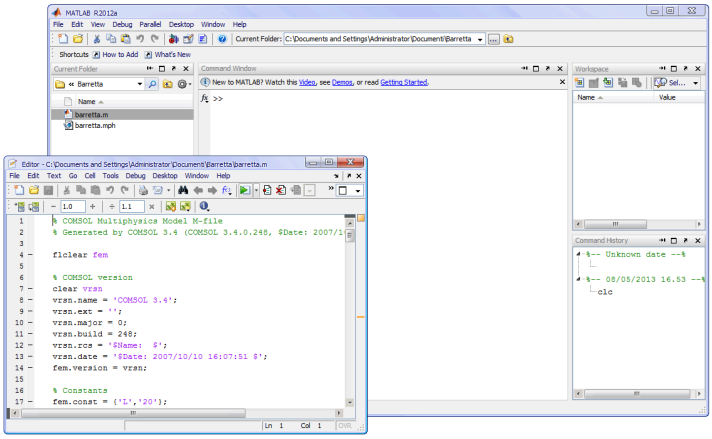


Figura 4.20: Apertura del file "barretta.m" in Matlab

Se si preme il tasto "*Play*" si avvierà il processo di risoluzione del modello all'interno di Matlab. Saranno mostrate diverse barre di caricamento, dopodiché sarà visualizzato lo stesso risultato già visto in Comsol all'interno di una finestra grafica di Matlab:

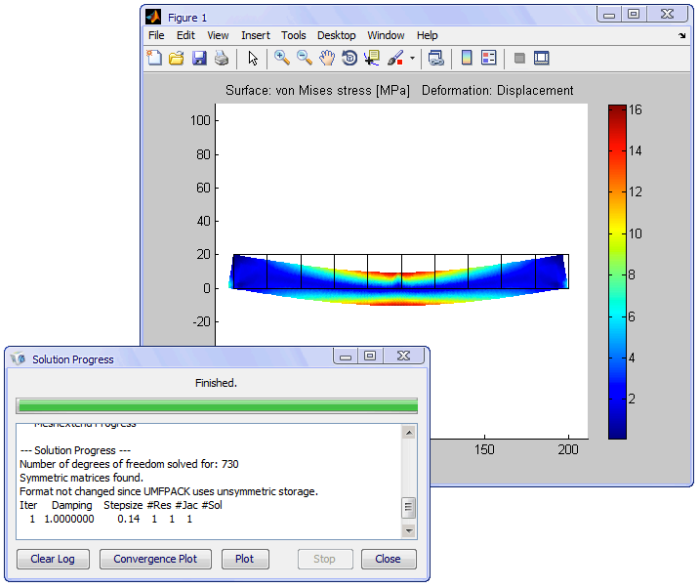


Figura 4.21: Soluzione della struttura in Matlab

4.4 Modellazione con visualizzazione 3D in Comsol

Viene descritto ora, a scopo di completezza, il procedimento per la modellazione ed esportazione di una struttura con visualizzazione 3D. Molti passaggi sono del tutto identici a quelli già visti nel capitolo precedente, cui si rimanderà per non creare inutili ripetizioni. La struttura che ci si propone di realizzare è una semplice trave incastrata a "T", con le caratteristiche geometriche (dalla norma UNI 5681) riportate in figure 3.22. Anche questo modello è presente nell'applicazione finale (Modello 10):

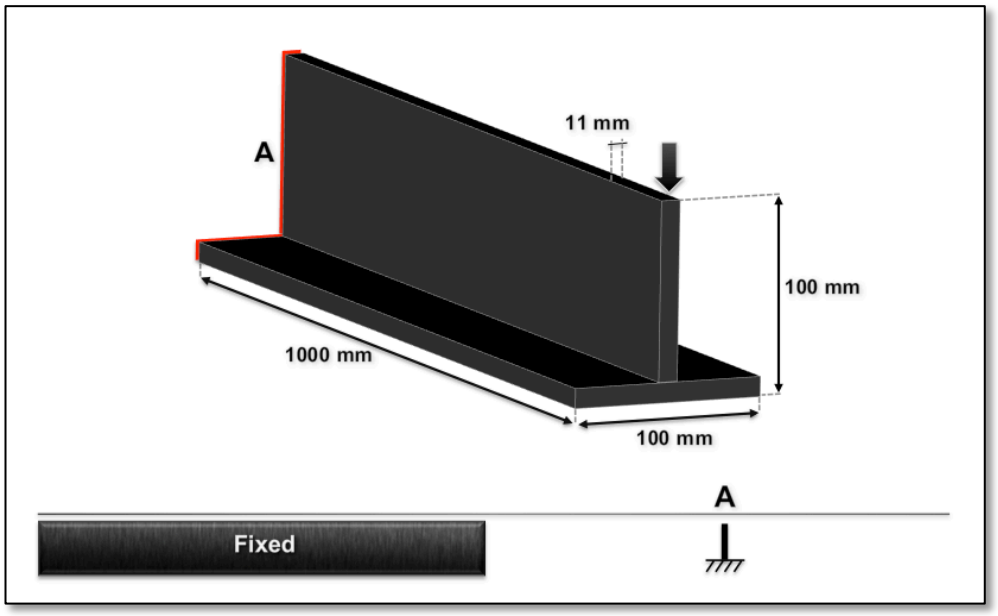


Figura 4.22: Schermata riassuntiva del modello 10 nell'applicazione finale

Si seguono le stesse indicazioni riportate nel paragrafo 3.1 per l'apertura di "Comsol with Matlab" e per l'impostazione del corretto sistema di misura. Nella finestra "*New*" si seleziona questa volta "*Space dimension: 3D*" e si sceglie, come fatto in precedenza, una analisi di tipo strutturale per un modello di tipo "*Solid, Stress-Strain*" e si imposta una analisi di tipo statica. Poiché i modelli 3D sono normalmente molto più "pesanti" a livello computazionale, è preferibile, in previsione della futura ottimizzazione dei tempi, scegliere un tipo di analisi lineare per gli elementi della mesh:

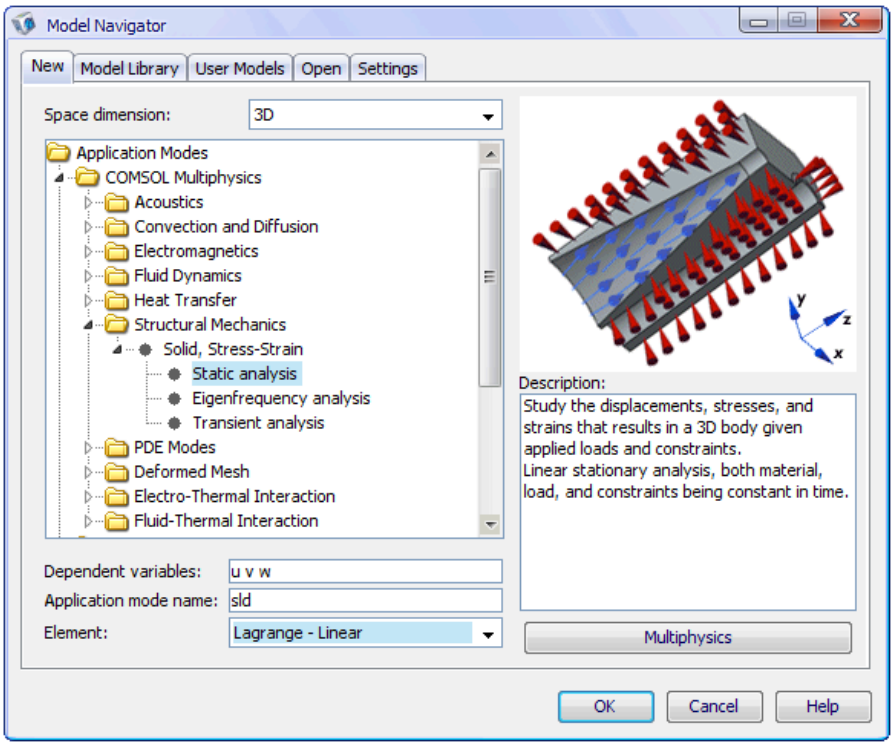


Figura 4.23: Schermata iniziale delle impostazioni di Comsol

Premendo su "OK" si aprirà una finestra grafica 3D, nella quale sono visualizzati i tre assi e i tre possibili piani di lavoro. Per disegnare la sezione della trave si seleziona:

Draw ➔ Work-Plane Settings…

Nella successiva finestra si seleziona il piano y-z, con origine sul piano 0. Premendo "OK" si ritornerà alla solita visualizzazione 2D, nella quale è possibile utilizzare i comandi di disegno all'interno del menu "*Draw*". Anche in questo caso è opportuno definire delle costanti corrispondenti ad altezza, larghezza e spessore della trave da realizzare. Utilizzando opportunamente il solo comando "*Rectangle*" due volte, si giunge al seguente risultato:

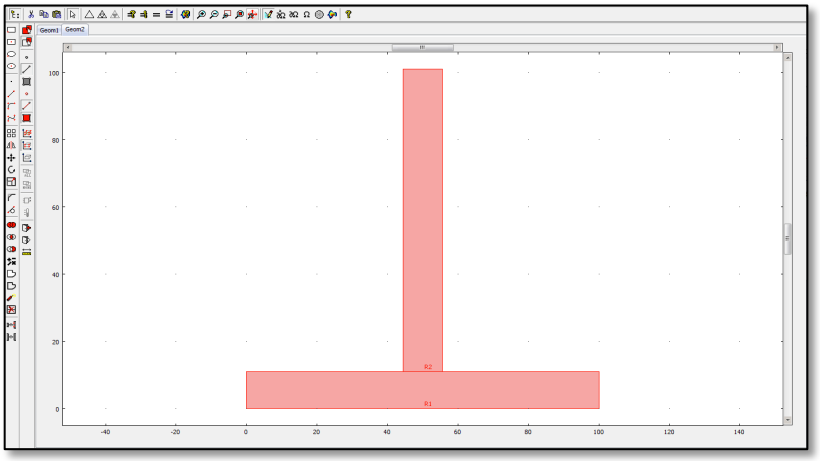


Figura 4.24: Sezione della trave

In caso di modelli più complessi, in particolar modo quando siano presenti raccordi o geometrie cave, può essere utile utilizzare il comando "*Create Composite Object*", disponibile all'interno del menu "*Draw*" e identificato dalla seguente icona:

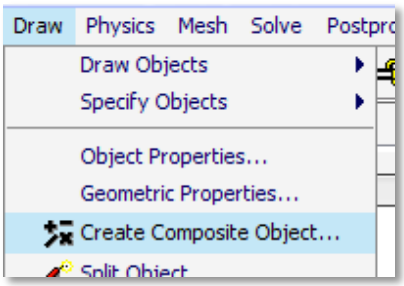


Figura 4.25: Comando per creare geometrie complesse

Accedendo a tale menu, si nota come sia possibile eseguire operazioni di unione e intersezione tra entità solide, tra curve o tra punti. Ciò può essere fatto utilizzando degli "*Shortcuts*", o scrivendo una semplice formula in cui si sommano o sottraggono le entità d'interesse. E' inoltre possibile eliminare i contorni interni alla geometria. Bisogna tuttavia prestare attenzione al comando "unione", perché può creare problemi nella successiva generazione della mesh in corrispondenza di eventuali spigoli vivi, come in questo caso. Pertanto tale funzione non è stata utilizzata per questo modello, ma si può tenere presente la possibilità di creare geometrie complesse in modo veloce utilizzando questi comandi.

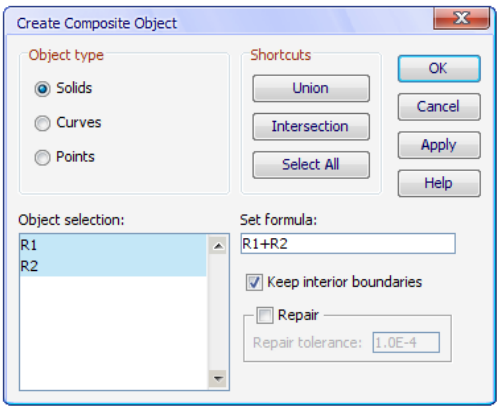


Figura 4.26: Esempio di unione di due solidi

Tornando al modello della trave, è ora necessario estruderla, tornando quindi alla visualizzazione 3D. Si segue quindi il percorso:

Draw ➔ Extrude…

Qui si selezionano entrambi i rettangoli e s'impone il valore "*Distance*" pari a 1000.

Dopo aver premuto "OK" si tornerà alla visualizzazione 3D, ottenendo un risultato del tutto identico all'immagine seguente:

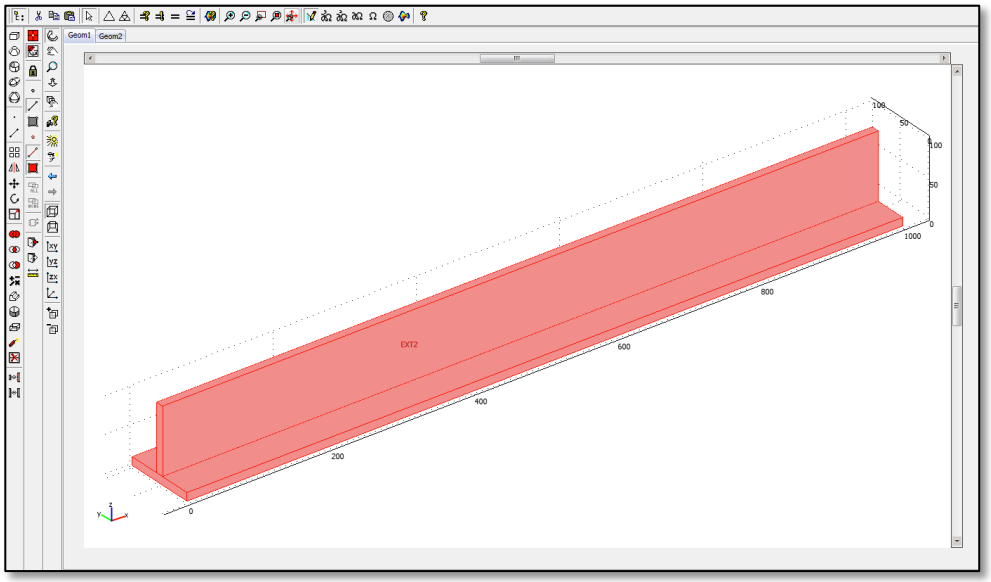


Figura 4.27: Risultato del comando "Extrude"

E' ora possibile, in modo del tutto analogo a quanto fatto nel capitolo precedente, impostare le caratteristiche del materiale, dal menu:

Physics ➔ Subdomain Settings ➔ Material…

S'impostano anche in questo caso i parametri tipici dell'acciaio. Si passa quindi alla definizione dei vincoli. Poiché la trave è incastrata in una delle sue estremità si va al menu:

Physics ➔ Boundary Settings…

Qui si selezionano le entità geometriche dell'estremo destro della trave, e s'impone:

- R_x = **0 mm**
- R_y = **0 mm**
- R_z = **0 mm**

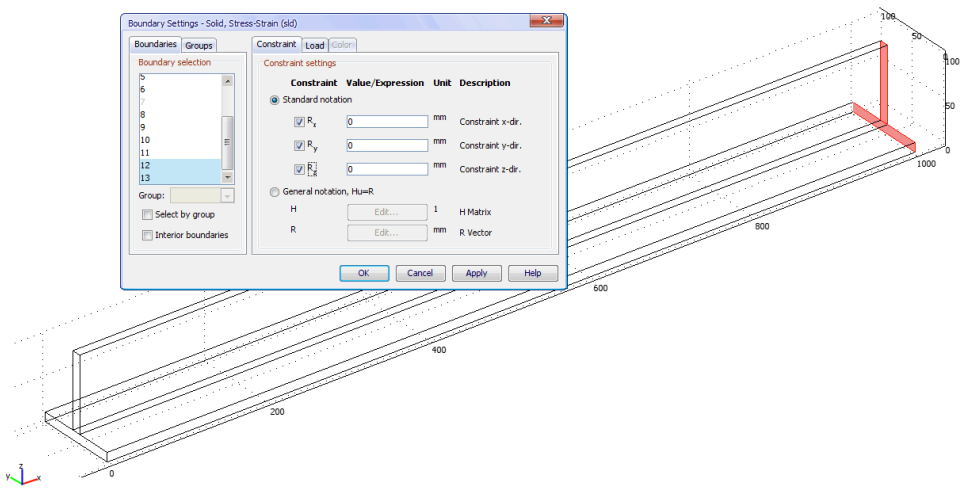


Figura 4.28: Impostazione del vincolo di incastro sulla faccia posteriore della barra

Diversamente da quanto fatto per il modello in visualizzazione 2D, si supponga di voler imporre uno spostamento dell'intera estremità libera della barra. Dal menu "*Boundary Settings…*" si selezionano le facce corrispondenti e si mette la spunta su "*General notation, Hu=R*". Sarà necessario impostare le matrici H e R del sistema per la risoluzione generale del problema, in modo da generare, ad esempio, uno spostamento verticale verso il basso pari a 3 mm. La matrice H riassume i gradi di libertà della struttura, R è il vettore degli spostamenti.

S'impostano quindi le matrici come nell'immagine seguente:

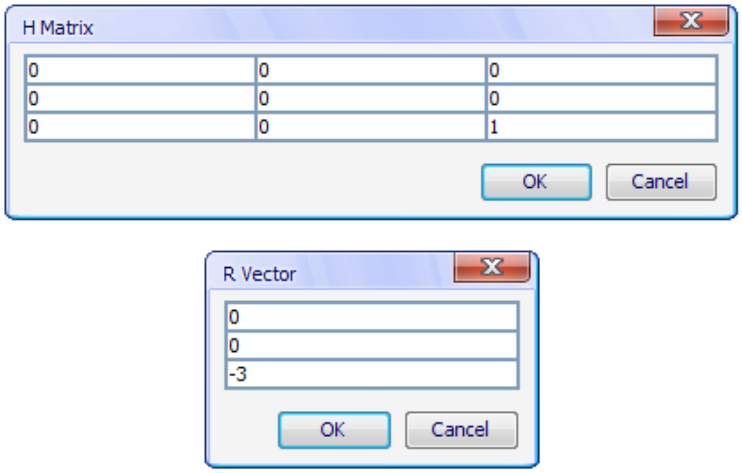


Figura 4.29: Impostazione delle matrici H e R

Infine, è necessario generare la *mesh* per la struttura modellata. Per selezionare tutti gli elementi della trave si segue il percorso:

Edit ➔ Select All

In seguito si accede al menu:

Mesh ➔ Free Mesh Parameters...

Sempre allo scopo di velocizzare il completamento delle operazioni che porteranno alla risoluzione del problema agli elementi finiti, è qui necessario adottare una mesh molto rada, il che, come si può facilmente intuire, non ha un effetto positivo sulla precisione e l'attendibilità del risultato finale.

Anche per questo motivo tutti i modelli 3D contenuti all'interno dell'applicazione finale saranno presentati solo come esempio delle possibili potenzialità del metodo proposto. Si seleziona pertanto una mesh di tipo "*Extremely coarse*" e si preme il tasto "*Mesh Selected*". Il risultato è il seguente:

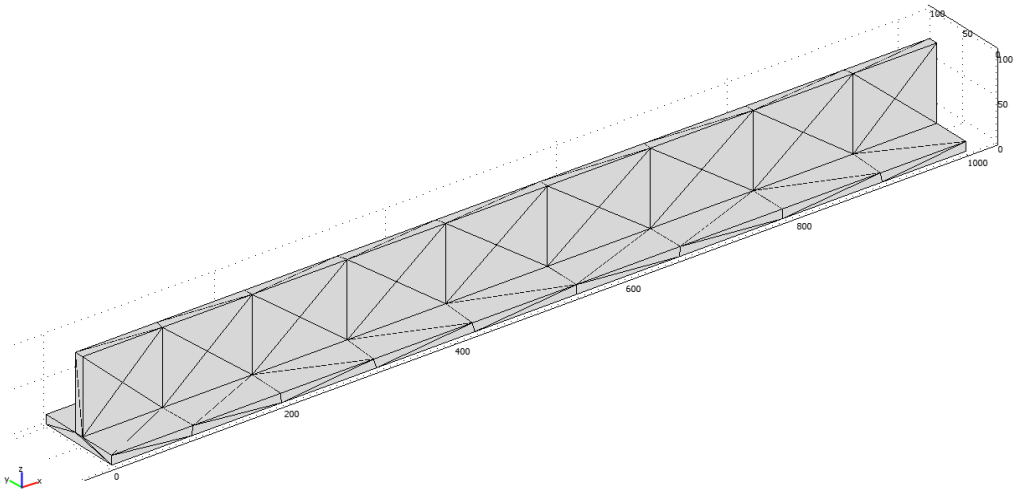


Figura 4.30: Mesh rada applicata alla struttura

L'ultima fase consiste nell'avviare la simulazione. Si va al menu:

Solve ➔ Solve Problem

La soluzione proposta dal software sarà la seguente:

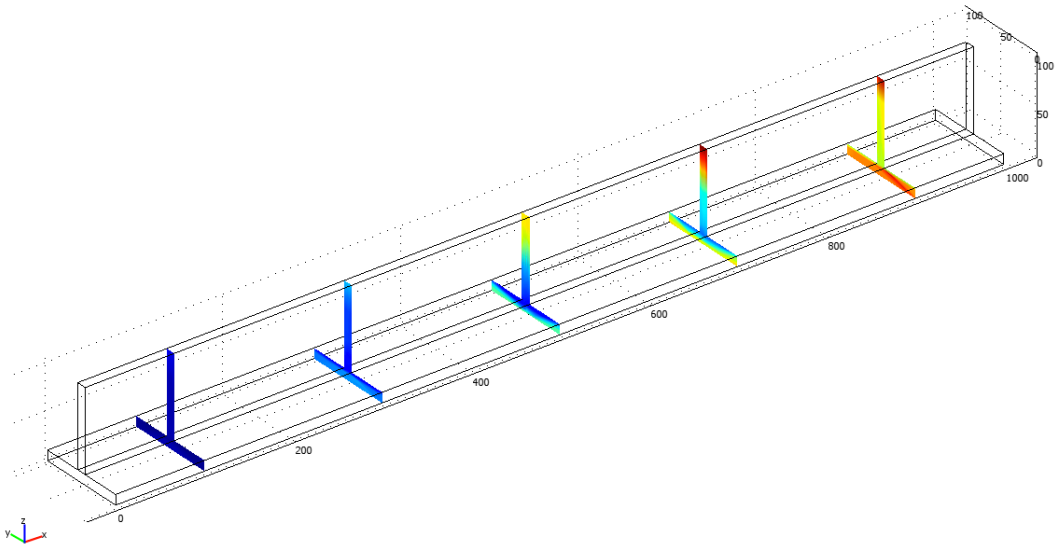


Figura 4.31: Soluzione proposta da Comsol

Come si nota, si tratta di una visualizzazione di "*slices*" della trave, all'interno delle quali viene rappresentato, con la solita mappa "*jet*" a 1024 colori, lo sforzo di von Mises. Si può tuttavia forzare il software a generare una visualizzazione più pratica, imponendogli di rappresentare lo stato di sforzo di von Mises non in sezione, bensì sulle superfici. Per farlo si segue il percorso:

Postprocessing ➔ Plot Parameters…

Nella finestra "*General*" si toglie la spunta a "*Slice*" e si mette invece a "*Boundary*" e a "*Deformed shape*". Si va poi alla finestra "*Boundary*". Come si noterà, la quantità predefinita è "*Total displacement*", e anche accedendo al menu a tendina non sembra possibile selezionare come quantità lo sforzo di von Mises.

Tuttavia ciò può essere comunque fatto imponendo nel campo "*Expression*" la dicitura "*mises_sld*", cosa che farà aggiornare anche il campo dell'unità di misura, che passerà automaticamente da mm a MPa:

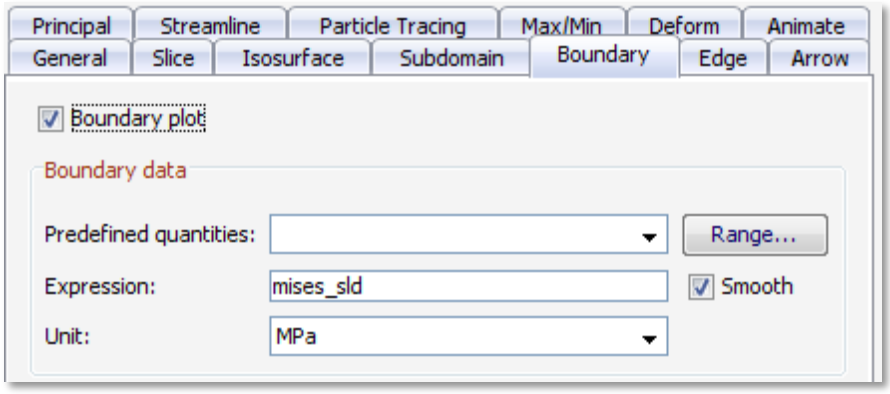


Figura 4.32: Impostazione della corretta espressione per le superfici

Premendo su "OK" se ne otterrà il risultato, molto più efficace, rappresentato nell'immagine seguente:

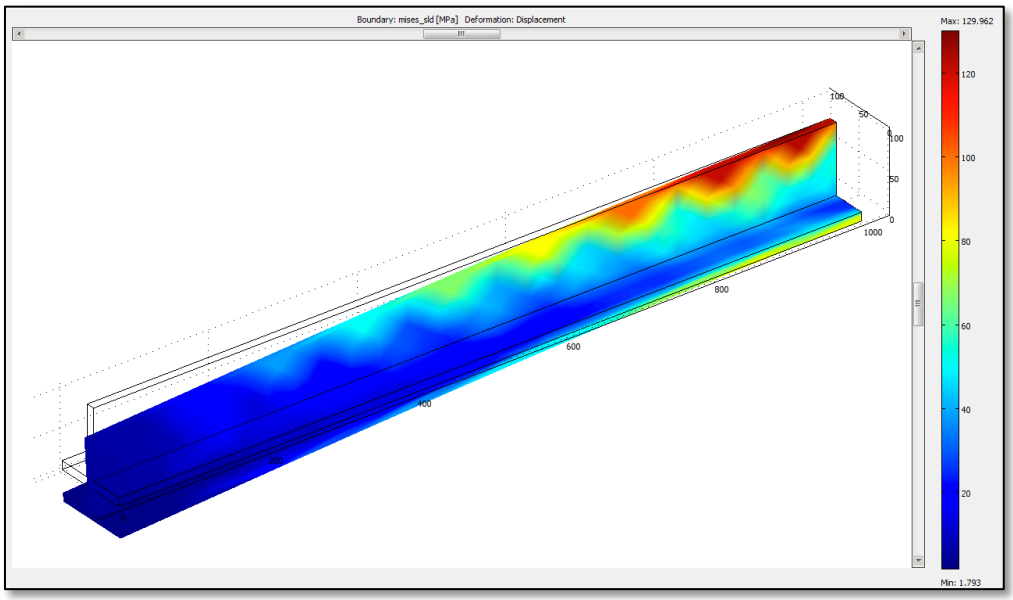


Figura 4.33: Impostazione della corretta espressione per le superfici

4.5 Salvataggio del file e apertura in Matlab

Le fasi di salvataggio e successiva apertura del file in Matlab sono identiche a quelle già esposte per il modello 2D, nei paragrafi 3.3 e 3.4.

Dopo aver salvato il file, sarà quindi possibile aprirlo direttamente da Matlab, ottenendo lo stesso risultato di Comsol all'interno di una finestra grafica, come nell'immagine seguente. Nel prossimo capitolo si approfondirà l'analisi degli script generati da Comsol e saranno spiegate nel dettaglio le tecniche seguite per velocizzarli ulteriormente.

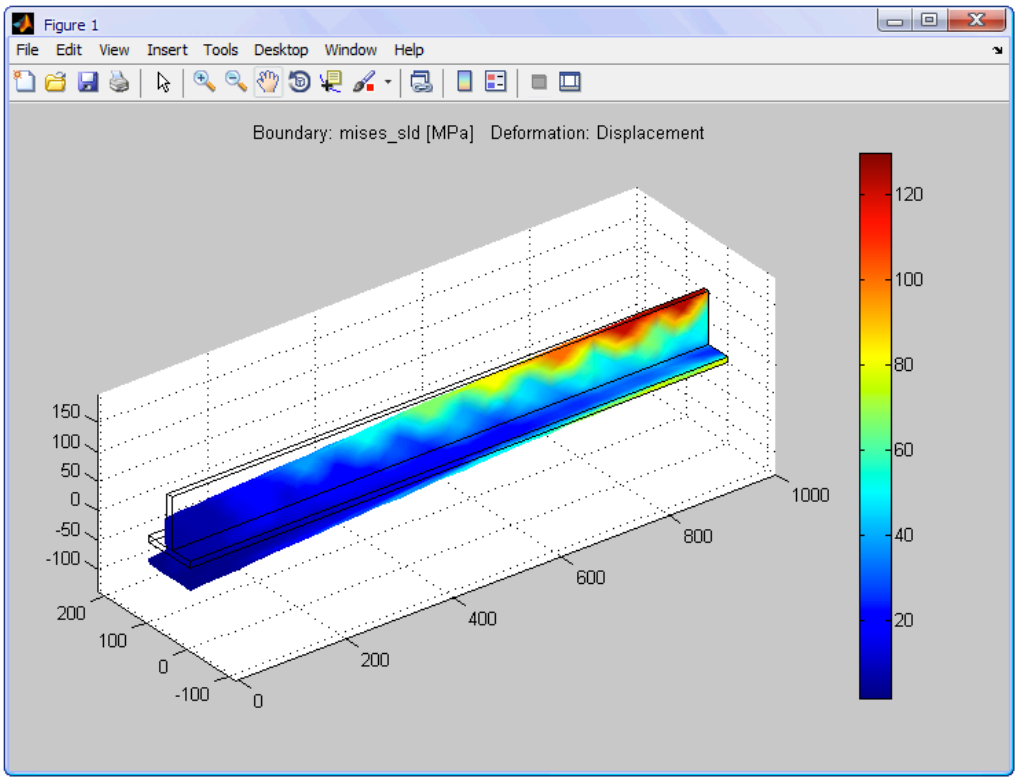


Figura 4.34: Apertura del file .m della trave in Matlab

Capitolo 5: Editing e ottimizzazione

In questo capitolo si spiegherà nel dettaglio quali sono le modifiche indispensabili da apportare ai file in formato .m generati con Comsol. Questa fase è assolutamente necessaria allo scopo di riordinare il contenuto di tali file, ma allo stesso tempo è fondamentale per capire come viene generata la geometria della struttura e come viene risolta. Questa fase di apprendimento sarà essenziale per i passi successivi, nei quali si cominceranno a considerare i singoli modelli come facenti parti di un'unica applicazione.

5.1 Interpretazione dei file .m

Nel paragrafo 3.4 è stata descritta la procedura per aprire il file "barretta2D.m" generato in precedenza con Comsol. Se si osserva il contenuto del file, dopo le prime righe d'intestazione, si giunge alla dichiarazione delle costanti. Ciò è essenziale: si può facilmente intuire come semplicemente agendo su questo parametro si possa completamente variare la geometria del modello finale.

```
% Constants
fem.const = {'L','20'};
```

Le costanti sono poi immediatamente utilizzate per la costruzione della geometria:

```
% Geometry
g1=square2('L','base','corner','pos',{'0','0'},'rot','0','const',fem.const);
g2=square2('L','base','corner','pos',{'L','0'},'rot','0','const',fem.const);
g3=square2('L','base','corner','pos',{'2*L','0'},'rot','0','const',fem.const);
g4=square2('L','base','corner','pos',{'3*L','0'},'rot','0','const',fem.const);
g5=square2('L','base','corner','pos',{'4*L','0'},'rot','0','const',fem.const);
g6=square2('L','base','corner','pos',{'5*L','0'},'rot','0','const',fem.const);
g7=square2('L','base','corner','pos',{'6*L','0'},'rot','0','const',fem.const);
g8=square2('L','base','corner','pos',{'7*L','0'},'rot','0','const',fem.const);
g9=square2('L','base','corner','pos',{'8*L','0'},'rot','0','const',fem.const);
g10=square2('L','base','corner','pos',{'9*L','0'},'rot','0','const',fem.const);
```

A questo punto è opportuno svolgere un'analisi approfondita dei diversi parametri, poiché sono importanti. Si consideri ad esempio la dichiarazione della variabile "g1". S'intuisce che debba trattarsi di un quadrato bidimensionale, come indica anche la dicitura "square2". Per ogni entità è poi possibile, tra parentesi tonde, modificare diversi parametri:

['L', 'base', 'corner'] indica che il quadrato avrà lato pari alla costante L precedentemente dichiarata, e verrà costruito a partire dal suo angolo inferiore sinistro.

['pos', {'0','0'}, 'rot', '0'] indica invece la posizione da cui sarà generato il primo quadrato, in questo caso dall'origine.

Tutte le geometrie dichiarate saranno in seguito analizzate, disegnate e trasformate in struttura, come si evince dalle seguenti righe di codice:

```
% Analyzed geometry
clear s
s.objs={g1,g2,g3,g4,g5,g6,g7,g8,g9,g10};
s.name={'SQ1','SQ2','SQ3','SQ4','SQ5','SQ6','SQ7','SQ8','SQ9','SQ10'};
s.tags={'g1','g2','g3','g4','g5','g6','g7','g8','g9','g10'};

fem.draw=struct('s',s);
fem.geom=geomcsg(fem);
```

In seguito viene inizializzata la mesh per la struttura:

```
% Initialize mesh
fem.mesh=meshinit(fem, ...
                  'hauto',5, ...
                  'point',[], ...
                  'edge',[], ...
                  'subdomain',[1,2,3,4,5,6,7,8,9,10]);
```

Qui, in particolare, si nota il parametro 'hauto', cui è attribuito un intero variabile tra 1 e 9, che controlla la finezza della mesh. A "9" corrisponde una mesh molto fitta, mentre per un valore pari a "1" corrisponde la mesh più rada possibile. Il valore "5" che qui si nota, corrisponde ad una mesh intermedia. Ovviamente all'aumentare della complessità della mesh corrisponde un tempo di risoluzione maggiore.

Si passa quindi allo step chiamato "*Application mode 1*", ossia a una fase che include la definizione degli elementi della mesh, i vincoli, il materiale ed il carico. Andando in ordine, si trovano prima di tutto le impostazioni sul tipo di analisi da eseguire, in questo caso "Plane Strain" di tipo quadratico. Tali informazioni sono contenute nelle variabili con suffisso "appl.", come si nota dalle righe di codice seguenti:

```
% Application mode 1
clear appl
appl.mode.class = 'FlPlaneStrain';
appl.gporder = 4;
appl.cporder = 2;
appl.assignsuffix = '_pn';
```

Segue poi la dichiarazione dei vincoli e dei carichi. Poiché sono tutte entità applicate su punti della struttura, sono designate col suffisso "pnt.". Il vincolo per lo spostamento verticale è espresso dalla variabile "pnt.Hy", quello orizzontale da "pnt.Hx" e il carico da "pnt.Fy". A ogni nodo viene inoltre attribuito un indice: in questo caso "1" per i nodi privi sia di carichi sia di vincoli, "2" per i nodi vincolati solo in direzione y, "3" per i nodi vincolati sia in direzione x sia y, "4" per i nodi sottoposti a carico verticale.

Ciò è interamente immagazzinato nella variabile "pnt.ind":

```
clear pnt
pnt.Hy = {0,1,1,0};
pnt.Hx = {0,0,1,0};
pnt.Fy = {0,0,0,-500};
pnt.ind = [2,1,1,1,1,1,1,1,1,1,1,4,1,1,1,1,1,1,1,1,3,1];
appl.pnt = pnt;
```

L'algoritmo procede poi con la definizione delle caratteristiche del materiale (in questo caso erano stati impostati i parametri tipici dell'acciaio) e dello spessore della struttura. Tali caratteristiche sono indicate dal suffisso "equ." E dopo essere state dichiarate, sono attribuite all'intera struttura attraverso la variabile "equ.ind", un vettore unitario di lunghezza pari al numero di entità geometriche da cui la struttura è costituita (in questo caso dieci quadrati):

```
clear equ
equ.E = 210000;
equ.thickness = 20;
equ.rho = '7870 [Kg/m^3]';
equ.ind = [1,1,1,1,1,1,1,1,1,1];
appl.equ = equ;
```

Seguono poi impostazioni che riguardano il sistema di misura adottato e sulle eventuali descrizioni delle costanti utilizzate che non si riportano per brevità, data la loro utilità relativa. E' sempre presente anche una sezione chiamata "ODE Settings", che però non sarà mai sfruttata in questo lavoro, poiché comprende impostazioni destinate a eventuali analisi tempo-variabili, che richiedono l'utilizzo di equazioni differenziali. Nella parte conclusiva dello script la struttura viene infine risolta con i seguenti comandi:

```
% Multiphysics
fem=multiphysics(fem);

% Extend mesh
fem.xmesh=meshextend(fem);

% Solve problem
fem.sol=femstatic(fem, ...
'solcomp',{'v','u'}, ...
'outcomp',{'v','u'});

% Save current fem structure for restart purposes
fem0=fem;
```

L'ultima fase è costituita dal plottaggio della struttura all'interno di una finestra grafica di Matlab. Ciò viene fatto attraverso la funzione "postplot" che, come si vedrà in seguito, riassume tutte le modifiche che vengono fatte all'interno del menu "Postprocessing" di Comsol. A ogni modifica effettuata corrisponde un nuovo postplot all'interno dello script, in quanto, come già precisato all'inizio del capitolo 3, tutte le modifiche vengono archiviate all'interno dei file .m generati dal software.

```
% Plot solution
postplot(fem, ...
         'tridata',{'mises_pn','cont','internal','unit','MPa'}, ...
         'trimap','jet(1024)', ...
         'deformsub',{'u','v'}, ...
         'title','Surface: von Mises stress [MPa]   Deformation:
Displacement', ...
         'axis',[-10,210,-89.84412470023966,109.84412470023996]);
```

5.2 Ottimizzazione del file .m

Terminata una prima "lettura" del file .m, è ora possibile cominciare la fase di editing, che consiste sostanzialmente nel "pulire" il file da tutto ciò che non è essenziale, lasciando solo le parti descritte nel paragrafo precedente. Infatti, poiché il file .m contiene una registrazione completa di tutte le modifiche che vengono apportate al modello di Comsol mano a mano che questo viene realizzato (errori compresi), questo conterrà una grande quantità di codice "inutile". La maggior parte di tale codice riguarda di solito i postplot: ogni volta che si accede al Postprocessing in Comsol allo scopo di attivare o disattivare alcune impostazioni di visualizzazione, nel file .m viene automaticamente salvato un nuovo plot della figura.

E' ovvio come una tale ridondanza sia del tutto inutile e sconveniente in termini di tempo di risoluzione, poiché prima di visualizzare il risultato finale verranno plottati e immediatamente sovrascritti anche tutti i plot precedenti. Se si salva il file .m dopo aver effettuato tutte le modifiche desiderate all'interno di Comsol, e solo quando si è certi che non se ne effettueranno altre, si potrà ritenere valida solo l'ultima porzione di codice, ed in particolare solo l'ultimo dei postplot presenti nel file. La stessa ridondanza di codice si trova quando si cambia un vincolo o il carico: in questo caso è ripetuta l'intera "*Application mode 1*". Infine, Comsol salva una variabile con un nome temporaneo, per resettarlo subito dopo (ad esempio: fem=xfem; xfem=fem;).

Ricapitolando, è quindi necessario ripulire il file da:

- Impostazioni per le "ODE"
- Passaggi di variabili non necessari
- "Postplot" ripetuti
- "Application mode 1" ripetute

Un altro "inconveniente" è la visualizzazione di barre di caricamento durante la risoluzione della struttura in Matlab. Tale problema può essere tuttavia facilmente risolto, semplicemente aggiungendo la dicitura "report, off" alle funzioni meshinit e femstatic:

```
fem.mesh=meshinit(fem, ...
                  'hauto',5, ...
                  'point',[], ...
                  'edge',[], ...
                  'subdomain',[1,2,3,4,5,6,7,8,9,10], 'report', 'off');
```

Avviando il file editato, è possibile notare già a occhio un notevole incremento di velocità nella visualizzazione del risultato finale. E' possibile "testare" il file modificato avviandolo attraverso il "*Profiler*" di Matlab:

Desktop ➔ Profiler

Se all'interno del capo contrassegnato da "*Run this code*" si scrive "barretta" e si preme invio, verrà avviato il file appena editato e mostrata la seguente schermata (E' sempre bene ripetere più di una volta questo processo per ciascun file, in modo da avere un risultato attendibile):

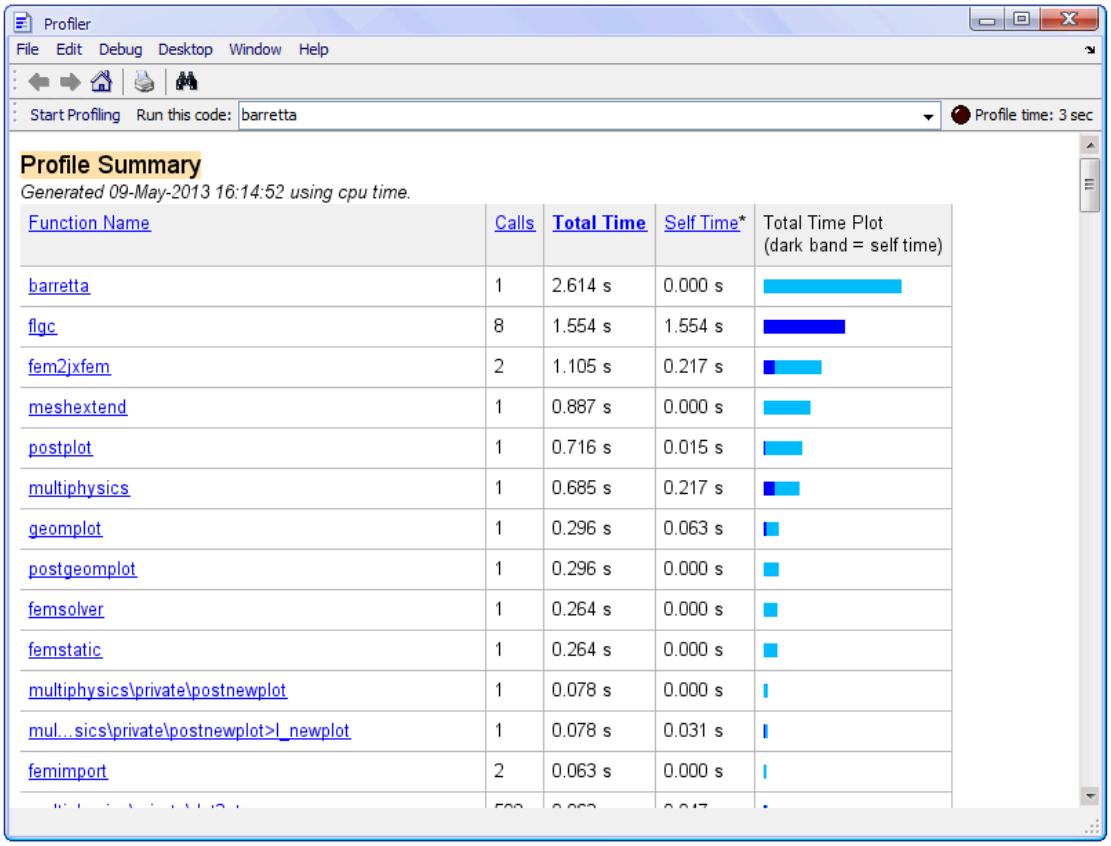


Figura 5.1: Profiler per il file "barretta.m" ripulito del codice superfluo

Osservando l'istogramma dei tempi in Figura 4.1 si nota innanzitutto che per la risoluzione del problema agli elementi finiti occorrono 2,614 secondi (il risultato ovviamente varia al variare della potenza del calcolatore che si sta utilizzando, in questo caso Windows XP virtualizzato su un iMac, soluzione adottata al momento del test ma tutt'altro che ideale, equipaggiato con un processore Intel Core 2 Duo da 3,06 GHz e 4 GB di RAM dedicati). Ciò che si nota è anche che grandi perdite di tempo si hanno nella dichiarazione della mesh, con i comandi "fem2jxfem" e "multiphysics", che fanno perdere circa quattro decimi di secondo complessivamente. La funzione più "pesante" tuttavia è "flgc", della quale si parlerà in seguito. Il risultato può sembrare deludente per il momento, ma in realtà si è già ottenuto un certo risparmio di tempo rispetto al file originale e non editato (chiamato "barretta_original.m"), di cui si riporta l'analisi dei tempi nella Figura 4.2.

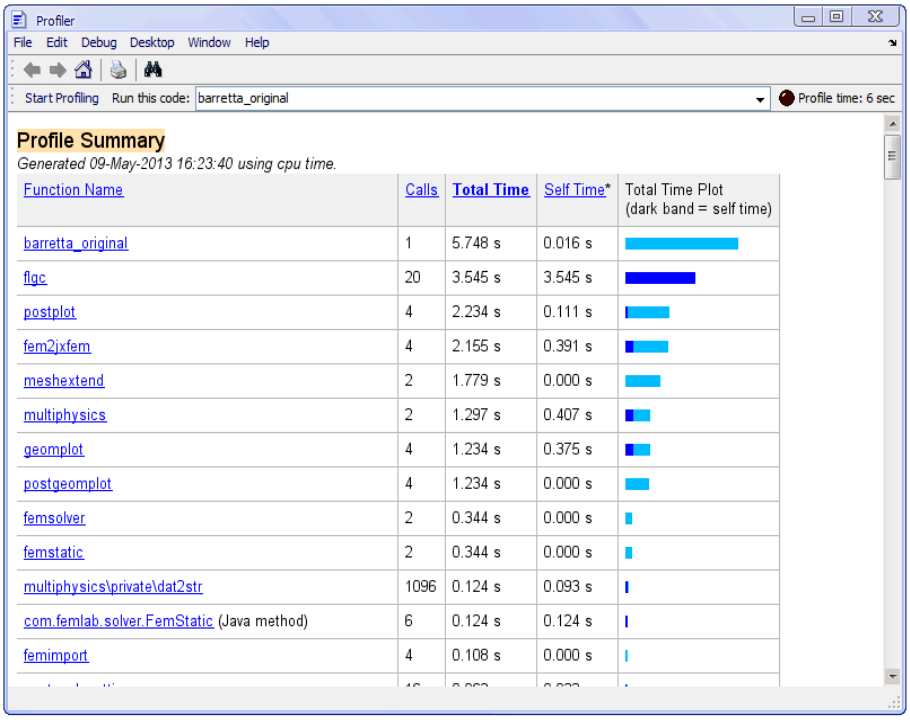


Figura 5.2: Profiler per il file "barretta_original.m"

Come si nota, il file originale (che è stato chiamato "barretta_original.m"), risulta di 3,134 secondi più lento rispetto alla nuova versione. Eliminando dal codice tutto ciò che non era strettamente necessario, si è già ottenuto un guadagno di tempo superiore al 52%. Da un'analisi più approfondita delle voci che compaiono nel "*Profiler*" per il file non modificato, è ancora più evidente la perdita di tempo generata dalla chiamata, per 20 volte, della funzione "flgc". Cliccando sul nome di questa funzione dal "*Profiler*" stesso, si aprirà la seguente finestra:

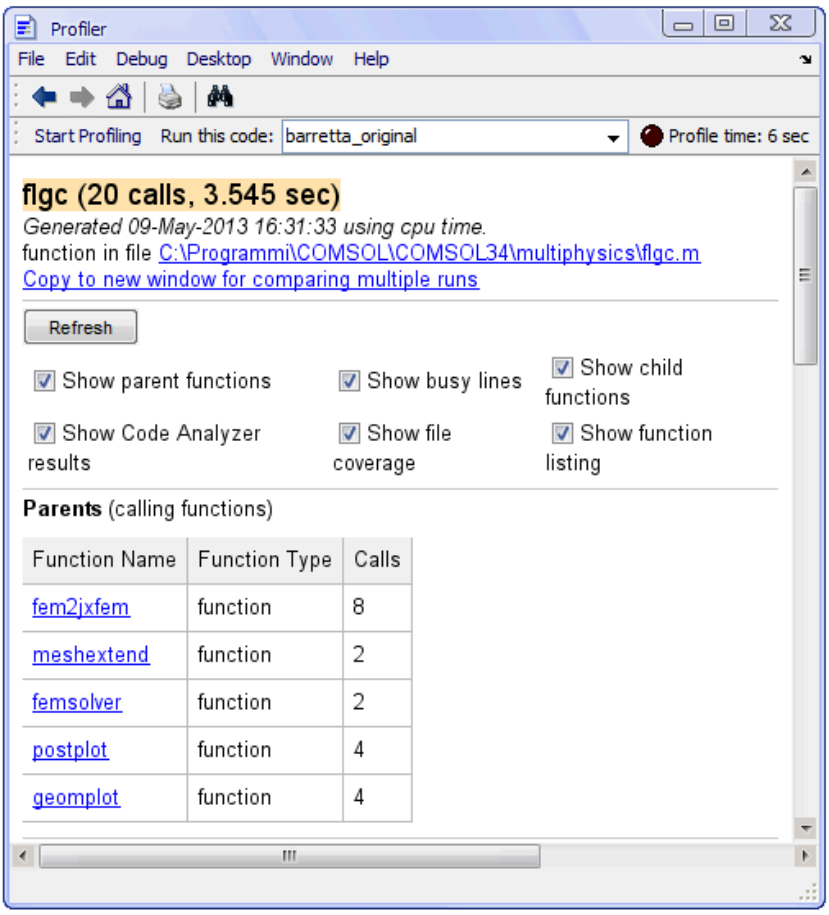


Figura 5.3: Dettaglio della funzione "flgc.m"

Seguendo il percorso indicato in figura 4.3, è possibile andare a recuperare tale file nella cartella:

C:\Programmi\COMSOL\COMSOL34\multiphysics

Aprendo il file se ne può quindi studiare il codice:

```
function flgc(forcegc)
%FLGC Calls the Java garbage collector
%   FLGC calls the Java garbage collector to try free unused memory

if ~isscript || (nargin > 0 && forcegc)
  java.lang.System.gc;
end
```

Dalla descrizione legge "FLGC Calls the Java garbage collector". Il file dunque è un "ripulitore di memoria", che provvede a cancellare dalla memoria del sistema, a intervalli regolari, tutte le funzioni o gli oggetti non utilizzati. In questo caso è inutile, quindi si può modificare, lasciandolo nella forma di funzione vuota:

```
function flgc(forcegc)
%FLGC Calls the Java garbage collector
%   FLGC calls the Java garbage collector to try free unused memory
```

Se ora si prova a riavviare il Profiler per il file "barretta.m" si ottiene il seguente risultato:

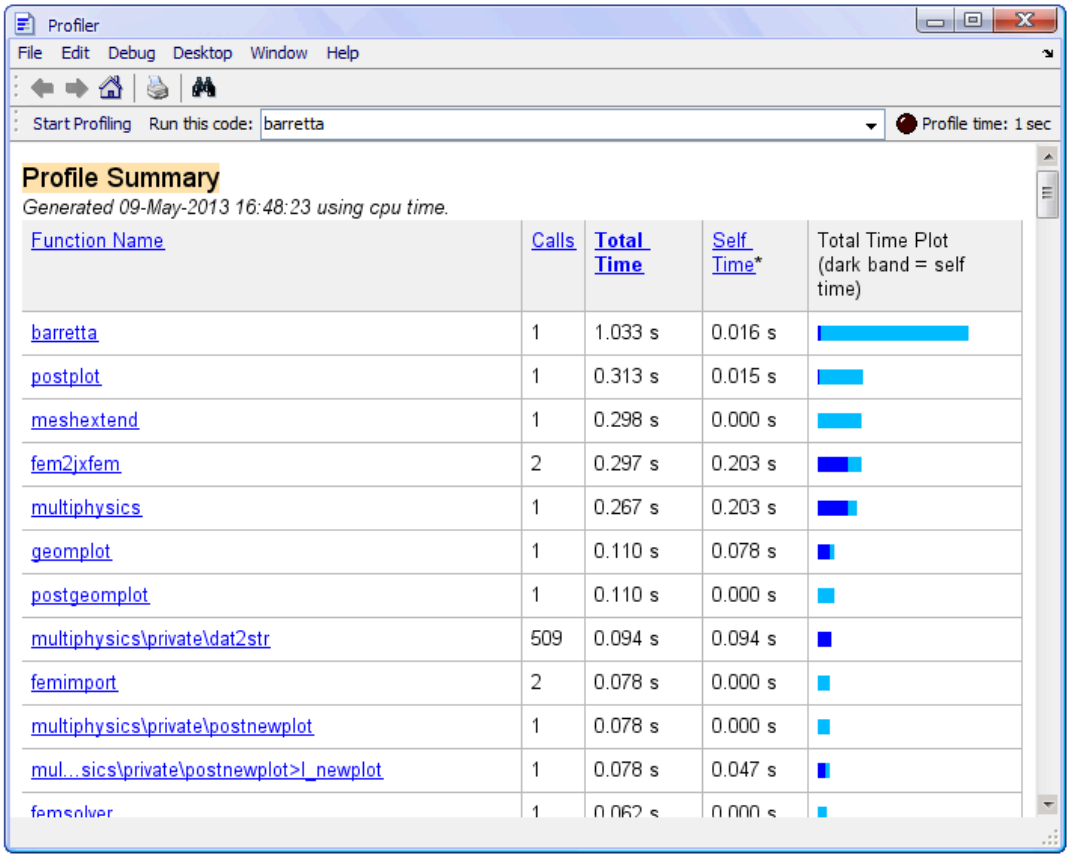


Figura 5.4: Profiler dopo l'editing della funzione flgc.m

Si nota che dall'istogramma è scomparsa la funzione flgc, e si ottiene un tempo totale di risoluzione di 1,033 secondi, che rispetto al file "barretta_original.m" costituisce un guadagno di tempo superiore all'82%.

Tuttavia, poiché ogni "postplot" costituirà un singolo "fotogramma" della simulazione *real-time*, sarà necessario ridurre ancora di molto il tempo di risoluzione, spingendosi auspicabilmente ben sotto al secondo, in modo da avere una simulazione quanto più fluida possibile, senza eccessivi e fastidiosi "scatti". Una possibile soluzione può derivare ancora una volta dall'analisi del file "barretta.m". Come si è già visto nel paragrafo 4.1, in esso esistono diverse "fasi": le prime in particolare riguardano la dichiarazione di variabili legate alla geometria, alla loro interpretazione come struttura solida, alla applicazione di mesh, vincoli e carichi. Poiché lo scopo finale è quello di applicare un carico o una deformazione variabili e di vederne l'immediato risultato in tempo reale, se ne deduce che sarebbe vantaggioso poter dichiarare a priori geometria, schema di vincolo, carico e mesh, e solo in un secondo momento operare una modifica del carico stesso. In questo modo l'onere computazionale sarebbe suddiviso in due parti: una parte d'inizializzazione, relativamente lenta, e una parte di modifica delle condizioni al contorno e di visualizzazione del risultato, auspicabilmente così veloce da essere in sostanza *real-time*.

Si parte quindi nuovamente dal file "barretta.m", se ne salva il contenuto fino alla funzione "fem.sol=femstatic" in un file separato, chiamato "barretta_init.m", poichè in sostanza contiene tutta la parte di inizializzazione. E' ora necessario apportare alcune modifiche al suo interno: innanzi tutto il file .m deve passare alla forma di "funzione", poiché dovrà essere in grado di scambiare alcuni dati con un secondo file, quello che si occuperà delle modifiche al contorno.

Inoltre è necessario dichiarare delle variabili globali, ossia valide per file non connessi direttamente tra loro, come si nota dall'immagine seguente:

```
function [solVertices solTriangles] = barretta_init()

global ComsolModel_outFem

% Constants
fem.const = {'L','20'};

% Geometry
g1=square2('L','base','corner','pos',{'0','0'},'rot','0','const',fem.const);
g2=square2('L','base','corner','pos',{'L','0'},'rot','0','const',fem.const);

[…]
```

La variabile globale "ComsolModel_outFem" servirà a poter accedere alla mesh creata in questo file da quello che si occuperà dell'elaborazione delle condizioni al contorno. Le variabili "solVertices" e "solTriangles" serviranno invece al trasferimento delle entità geometriche. Altre modifiche devono tuttavia essere apportate alla sezione finale del codice: qui infatti deve essere calcolato ed estrapolato il vettore contenente la soluzione di von Mises per la struttura. Per farlo si può utilizzare la funzione "posteval", che serve proprio a questo scopo:

```
% Extract results
[pd_solSigma] = posteval(fem,'mises_pn',...
    'refine',1,'cont','on','edim',2);

[solTriangles] = pd_solSigma.t;
[solVertices] = pd_solSigma.p;

ComsolModel_outFem = fem;
```

All'interno del vettore "pd_solSigma", estrapolato dalla soluzione FEM, è possibile ricavare anche le soluzioni per triangoli e vertici della struttura utilizzando i pedici ".t" e ".p" rispettivamente. Si passa quindi alla definizione del file che si occuperà della modifica delle condizioni al contorno in tempo reale, che qui è stato chiamato "barretta_update.m":

```
function barretta_update()

global ComsolModel_outFem

fem = ComsolModel_outFem;

% Plot solution
postplot(fem, ...
         'tridata',{'mises_pn','cont','internal','unit','MPa'}, ...
         'trimap','jet(1024)', ...
         'deformsub',{'u','v'}, ...
         'title','Surface: von Mises stress [MPa]   Deformation: Displacement', ...
         'axis',[-10,210,-89.84412470023966,109.84412470023996]);

end
```

Ciò che è stato fatto fino a questo momento è dunque solamente separare la parte geometrica da quella di plottaggio, allo scopo di determinare quanto tempo si guadagni con questa tecnica. Se arrivati a questo punto si scrive nella "*Command Window*" di Matlab la seguente istruzione:

[solTriangles solVertices]=barretta_init()

Verrà avviato il file di inizializzazione e la geometria sarà salvata nei vettori solTriangles e solVertices.

Fatto ciò, si può lanciare il secondo file direttamente dal Profiler di Matlab, allo scopo di saggiarne le performance. Il risultato è il seguente:

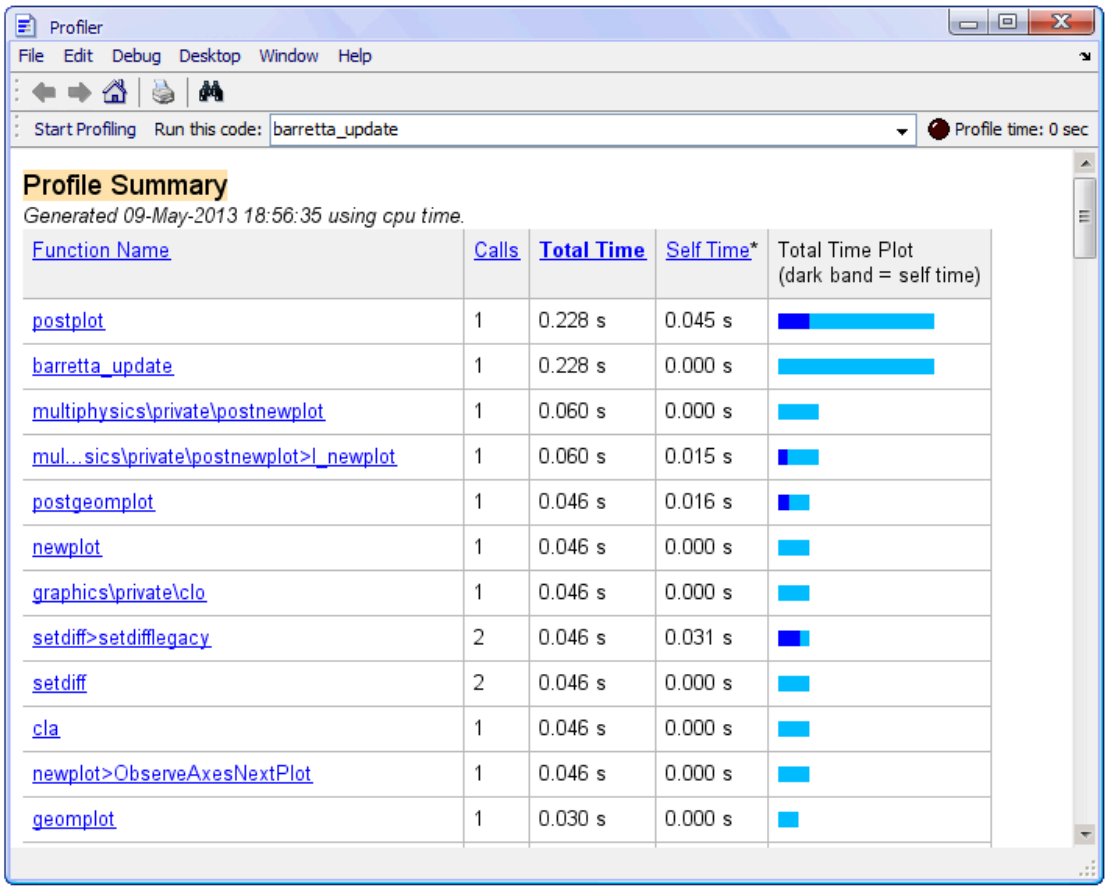


Figura 5.5: Profiler dopo la suddivisione del file originario

Il tempo totale è ulteriormente sceso a 0,228 secondi, con un guadagno rispetto al file originale salito questa volta al 96% (si ricorda che la prima simulazione col Profiler aveva restituito un risultato di 5,748 secondi).

Dimostrata la bontà del metodo, è ora necessario ultimare la modifica dei due file in modo da avviare la prima vera simulazione in tempo reale.

Si torna quindi al file "barretta_init.m" e si dichiara un'ulteriore variabile globale, chiamata "ComsolModel_SourceElements", che servirà per la modifica diretta della mesh. L'obiettivo ora è infatti quello di operare una modifica della mesh partendo dal file "barretta_update.m", dopo aver dichiarato la geometria, lanciando il file "barretta_init.m". Per farlo, è innanzitutto necessario determinare quali elementi bisognerà sostituire all'interno del vettore "fem.xmesh" per variare in tempo reale la forza applicata alla struttura. La forza che qui si applicava era di -500 N. Per poter meglio identificare questo valore, e per poterlo sostituire senza dubbi in seguito, lo può sostituire con un'espressione ben distinguibile, ad esempio "y*x":

```
% Constraints
clear pnt
pnt.Hy = {0,1,1,0};
pnt.Hx = {0,0,1,0};
pnt.Fy = {0,0,0,'y*x'};    %<-------!!!!
pnt.ind =
[2,1,1,1,1,1,1,1,1,1,1,4,1,1,1,1,1,1,1,1,3,1];
appl.pnt = pnt;
```

Ora è necessario visualizzare a schermo gli elementi della matrice "fem.xmesh". A questo scopo, al termine dello script si può aggiungere l'istruzione "get(fem.xmesh,'elements')":

```
% Extract results
[pd_solUx,pd_solUy,pd_solSigma] =
posteval(fem,'u','v','mises_ps',...
    'refine',1,'cont','on','edim',2);

[solTriangles] = pd_solSigma.t;
[solVertices] = pd_solSigma.p;

ComsolModel_outFem = fem;
xmesh_elements = get(fem.xmesh,'elements');  %<----!!!!
```

Sopprimendo solo temporaneamente la dicitura "function" al file e lanciandolo, è possibile analizzare nel dettaglio la matrice "xmesh_elements", ce è formata da una colonna di 13 righe di codice, come si nota dall'immagine seguente:

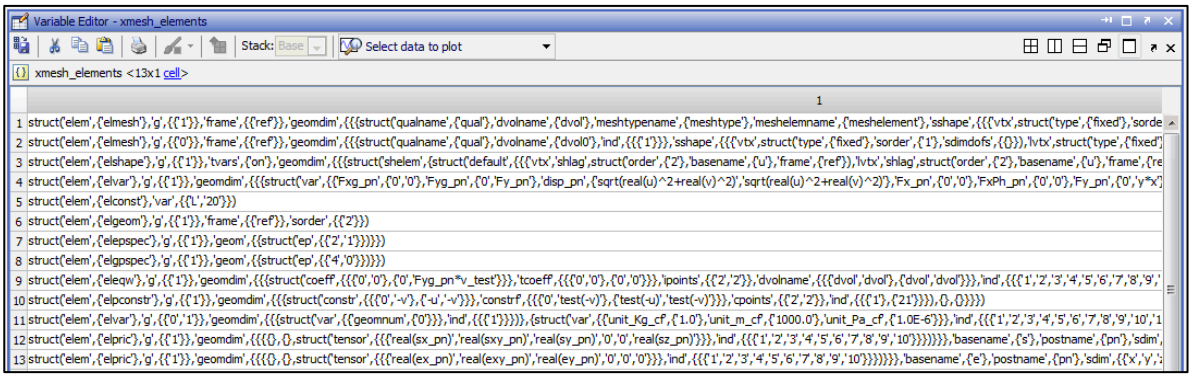


Figura 5.6: Matrice degli xmesh_elements

Copiando tale matrice in Excel e svolgendo una ricerca, si può rintracciare facilmente il fattore "y*x" imposto in precedenza, che in questo caso si trova nella riga 4. Se ne deduce che se si vuole operare una modifica diretta di tale fattore, sarà sufficiente agire con un'operazione di sostituzione solo su quella riga della matrice. Il file "barretta_init.m" può dirsi pertanto completato aggiungendo come ultima istruzione, la seguente:

```
ComsolModel_SourceElements  = xmesh_elements{4,1};
```

Che definisce la riga della matrice all'interno della quale il file "barretta_update.m" dovrà cercare il fattore 'y*x' e sostituirlo con un valore numerico, in tempo reale.

Si apre quindi il file "barretta_update.m", e si dichiara anche in questo caso la nuova variabile globale "ComsolModel_SourceElements". Infine, sono state aggiunte le seguenti righe di codice:

```
newElements =
strrep(ComsolModel_SourceElements,'y*x','500');
    fem.xmesh.replaceElems(newElements,[3]);
```

La funzione "strrep" di Matlab è in grado di cercare un determinato elemento all'interno di una matrice o di un vettore e di sostituirlo con un altro. In questo caso, come si nota, è stata impostata per cercare la stringa 'y*x' e per sostituirla con il valore 500 (Newton). Se la sostituzione di variabile avrà successo quindi, sarà visualizzata una deformata verso l'alto invece che verso il basso. Tuttavia la mera sostituzione di variabile non è purtroppo sufficiente, poiché è necessario riavviare il solutore dopo ogni cambiamento. Ciò può essere fatto attraverso il comando "MeshExtendRun" di Comsol, come si può vedere dalle seguenti righe di codice, che sono state poi incluse al file "barretta_update.m":

```
[solProp] = com.femlab.util.Prop;
solProp.initVectorInt('sorder',2);
MeshExtendRun =
com.femlab.xmesh.MeshExtendRunnable(fem.xmesh,solProp);
MeshExtendRun.smartRunner(false, 500);

fem.sol=femstatic(fem, ...
    'solcomp',{'v','u'}, ...
    'outcomp',{'v','u'},'report','off');
```

Come si nota anche l'istruzione "fem.sol=femstatic" deve essere necessariamente ripetuta. L'aspetto finale del file "barretta_update.m" è il seguente:

```
global ComsolModel_outFem
global ComsolModel_SourceElements

fem = ComsolModel_outFem;
newElements = strrep(ComsolModel_SourceElements,'y*x','500');
fem.xmesh.replaceElems(newElements,[3]);
[solProp] = com.femlab.util.Prop;
solProp.initVectorInt('sorder',2);
MeshExtendRun = com.femlab.xmesh.MeshExtendRunnable(fem.xmesh,solProp);
MeshExtendRun.smartRunner(false, 500);
fem.sol=femstatic(fem, ...
    'solcomp',{'v','u'}, ...
    'outcomp',{'v','u'},'report','off');

% Plot solution
postplot(fem, ...
         'tridata',{'mises_pn','cont','internal','unit','MPa'}, ...
         'trimap','jet(1024)', ...
         'deformsub',{'u','v'}, ...
         'title','Surface: von Mises stress [MPa]   Deformation:
      Displacement', ...
         'axis',[-10,210,-89.84412470023966,109.84412470023996]);
```

Come fatto in precedenza, si lancia il file d'inizializzazione dalla "*Command Window*" utilizzando il comando:

[solTriangles solVertices]=barretta_init()

Successivamente si può avviare, direttamente dal Profiler, il file di update. Il risultato è il seguente:

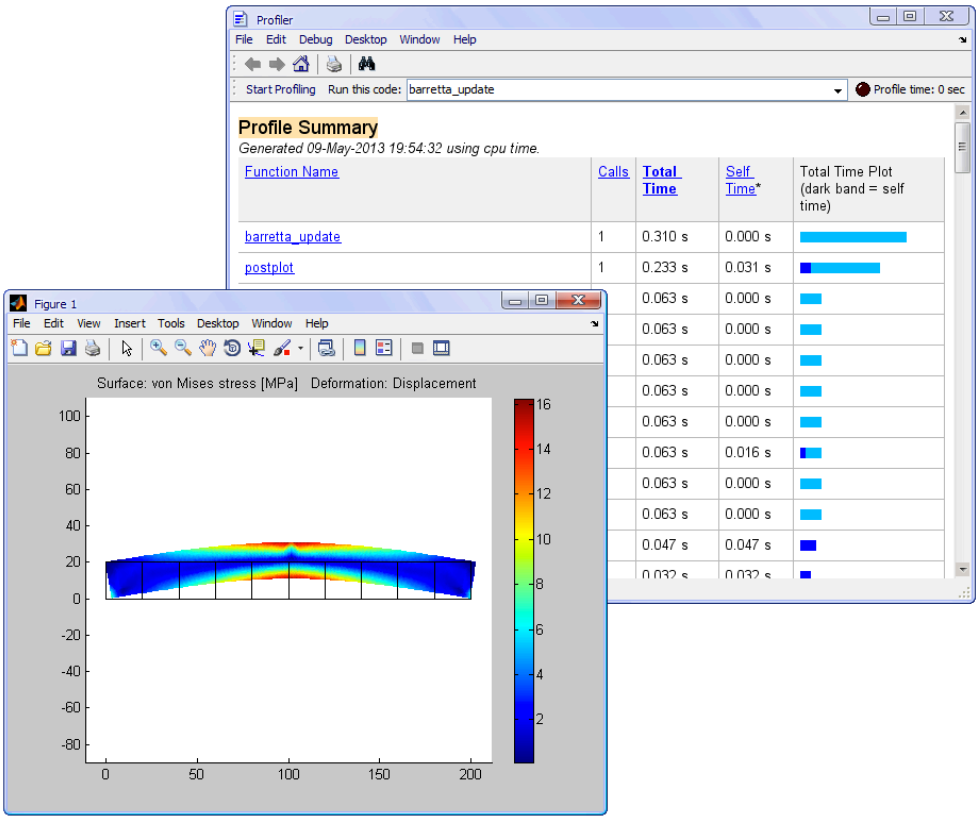


Figura 5.7: Risultato finale

Come ci si aspettava, ora la deformata è verso l'alto, chiaro segno che la sostituzione di variabile ha avuto pieno successo. Si nota tuttavia un leggero peggioramento nel tempo di risoluzione, che è passato a 0,310 secondi, con un aggravio di circa 0.082 secondi rispetto al test precedente.

Tuttavia osservando l'istogramma del "*Profiler*" si notano ancora diversi tempi recuperabili, in particolare quelli dati dall'introduzione di una nuova finestra grafica.

5.3 Ottimizzazione della visualizzazione

Nel paragrafo precedente è stato descritto il processo di ottimizzazione dei file, che ha notevolmente ridotto il tempo totale di risoluzione. Nell'immagine riassuntiva seguente si possono notare i progressi fatti passo-passo:

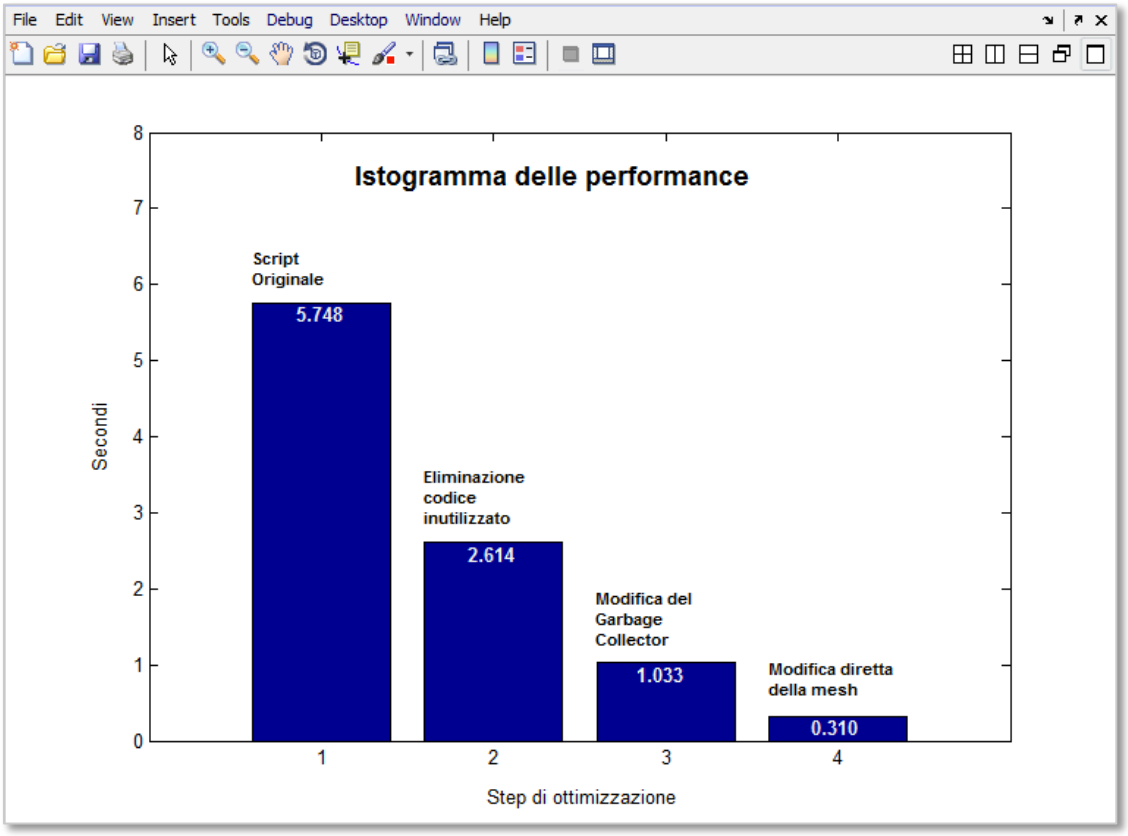


Figura 5.8: Istogramma riassuntivo del progresso delle performance, ottenuto con Matlab

Si tratta obbiettivamente di un buon progresso, tuttavia i circa tre decimi di secondo necessari dell'ultima ottimizzazione non sono ancora del tutto sufficienti a garantire una simulazione *real-time*, che sarebbe visualizzata a soli 3 fps.

Come accennato, ancora molto tempo viene perso per la visualizzazione del risultato. In particolare, Matlab perde circa 0,2 secondi per aprire una nuova finestra grafica e per impostarne correttamente dimensioni e assi. Partendo da queste considerazioni si può dunque cercare di impedire al software di agire autonomamente, dandogli in input la maggior parte degli elementi necessari per la visualizzazione del risultato.

E' stato dunque creato un nuovo script di Matlab, chiamato "play.m" (poiché questo sarà il file di avvio di una vera e propria applicazione). Esso comincia con poche righe di codice in grado di pulire la "Command Window", di cancellare eventuali variabili dichiarate in precedenza e di chiudere tutte le figure eventualmente aperte. Queste operazioni sono rispettivamente svolte attraverso i comandi "clc", "clear all" e "close all". Inoltre, poiché si ricorrerà all'utilizzo di più file concatenati, è sempre bene accertarsi che questi facciano tutti parte della directory di lavoro di Matlab. Questa operazione può essere automatizzata attraverso il comando "addpath(genpath(pwd))", che aggiunge automaticamente tutti i file contenuti in cartelle ed eventuali sottocartelle della directory selezionata all'area di lavoro.

Infine, per visualizzare il risultato a tutto schermo, è possibile acquisire le dimensioni dello schermo del computer attraverso il comando "screen_size"

```
%PLAY FROM THIS FILE

clc
clear all
close all

addpath(genpath(pwd));   %add folder and subfolders in the path

screen_size = get(0, 'ScreenSize');                          %get screen size
f1=figure(1);                                                %new figure
set(f1, 'Position', [0 0 screen_size(3) screen_size(4)]);    %fullscreen

[solTriangles solVertices]=barretta_init();
barretta_update;
```

Come si nota dallo script, dopo aver acquisito le dimensioni dello schermo, queste sono utilizzate per generare una finestra grafica con le stesse dimensioni. Le ultime due istruzioni inserite nel file riguardano invece il semplice avvio in sequenza del file di inizializzazione "barretta_init.m" e di quello di aggiornamento della mesh "barretta_update.m".

Arrivati a questo punto si può far partire l'applicazione premendo il tasto di avvio direttamente dal file "play.m", e questo è il risultato a tutto schermo ottenuto:

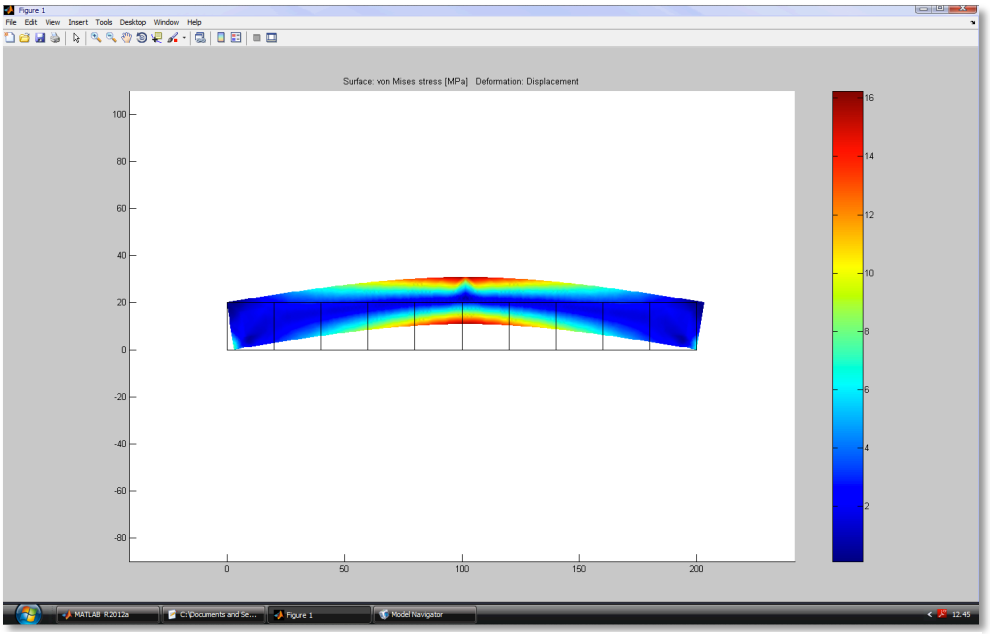


Figura 5.9: Visualizzazione a tutto schermo

Per quanto riguarda la parte grafica, il risultato è quello sperato, mentre per quanto riguarda le performance è difficile dare a questo punto una stima corretta. Sia utilizzando il "*Profiler*", sia contatori quali il "tic, toc" di Matlab, si ottengono risultati contraddittori, con una stima del tempo di risoluzione di oltre 0,6 secondi. Questo risultato è del tutto inattendibile poiché rispetto alla simulazione svolta al paragrafo precedente non sono stati aggiunti elementi in grado di produrre un rallentamento di tale portata. L'aumento del tempo di risoluzione è probabilmente dovuto al fatto di aver concatenato più file all'interno del file "play.m", il che tende a "confondere" il "*Profiler*" nel calcolo del tempo di risoluzione della struttura. Per riuscire a dare una stima della reale velocità dello script, si dovrà quindi attendere la fase di simulazione *real-time* vera e propria, della quale ci si occuperà approfonditamente nel prossimo capitolo.

Capitolo 6: Simulazione *Real-Time*

In questo capitolo ci si occuperà della ricerca di un metodo che consenta di eseguire una simulazione in tempo reale nel modo più efficace e semplice possibile. Dopo una fase iniziale di comprensione e ottimizzazione degli script generati con Comsol, lenta ma necessaria, è ora indispensabile dimostrare che l'aggiornamento della mesh può essere fatto in tempi tali da garantire una visualizzazione sufficientemente fluida.

Inizialmente, poiché questo era l'obiettivo primario della tesi, si era pensato di concentrarsi sull'implementazione esclusiva del sistema *haptic* nell'applicazione per gestire la deformazione dei modelli. In seguito tuttavia, si è deciso di proseguire con uno sviluppo parallelo che consentisse agli utenti di interagire in tempo reale con le strutture anche in mancanza di un dispositivo *haptic* (come la maggior parte delle volte accade), usando un mouse o di un trackpad. Ciò aggiunge un'intrinseca potenzialità al metodo: pochi laboratori e aule didattiche hanno, infatti, la fortuna di possedere un sistema *haptic*, ma tutti possiedono o possono procurarsi in modo semplice e relativamente economico un computer. Ovviamente, se si ha la fortuna di poter utilizzare un hardware come il *Phantom*, l'esperienza di "contatto" con la struttura virtuale aggiunge una nuova dimensione alla simulazione, che in questo modo potrà comunque essere garantita anche in sua assenza, cosa altrimenti impossibile.

6.1 Installazione della *Haptik Library*

Poiché lo scopo è ora quello di far dialogare Matlab con il sistema *haptic*, è stata installata la *Haptik Library*, una libreria totalmente open source e liberamente scaricabile dal sito http://sirslab.dii.unisi.it/haptiklibrary/ dell'università di Siena:

Figura 6.1: Schermata di installazione della Haptik Library

Dopo aver installato la libreria nella directory di default, è necessario individuare e isolare i soli file necessari a interfacciarla con Matlab, poiché essa è pensata per funzionare anche con altri linguaggi di programmazione, quali C e Java. Andando nella directory d'installazione, è ben riconoscibile una sottocartella chiamata "Matlab". Al suo interno sono presenti diversi file, dei quali solo alcuni sono strettamente necessari. L'intero contenuto di questa cartella è stato quindi copiato all'interno della directory di lavoro di Matlab.

In questo caso particolare, come si nota dall'immagine sottostante, i file sono stati copiati all'interno della cartella contenente i file "barretta_init.m" e "barretta_update.m":

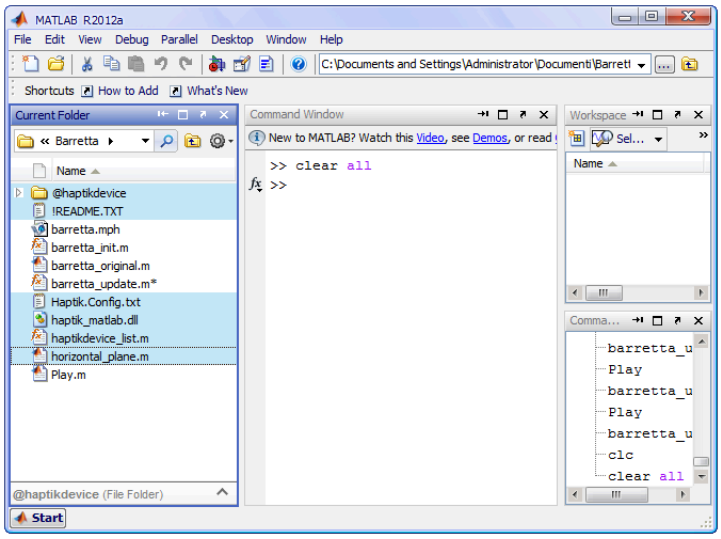


Figura 6.2: File della Haptik Library copiati all'interno del path di Matlab

Tra i vari file è presente una cartella chiamata "@haptikdevice", che contiene tutte le funzioni che garantiscono il corretto funzionamento della libreria. Queste sono anche elencate a pagina 121 del "*Developer's Manual*" della libreria:

Chapter 9

Matlab & Simulink

9.1 Matlab

Haptik can be used from matlab command line and from .m files through a mex interface (haptik_matlab.dll). A simple matlab class has been written on top of this to make easy to use.

`haptikdevice_list`
print a list of current devices
`h = haptikdevice`
open for polling the default device
`h = haptikdevice(id)`
open for polling a specific device
`h = haptikdevice(@mycallback,rate)`
open default device and use the mycallback(.m) function at the specified rate in Hz
`h = haptikdevice(id,@mycallback,rate)`
open a specific device and use the mycallback(.m) function at the specified rate in Hz
`[matrix,button] = read(h)`
get orientation from device (matrix(4,1) matrix(4,2) and matrix(4,3) contains position)
`position = read_position(h)`
get position
`button = read_button(h)`
get button status
`write(h,ff)`
send forces and torques to device. FF can be 2x3, 3x2, 1x3, 3x1, 1x6, 6x1.
`close(h)`
close the device

Figura 6.3: Elenco di tutte le funzioni della Haptik Library utilizzabili in Matlab

Come si nota, installando la Haptik Library e disponendo di un dispositivo *haptic*, l'utente potrà, ad esempio, conoscere l'esatta posizione dell'*end-effector*. Si nota anche, in cima all'elenco, la funzione "haptikdevice_list" che fornisce un elenco completo di tutti i dispositivi in grado di interagire con la libreria per mezzo di Matlab. Al primo avvio, nel caso testato in laboratorio, la libreria ha riconosciuto non solo la presenza del "Phantom Desktop" ma anche l'interfaccia "*Mouse Spectre*" creata, a quanto si legge, con uno scopo molto simile a quello esposto nel paragrafo introduttivo di questo capitolo: fornire uno strumento alternativo all'utente in mancanza di un dispositivo *haptic*. Dopo aver svolto alcuni test, si è tuttavia notato un serio conflitto tra il dispositivo *haptic* e l'interfaccia "*Spectre*", pertanto si è deciso di intervenire sul file "Haptik.Config.txt" per cercare di porvi rimedio. Come s'intuisce dall'estensione, si tratta di un file di testo "speciale", che è utilizzato per caricare le corrette librerie .dll, in relazione all'hardware *haptic* disponibile (o non disponibile) al momento. Al suo interno sono anche contenute istruzioni sulle impostazioni di eventuali *offset*, sulla massima forza restituita dal dispositivo, sulla frequenza di aggiornamento e sull'interfaccia "*Spectre*", che dovrebbe intervenire solo in assenza di un dispositivo *haptic*, ma che in questo caso sembra andare in conflitto con il *Phantom*. Pertanto dopo numerose prove si è infine deciso di eliminare tutte le voci che non avessero a che fare con il *Phantom Haptic*, compresi i richiami ai plugin per altri tipi di dispositivi *haptic*. Il file si riduce pertanto al seguente:

```
HaptikLibrary.numberOfPlugins =          5
HaptikLibrary.plugin0_0 =          Haptik.Phantom42OH.dll
HaptikLibrary.plugin0_1 =          Haptik.Phantom42.dll
HaptikLibrary.plugin0_2 =          Haptik.Phantom40.dll
HaptikLibrary.plugin0_3 =          Haptik.Phantom31.dll
HaptikLibrary.allowMessageBoxes = FALSE
HaptikLibrary.preventSplashScreen = TRUE
Phantom.LeftHanded     =          FALSE
Phantom.MaxForce       =          10.000000
Remote.Port     =          60000
Remote.Address = majorana
```

6.2 Deformazione *real time* con un dispositivo *haptic*

Dopo aver preso conoscenza con le potenzialità e i limiti della Haptik Library, si può tornare al problema principale: deformare un modello creato con Comsol in tempo reale usando il sistema *haptic*. Come si è già visto, attraverso l'utilizzo della Haptik Library si ha la possibilità di calcolare istantaneamente la posizione del *Phantom*, in coordinate cartesiane. Le coordinate, tuttavia, includono anche un campo negativo, pertanto sono state normalizzate in modo da avere come valore minimo "0". Nel file "barretta_update.m" sono inoltre presenti le righe di codice che garantiscono la sostituzione diretta della mesh, seguite dal "postplot". Questa parte di codice è stata inserita in un ciclo "while", per creare un "loop" infinito, che continua a eseguire sempre la medesima sostituzione: l'algoritmo cerca all'interno della mesh una sequenza di caratteri uguale a "y*x" e la sostituisce con "500", e il risultato visualizzato è, per il momento, sempre quello della barretta sollecitata con 500 N.

```
play=1;
while play==1
newElements = strrep(ComsolModel_SourceElements,'y*x','500');
fem.xmesh.replaceElems(newElements,[3]);

[solProp] = com.femlab.util.Prop;
solProp.initVectorInt('sorder',2);
MeshExtendRun = com.femlab.xmesh.MeshExtendRunnable(fem.xmesh,solProp);
MeshExtendRun.smartRunner(false, 500);
fem.sol=femstatic(fem, ...
    'solcomp',{'v','u'}, ...
    'outcomp',{'v','u'},'report','off');
% Plot solution
postplot(fem, ...
         'tridata',{'mises_pn','cont','internal','unit','MPa'}, ...
         'trimap','jet(1024)', ...
         'deformsub',{'u','v'}, ...
         'deformscale',15, ...
         'title','Surface: von Mises stress [MPa]   Deformation:
Displacement',...
         'axis',[-10,210,-89.84412470023966,109.84412470023996]);
drawnow
end
```

Lo scopo è fare ora in modo che, al posto di 500, l'algoritmo utilizzi un numero dipendente dalla coordinata y del sistema *haptic* per la sostituzione diretta. All'inizio del ciclo "while" si è imposto quindi:

```
s = 1;
hap_pos = read_position(h);
hap_posy =hap_pos(2)*s;
```

Dove “s” è un fattore di scala, posto per il momento pari all’unità, con il quale sarà possibile scalare opportunamente l’ordinata del sistema haptic. Al sistema *haptic* è stata invece attribuita la variabile “h”. Il comando "strrep", utilizzato per rintracciare la scrittura “y*x” all’interno della mesh, esige tuttavia in ingresso una stringa di caratteri, mentre la quantità “hap_posy” è un numero.

E’ quindi necessario eseguire una conversione di formato attraverso il seguente comando:

```
hap_y = num2str(hap_posy)
```

A questo punto l'ordina del Phantom calcolata in tempo reale è un numero e verrà sostituita in tempo reale al posto della stringa "y*x" nella mesh della struttura. La trave sarà sottoposta a una forza pari al valore dell'ordinata del Phantom, che come detto sarà eventualmente possibile scalare modificando il fattore di scala "s".

```
fem = ComsolModel_outFem;
h=haptikdevice;
s=1;

play=1;
while play==1

    hap_pos = read_position(h);
    hap_posy = hap_pos(2)*s;
    hap_y = num2str(hap_posy);
    newElements = strrep(ComsolModel_SourceElements,'y*x',hap_y);
    fem.xmesh.replaceElems(newElements,[3]);

    (…)
```

L'attribuzione della variabile "h" al dispositivo *haptic* non può essere fatta all'interno del ciclo "while", per alcune problematiche che saranno esposte nel paragrafo finale di questo capitolo. Una soluzione è quindi quella di attribuire la variabile prima di tale ciclo, attraverso il comando "h=haptikdevice". A questo punto è possibile eseguire una prova, avviando la simulazione dal file "play.m".

Il risultato è sintetizzato nell'immagine seguente:

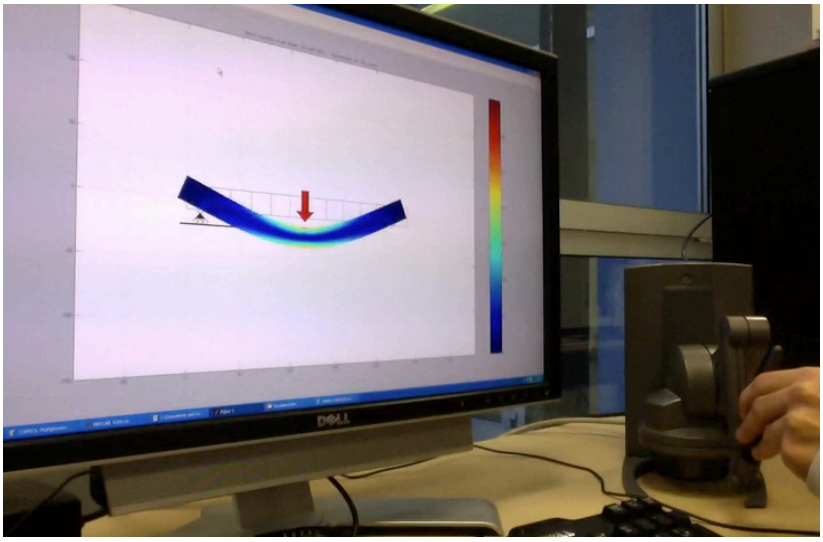


Figura 6.4: Simulazione in tempo reale con l'utilizzo del sistema haptic

Per il momento l'utente è solo in grado di utilizzare il sistema *haptic* come un cursore, e non è in grado di sentire alcuna risposta da parte della struttura. La Haptik Library consente però di gestire anche le forze, che in questo caso dovranno essere compatibili con la deformazione della struttura, in modo che chi utilizza il sistema abbia un input non solo visivo, ma anche tattile. Ciò aggiungerà una nuova dimensione sensoriale alla simulazione, e l'utente avrà l'impressione di deformare una struttura reale, "sentendola" attraverso il sistema *haptic* come se stesse toccando e deformando una barretta reale con una penna, anche se con forze scalate.

Le forze sono gestite attraverso il comando "write", come nel codice riportato di seguito, che deve essere inserito all'interno del ciclo while, poiché le forze cambiano a ogni ciclo compatibilmente con gli spostamenti del puntatore, e pertanto devono essere aggiornate a ogni ciclo per garantire un effetto realistico:

```
% Force feedback settings:
    if hap_posy>=0
      write(h,[0 -hap_posy 0]);
    elseif hap_posy<0
      write(h,[0 -hap_posy 0]);
    end
```

Dopo numerose prove atte a tarare opportunamente la quantità "hap_posy" è stato possibile apprezzare una risposta realistica. Si è avuta effettivamente l'impressione di deformare una struttura reale, sentendone la risposta elastica. Un inconveniente è dato dalla "lentezza" di risoluzione: il sistema *haptic* lavora normalmente a 1 *kHz*, e ciò significa che, per funzionare in modo fluido, richiederebbe un aggiornamento ogni millesimo di secondo. Purtroppo la risoluzione di un problema agli elementi finiti continua a essere un procedimento oneroso per un calcolatore e in questo caso, come si è visto, non si ha modo di svolgerla in meno di due decimi di secondo. Ciò comporta un certo "gap" nell'aggiornamento delle forze restituite dal sistema *haptic*, che per l'utente si traduce in una serie di lievi "scatti" durante il suo utilizzo, che però non pregiudicano il funzionamento del dispositivo e l'esperienza dell'utente. Facendo un bilancio complessivo del risultato ottenuto si ritiene che esso sia quindi positivo, con ovvi margini di miglioramento nel caso si utilizzi un calcolatore più potente. In questo caso, il calcolatore a disposizione in laboratorio non aveva caratteristiche eccellenti, e il sistema *haptic* non è stato testato su altri computer, poiché i moderni portatili e anche molti pc

fissi non possiedono più la porta seriale, necessaria al collegamento del sistema *haptic*. Una stima del miglioramento di prestazioni in relazione ad un hardware più potente è comunque stata stabilita nel prossimo paragrafo. Con l'implementazione del controllo via mouse, infatti, non è più richiesta la presenza di collegamenti "speciali" e pertanto si può testare l'algoritmo anche su computer più potenti senza problemi particolari.

6.3 Deformazione *real time* di un modello con il mouse

Arrivati a questo punto del lavoro, è stato ritenuto opportuno chiedersi quanto, dal punto di vista di realizzazione di un'applicazione con scopi didattici, un'interfaccia basata esclusivamente su un hardware costoso, delicato e difficile da reperire come un sistema *haptic* potesse essere efficace. Normalmente in classe tutti gli studenti possono avere a disposizione un computer, ma non sono presenti dispositivi *haptic*, a meno che la lezione non si svolga in un laboratorio di prototipazione virtuale, come il VPLab del Politecnico di Milano, all'interno del quale è stata svolta questa tesi. Da queste riflessioni è nata l'idea, già accennata nei paragrafi precedenti, di separare il lavoro in due rami paralleli. Il primo è dedicato esclusivamente all'esperienza con sistemi *haptic*, l'altro a un'esperienza più comune e più alla portata di tutti, poiché sfrutta un mouse o un trackpad per deformare in tempo reale la struttura. L'operazione, una volta impostati gli script necessari alla simulazione come visto al paragrafo precedente, è molto semplice, poiché consiste semplicemente nella modifica di poche righe del codice già utilizzato per gli esempi precedenti.

Il file “barretta_update.m” è stato quindi modificato come segue:

```
%... (intestazione della funzione)

%limits:
min_x=-50;
min_y=-100;
max_x=250;
max_y=100;

fem = ComsolModel_outFem;
s=60;
Fmax=20000;
Fm=num2str(Fmax);

%Extract color of SigmaMax
newElements = strrep(ComsolModel_SourceElements,'y*x', Fm);
fem.xmesh.replaceElems(newElements,[3]);

[solProp] = com.femlab.util.Prop;
solProp.initVectorInt('sorder',2);
MeshExtendRun = com.femlab.xmesh.MeshExtendRunnable(fem.xmesh,solProp);
MeshExtendRun.smartRunner(false, 500);

fem.sol=femstatic(fem, ...
    'solcomp',{'v','u'}, ...
    'outcomp',{'v','u'},'report','off');


% Extract results
[pd_solSigma] = posteval(fem,'mises_pn','refine',1,'cont','on','edim',2);
color = max(pd_solSigma.d)+10;   %<----
play=1;

 %Continua a pag. seguente...
```

```
%...Continua da pag. precedente

while play==1
    mouse_pos = get(0, 'pointerlocation');
    mouse_posy = (mouse_pos(2)-470)*s;

    if mouse_posy>=Fmax
        mouse_posy=Fmax;
    elseif mouse_posy<=-Fmax
        mouse_posy=-Fmax;
    end

    mouse_y = num2str(mouse_posy);
    newElements = strrep(ComsolModel_SourceElements,'y*x', mouse_y);
    fem.xmesh.replaceElems(newElements,[3]);

    [solProp] = com.femlab.util.Prop;
    solProp.initVectorInt('sorder',2);
    MeshExtendRun = com.femlab.xmesh.MeshExtendRunnable(fem.xmesh,solProp);
    MeshExtendRun.smartRunner(false, 500);

    fem.sol=femstatic(fem, ...
        'solcomp',{'v','u'}, ...
        'outcomp',{'v','u'},'report','off');

    postplot(fem, ...
        'tridata',{'mises_pn','cont','internal','unit','MPa'}, ...
        'tridlim',[0 color], ...
        'trimap','jet(1024)', ...
        'deformsub',{'u','v'}, ...
        'deformscale',15, ...
        'axis',[min_x max_x min_y max_y]);

    grid on
    axis on

    drawnow
end

end
```

In questo script, come si può notare, sono presenti diverse modifiche. Partendo dall'inizio, si nota l'impostazione dei limiti del grafico, che possono quindi essere cambiati secondo le esigenze, e che devono essere richiamati nel "postplot" alla voce "axis". Prima del ciclo while è stato inserito uno script che lancia, senza visualizzarla, una prima simulazione che impone alla struttura il valore massimo che si vuole impostare per la forza "Fmax", in questo pari a 20000 N. In questo modo il software calcola automaticamente la soluzione, e da questa viene estrapolato lo sforzo massimo generato nella struttura attraverso il successivo comando "posteval", allo scopo di utilizzarlo nel "postplot" con il comando "tridlim", che definisce l'ampiezza della scala di colori del grafico. Nel ciclo while si può notare che per sfruttare le coordinate del mouse anziché quelle del sistema *haptic* è stato usato il comando "**get(0, 'pointerlocation')**", che in modo analogo al comando "read_position(h)" visto per la parte *haptic*, fornisce la posizione di ascissa ed ordinata del mouse ad ogni ciclo. Procedendo nella lettura del file si possono notare le seguenti stringhe:

```
if mouse_posy>=Fmax
        mouse_posy=Fmax;
    elseif mouse_posy<=-Fmax
        mouse_posy=-Fmax;
end
```

Queste istruzioni servono a limitare il campo di movimento del mouse: ogni volta che viene superato il limite, pari a Fmax o a -Fmax, la forza applicata sulla struttura è pari al massimo a quella del limite impostato.

Le righe di codice successive, fino al postplot, sono invariate rispetto a quelle viste al paragrafo precedente a parte l'introduzione del calcolo, ciclo per ciclo, dello sforzo massimo di von Mises generato nella struttura che è salvato nella variabile

"SolSigmaMax". Segue il postplot, nel quale sono presenti i nuovi campi "tridlim" e "axis", dei quali si è già parlato in precedenza, e il campo "deformscale", che gestisce l'amplificazione della deformata, in questo caso posta pari a 15 poiché una forza di 20000 N, pari a circa 2 tonnellate, produrrebbe una deformazione reale (che si può ottenere con un "deformscale" pari a 1) di soli 1,12 mm, non apprezzabile nel grafico. L'ultima novità presente è ancora nel postplot, alla voce "title", con la quale si riesce a inserire una scritta sopra il grafico che riporta istantaneamente il valore dello sforzo massimo e la forza applicata, come nell'immagine seguente:

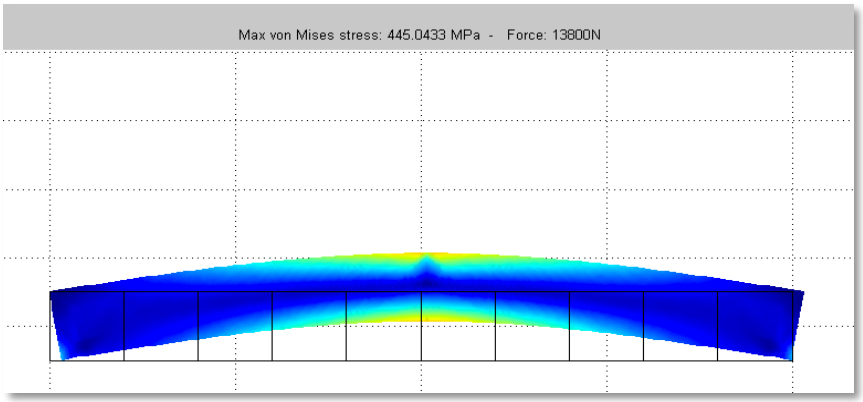


Figura 6.5: Visualizzazione in tempo reale di sforzi e forza applicata

6.4 Performance

Come accennato in precedenza, nel test effettuato in laboratorio con il sistema *haptic* "*Phantom*" si era vincolati ad utilizzare un computer non particolarmente performante (Intel Xeon, 3.25 GHz, 3,25 Gb RAM), ma dotato dell'indispensabile porta seriale. Tuttavia, avendo creato degli script in grado di funzionare anche solo con l'utilizzo di un mouse o di un trackpad, si può valutare l'incremento di performance che si può raggiungere con calcolatori più potenti. Per farlo, è sufficiente utilizzare i comandi

"tic…toc" di Matlab, posti rispettivamente all'inizio e alla fine del ciclo while del file "barretta_update.m".

In questo modo viene avviato un contatore in corrispondenza del comando "tic" che viene poi arrestato in corrispondenza del "toc", fornendo un'idea piuttosto precisa del tempo impiegato per generare un singolo "frame":

```
play=1;
while play==1
    tic
    mouse_pos = get(0, 'pointerlocation');
    mouse_posy = (mouse_pos(2)-470)*s;
.
.
.
drawnow
    toc
end
```

Lanciando nuovamente la simulazione, e interrompendola dopo qualche secondo, se si guarda la "Command Window" di Matlab si avrà:

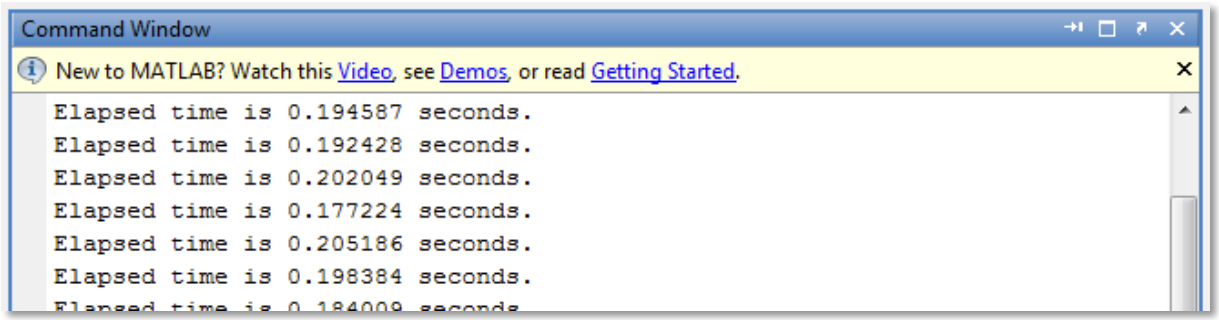


Figura 6.6: Tempo di ciclo con un processore Intel i7 e Windows 7 virtualizzato

Come si nota il tempo impiegato per la risoluzione della struttura e la visualizzazione varia da 0,18 a 0,20 secondi, garantendo una media intorno ai 5/6 fotogrammi al secondo.

Non si tratta ovviamente di un risultato all'altezza delle migliori applicazioni *real-time*, ma considerando la semplicità degli strumenti utilizzati, si ritiene che si tratti comunque di un risultato accettabile in termini di usabilità. Il risultato riportato in Figura 6.5 si riferisce a una prova effettuata con un MacBook Pro, utilizzando Windows 7 virtualizzato, con 4 Gb di memoria RAM dedicata. Ovviamente, anche in termini di utilizzo di CPU e GPU non si tratta di una scelta ottimale, poiché un personal computer di pari caratteristiche sfrutterebbe molto meglio le risorse a disposizione rispetto a un sistema operativo virtualizzato. Oltre a questa, sono state eseguite altre tre prove: una su un MacBook Pro di ultima generazione con 16 Gb di RAM, una su un iMac, con caratteristiche inferiori e con Windows XP virtualizzato, e l'ultima sul computer disponibile in laboratorio, con caratteristiche ancora inferiori, utilizzando direttamente il sistema *haptic*. Nella tabella sottostante sono riassunti i dati delle prove svolte:

Sistema Operativo	**Cpu**	**RAM**	**Tempo medio di soluzione**	**fps (medi)**
Windows XP SP3	2,66 GHz Intel Xeon	3,25 Gb	0,259 s	3,851
Windows XP (virtualizzato)	3,06 GHz Intel Core2Duo	2 Gb dedicati su 6 (1067 MHz DDR3)	0,240 s	4,155
Windows 7 (virtualizzato)	2,7 GHz Intel Core i7	4 Gb di 8 (1333 MHz DDR3)	0,184 s	5,417
Windows 8 (virtualizzato)	2,6 GHz Intel Core i7	8 Gb di 16 (1600 MHz DDR3)	0,170 s	5,882

Tabella 6.1: Riassunto delle prestazioni ottenute su diversi calcolatori

I risultati confermano dunque la possibilità di migliorare l'esperienza utente al migliorare delle caratteristiche del calcolatore utilizzato, oltre che al variare del sistema operativo. Si ritiene inoltre che performance superiori si potrebbero ottenere con un'interfaccia di Comsol creata ad-hoc per le simulazioni real-time.

6.5 Aggiunta di elementi grafici

Per migliorare l'esperienza dell'utente è possibile inserire alcuni elementi grafici che consentano di comprendere meglio quale sia il punto di applicazione della forza e i vincoli utilizzati. Attraverso l'utilizzo di elementi semplici quali linee, rettangoli e circonferenze, si è dimostrato con molte prove che il tempo di soluzione peggiora solo di pochi decimillesimi di secondo, pertanto è comunque conveniente adottarli se possono essere utili a evitare incomprensioni. Purtroppo, poiché l'immagine viene continuamente sovrascritta, questi elementi dovranno essere inseriti tutti all'interno del ciclo while, in modo da poter essere ridisegnati ad ogni step.

Si parte quindi dalla rappresentazione grafica dei vincoli. In questo caso l'utente potrebbe decidere di modificare il file "barretta_init.m" in modo da ottenere una configurazione carrello-cerniera o cerniera-cerniera. Se si apre il file in questione, si nota che basta una semplice modifica alla sezione "constraints" per passare da una configurazione all'altra:

```
% Constraints
clear pnt
pnt.Hy = {0,1,1,0};   %cerniera
pnt.Hx = {0,0,1,0};   %carrello
```

```
% Constraints
clear pnt
pnt.Hy = {0,1,1,0};   %cerniera
pnt.Hx = {0,1,1,0};   %cerniera
```

Poiché il particolare tipo di configurazione scelto dovrà essere "comunicato" al file "barretta_update.m", è opportuno inserire una apposita variabile (qui chiamata "constr") da utilizzare come output per il file di inizializzazione, e come input per quello di aggiornamento:

```
function [solTriangles,solVertices,constr] = barretta_init()

...

% Constraints
clear pnt
pnt.Hy = {0,1,1,0};
pnt.Hx = {0,0,1,0};

constr=cell2mat(pnt.Hx(2));

...
```

Poiché la variabile "pnt.Hx" è di tipo "cell", è necessario convertirla in una matrice di numeri, attraverso l'apposita funzione di Matlab "cell2mat". Il secondo elemento di "pnt.Hx" è quello di interesse, e la variabile "constr" varierà insieme ad esso. Ciò tornerà utile per "far capire" al file "barretta_update.m" qual è lo schema di vincolo scelto dall'utente.

Il file di aggiornamento è stato poi modificato come segue:

```
postplot(fem, ...
        'tridata',{'mises_pn','cont','internal','unit','MPa'}, ...
        'tridlim',[0 color], ...
        'trimap','jet(1024)', ...
        'deformsub',{'u','v'}, ...
        'deformscale',def_scale, ...
        'axis',[min_x max_x min_y max_y],...
        'title',(['Max von Mises stress: ' num2str(SolSigmaMax),...
        ' MPa  -   Force: ' num2str(mouse_posy) 'N']));


    if constr==1    %caso cerniera/cerniera
        [Xs Ys Zs]=sphere(30);
        ang=0:0.01:2*pi;
        xp=3*cos(ang);
        yp=3*sin(ang);
        fill(200+xp,-10+yp,'k','LineWidth',2,'FaceAlpha',0); %cerniera1
        hs=surf(200+3*Xs,(3*Ys-10),3*Zs);
        set(hs,'EdgeColor','none', ...
            'FaceColor','w', ...
            'FaceLighting','phong', ...
            'AmbientStrength',0.6, ...
            'DiffuseStrength',0.8, ...
            'SpecularStrength',0.9, ...
            'SpecularExponent',50, ...
            'BackFaceLighting','lit');

fill(xp,-10+yp,'k','LineWidth',2,'FaceAlpha',0);     %cerniera2
        hs=surf(3*Xs,(3*Ys-10),3*Zs);
        set(hs,'EdgeColor','none', ...
            'FaceColor','w', ...
            'FaceLighting','phong', ...
            'AmbientStrength',0.6, ...
            'DiffuseStrength',0.8, ...
            'SpecularStrength',0.9, ...
            'SpecularExponent',50, ...
            'BackFaceLighting','lit');       %Continua a pag. seguente...
```

```
%...Continua da pag. precedente

elseif constr==0    %caso carrello/cerniera

            Ixr=(postint(fem,'u', ...
            'unit','mm', ...          %spostamento x del carrello
            'dl',1, ...
            'edim',0))*def_scale;
        [Xs Ys Zs]=sphere(30);
        ang=0:0.01:2*pi;
        xp=3*cos(ang);
        yp=3*sin(ang);
        fill(200+xp,-10+yp,'k','LineWidth',2,'FaceAlpha',0);  %cerniera
        hs=surf(200+3*Xs,(3*Ys-10),3*Zs);
        set(hs,'EdgeColor','none', ...
            'FaceColor','w', ...
            'FaceLighting','phong', ...
            'AmbientStrength',0.6, ...
            'DiffuseStrength',0.8, ...
            'SpecularStrength',0.9, ...
            'SpecularExponent',50, ...
            'BackFaceLighting','lit');
        fill(xp+Ixr,-10+yp,'k','LineWidth',2,'FaceAlpha',0);  %carrello
        hs=surf(3*Xs+Ixr,(3*Ys-10),3*Zs);
        set(hs,'EdgeColor','none', ...
            'FaceColor','w', ...
            'FaceLighting','phong', ...
            'AmbientStrength',0.6, ...
            'DiffuseStrength',0.8, ...
            'SpecularStrength',0.9, ...
            'SpecularExponent',50, ...
            'BackFaceLighting','lit');
        xp=2*cos(ang)+Ixr;
        yp=2*sin(ang);
        fill(-4.6+xp,-20.5+yp,'w','LineWidth',1,'FaceAlpha',1);
        fill(xp,-20.5+yp,'w','LineWidth',1,'FaceAlpha',1);
        fill(4.6+xp,-20.5+yp,'w','LineWidth',1,'FaceAlpha',1);
        x = [(-4.5)+Ixr (4.5)+Ixr Ixr];
        y = [-18 -18 -13];
        fill(x, y, 'k');
        line([-20 20], [-23 -23],'LineWidth',3, 'color', 'k'); %terreno
    end

    grid on
    axis on

    drawnow
    hold off
```

Le modifiche introdotte sono molte, anche se ricorsive. Procedendo in ordine si può notare, dopo il "postplot", il comando "hold on", con il quale si impone che tutte le figure create si sovrappongano all'immagine di volta in volta già esistente. Segue il primo dei due possibili schemi di vincolo per questo caso, quella della doppia cerniera. Quando la variabile "constr" assume valore "1", vengono disegnate due semplici sfere bianche alle estremità inferiori della barretta. Se la variabile "constr" è invece "0", viene disegnata una cerniera sull'estremo inferiore destro, e un carrello su quello sinistro. In questo caso viene anche calcolato lo spostamento del carrello, attraverso la funzione "postint", alla quale è stata data in ingresso il codice numerico corrispondente al nodo 1 (dove è inserito il carrello) e in uscita è stata chiesta l'ascissa di quel punto, immagazzinata poi nella variabile "Ixr". Quest'ultima viene utilizzata successivamente per aggiornare la posizione del carrello, che si sposta con l'estremità della barretta. Si nota anche che la variabile "Ixr" è moltiplicata per il fattore di scala "def_scale", che era stato dichiarato all'inizio dell'algoritmo, e che in questo caso assume un valore pari a 15. Tra le ultime righe di codice si nota il comando "drawnow", che obbliga il software a visualizzare il risultato a ogni step, e l'importantissimo "hold off" finale, che consente di ripristinare la situazione originaria per lo step successivo, senza sovrapposizioni errate. I due risultati grafici sono infine i seguenti:

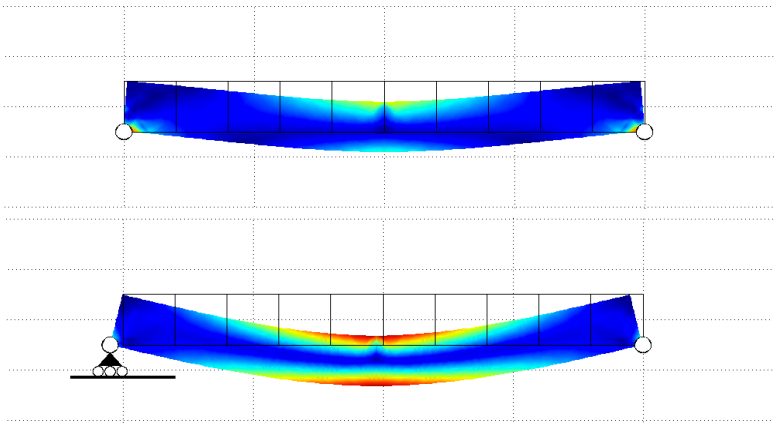


Figura 6.7: Schemi di vincolo cerniera/cerniera e carrello/cerniera

Un'altra impostazione che l'utente può facilmente modificare è il punto di applicazione della forza. Si considerino le seguenti righe di codice del file "barretta_init.m":

```
pnt.Fy = {0,0,0,'y*x'};
pnt.ind = [2,1,1,1,1,1,1,1,1,1,1,4,1,1,1,1,1,1,1,1,3,1];
appl.pnt = pnt;
```

Nel vettore "pnt.ind", come già visto nel Capitolo 4, sono contenute tutte le informazioni sui nodi della struttura, ed in particolare su vincoli e carichi. In questo caso il "2" sul primo elemento sta ad indicare che su quel nodo sarà presente un vincolo di tipo carrello, e allo stesso modo il "3" sul penultimo nodo, il ventunesimo, che lì è applicato il vincolo di tipo cerniera. Se si apre il modello salvato in precedenza nel formato .mph di Comsol e si fa una veloce verifica, si scopre infatti che si ha una diretta corrispondenza tra nodi della struttura e posizioni sul vettore "pnt.ind":

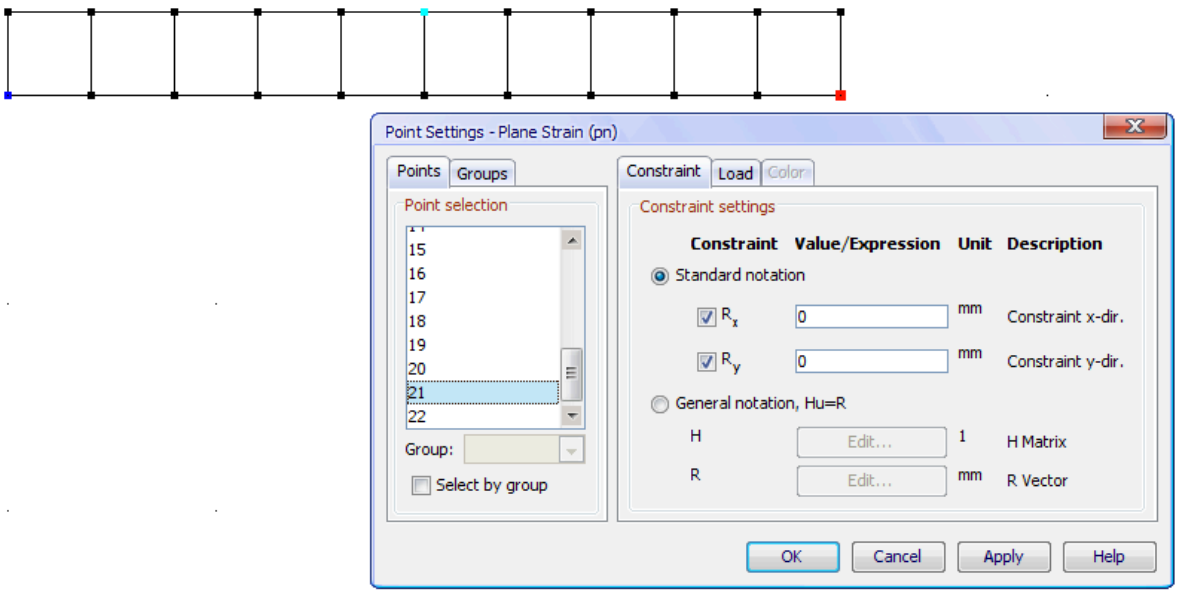


Figura 6.8: Corrispondenza tra cerniera e ventunesimo nodo della struttura

Nella Figura 5.8 si nota che il nodo 21 della struttura è proprio quello corrispondente al punto di applicazione della cerniera. Se si procede in Comsol allo stesso modo, si può determinare il numero corrispondente ai nodi su cui l'utente può spostare l'applicazione della forza, spostando l'elemento "4" del vettore su una delle seguenti posizioni:

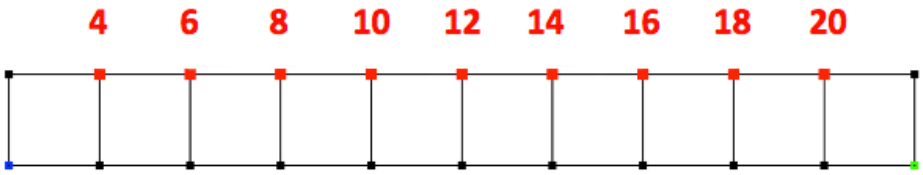


Figura 6.9: Numero di ciascun nodo su cui la forza è applicabile

Ora quindi è molto semplice inserire all'interno del file "barretta_init.m" una nuova variabile, chiamata ad esempio "load", che potrà in seguito essere passata al file di aggiornamento allo stesso modo visto con la variabile "constr":

```
pnt.Fy = {0,0,0,'y*x'};
pnt.ind = [2,1,1,1,1,1,1,1,1,1,1,4,1,1,1,1,1,1,1,1,3,1];
if pnt.ind(4)==4
    load=4;
elseif pnt.ind(6)==4
    load=6;
elseif pnt.ind(8)==4
    load=8;
elseif pnt.ind(10)==4
    load=10;
elseif pnt.ind(12)==4
    load=12;
elseif pnt.ind(14)==4
    load=14;
elseif pnt.ind(16)==4
    load=16;
elseif pnt.ind(18)==4
    load=18;
elseif pnt.ind(20)==4
    load=20;
end
appl.pnt = pnt;
```

Dopo aver modificato opportunamente i file "barretta_init.m", "play.m" e barretta_update.m" in modo che sia possibile il passaggio della nuova variabile "load" (analogamente a quanto già visto per la costante "constr"), è possibile concentrarsi sul solo file "barretta_update.m", all'interno del quale dovranno essere inserite delle righe di codice che sfruttino opportunamente il valore della nuova variabile. Per identificare il punto scelto dall'utente si può ad esempio disegnare una freccia che cambi verso a seconda che la barretta si trovi in una condizione di flessione verso il basso o verso l'alto. Per farlo sono sufficienti pochi passaggi, sempre all'interno del ciclo while:

```
%Freccia:

        x_arr=(load-2)*10;

        if Iy<0
            xf = [x_arr-10+Ix x_arr+10+Ix x_arr+Ix];
            yf = [15+Iy 15+Iy 5+Iy]+10;
            fill(xf, yf, 'r');
            xr = [x_arr-4+Ix x_arr+4+Ix x_arr+4+Ix x_arr-4+Ix];
            yr = [15+Iy 15+Iy 35+Iy 35+Iy]+10;
            fill(xr, yr, 'r');
        elseif Iy>=0
            xf = [x_arr-10+Ix x_arr+10+Ix x_arr+Ix];
            yf = [25+Iy 25+Iy 35+Iy]+10;
            fill(xf, yf, 'r');
            xr = [x_arr-4+Ix x_arr+4+Ix x_arr+4+Ix x_arr-4+Ix];
            yr = [5+Iy 5+Iy 25+Iy 25+Iy]+10;
            fill(xr, yr, 'r');
        end
```

Andando con ordine, si trova prima di tutto la dichiarazione di una nuova variabile "x_arr", che definisce l'ascissa della freccia. Questa è calcolata dal valore della variabile "load", che viene diminuita di 2 e moltiplicata per 10.

Supponendo, infatti, che l'utente disponga il "4" del vettore "pnt.ind" nella sua ottava posizione (indicata con "*" nell'immagine seguente):

```
                              *
pnt.ind = [2,1,1,1,1,1,1,4,1,1,1,1,1,1,1,1,1,1,1,3,1];
if pnt.ind(4)==4
    load=4;
elseif pnt.ind(6)==4
    load=6;
elseif pnt.ind(8)==4        %<-----!!!
    load=8;

...
```

In questo caso alla variabile "load" verrà attribuito il valore "8" all'interno del ciclo "if". Dalla Figura 5.9 si nota che il nodo "8" nella struttura corrisponde al vertice superiore destro del terzo quadrato di lato pari a 20 mm della struttura. Dunque, poiché il primo quadrato era stato generato dall'origine degli assi, l'ascissa di tale punto deve essere x_arr=60. Svolgendo l'operazione:

$$\mathbf{x\_arr = (load-2)*10}$$

Si ottiene, infatti:

$$\mathbf{x\_arr = (8-2)*10 = 60}$$

Procedendo con l'analisi del codice, si trovano semplici istruzioni che definiscono un rettangolo e un triangolo, che opportunamente sistemati formano una freccia rossa, attraverso il comando "fill(x, y, 'r')".

I valori "Ix" e "Iy" sono stati calcolati attraverso il comando "postint", alla stregua di "Ixr", col seguente codice:

```
Ix=(postint(fem,'u', ...
        'unit','mm', ...
        'dl',[load], ...
        'edim',0))*def_scale;

Iy=(postint(fem,'v', ...
        'unit','mm', ...
        'dl',[load], ...
        'edim',0))*def_scale;
```

A queste nuove variabili corrispondono rispettivamente l'ascissa e l'ordinata del punto sollecitato. Sommando tali valori alle ascisse e alle ordinate del rettangolo e del triangolo che costituiscono la freccia, si ottiene una geometria solidale con il movimento del punto in questione. Dalle istruzioni necessarie per generare la freccia, si nota infine che sono contenute in un ciclo "if". Se lo spostamento lungo y è positivo, la freccia sarà rivolta verso l'alto, se negativo verso il basso.

Si può comprendere meglio quanto e stato fatto finora osservando l'immagine successiva:

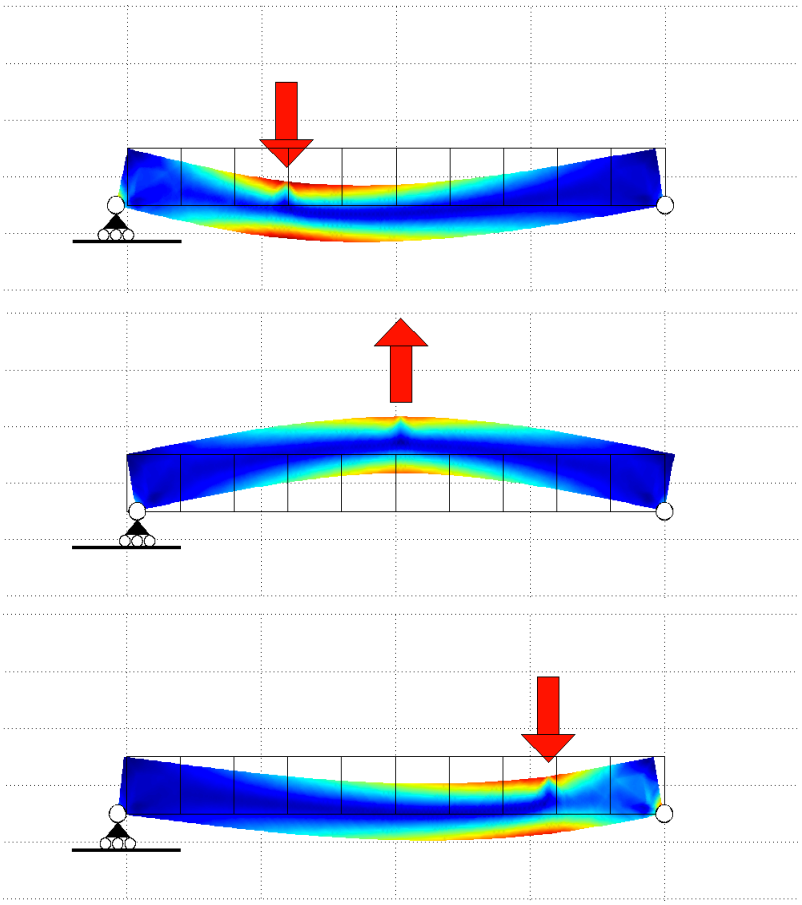


Figura 6.10: Freccia e suo spostamento nel caso di scelta dei nodi 8, 12 o 18 (dall'alto verso il basso)

6.6 Uscita dall'applicazione

L'ultimo problema che resta da risolvere è come "chiudere" l'applicazione fin qui creata una volta che è stata avviata. Fino a questo momento, infatti, una volta avviato il ciclo "while" all'interno del file "barretta_update.m" è impossibile uscirne, se non con una chiusura forzata, utilizzando la combinazione di tasti "Ctrl+C". Il modo più intuitivo sarebbe invece quello di uscire dall'applicazione quando si chiude la finestra grafica. Tuttavia, se ciò viene fatto allo stato attuale, semplicemente la finestra viene subito rimpiazzata da una nuova. Dopo vari tentativi e ricerche, si è scoperto che Matlab è in grado di registrare l'apertura o la chiusura di un plot attraverso il comando "ishandle". Per capirne il funzionamento si può eseguire un veloce test: si apre una nuova finestra grafica direttamente dalla "Command Window" di Matlab, con il comando "figure(1)", che farà aprire una finestra grafica vuota. Se successivamente si digita il comando "ishandle(1)", Matlab restituisce come output il valore "1", che sta per "true". Il software comunica quindi che la figura con indice "1" in quel momento è aperta. Se si digita ora il comando "close all" la figura verrà chiusa, e se si ripete il comando "ishandle(1)", la risposta sarà "0" ossia "false".

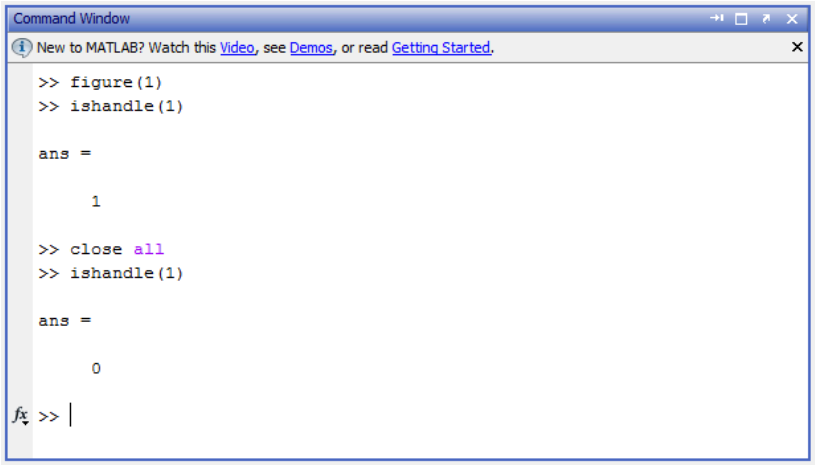


Figura 6.11: Test di verifica del funzionamento del comando "ishandle" di Matlab

L'idea è quindi quella di inserire un "if" all'interno del ciclo "while" del file di aggiornamento, in modo tale che al verificarsi del caso "0", l'algoritmo s'interrompa. Tuttavia, rimane il problema di come "dire" al programma di chiudersi. Dopo aver svolto molti tentativi, si è infine scoperto che in realtà il modo più semplice per dare questo tipo di input al software, è quello di provocare quello che si potrebbe definire un "errore controllato". Si ricorre a una funzione esterna, qui chiamata "stopall.m" all'interno della quale è presente il comando "error()", che in questa forma è sufficiente a provocare l'uscita dal loop infinito. Prima di lanciare questo comando è inoltre possibile comunicare all'utente la chiusura dell'applicazione attraverso un messaggio di testo, che verrà visualizzato all'interno della "Command Window". Di seguito si riporta il codice completo della funzione "stopall.m".

```
function stopall()

clc
clear

disp('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%')
disp('    Thank you for using this program!      ')
disp('  Please ignore the error messages below.  ')
disp('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%')
disp(' ')

error()

end
```

Per far avviare questa nuova funzione è sufficiente inserire il seguente "if" all'interno del ciclo "while" del file "barretta_update.m":

```
if ishandle(1)==0
    stopall;
end
```

Se durante l'esecuzione dell'algoritmo si prova ora a chiudere la finestra grafica, si otterrà l'immediata interruzione del *loop*, e la visualizzazione di diversi messaggi d'errore all'interno della "*Command Window*", preceduti dal messaggio inserito nella funzione d'errore:

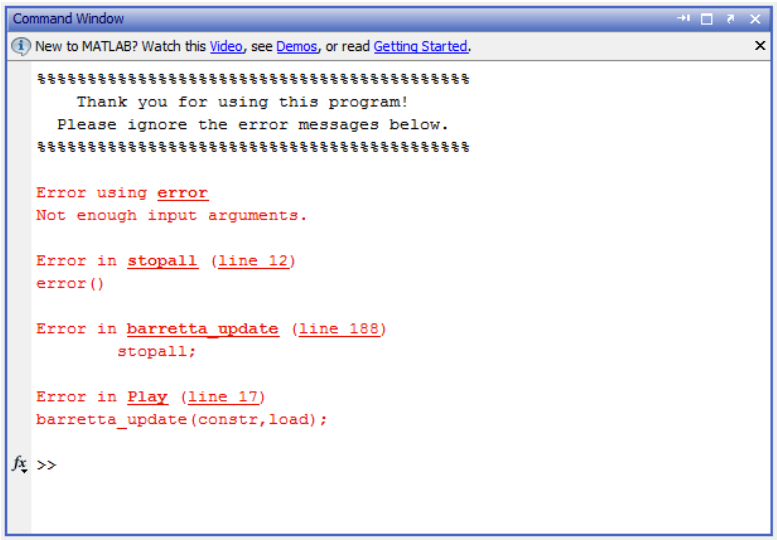


Figura 6.12: Messaggio finale e interruzione dell'algoritmo alla chiusura della finestra grafica

6.7 Problemi riscontrati

Lanciando nuovamente il file "haptikdevice_list.m", si nota che ora il *Phantom* viene correttamente riconosciuto. Sussistono tuttavia altri problemi irrisolti e non compresi fino in fondo:

a) La libreria sembra non accettare il comando "clear all" di Matlab, che si utilizza normalmente per cancellare tutte le variabili dichiarate. Il problema si manifesta dopo aver attribuito un nome al dispositivo hapti (ad esempio: "h=haptikdevice"). Se in seguito si esegue un "clear all" il software si blocca totalmente, e compare fissa la scritta "busy", come se si stesse svolgendo un qualche tipo di operazione complessa. L'unico modo per tornare a lavorare è riavviare il programma forzandone l'uscita attraverso il "*Task Manager*" di Windows. L'unica alternativa è il comando "clear", che non ha tuttavia la stessa potenza di "clear all".

b) Dopo aver finito di lavorare con il sistema *haptic*, questo rimane comunque attivo, anche dopo aver chiuso il file .m che si stava utilizzando. Se si lancia l'esempio fornito con la Haptik Library "horizontal_plane.m" (con il quale l'utente è invitato a "sentire" per mezzo del dispositivo un piano orizzontale virtuale), e successivamente si chiude il file, il dispositivo continuerà a restituire una forza, che la maggior parte delle volte è totalmente casuale. Inoltre, se si riavvia il file appena chiuso, Matlab si blocca. Anche in questo caso l'unica soluzione è la chiusura forzata dell'applicazione, dopo la quale anche il dispositivo *haptic* cessa di restituire forze.

c) Matlab si blocca in modo irreversibile quando si attribuisce per due volte un nome al dispositivo *haptic* per mezzo del comando "haptikdevice". Non potendo utilizzare "clear all", il dispositivo va dichiarato quindi solo ed esclusivamente all'inizio, e in ogni caso per terminare completamente l'esecuzione di uno script è necessario un riavvio forzato del software.

d) Matlab normalmente non riconosce il dispositivo *haptic* collegato al computer al primo avvio di Matlab. Con l'ausilio del "*Task Manager*" si è notato che al primo avvio dell'applicazione "*Comsol with Matlab*", Matlab viene lanciato in realtà due volte. Una delle due applicazioni è "silente", mentre l'altra si apre regolarmente. E' necessario chiuderle entrambe dal manager di Windows e riavviare l'applicazione prima di procedere. In generale il dispositivo *haptic* non viene riconosciuto tutte le volte che è aperta più di una finestra principale di Matlab.

6.8 Considerazioni finali

Dopo aver affrontato e risolto la maggior parte dei problemi che si sono presentati, è possibile fare alcune considerazioni sul lavoro fin qui svolto. Il primo obiettivo della tesi era trovare un modo per svolgere un'analisi FEM in tempo reale. Non esistendo applicazioni progettate appositamente per questo scopo, è stata scelta una soluzione di compromesso, che consentisse di lavorare con strumenti già incontrati durante il corso di studi, in modo non solo da accelerare il lavoro ma anche, in prospettiva futura, di risultare familiare a studenti e docenti. Il secondo obiettivo era quello di trovare un sistema per "esplorare" le strutture create con un dispositivo *haptic*, in modo da fornire all'utente un feedback tattile della simulazione, oltre che visivo. Ciò è stato possibile attraverso la cooperazione di Comsol, Matlab e Haptik Library.

Solo dopo aver raggiunto entrambi i traguardi, si è infine pensato di introdurre la possibilità di controllare la struttura anche con il mouse, in modo da rendere l'applicazione più facilmente fruibile, dato il costo e la scarsa reperibilità di dispositivi *haptic*. Nell'immagine seguente è rappresentato lo schema logico di funzionamento dell'applicazione, allo stato attuale del lavoro:

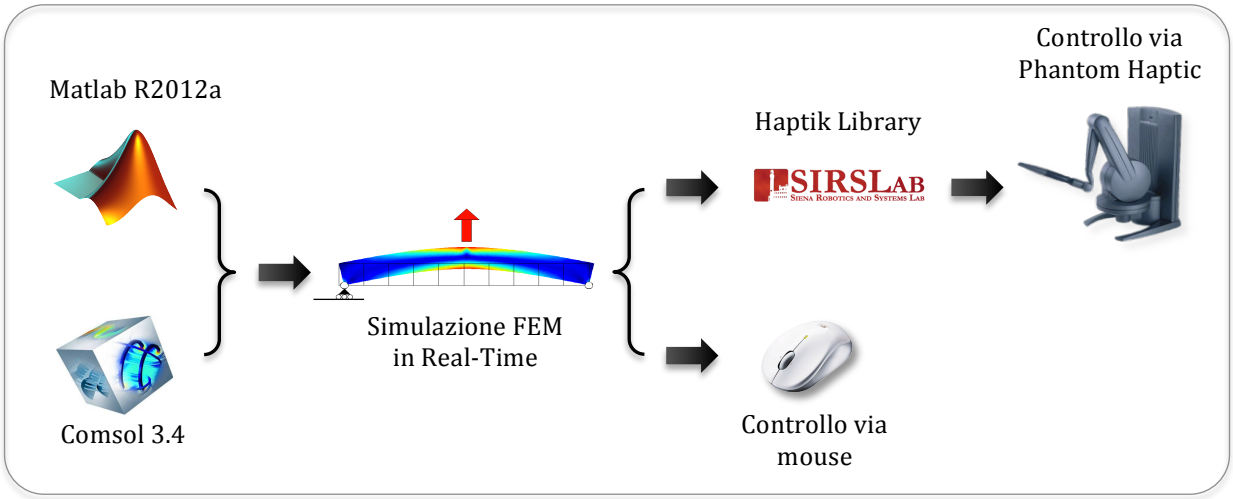


Figura 6.13: Schema dei passaggi logici che governano il funzionamento del processo

Il metodo scelto, come si è visto nei paragrafi precedenti, è tuttavia ancora lontano dall'essere perfetto. Pur avendo automatizzato molte operazioni e nonostante si sia notevolmente migliorato il tempo medio di risoluzione della struttura, il problema principale rimane la fluidità della simulazione. Il miglior risultato ottenuto si è attestato finora a meno di 6 fotogrammi al secondo, che è ancora un risultato lontano dai 10-15 fps che garantirebbero una discreta fluidità. Tuttavia, poiché il risultato migliora al migliorare delle caratteristiche del calcolatore, e poiché i mezzi utilizzati non sono ottimali e non sono pensati per funzionare in tempo reale, si è ritenuto di aver comunque raggiunto un buon risultato. Nei capitoli successivi si applicheranno le conoscenze fin qui acquisite per creare modelli sempre più complessi e d'interesse per l'insegnamento di scienza delle costruzioni e di altre materie che abbiano a che fare con questa tipologia di argomenti. Si vedranno inoltre i passi principali che hanno portato alla realizzazione di una vera e propria interfaccia grafica per l'applicazione finale, allo scopo di migliorare l'usabilità e di consentire a docenti e studenti di utilizzare i modelli più complessi in modo del tutto automatico, consentendo comunque l'impostazione dei parametri di maggior interesse per ciascuna struttura.

Capitolo 7: Interfaccia utente

Dopo aver realizzato una simulazione FEM in tempo reale controllandone la deformazione con un dispositivo *haptic*, si è cominciato a pensare all'usabilità dell'applicazione. Come si è visto nel capitolo precedente, per il momento l'utente può modificare i parametri più importanti della struttura, come il materiale, il carico o i vincoli, solo agendo direttamente sui file .m di Matlab. L'applicazione finale dovrà essere però pensata per utilizzo didattico, dunque è semplice immaginare quanto questo metodo possa essere poco agevole e lento, così come sia complesso capire quali modifiche si stiano apportando ai file. Pertanto si è deciso di creare una vera e propria interfaccia grafica (GUI, "*Graphical User Interface*") e di valutare l'efficacia di questa soluzione. Nei paragrafi seguenti saranno illustrati i passaggi fondamentali che hanno portato alla realizzazione dell'interfaccia dell'applicazione finale, che è stata chiamata "HaptikFem", nome che ne riassume il senso ("Haptik" richiama sia al funzionamento con i dispositivi *haptic* che alla Haptik Library, "FEM" fa riferimento all'analisi agli elementi finiti).

7.1 Scelta dell'interfaccia

E' possibile creare con Matlab delle semplici interfacce grafiche dotate di pulsanti e caselle, che possono essere utilizzate dagli utenti come in un qualsiasi programma per PC allo scopo di inserire dati numerici e visualizzare grafici. Un'interfaccia di questo tipo, tuttavia, non si adatta bene allo scopo che ci si sta prefiggendo, poiché l'editor per crearle non è sviluppato con lo scopo di cooperare con Comsol o con un sistema *haptic*, tantomeno in tempo reale.

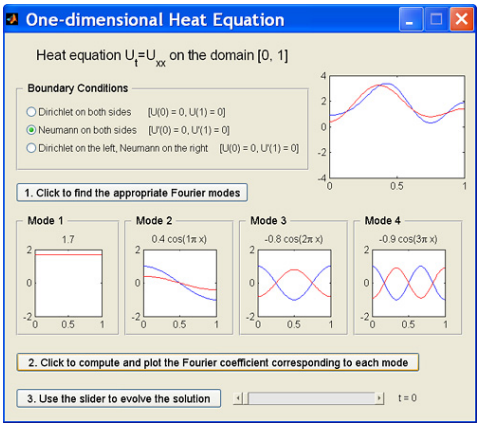


Figura 7.1: Esempio di GUI creata con Matlab

Ciò si è invece scelto di fare, è creare piuttosto GUI molto semplice, costituita da una successione d'immagini. L'interfaccia è pensata per dare l'opportunità all'utente di scegliere alcuni tra i più importanti parametri della struttura, senza dover mai accedere ai file del programma. Lo scopo che ci si è prefisso fin dall'inizio, è stato quello di "guidare" l'utente, anche il meno esperto, attraverso un percorso a tappe, fino al risultato finale. In Figura 6.1 si nota invece l'aspetto della maggior parte delle GUI: l'utente ha a disposizione un'unica schermata, all'interno della quale si scontra con una numerosa serie di pulsanti, scelte e grafici. Per questo l'obiettivo è sempre stato quello di porsi dal lato dell'utente, più che da quello del programmatore. Per chiarire meglio il percorso che si è scelto di seguire, si può osservare lo schema logico riassuntivo dell'interfaccia che è stata creata, rappresentato nell'immagine seguente. Si notano otto livelli, che guidano l'utente nelle diverse scelte, fino alla fase di simulazione. In caso di errore l'utente è sempre in grado, ovunque si trovi, di tornare allo *step* precedente. Nei paragrafi seguenti saranno analizzate nel dettaglio le caratteristiche di ciascun livello.

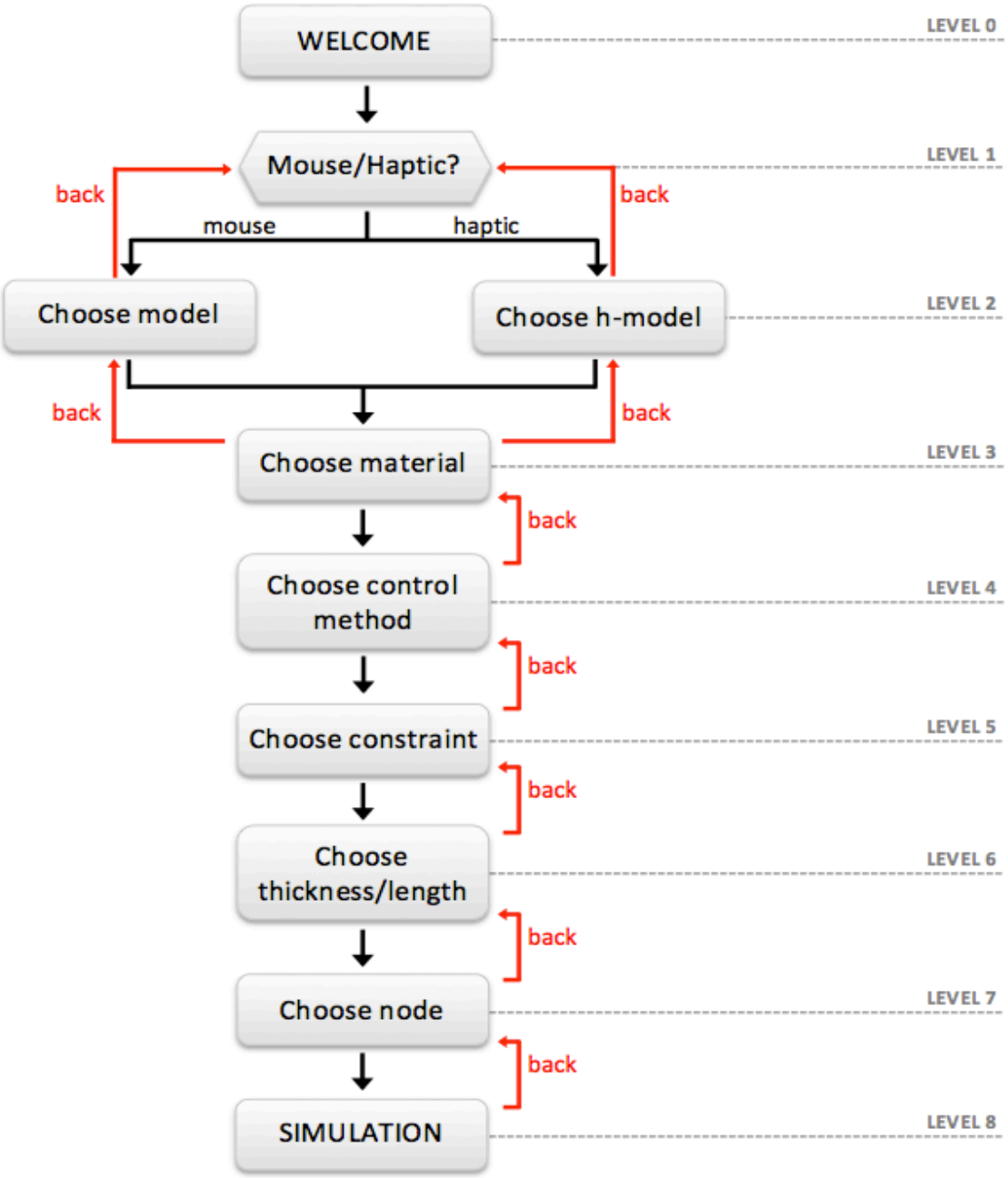


Figura 7.2: Schema logico a blocchi dei livelli dell'applicazione finale

7.2 Livello 0: inizializzazione

Nel livello 0 dell'applicazione vengono svolte alcune operazioni preliminari nascoste alla vista dell'utente, al quale viene solo mostrata la seguente schermata di benvenuto:

Figura 7.3: Schermata iniziale di benvenuto

L'immagine di benvenuto, così come tutte le altre, è stata realizzata con l'ausilio di Word e di Photoshop. Il formato delle immagini è .jpg e hanno dimensioni di 1507x967 pixel. Quando l'utente avvia il file "Play.m" contenuto nella cartella principale dell'applicazione, vengono anche avviati diversi algoritmi di inizializzazione:

```
clc
clear
close all

addpath(genpath(pwd));    %add folder and subfolders in the path
```

Le prime righe di codice del file d'inizializzazione puliscono la "*Command Window*" e il "*Workspace*" di Matlab, cancellando eventuali variabili dichiarate in precedenza e chiudendo tutte le finestre grafiche ancora aperte.

```
%welcome!

tic
while toc<5
    screen_size = get(0, 'ScreenSize');
    if screen_size(3)<=1280 && screen_size(4)<=800
        font=2;
    else
        font=0;
    end
    f1=figure(1);
    set(f1,'Units', 'Normalized', 'OuterPosition', [0 0 1 1]);
    welcome = imread('welcome.jpg');   %(1507x967)
    sz = size(welcome);
    imagesc([0 sz(2)], [0 sz(1)], welcome);   %plot wallpaper
    axis off
    grid off
    drawnow
end
```

Le righe di codice seguenti avviano un ciclo "while" che si occupa non solo di visualizzare per una durata predeterminata (in questo caso 5 secondi) la schermata di benvenuto, ma creano una nuova finestra grafica a tutto schermo, normalizzano le unità di misura e individuano la risoluzione del monitor che si sta utilizzando allo scopo di adeguare la dimensione dei font dei caratteri che verranno utilizzati in seguito.

```
%is there any haptic device connected?

try   %if yes (no errors)
    haptikdevice_list;
    haptic=haptikdevice; %haptikdevice;
    hap=1;
catch  %if no (error in haptikdevice_list)
    hap=0;
    haptic=0;
end

menu(0,hap,font,b,rr,haptic);    %Start the program
```

Le ultime righe del codice si occupano dell'individuazione di un eventuale dispositivo *haptic* collegato al computer. Per farlo, viene per prima cosa richiamata la funzione "haptikdevice_list" della Haptik Library, che restituisce un errore solo quando non c'è alcun sistema *haptic* connesso. Se la funzione restituisce un errore, una variabile "hap" assume valore "1", in caso contrario "0". Questa variabile sarà poi inviata al file "menu.m", che viene avviato alla conclusione del ciclo while. Il file "menu.m" è il file che si occupa della gestione e sincronizzazione delle varie schermate. Per brevità si rimanda alla sua analisi diretta; il file in questione è contenuto all'interno della sottocartella "menu" dell'applicazione.

7.3 Livello 1: scelta del dispositivo

Com'è già stato anticipato nei capitoli precedenti, l'applicazione è stata pensata in modo tale da poter funzionare anche in assenza di un dispositivo *haptic*. La variabile "hap" vista al paragrafo precedente è utilizzata nel file "devices.m", lanciato a sua volta dal file "menu.m". Secondo il valore assunto dalla variabile, viene lanciata una delle seguenti schermate:

Figura 7.4: Schermata della scelta del dispositivo in presenza e assenza di un sistema haptic

Se al computer è collegato un dispositivo *haptic* (nella fattispecie il sistema "*Phantom*", che è stato testato in laboratorio) verrà visualizzata la schermata a sinistra, in caso contrario quella a destra. Nella parte superiore di questa finestra grafica, così come nelle seguenti, sono riportate tutte le indicazioni utili per proseguire. In questa fase in particolare, l'utente può selezionare il dispositivo semplicemente utilizzando la freccia destra o sinistra della tastiera. Premendo invio si può invece proseguire agli *step* successivi. L'algoritmo dedicato alla selezione, di cui se ne riporta solo una parte indicativa, è in realtà abbastanza semplice, e si limita a spostare un rettangolo rosso semitrasparente (comando "patch" di Matlab) in concomitanza della pressione dei pulsanti "left/right" sulla tastiera, per indicare all'utente la scelta fatta. Ovviamente in caso di assenza di un dispositivo haptic sarà selezionabile solo l'opzione "mouse":

```
if move==1 && j==0 %haptic --> mouse
     imagesc([0 sz(2)], [0 sz(1)], device);
     image(flipdim(device,1));
     p=patch([754 1506 1506 754],[1 1 800 800],'r');
     set(p,'FaceAlpha',0.5);
     set(gca,'ydir','normal');
     hd=1;
     axis off
     j=1;
```

L'effetto ottenuto è il seguente:

Figura 7.5: Selezione del dispositivo mediante l'utilizzo dei comandi da tastiera

Come si vedrà nei paragrafi seguenti, lo stile delle schermate sarà in linea con quello di Figura 6.5. Tutte le istruzioni utili saranno inserite nella parte alta, e l'utente potrà selezionare l'opzione desiderata utilizzando la tastiera. Si è scelto di non utilizzare il mouse per selezionare le opzioni, sia per definirlo esclusivamente come "device" da utilizzarsi solo nella fase di simulazione, sia per difficoltà di programmazione legate al suo utilizzo all'interno delle schermate. Alla pressione del pulsante "invio" sulla tastiera, si passa alla fase successiva: la scelta del modello.

7.4 Livello 2: scelta del modello

Quando l'utente sceglie il dispositivo da utilizzare, una variabile assume un valore nullo o unitario. Tale variabile è poi opportunamente gestita dal file "menu.m", che lancia l'algoritmo che introduce l'utente alla scelta dei modelli.

In particolare, se l'utente ha scelto di utilizzare un dispositivo *haptic*, verranno lanciati ulteriori algoritmi opportunamente modificati per poter funzionare con questo tipo di hardware (sono stati chiamati "h-models" e sono contenuti nella cartella "haptic" dell'applicazione); in caso contrario verranno lanciati algoritmi "classici", che non ricorrono alle speciali funzioni della Haptik Library (sono chiamati semplicemente "models" e sono contenuti nella cartella "mouse" dell'applicazione). L'utente può scegliere tra numerosi modelli, sette con visualizzazione bidimensionale e quattro con visualizzazione e modellizzazione tridimensionale, dei quali si parlerà approfonditamente nel capitolo successivo:

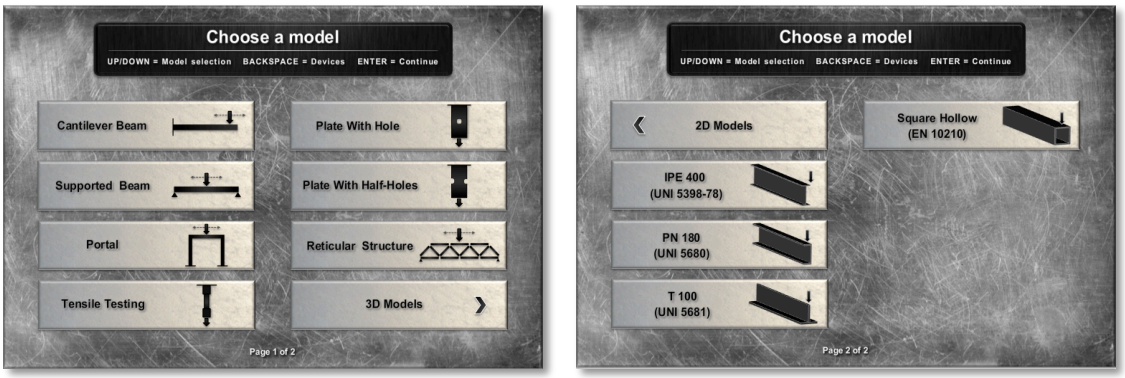


Figura 7.6: Selezione del modello 2D (a sinistra) o 3D (a destra)

Anche in questo caso all'utente sono fornite tutte le indicazioni utili alla selezione del modello nella parte superiore della schermata. In questo caso specifico, la selezione di un modello viene fatta premendo le frecce "up/down" della tastiera, la conferma avviene premendo "invio", e l'utente può tornare al livello 1 semplicemente premendo il "backspace".

7.5 Livello 3: scelta del materiale

L'utente viene successivamente guidato nella scelta del materiale con cui modellizzare la struttura scelta. A differenza di quanto accade con i classici software di simulazione agli elementi finiti, nei quali è necessario impostare manualmente i parametri caratteristici, in quest'applicazione l'utente può scegliere tra cinque dei materiali più utilizzati in campo ingegneristico, le cui caratteristiche sono preimpostate nel corrispondente algoritmo (file "materials.m" nella cartella "menu") e riassunte nella schermata di selezione. In ogni caso l'utente può facilmente bypassare questa fase accedendo al file corrispondente al modello scelto e modificando a piacimento le caratteristiche di uno o più materiali. L'operazione, come si nota dalla porzione di codice sottostante, è molto semplice ed è costituita da un lungo ciclo "if":

```
if mat==1       %steel
    equ.E = 210000;
    equ.rho = '7870 [kg/m^3]';

elseif mat==2   %aluminum
    equ.E = 73100;
    equ.rho = '2780 [kg/m^3]';

elseif mat==3   %concrete
    equ.E = 30000000;
    equ.rho = '2480 [kg/m^3]';

elseif mat==4   %composite
    equ.E = 130000;
    equ.rho = '1560 [kg/m^3]';

elseif mat==5   %polymeric
    equ.E = 2300;
    equ.rho = '1200 [kg/m^3]';
end
```

I materiali preimpostati tra cui l'utente può scegliere sono:

- Acciaio per uso strutturale Fe510/S355
- Lega di alluminio Al 2024-T3
- Cemento armato C20/25
- Materiale composito a fibre corte non orientate (isotropo)
- Policarbonato

Per la scelta del materiale l'utente può utilizzare, come già visto in precedenza, le frecce "left/right" della tastiera e confermare premendo "invio", o tornare alla scelta dei modelli premendo il tasto "backspace", come si può notare dall'immagine seguente:

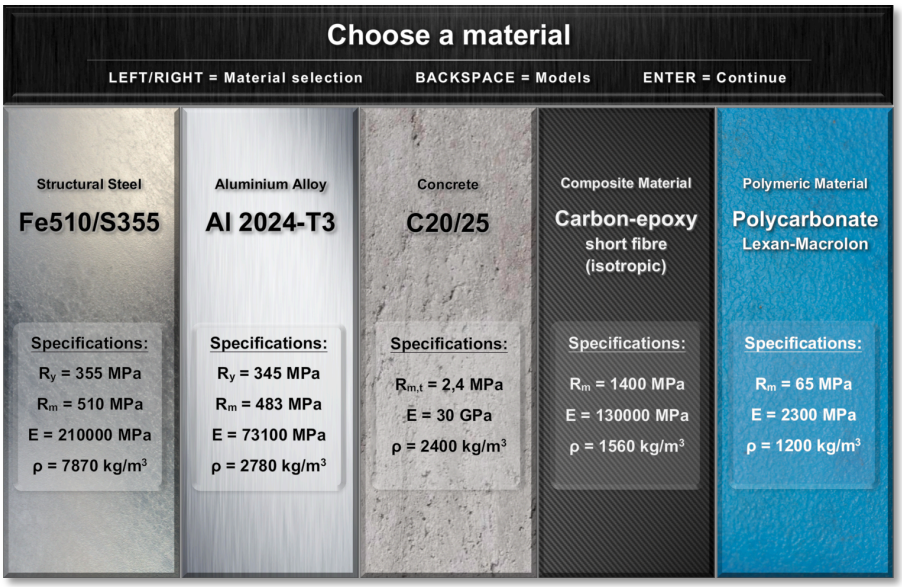


Figura 7.7: Schermata di selezione del materiale

7.6 Livello 4: scelta del metodo di controllo

Si prosegue con la scelta del metodo di controllo con cui eseguire la simulazione. Sono disponibili due opzioni:

- Controllo in forza
- Controllo in spostamento

Con il controllo in forza verrà applicato un carico, che l'utente potrà variare con lo spostamento del sistema *haptic* o del mouse, e verranno visualizzati a schermo lo sforzo massimo raggiunto dalla struttura in ogni istante e lo spostamento generato dal carico nel punto di applicazione. Con il controllo in spostamento s'impone invece una traslazione di un nodo della struttura e se ne valutano i corrispondenti sforzi e deformazioni. Logiche simili sono comunemente utilizzate nelle prove meccaniche che si eseguono nei laboratori di caratterizzazione dei materiali. Un caso tipico è quello della prova di trazione monoassiale [10]:

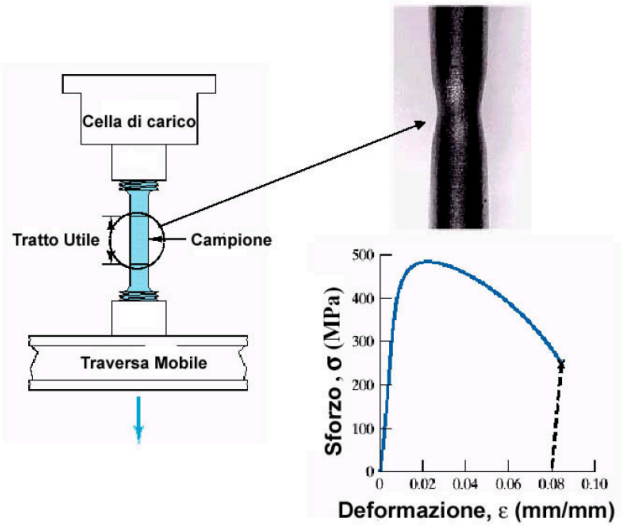


Figura 7.8: Prova di trazione monoassiale in controllo di spostamento

La prova consiste nell'applicazione di un carico monoassiale su un provino normato, e nella valutazione dello spostamento e della forza generata: il concetto è del tutto identico a quello che nell'applicazione è chiamato "controllo in forza". Nell'immagine seguente è riportata la schermata di selezione del metodo di controllo:

Figura 7.9: Schermata di selezione del metodo di controllo

7.7 Livello 5: scelta dello schema di vincolo della struttura

La fase successiva consente all'utente di scegliere quali vincoli applicare nei punti di appoggio della struttura. Come si è già visto nel Capitolo 4 è semplice, a livello algoritmico, modificare le condizioni di vincolo nei singoli punti. In questo caso, sono state inserite, a seconda del modello, una o più opzioni.

Quando l'utente sceglie uno schema di vincolo e preme "invio", interviene la seguente porzione del file "init.m" del modello scelto:

```
elseif r5==4   %hinge/roller
    pnt.Hy = {0,0,1,1};
    pnt.Hx = {0,0,1,0};
    if res(4)==1        %force
    pnt.Fy = {0,'y*x',0,0};
    pnt.ind = nodo;
    appl.pnt = pnt;
    elseif res(4)==2    %disp
    pnt.constrtype =
{'standard','general','standard','standard'};
    pnt.R = {0,{0;'y*x'},0,0};
    pnt.H = {0,{0,0;0,1},0,0};
    pnt.ind = nodo;
    appl.pnt = pnt;
    end

elseif r5==5  %hinge/hinge
    pnt.Hy = {0,0,1};
    pnt.Hx = {0,0,1};
    if res(4)==1        %force
    pnt.Fy = {0,'y*x',0};
    pnt.ind = nodo;
    appl.pnt = pnt;
    elseif res(4)==2    %disp
    pnt.constrtype =
{'standard','general','standard'};
    pnt.R = {0,{0;'y*x'},0};
    pnt.H = {0,{0,0;0,1},0};
    pnt.ind = nodo;
    appl.pnt = pnt;
    end
```

Come si può notare, la scelta fatta dall'utente genera un valore specifico per la variabile "r5". Secondo il valore che essa assume, viene applicato alla struttura, nello specifico punto d'appoggio, il vincolo corrispondente. Ad esempio, se la variabile "r5" assume valore "4", nei due punti d'appoggio della struttura cui si riferisce il codice riportato in precedenza, saranno inseriti rispettivamente un carrello e una cerniera; nel caso in cui

assuma valore "5", due cerniere. Nell'immagine seguente si può notare la schermata di selezione degli schemi di vincolo disponibili per una struttura a portale. Nella stessa schermata sono anche riassunte tutte le caratteristiche geometriche della struttura, che consentono all'utente di comprendere meglio quali siano le esatte proporzioni del modello, e i punti (in rosso e indicati con lettere maiuscole) nei quali i vincoli saranno applicati. In questo caso per selezionare una delle opzioni si possono utilizzare i tasti "up/down" della tastiera, e confermare premendo il tasto "invio":

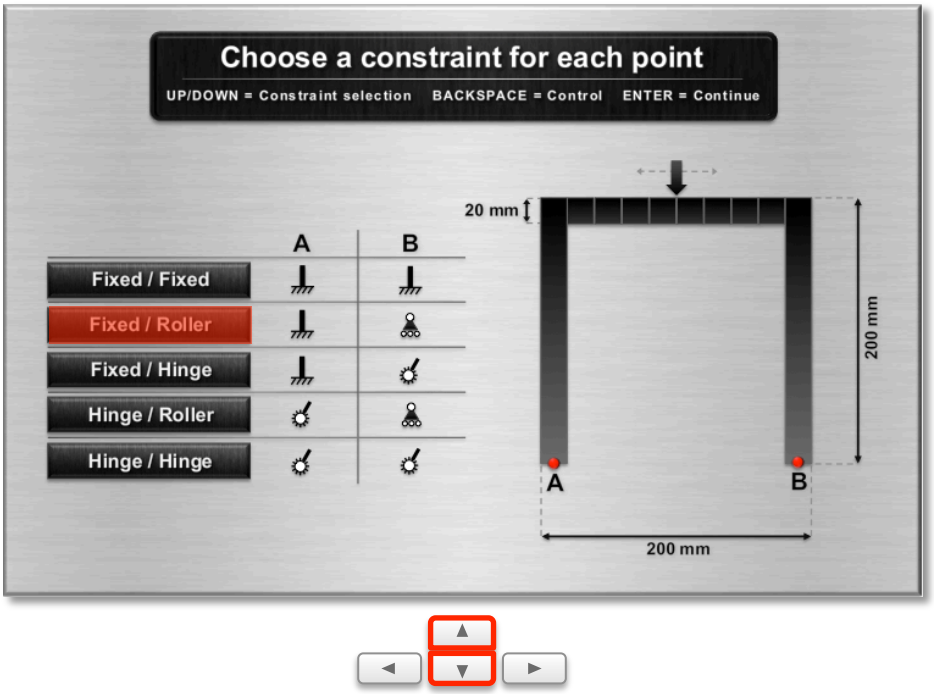


Figura 7.10: Schermata di selezione dello schema di vincolo

7.8 Livello 6: scelta di spessore/lunghezza della struttura

Uno degli ultimi passaggi permette all'utente di scegliere, a seconda del modello scelto, lo spessore o la lunghezza della struttura. Nei modelli in cui sono presenti fori, sarà possibile anche modificarne il diametro.

A livello algoritmico le modifiche continuano a coinvolgere il solo file "init.m" del modello scelto, che è amministrato a sua volta dai file "menu.m" e "thick.m" (o "length.m" se viene modificata la lunghezza della struttura, cosa che accade ad esempio nelle strutture 3D):

```
thick=num2str(t);
equ.thickness = t;
```

Nelle immagini seguenti si possono vedere le diverse schermate che, a seconda del caso, permettono l'impostazione dello spessore, della lunghezza o del diametro. L'utente può modificare i valori numerici usando i tasti "left/right" della tastiera:

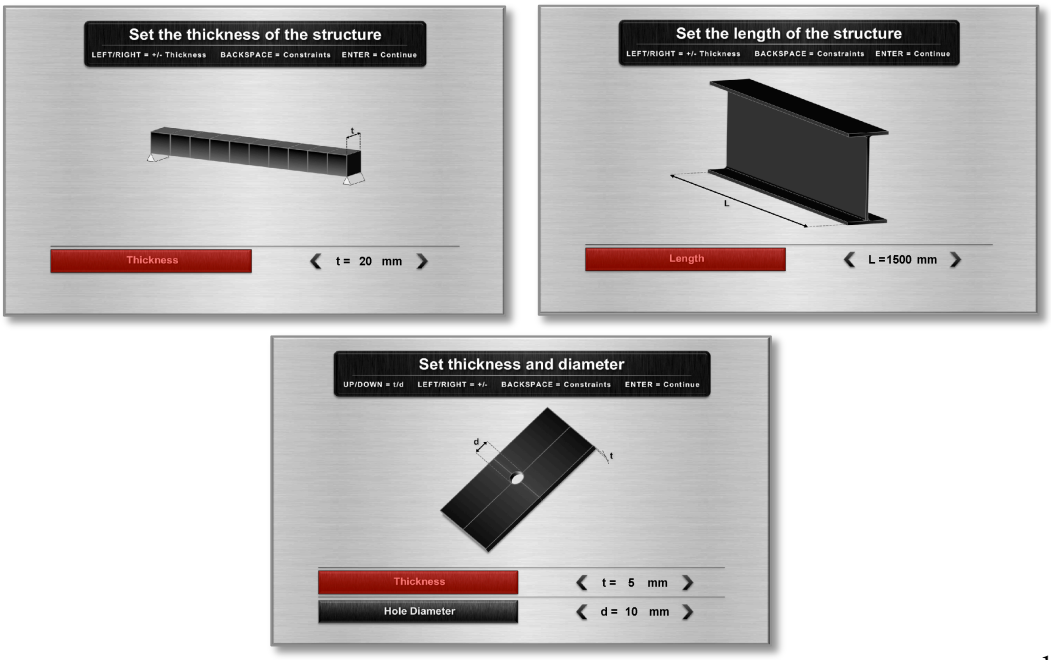


Figura 7.11: Schermate di impostazione di spessore/lunghezza/diametro

7.9 Livello 7: scelta del nodo

Prima di passare alla fase di simulazione vera e propria, all'utente viene richiesto di scegliere su quale nodo della struttura applicare la forza, o imporre uno spostamento. Anche questa scelta comporta una modifica del file "init.m" del modello scelto, e non va quindi ad appesantire la fase di simulazione, ma solo a compilare in modo opportuno il file d'inizializzazione:

```
node = [2,1,1,1,1,1,1,1,1,1,1,4,1,1,1,1,1,1,1,1,3,1];    % node 12
num=12;     %node number
```

Il processo attraverso il quale viene "spostato" il punto di applicazione da un nodo all'altro è identico a quello già illustrato nel Paragrafo 5.5. Nel caso delle righe di codice qui riportate, ad esempio, il numero "4", che indica il punto di applicazione della forza o dello spostamento imposto, è il dodicesimo elemento del vettore chiamato "node". Pertanto sarà il nodo numero 12 a subire l'applicazione del carico. Per quanto riguarda la parte grafica, la struttura, ora completamente definita, viene disegnata per la prima volta dal software. Per il momento essa appare di colore grigio, e sui nodi principali sono presenti dei pallini verdi e una freccia mobile. L'utente può spostare la freccia, e dunque cambiare nodo, con i tasti "left/right" della tastiera, e confermare con "invio". Il numero identificativo del nodo scelto è visualizzato in una casella in alto, mentre tutte le altre indicazioni utili sono raccolte in una barra inferiore. Nelle immagini che seguono si può notare come la grafica differisca in modo evidente dalle classiche GUI di Matlab e da Comsol (confrontare le Figure 6.12 e 6.13).

Nell'HaptikFem l'immagine occupa tutta l'area disponibile, i comandi sono pochi ed essenziali, e la rappresentazione della struttura, dell'ambiante e dei vincoli è in grado di far comprendere anche all'utente meno esperto come sarà sollecitata la struttura. Già in questa fase l'utente possiede tutti gli elementi necessari per immaginare quale sarà la deformata, come si sposteranno i punti di appoggio e quale sarà probabilmente il punto più sollecitato. In Comsol normalmente solo un utente con un certo grado d'esperienza riesce a fare valutazioni simili, poiché la rappresentazione grafica è molto tecnica, i vincoli e il carico sono indicati solo con dei piccoli punti colorati, e non è presente alcun elemento che indichi come possa essere utilizzata nel mondo reale la struttura con la quale si ha a che fare:

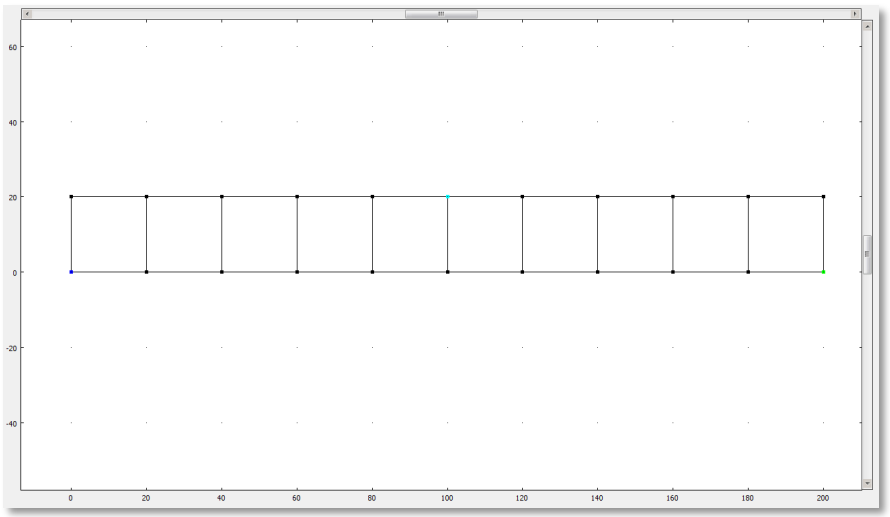


Figura 7.12: Tipica schermata d'inizializzazione di un modello in Comsol

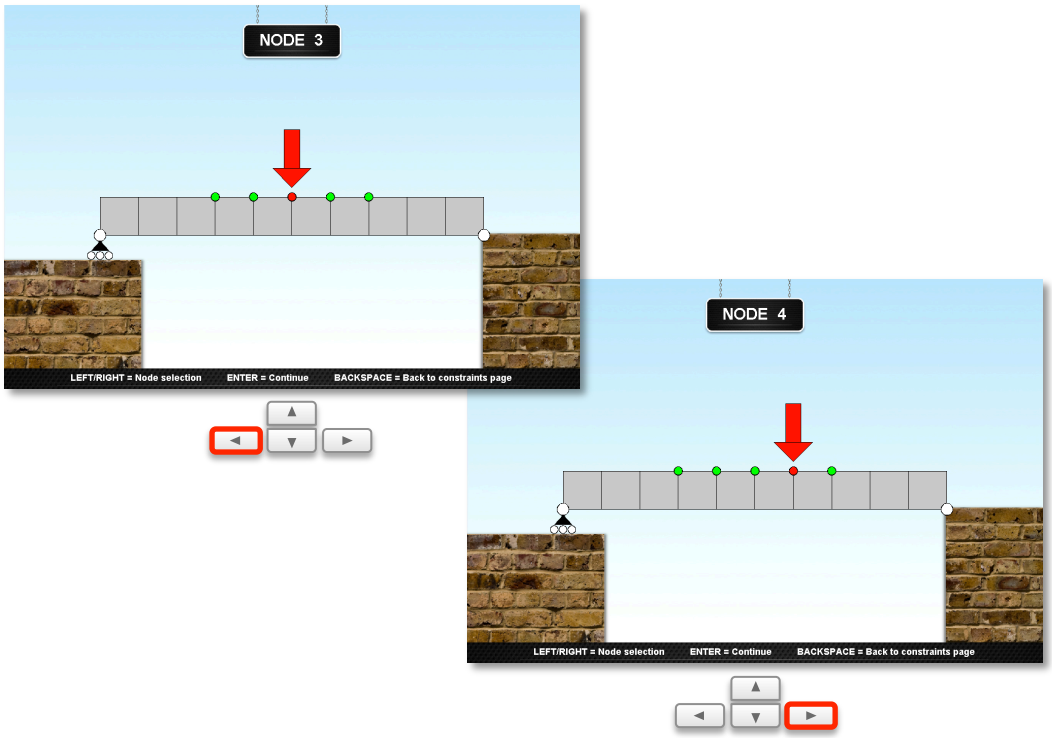


Figura 7.13: Schermata di inizializzazione nell'HaptikFem

7.10 Livello 8: simulazione

La fase conclusiva è ovviamente quella della simulazione *real-time*. Ora che il file "init.m" è stato completamente compilato grazie ai passaggi intermedi, l'esecuzione passa al file "mod.m" relativo al modello scelto, che si occupa della sostituzione diretta della mesh attraverso i dati provenienti dal mouse o dal dispositivo *haptic*. Nella schermata di simulazione con l'utilizzo del mouse, l'utente ha a disposizione un cursore (sul lato destro della schermata) che può essere spostato in alto o in basso attraverso un "*click and drag*". In base al movimento, sarà applicato uno spostamento o una forza a esso proporzionale, e la struttura si deformerà di conseguenza.

Nella parte alta della schermata è presente una barra nella quale sono riassunti tutti i dati utili, alcuni dei quali, come lo sforzo massimo e le coordinate del nodo sollecitato, sono aggiornati in tempo reale:

Figura 7.14: Barra superiore con tutte le informazioni utili aggiornate in real time

Nella barra, in ordine da sinistra verso destra, sono presenti le seguenti sezioni:

- DEVICE: in questa sezione viene indicato il dispositivo scelto per effettuare la simulazione (mouse o haptic).
- MATERIAL: riassunto delle caratteristiche del materiale scelto, in particolare dei limiti di snervamento e rottura.
- MAX VON MISES STRESS: viene indicato il valore massimo dello sforzo di von Mises registrato all'interno della struttura. Il valore viene aggiornato in tempo reale, e diventa rosso quando supera il limite di rottura. In basso è indicato lo spessore (o la lunghezza) scelto dall'utente per la struttura.
- NORMAL/SCIENTIFIC VIEW: durante la simulazione l'utente può passare dalla visualizzazione "classica" (con sfondo e rappresentazione realistica dei punti di appoggio) a "scientifica" (con griglia attivata e sfondo bianco), semplicemente premendo i tasti "up/down" della tastiera. La modalità scelta si illumina di verde, come si nota in Figura 7.14.
- NODE DISPLACEMENT: in questa sezione vengono riassunte le coordinate del nodo scelto, aggiornate in tempo reale. In questo caso è indicata anche la coordinata "xr", che indica lo spostamento di un carrello (se presente nello schema di vincolo scelto).

Nelle immagini che seguono si può apprezzare meglio l'interfaccia della fase di simulazione:

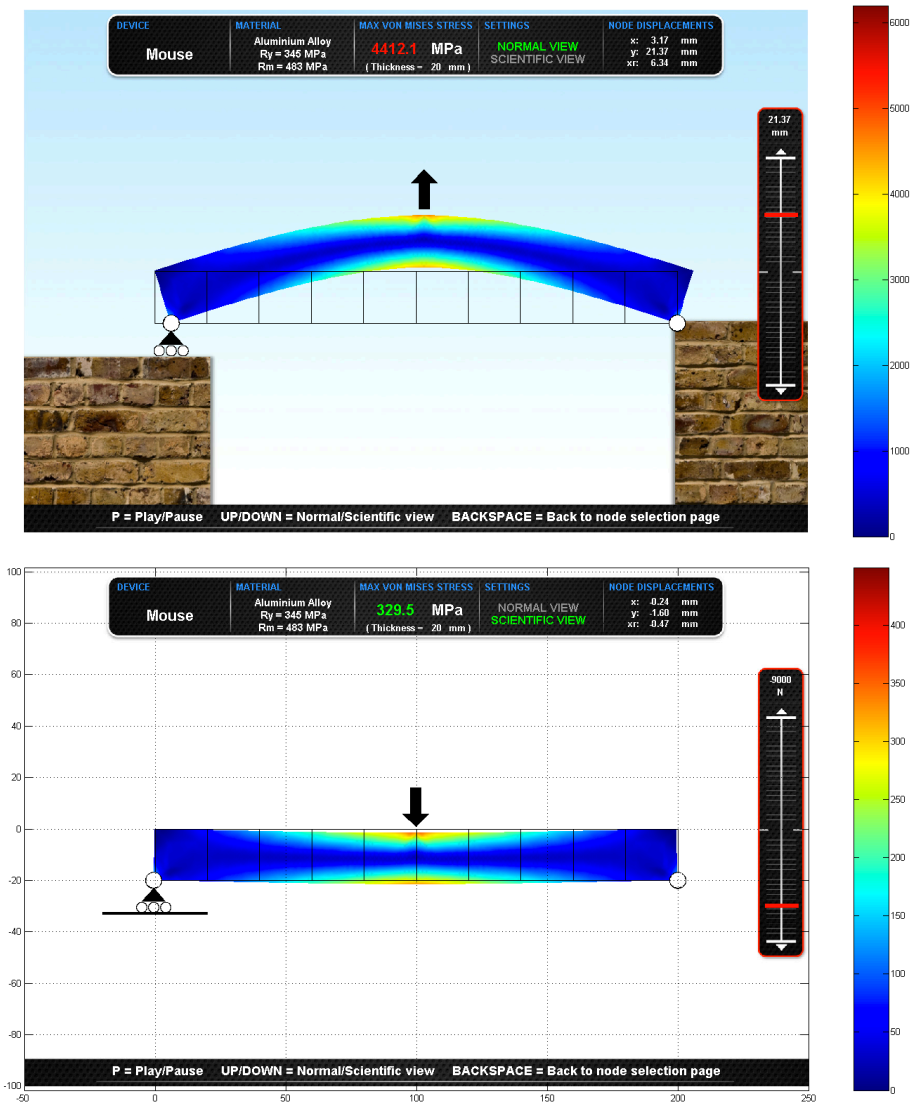


Figura 7.15: Schermata di simulazione "classica" in controllo di spostamento (in alto) e schermata "scientifica" in controllo di forza (in basso). A destra si noti il cursore.

Capitolo 8: Modelli

Dopo aver descritto le tecniche utilizzate per realizzare l'applicazione e la sua interfaccia, si passa ora a fornire le caratteristiche di ciascuno dei modelli implementati nell'applicazione "HaptikFem". I modelli sono in totale undici, dei quali sette visualizzati in 2D e quattro in 3D. Le strutture sono state modellizzate tenendo conto di quelli che sono gli esempi più semplici e classici utilizzati nei primi corsi di meccanica delle strutture nei corsi d'ingegneria. Si tratta di strutture piuttosto semplici, create con l'intento di far apprendere agli studenti i nuovi concetti nel modo più veloce possibile.
All'interno dell'applicazione, le simulazioni sono tutte in campo elastico, mai plastico. Ciò è dovuto alle difficoltà che una simulazione con deformazioni permanenti in *real-time* comporterebbe in termini di tempo di calcolo. In questo caso, infatti, la mesh deformata dovrebbe essere salvata a ogni *step* e rielaborata al successivo, con un onere computazionale molto elevato. Alcuni modelli sono visualizzati in 2D, altri in 3D: in realtà Comsol tratta tutti i modelli come se fossero in 3D (infatti, per tutti i modelli è possibile scegliere lo spessore, anche per quelli visualizzati in 2D), ma lo stile di visualizzazione bidimensionale consente di risparmiare tempo nella risoluzione delle strutture, le cui mesh sono solitamente composte da molti meno elementi di quelle 3D. Le strutture 2D introdotte affrontano i problemi sulla flessione, trazione, strizione e il calcolo in tempo reale del coefficiente d'intensificazione degli sforzi in prossimità di fori. I quattro modelli con visualizzazione 3D sono stati inseriti in realtà più per una ragione di completezza e con l'intento di far comprendere quali siano le possibili potenzialità dell'applicazione, poiché con un calcolatore non professionale i risultati non sono ottimali. Queste strutture affrontano tutte il problema della flessione per travi a incastro con forme delle sezioni differenti. Nei paragrafi seguenti verrà fornita una breve descrizione per ciascun modello implementato.

8.1 Modello 1: trave a incastro (2D)

Come già accennato, il primo modello è anche il più semplice. Ciò che si è cercato di fare, infatti, è di inserire modelli con complessità via via crescente. La struttura del modello realizzato con Comsol è costituita da dieci geometrie quadrate identiche, di lato pari a 20 mm, ed accostate tra loro a formare una barretta.

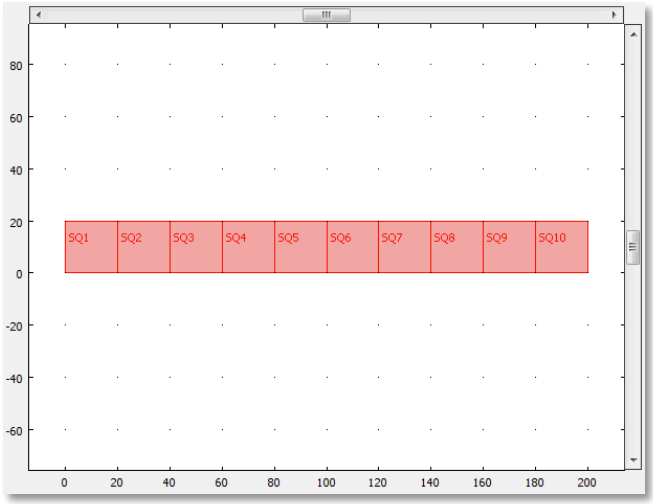


Figura 8.1: Modello 1 realizzato in Comsol

La mesh di questo semplice modello è composta da soli 160 elementi bidimensionali. L'estremità sinistra è incastrata per l'intera lunghezza del lato, e l'utente può applicare un carico o imporre uno spostamento sul lato superiore, in corrispondenza delle linee divisorie.

L'aspetto finale della struttura all'interno di "HaptikFem" è quello di figura 7.2:

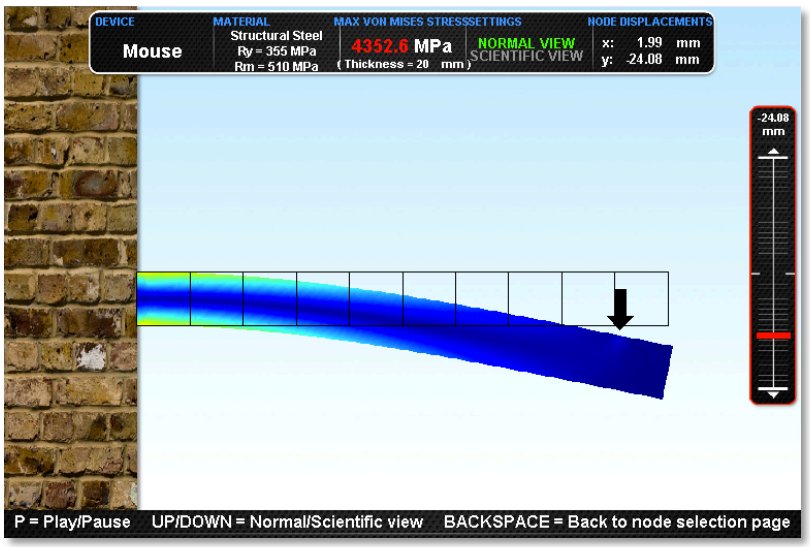


Figura 8.2: Modello 1 in funzione con uno spostamento imposto

Da Figura 7.2 si possono notare anche la deformazione della barretta, coerente con quanto ci si sarebbe aspettato da una trave a incastro, e le zone più sollecitate all'estradosso e all'intradosso in corrispondenza del vincolo. Di seguito è stata inserita breve tabella che riassume le caratteristiche principali del modello e gli obiettivi che esso si prefigge di raggiungere:

Modello	Trave a incastro
Stile di visualizzazione	2D
Numero di elementi della mesh	160 (*Normal*)
Schema di vincolo	Incastro nell'estremità
Obiettivi	Comprensione degli sforzi e della deformata di una trave a incastro sottoposta a flessione.

8.2 Modello 2: trave a doppio appoggio (2D)

Il secondo modello deriva direttamente dal primo. La struttura è composta di dieci geometrie quadrate accostate con lato di 20 mm; la mesh è ancora composta da 160 elementi. In questo caso la barretta è però vincolata a terra alle estremità. L'utente potrà decidere la natura dei punti d'appoggio.

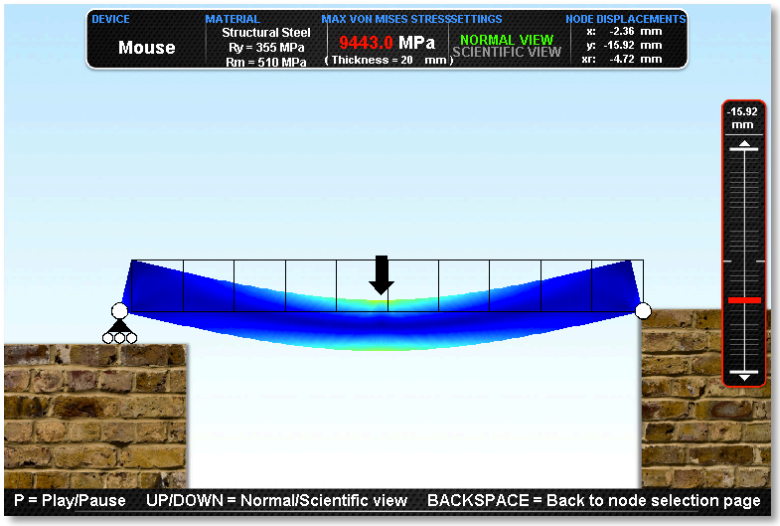


Figura 8.3: Modello 2 in funzione con uno spostamento imposto

Di seguito si riporta la tabella riassuntiva:

Modello	Trave a doppio appoggio
Stile di visualizzazione	2D
Numero di elementi della mesh	160 (*Normal*)
Schema di vincolo	carrello/cerniera cerniera/cerniera
Obiettivi	Comprensione degli sforzi e della deformata di una trave a doppio appoggio sottoposta a flessione. Comprensione delle differenze al variare dello schema di vincolo, analisi degli spostamenti della struttura e del carrello.

8.3 Modello 3: portale (2D)

La complessità aumenta nel terzo modello, che rappresenta un portale. E' una struttura formata da due colonne rettangolari di dimensioni 20x200 mm e da una traversa composta da otto geometrie quadrate di lato pari a 20 mm. La mesh è consiste di 282 elementi.

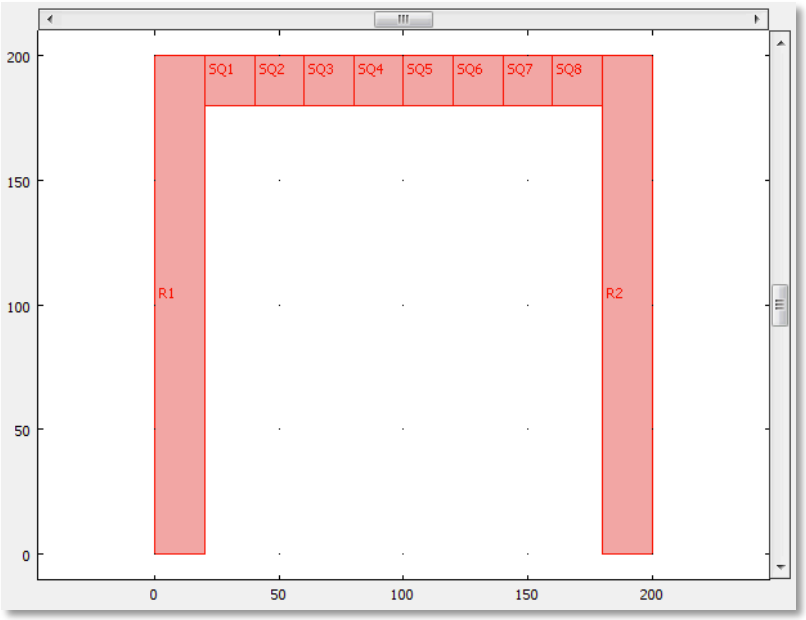


Figura 8.4: Modello 3 modellizzato in Comsol

La struttura è vincolata al terreno per mezzo delle estremità inferiori delle due colonne. Gli schemi di vincolo tra i quali l'utente può scegliere sono ben cinque. Uno degli obiettivi principali del modello, infatti, è quello di imparare a osservare e comprendere gli sforzi e le deformazioni al variare dello schema di vincolo adottato.

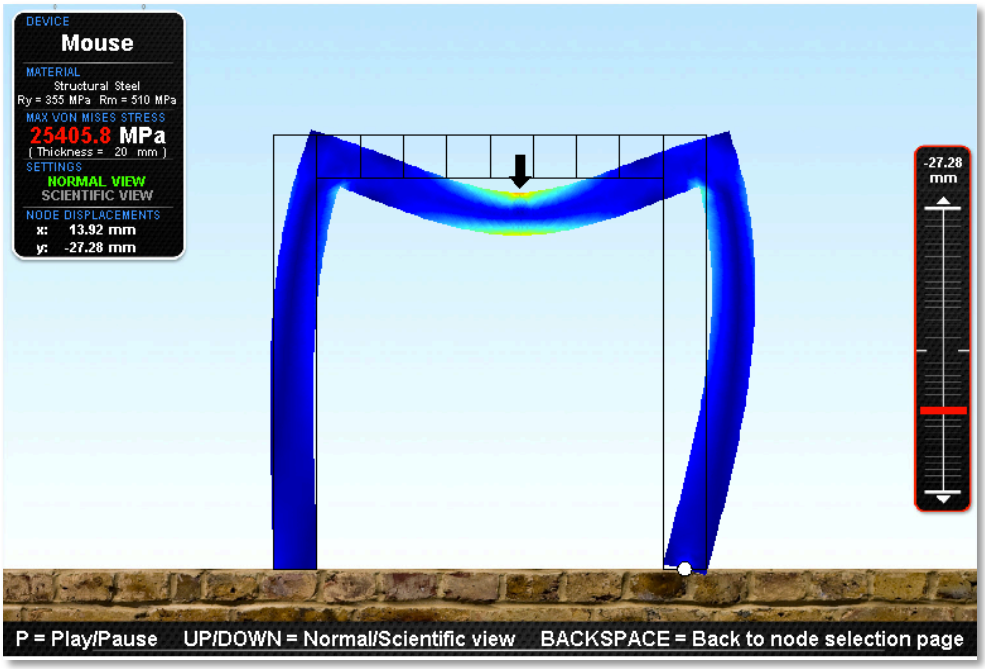


Figura 8.5: Modello 3 in funzione con uno schema di vincolo incastro/cerniera

Nella tabella sottostante sono riportati tutti gli schemi di vincolo e le altre caratteristiche salienti del terzo modello:

Modello	Portale
Stile di visualizzazione	2D
Numero di elementi della mesh	282 (*Normal*)
Schema di vincolo	incastro/incastro carrello/incastro incastro/cerniera carrello/cerniera cerniera/cerniera
Obiettivi	Comprensione degli sforzi e della deformata di una struttura a portale sottoposta a flessione. Comprensione delle differenze al variare dello schema di vincolo, analisi degli spostamenti della struttura e del carrello.

8.4 Modello 4: prova di trazione su provino (2D)

Il quarto modello consiste di una geometria piuttosto semplice, che riproduce fedelmente un tipico provino piatto utilizzato nelle prove di trazione di dimensioni normate. Il provino è formato da uno stelo centrale di 60 mm di lunghezza e largo 10 mm, cui sono raccordate due zone di afferraggio, larghe 20 mm e alte 40 mm. La lunghezza complessiva del provino è di 165 mm.

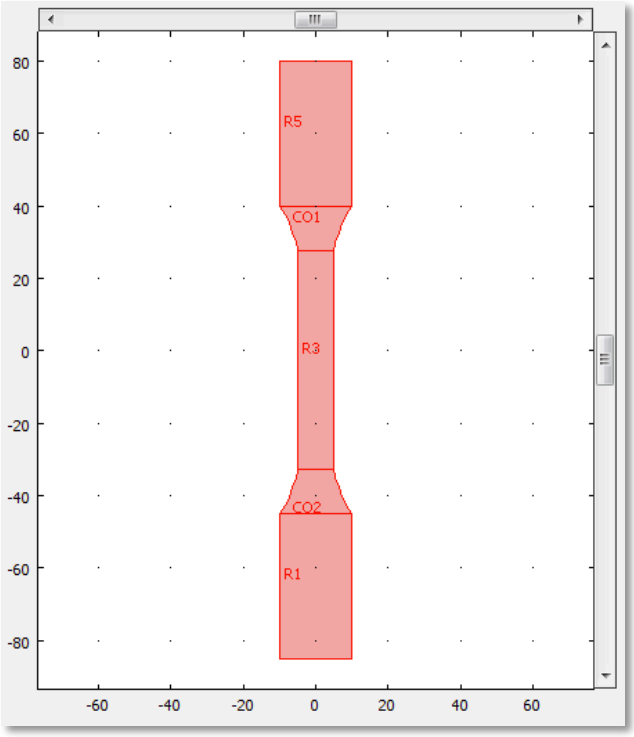


Figura 8.6: Geometria del provino realizzata in Comsol

La particolarità di questo modello è costituita dal fatto che lo stelo centrale è sottoposto a strizione quando il provino viene sottoposto a trazione. Valutando la larghezza della porzione centrale dello stelo, l'utente può visualizzare a schermo l'entità della deformazione in tempo reale.

L'obiettivo principale è quindi quello di comprendere quale sia l'entità della strizione provocata da una forza applicata o da uno spostamento imposto sull'afferraggio inferiore. Un altro aspetto interessante è costituito dalla presenza degli ampi raccordi, che permettono una distribuzione uniforme degli sforzi, senza punti di accumulo di sollecitazioni.

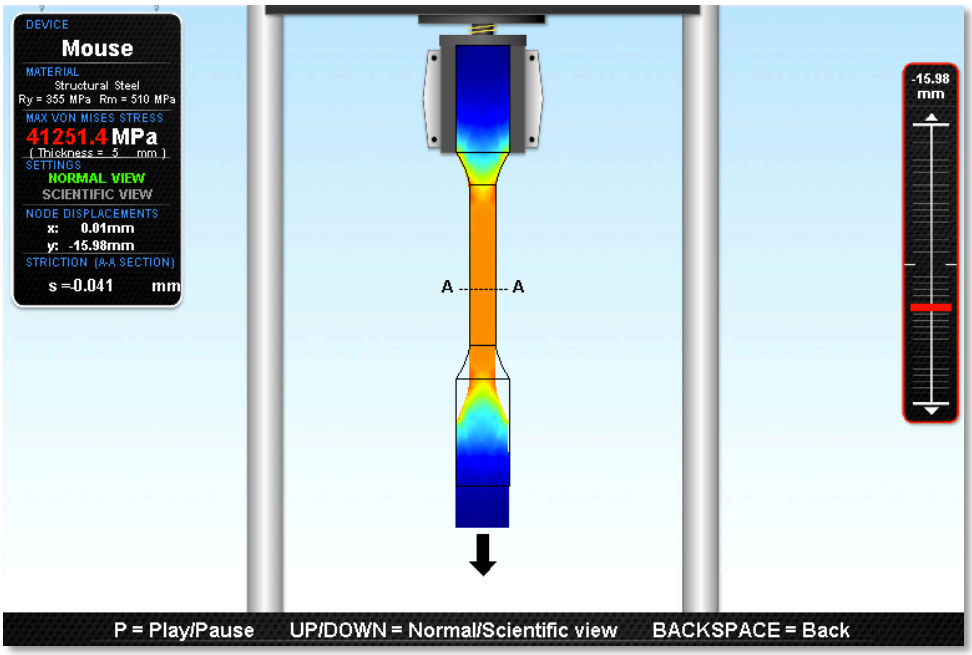


Figura 8.7: Simulazione di una prova di trazione. Si noti nella tabella il calcolo in tempo reale della strizione generata dallo spostamento imposto

Modello	Provino piatto per prova di trazione (normato)
Stile di visualizzazione	2D
Numero di elementi della mesh	130 (*Normal*)
Schema di vincolo	Afferraggio superiore
Obiettivi	Comprensione degli sforzi e della deformata di un provino piatto in una prova di trazione. Comprensione e valutazione dell'entità della strizione in relazione al carico o allo spostamento imposto all'afferraggio inferiore.

8.5 Modello 5: prova di trazione su lastra forata (2D)

Il quinto modello affronta un'altra delle problematiche principali nello studio di meccanica delle strutture: il calcolo del coefficiente di concentrazione degli sforzi. E' utilizzato in presenza di particolari geometrie, come fori, scanalature o cricche, in corrispondenza delle quali si ha un accumulo degli sforzi. Infatti, la presenza di una discontinuità provoca un incremento locale dello sforzo che, anche se nominalmente rimane inferiore del limite di snervamento del materiale, in quei punti può tuttavia raggiungerlo e superarlo, provocando una deformazione plastica locale o la propagazione di una cricca.

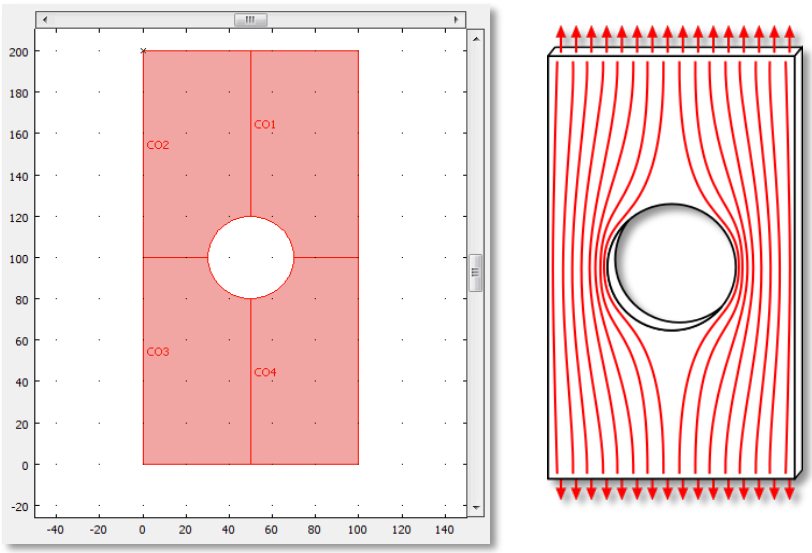


Figura 8.8: Modello 5 realizzato in Comsol (a sinistra) e schematizzazione esemplificativa della concentrazione degli sforzi in corrispondenza del foro (a destra).

Come si nota da Figura 8.8, il modello schematizza la classica piastra rettangolare forata utilizzata come esempio in tutti i libri e i corsi di meccanica delle strutture. L'utente è in grado di regolare lo spessore della lamina e il diametro del foro; le dimensioni della lamina sono di 200x100 mm.

Quando l'utente varia il diametro e avvia la simulazione, può vedere come il coefficiente di concentrazione degli sforzi varia di conseguenza e aumenta progressivamente all'aumentare del diametro. Il coefficiente di concentrazione (indicato con K) viene calcolato in tempo reale come rapporto tra lo sforzo massimo nell'intorno del foro e lo sforzo minimo all'interno della lamina:

$$K = \sigma_{max} / \sigma_{min}$$

Una volta avviata la simulazione, il risultato sarà simile a quello della figura seguente:

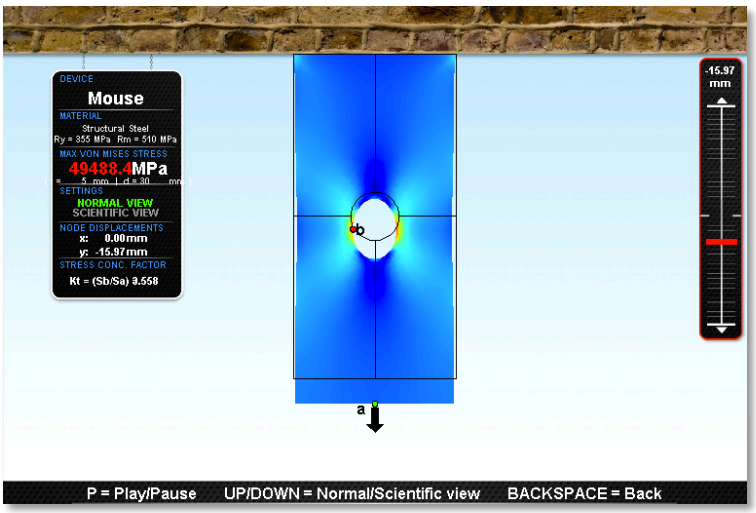


Figura 8.9: Simulazione della trazione del modello 5. Si noti il calcolo in tempo reale del fattore di concentrazione nella tabella a sinistra

Modello	Piastra forata
Stile di visualizzazione	2D
Numero di elementi della mesh	636 (*Normal*)
Schema di vincolo	Incastro superiore
Obiettivi	Comprensione degli sforzi e della deformata di una lastra piatta in una prova di trazione. Comprensione e valutazione dell'entità del fattore di concentrazione degli sforzi al variare del diametro del foro. Comprensione della degli sforzi in prossimità del foro.

8.6 Modello 6: trazione su lastra con semi-fori (2D)

Il modello 6 è molto simile a quello visto nel paragrafo precedente, con l'unica differenza che in questo caso la lamina presenta due scanalature semicircolari sui bordi. Anche in questo caso l'obiettivo principale è far comprendere all'utente il meccanismo di concentrazione degli sforzi in prossimità dei mezzi fori. La lamina presenta le stesse dimensioni (200x100 mm) di quella vista al Paragrafo 8.5.

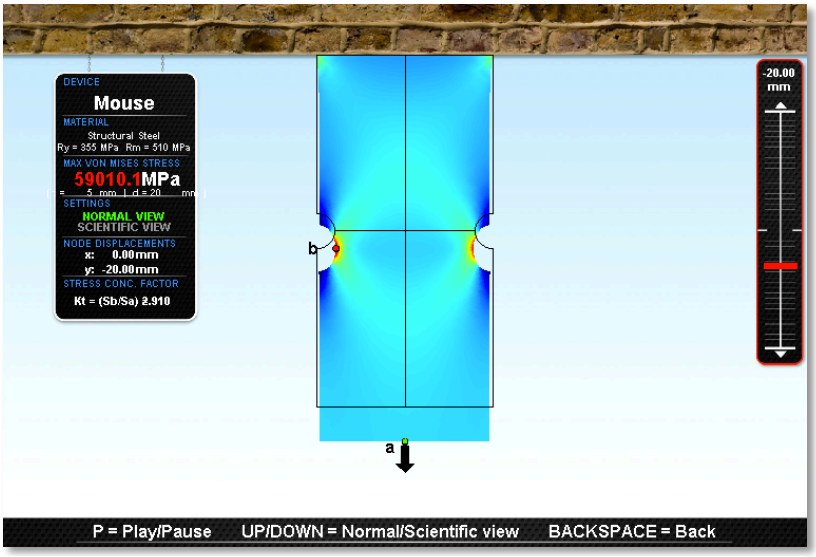


Figura 8.10: Simulazione della trazione del modello 6. Si noti il calcolo in tempo reale del fattore di concentrazione nella tabella a sinistra

Modello	Piastra con due semi-fori
Stile di visualizzazione	2D
Numero di elementi della mesh	834 (*Normal*)
Schema di vincolo	Incastro superiore
Obiettivi	Comprensione degli sforzi e della deformata di una lastra piatta in una prova di trazione. Comprensione e valutazione dell'entità del fattore di concentrazione degli sforzi al variare del diametro dei fori. Comprensione della degli sforzi in prossimità dei fori.

8.7 Modello 7: struttura reticolare (2D)

L'ultimo dei modelli con visualizzazione 2D è anche il più complesso, ed è stato pensato sia per facilitare la comprensione della deformata e della distribuzione degli sforzi in una struttura reticolare, sia per dimostrare le potenzialità dell'applicazione, che in questo caso deve amministrare una mesh di oltre 1700 elementi.

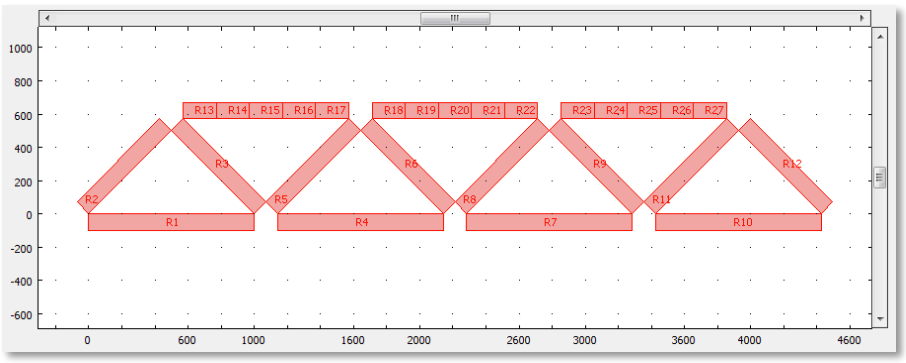


Figura 8.11: Modellazione della struttura reticolare in Comsol

Come si nota da Figura 8.11, la struttura realizzata tenta di riprodurre un ponte a quattro campate in scala ridotta. La lunghezza complessiva della struttura è pari a 4,424 m e l'altezza è di 0,978 m. La lunghezza di ciascuna campata è di 1 m. Le varie parti della struttura hanno in comune solo un punto di connessione, che approssima il comportamento di una cerniera.

Se si interponessero delle vere e proprie cerniere nei punti di connessione della struttura infatti, essa risulterebbe più volte iperstatica e non si deformerebbe nel modo atteso, visibile nella figura seguente:

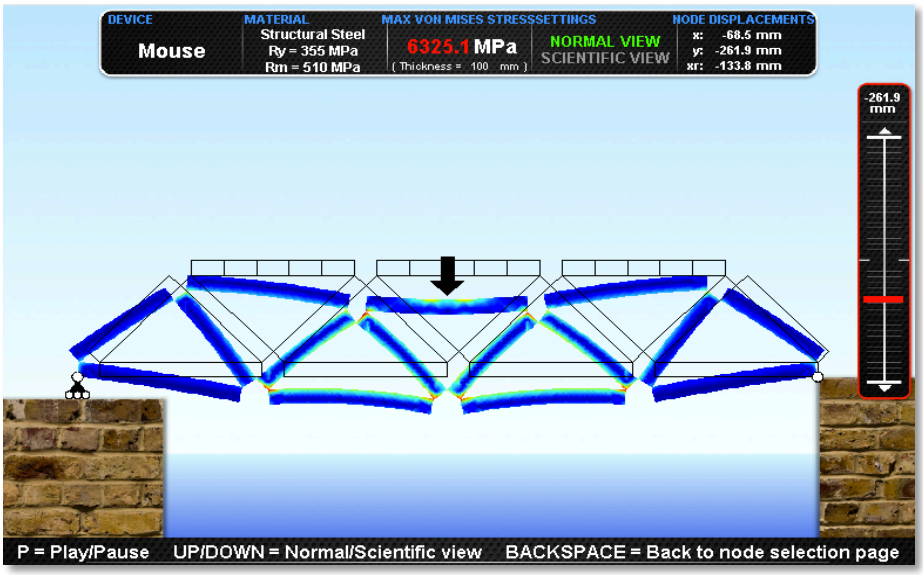


Figura 8.11: Simulazione real time della deformazione della struttura

Modello	Struttura reticolare
Stile di visualizzazione	2D
Numero di elementi della mesh	1760 (*Normal*)
Schema di vincolo	carrello/cerniera cerniera/cerniera
Obiettivi	Comprensione degli sforzi e della deformata in una struttura reticolare. Comprensione e valutazione dell'entità dello spostamento del carrello.

8.8 Modello 8: trave a incastro IPE 400 (3D)

Si passa quindi ai modelli caratterizzati da visualizzazione 3D. Come già accennato in precedenza, questi modelli sono stati introdotti più per dimostrare le potenzialità dell'applicazione che per un reale scopo didattico, anche se in ogni caso possono contribuire a una migliore comprensione della distribuzione degli sforzi all'interno di travi a incastro di sezioni differenti. La prima di queste travi è una classica IPE 400, modellizzata seguendo le dimensioni dettate dalla norma UNI 5398-78. La trave è alta complessivamente 400 mm, mentre la lunghezza è definibile direttamente dall'utente.

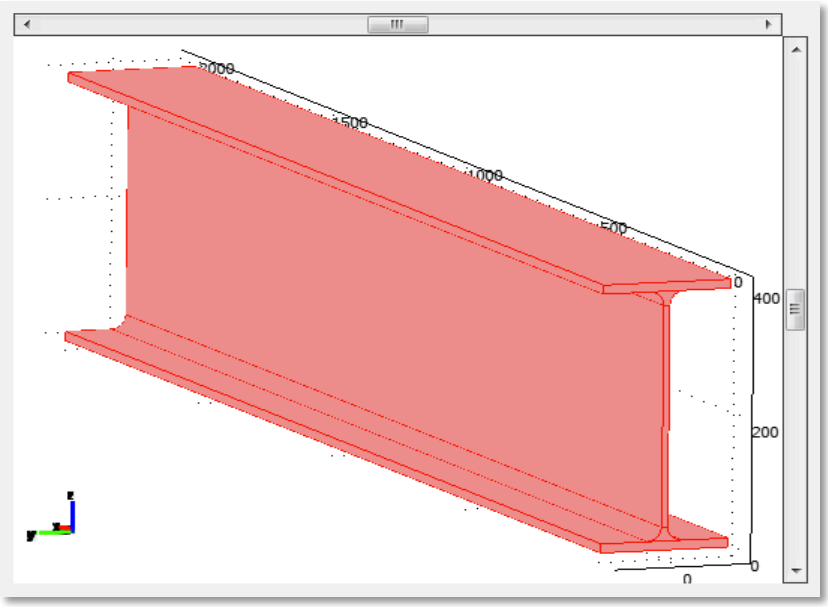


Figura 8.12: Modellazione della trave IPE 400 in Comsol

Nella fase di simulazione l'utente può osservare la diversa entità degli sforzi e la loro distribuzione nelle tre dimensioni al variare della lunghezza della trave. Essendo le strutture con visualizzazione 3D formate da un numero considerevole di elementi della mesh, e dovendo il software gestire non due, bensì tre dimensioni di visualizzazione, la simulazione risulta leggermente più lenta con calcolatori di prestazioni intermedie.

Per questo motivo si è cercato di alleggerire al massimo i modelli utilizzando mesh molto rade ("*Coarser*" in Comsol) e pochi elementi grafici:

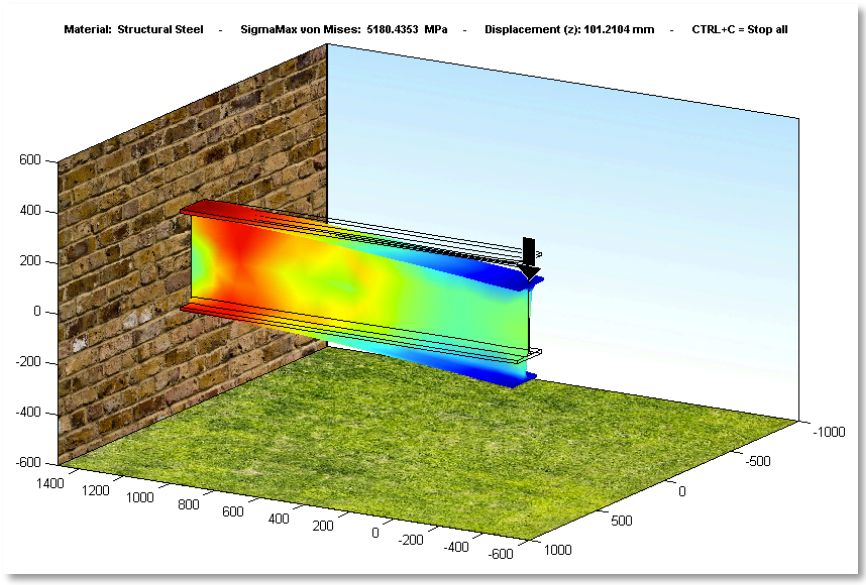


Figura 8.13: Flessione della trave IPE 400 nell'applicazione HaptikFem

Modello	Trave IPE 400
Stile di visualizzazione	3D
Numero di elementi della mesh	1286 (*Coarser*)
Schema di vincolo	Incastro
Obiettivi	Comprensione degli sforzi e della deformata di una trave IPE a incastro al variare della sua lunghezza. Comprensione della distribuzione degli sforzi nelle tre dimensioni.

8.9 Modello 9: trave a incastro PN 180 (3D)

I restanti modelli con visualizzazione 3D richiamano gli stessi concetti visti in precedenza, variando tuttavia la geometria della sezione della trave. La trave PN 180 è una trave di forma a "C" che, seguendo la norma UNI 5680, risulta alta 180 mm. L'ala superiore e quella inferiore sono collegate allo stelo per mezzo di un raggio di raccordo pari a 11 mm.

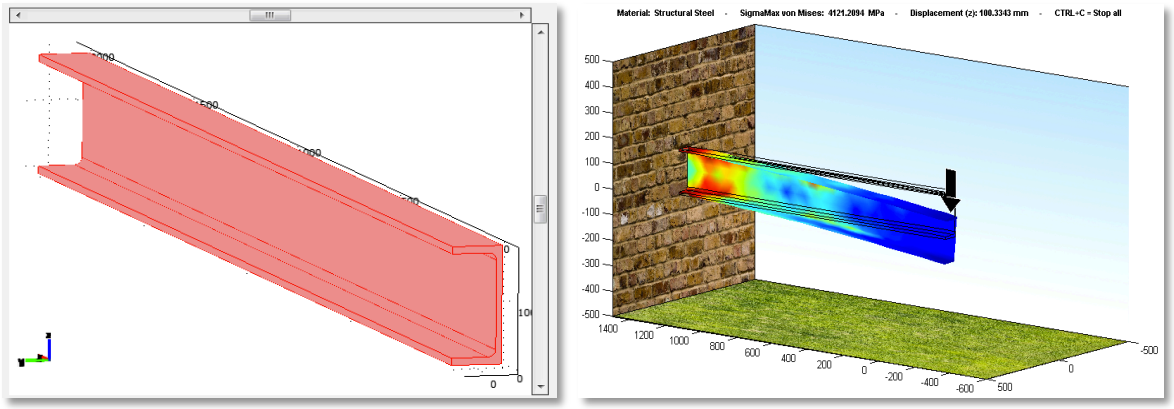


Figura 8.14: Modellazione della trave PN 180 in Comsol (a sinistra) e fase di simulazione (a destra)

Modello	Trave PN 180
Stile di visualizzazione	3D
Numero di elementi della mesh	3373 (*Extremely Coarse*)
Schema di vincolo	Incastro
Obiettivi	Comprensione degli sforzi e della deformata di una trave PN a incastro al variare della sua lunghezza. Comprensione della distribuzione degli sforzi nelle tre dimensioni.

8.10 Modello 10: trave a incastro T 100 (3D)

Il penultimo modello è una semplice trave a "T" di lunghezza variabile dall'utente. Secondo la norma UNI 5681, sia la base sia lo stelo sono lunghi 100 mm, mentre lo spessore delle lamine è pari a 11 mm. Come le precedenti strutture con visualizzazione 3D anche questa trave è incastrata a una delle estremità. Il modello risulta particolarmente semplice, costituito da un numero di elementi della mesh inferiore rispetto agli altri. Di conseguenza la fase di simulazione è leggermente più fluida, ed è possibile utilizzare una mesh meno rada.

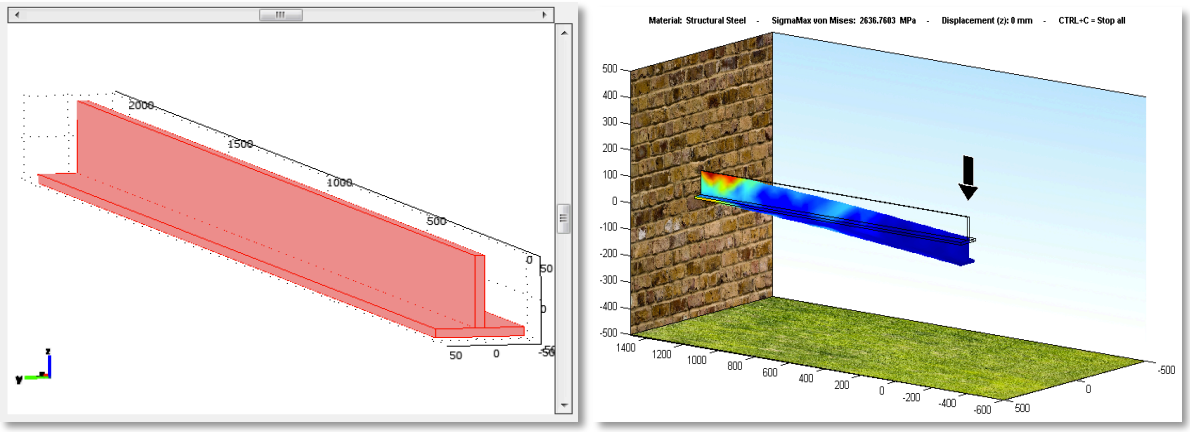


Figura 8.15: Modellazione della trave T 100 in Comsol (a sinistra) e fase di simulazione (a destra)

Modello	Trave T 100
Stile di visualizzazione	3D
Numero di elementi della mesh	501 (*Coarser*)
Schema di vincolo	Incastro
Obiettivi	Comprensione degli sforzi e della deformata di una trave T a incastro al variare della sua lunghezza. Comprensione della distribuzione degli sforzi nelle tre dimensioni.

8.11 Modello 11: tubo a sezione quadrata EN 10210 (3D)

L'ultimo modello è costituito da una struttura tubolare a sezione quadrata, incastrata a una delle estremità e sottoposta a flessione per mezzo di un carico o di uno spostamento imposto. Seguendo la norma EN 10210, ciascun lato è lungo 160 mm, e gli angoli sono raccordati con un raggio di raccordo di 3 mm. Lo spessore del tubo è di 3 mm. Essendo la struttura cava, l'utente è in grado di vedere non solo la distribuzione degli sforzi sulle superfici esterne, ma anche su quelle interne, ruotando la figura durante la fase di simulazione.

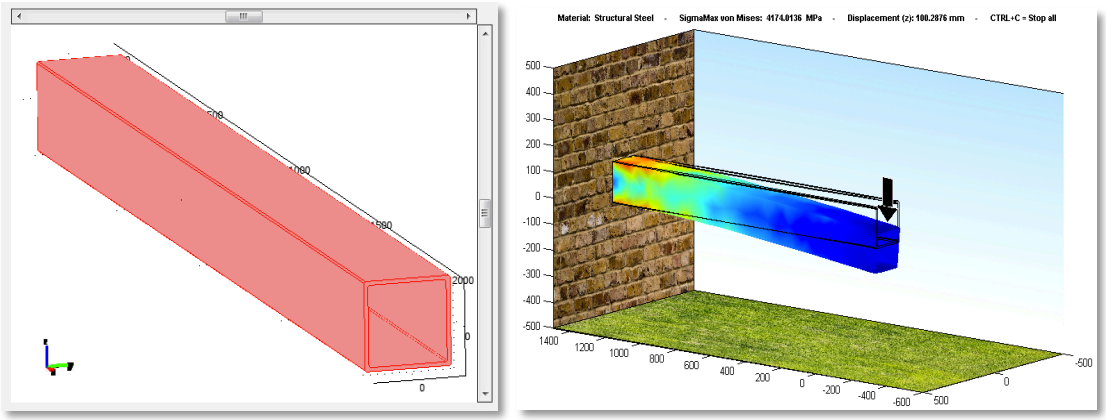


Figura 8.16: Modellazione della struttura tubolare in Comsol (a sinistra) e fase di simulazione (a destra)

Modello	Tubo a sezione quadrata EN 10210
Stile di visualizzazione	3D
Numero di elementi della mesh	3352 (*Extremely Coarse*)
Schema di vincolo	Incastro
Obiettivi	Comprensione degli sforzi e della deformata di una struttura tubolare a incastro al variare della sua lunghezza. Comprensione della distribuzione degli sforzi nelle tre dimensioni.

8.12 Comparazione dei modelli

Di seguito viene riportata una tabella comparativa tra tutti i modelli introdotti nell'applicazione "HaptikFem". Come si può osservare, al crescere della complessità della struttura, cresce anche il numero di elementi che costituiscono la mesh, ma decresce la complessità del tipo di elementi utilizzati e del solutore, che passa da analisi di tipo quadratico a lineare. Ciò è stato fatto per trovare il giusto compromesso tra precisione del risultato e fluidità della simulazione.

N° modello	Nome modello	Stile visualizz.	Nome modalità	Numero elementi	Tipo elementi
1	Trave a incastro	2D	ps	160	Lagrange quadratici
2	Trave a doppio appoggio	2D	ps	160	Lagrange quadratici
3	Portale	2D	ps	282	Lagrange quadratici
4	Provino piatto per prova di trazione	2D	ps	130	Lagrange quadratici
5	Piastra forata	2D	ps	636	Lagrange quadratici
6	Piastra con 2 semi-fori	2D	ps	834	Lagrange quadratici
7	Struttura reticolare	2D	ps	1760	Lagrange lineari
8	Trave IPE 400	3D	sld	1286	Lagrange lineari
9	Trave PN 180	3D	sld	3373	Lagrange lineari
10	Trave T 100	3D	sld	501	Lagrange lineari
11	Tubo a sezione quadrata EN 10210	3D	sld	3352	Lagrange lineari

Tabella 8.1: Tabella comparativa delle caratteristiche dei modelli presenti nell'applicazione "HaptikFem"

Capitolo 9: User test

L'ultima fase del lavoro prevede una serie di test con utenti, allo scopo di raccogliere dati, impressioni, consigli e di valutare l'usabilità dell'applicazione. Vedere "all'opera" i potenziali utilizzatori del prodotto da valutare, infatti, consente l'individuazione immediata di eventuali mancanze e anche dei punti di forza. I dati raccolti in fase di test servono a identificare le difficoltà incontrate dagli utenti, individuarne le cause e porvi rimedio, cosa che è stata fatta anche durante lo svolgimento della tesi sotto forma di prove e quesiti informali, allo scopo di apportare modifiche fondamentali. Nella fase di test descritta in questo capitolo, agli utenti è stata invece sottoposta la versione finale e già più volte corretta e aggiornata dell'applicazione: i risultati serviranno a un'ulteriore eventuale ottimizzazione, e a mettere in luce possibili problemi che ancora persistono.

9.1 Struttura del test

Il test è stato strutturato in forma di questionario a risposta multipla. Per facilitarne e velocizzarne lo svolgimento è stata utilizzata la sezione "Moduli" di "Google Docs" che consente di realizzare in modo semplice questionari o sondaggi online, e di raccogliere i dati in file compatibili con Word o Excel.

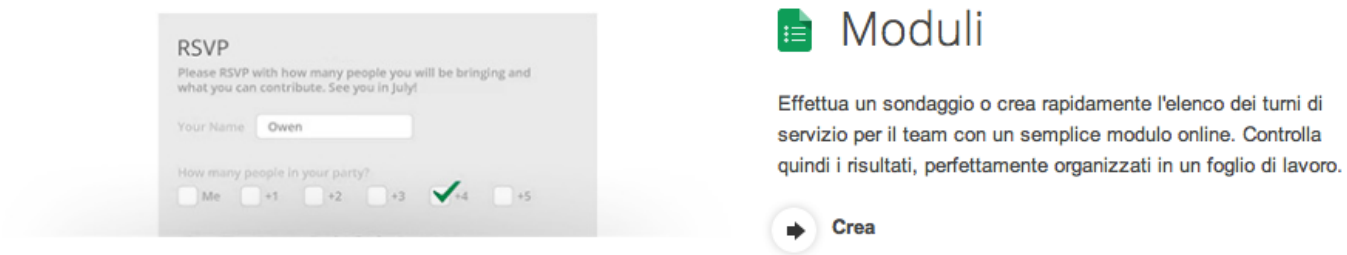


Figura 9.1: Sezione "Moduli" di Google Docs

L'obiettivo del test non era solo quello di vedere all'opera gli utenti con l'applicazione "HaptikFem", ma anche di fare un paragone diretto tra Comsol e HaptikFem. Per questo sono stati preparati due questionari, uno dedicato agli utenti che avrebbero utilizzato il metodo standard, con Comsol, l'altro per coloro che avrebbero utilizzato HaptikFem. Ogni punto del questionario, che è stato accuratamente corredato da specifiche istruzioni (in lingua inglese) in grado di guidare l'utente nell'analisi della struttura dopo una prima fase di "familiarizzazione", consiste di una domanda con cinque possibilità di risposta, delle quali solo una è corretta. Ciascun questionario è a sua volta suddiviso in due sezioni, una dedicata ai carichi e una ai vincoli. Molte domande, inoltre, fanno riferimento a disegni e schemi che sono stati forniti agli utenti in via cartacea. Il testo completo del questionario è consultabile nell'Appendice I; di seguito è riportato uno *screenshot* del questionario a puro scopo illustrativo:

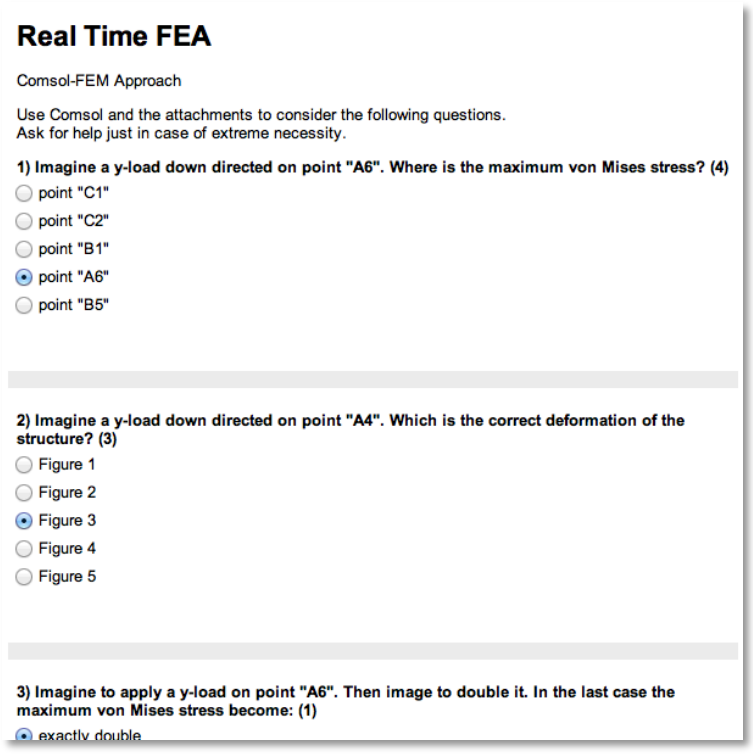


Figura 9.2: Screenshot di una porzione del questionario per gli utenti "Comsol"

9.2 Utenti

In totale sono stati selezionati 20 utenti, metà dei quali ha effettuato prima il test sull'applicazione HaptikFem, e metà su Comsol, per poi invertirsi. In totale quindi sono stati raccolti dati corrispondenti a 40 prove. L'alternanza è stata stabilita allo scopo di valutare meglio un eventuale tasso di apprendimento.

Il gruppo di utenti scelto era formato interamente da studenti universitari o giovani laureati, con le seguenti caratteristiche:

- Età media: 24,3 anni
- Deviazione standard dell'età: 2,105 anni
- Maschi: 15
- Femmine: 5
- Utenti provenienti da indirizzi di studio ingegneristici: 7
- Utenti provenienti da altri indirizzi di studio: 13

Si è tentato di sottoporre i test a un pubblico quanto più eterogeneo possibile sotto diversi punti di vista: età, sesso, indirizzo di studi, esperienza con i software utilizzati. Allo scopo di valutare la capacità di apprendimento, si è pensato inoltre di suddividere gli utenti in due sottogruppi, dei quali il primo (indicato in giallo e chiamato "Gruppo 1") ha svolto il test prima con l'HaptikFem e poi con Comsol. Il secondo gruppo (indicato in azzurro e chiamato "Gruppo 2") ha svolto invece i test al contrario: prima quello con Comsol e in seguito quello con l'HaptikFem. I punteggi conseguiti dal Gruppo 1 sono quindi indicativi per valutare l'efficacia dell'applicazione: se saranno mediamente superiori si potrà affermare che l'HaptikFem contribuisce a consolidare alcuni argomenti, che vengono poi applicati anche al successivo test con Comsol.

Si è cercato di uniformare il più possibile i due gruppi, distribuendo nel modo migliore possibile gli utenti che avevano già conoscenza dei software utilizzati. Nel dettaglio, i due gruppi avevano le seguenti caratteristiche:

	Gruppo 1	**Gruppo 2**
Età media	23,4	25,2
Deviazione standard età	2,27	1,55
N° studenti di ingegneria	3	4
Maschi/Femmine	7 Maschi - 3 Femmine	8 Maschi - 2 Femmine
Conoscenza software	4	3

Tabella 9.1: Caratteristiche dei due sottogruppi in cui gli utenti sono stati suddivisi

Di seguito viene è riportata invece la tabella riassuntiva dell'intero gruppo di utenti, formato da 20 volontari:

	Sesso	**Età**	**Studente di ingegneria?**	**Conoscenza software**
Utente 1	M	21	Ing. meccanica	Matlab
Utente 2	M	27	No	No
Utente 3	F	22	No	No
Utente 4	M	22	No	No
Utente 5	M	23	Ing. gestionale	No
Utente 6	M	25	No	No
Utente 7	M	27	Ing. meccanica	Matlab
Utente 8	M	24	No	No
Utente 9	F	22	No	No
Utente 10	F	21	No	No
Utente 11	M	28	No	No
Utente 12	M	26	Ing. meccanica	Comsol, Matlab
Utente 13	M	26	Ing. energetica	Matlab
Utente 14	F	23	No	No
Utente 15	M	24	No	No
Utente 16	M	25	No	No
Utente 17	F	23	Ing. aerospaziale	Comsol, Matlab
Utente 18	M	25	No	No
Utente 19	M	26	Ing. gestionale	Matlab
Utente 20	M	26	No	Comsol

Tabella 9.2: Profili degli utenti sottoposti al test

9.3 Risultati

Le seguenti tabelle illustrano le risposte date a ciascun quesito dai singoli utenti:

	Ingegneria?	Conoscenza dei software utilizzati	TEST con COMSOL (1 = corretto 0 = errato)										Punteggio totale per utente
			1	2	3	4	5	6	7	8	9	10	
U1	Ing. meccanica	Matlab	1	0	1	1	1	1	0	0	1	1	**7**
U2	No	No	0	0	1	1	0	1	0	1	0	1	**5**
U3	No	No	0	0	0	1	0	0	0	0	0	0	**1**
U4	No	No	0	0	0	0	0	0	1	1	1	0	**3**
U5	Ing. gestionale	No	0	0	1	1	0	0	0	1	0	1	**4**
U6	No	No	1	1	1	1	1	1	0	0	1	1	**8**
U7	Ing. meccanica	Matlab	1	0	1	1	1	0	1	0	1	1	**7**
U8	No	No	1	1	1	1	1	1	0	0	1	1	**8**
U9	No	No	0	0	1	0	0	1	0	1	1	0	**4**
U10	No	No	1	1	0	0	0	0	1	0	1	1	**5**
U11	No	No	0	1	0	0	1	0	0	0	0	0	**2**
U12	Ing. meccanica	Comsol, Matlab	1	0	1	1	1	0	0	1	0	0	**5**
U13	Ing. energetica	Matlab	1	0	1	1	1	0	1	0	1	1	**7**
U14	No	No	1	1	1	1	1	0	0	0	0	0	**5**
U15	No	No	0	0	1	1	0	0	0	0	0	0	**2**
U16	No	No	0	0	1	0	0	1	1	0	0	0	**3**
U17	Ing. Aerospaziale	Comsol, Matlab	1	0	0	1	0	1	1	1	0	0	**5**
U18	No	No	0	1	0	0	0	0	0	0	0	0	**1**
U19	Ing. gestionale	Matlab	1	1	0	1	1	0	0	0	1	1	**6**
U20	No	Comsol	1	0	1	1	1	0	0	1	1	0	**6**
			Somma dei punteggi per domanda										
		TOTALE →	**10**	**7**	**11**	**11**	**9**	**5**	**6**	**6**	**9**	**7**	

	Ingegneria?	Conoscenza dei software utilizzati	TEST con HAPTIKFEM (1 = corretto 0 = errato)										Punteggio totale per utente
			1	2	3	4	5	6	7	8	9	10	
U1	Ing. meccanica	Matlab	1	1	1	1	1	1	1	0	1	1	**9**
U2	No	No	1	1	1	0	1	0	1	1	1	1	**8**
U3	No	No	0	1	1	0	1	1	0	0	1	0	**5**
U4	No	No	1	0	0	0	1	0	0	0	1	1	**4**
U5	Ing. gestionale	No	1	1	1	1	0	1	1	1	1	1	**9**
U6	No	No	1	1	1	1	1	1	1	0	1	1	**9**
U7	Ing. meccanica	Matlab	1	1	1	1	1	1	1	1	0	0	**8**
U8	No	No	1	1	1	0	1	1	1	1	1	1	**9**
U9	No	No	1	0	0	1	1	1	1	0	1	1	**7**
U10	No	No	1	1	1	0	1	1	1	1	1	0	**8**
U11	No	No	0	0	1	1	0	0	0	0	0	1	**3**
U12	Ing. meccanica	Comsol, Matlab	1	0	1	0	1	1	1	1	1	0	**7**
U13	Ing. energetica	Matlab	1	1	1	1	1	1	1	0	1	1	**9**
U14	No	No	0	1	1	1	1	1	1	0	1	1	**8**
U15	No	No	1	0	0	0	1	0	0	0	0	1	**3**
U16	No	No	0	1	0	0	1	1	1	1	1	0	**6**
U17	Ing. Aerospaziale	Comsol, Matlab	1	0	1	1	1	1	1	0	1	1	**8**
U18	No	No	1	0	1	1	0	1	0	0	0	0	**4**
U19	Ing. gestionale	Matlab	0	1	1	0	1	1	1	0	1	1	**7**
U20	No	Comsol	1	1	1	1	1	0	1	1	0	1	**8**
			Somma dei punteggi per domanda										
		TOTALE →	**15**	**13**	**16**	**11**	**17**	**15**	**15**	**8**	**15**	**14**	

Tabelle 9.3: Riassuntivi delle risposte del test con Comsol (in alto) e con l'utilizzo dell'applicazione "HaptikFem" (in basso)

9.3.1 Confronto dei tempi di completamento del questionario

A tutti gli utenti è stato concesso del tempo per familiarizzare con le interfacce di Comsol e dell'HaptikFem. In questa fase non c'era un limite di tempo, e gli utenti potevano fare domande e osservazioni. Quando si sentivano pronti, potevano iniziare il test, tenendo presente che il tempo massimo a loro disposizione era di un'ora e che non avrebbero potuto chiedere spiegazioni sui quesiti, ma solo in caso di problemi con il software o di estrema necessità. In Tabella 9.4 sono riportati i tempi totalizzati:

	Studente di ingegneria?	**Conoscenza dei software utilizzati**	**Tempo Comsol (min)**	**Tempo HaptikFem (min)**
Utente 1	Ing. meccanica	Matlab	31	28
Utente 2	No	No	40	34
Utente 3	No	No	45	41
Utente 4	No	No	42	30
Utente 5	Ing. gestionale	No	35	25
Utente 6	No	No	50	41
Utente 7	Ing. meccanica	Matlab	37	22
Utente 8	No	No	41	38
Utente 9	No	No	56	44
Utente 10	No	No	44	31
Utente 11	No	No	32	24
Utente 12	Ing. meccanica	Comsol, Matlab	28	25
Utente 13	Ing. energetica	Matlab	39	36
Utente 14	No	No	58	45
Utente 15	No	No	44	40
Utente 16	No	No	52	38
Utente 17	Ing. Aerospaziale	Comsol, Matlab	25	25
Utente 18	No	No	36	34
Utente 19	Ing. gestionale	Matlab	40	36
Utente 20	No	Comsol	48	44

Tabella 9.4: Riassunto dei tempi totalizzati dagli utenti in ciascuna prova

Di seguito sono riportate le statistiche relative ai tempi conseguiti dagli utenti:

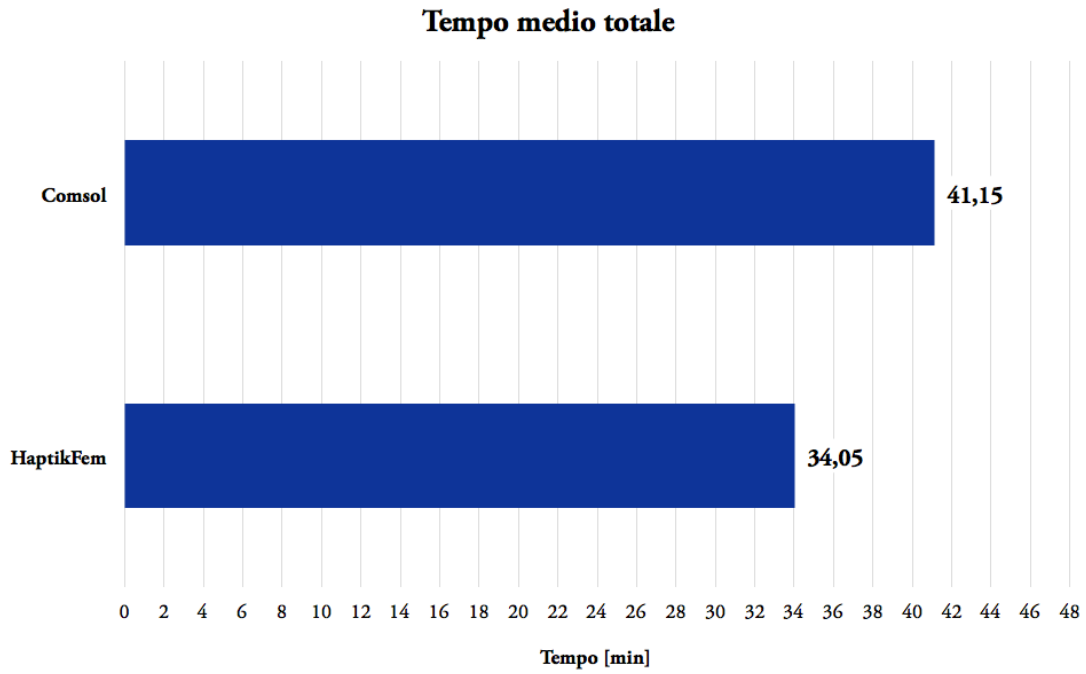


Figura 9.4: Istogramma dei tempi medi di completamento dei test conseguiti dagli utenti

Come si nota da Figura 9.4, mediamente gli utenti hanno impiegato 7,1 minuti in meno per completare il questionario che si riferisce all'HaptikFem.

Di seguito viene proposto un istogramma che confronta i tempi medi ottenuti dagli utenti con conoscenze ingegneristiche e/o dei software utilizzati con quelli ottenuti dagli utenti restanti:

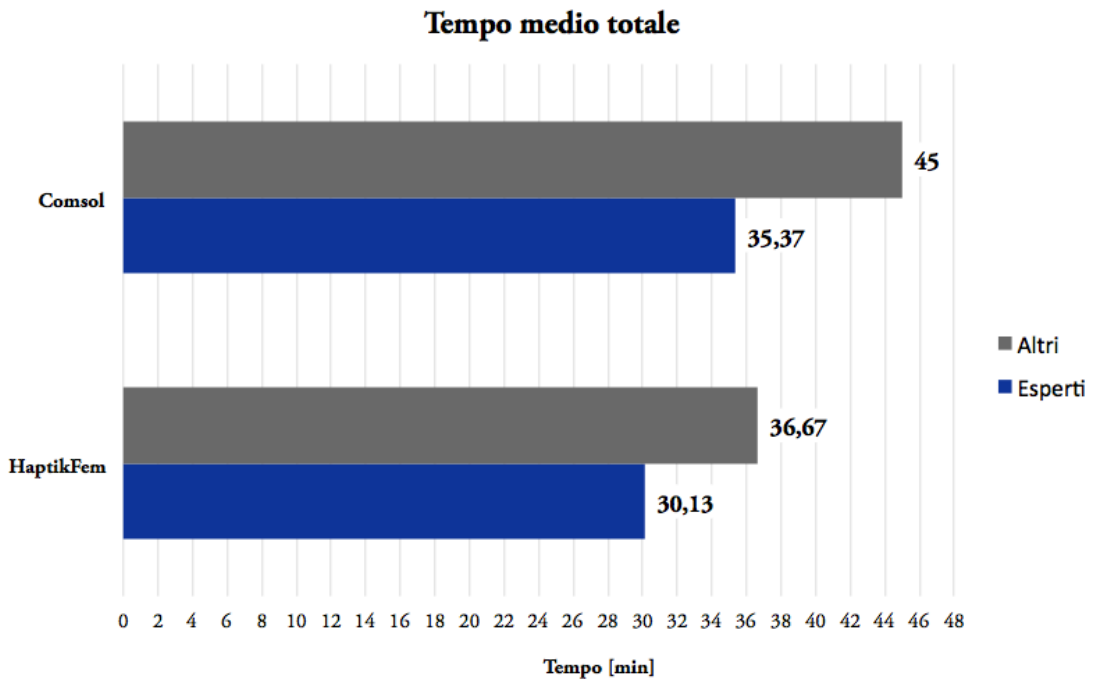


Figura 9.5: Confronto dei tempi medi impiegati da utenti con conoscenze ingegneristiche e/o dei software utilizzati con quelli ottenuti dagli altri utenti

Come si può facilmente osservare da Figura 9.5, gli utenti che avevano precedenti conoscenze ingegneristiche o di Comsol e Matlab hanno impiegato mediamente meno tempo per concludere il questionario. In particolare, gli utenti "esperti" hanno impiegato in media 9,63 minuti in meno a risolvere il questionario relativo a Comsol e 6,54 minuti in meno per quello relativo all'HaptikFem.

9.3.2 Confronto tra risposte corrette ed errate

La seconda analisi riguarda la percentuale di risposte esatte sul totale. Ciascuno dei 20 utenti poteva raggiungere un punteggio massimo di 10 punti per ciascun test, quindi ciascun test ha un valore complessivo di 200 punti. Di seguito sono riportati i grafici relativi ai due test effettuati:

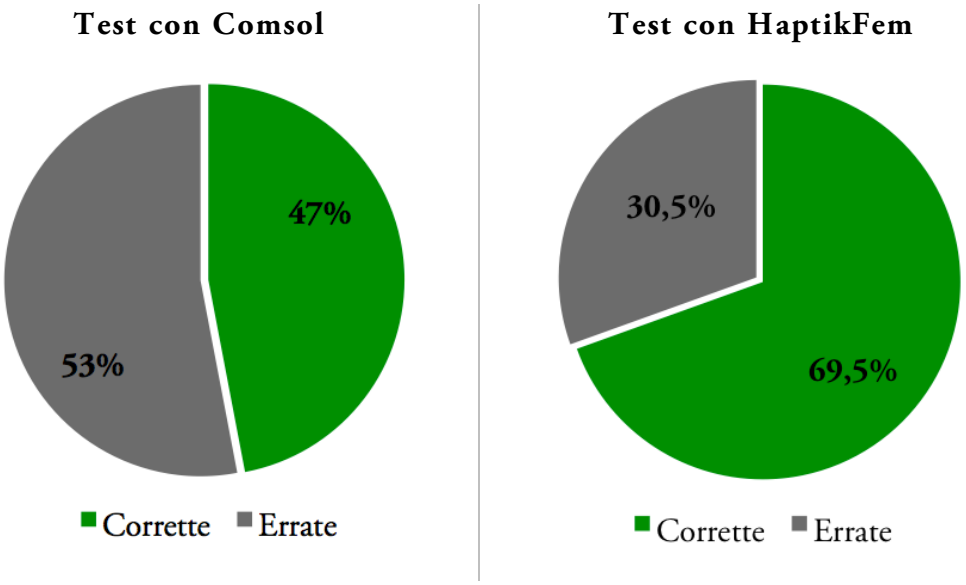


Figura 9.6: Confronto percentuale tra domande corrette ed errate nei due test

Come si può subito notare, complessivamente le risposte corrette con l'utilizzo dell'applicazione "HaptikFem" sono state maggiori del 22,5% rispetto al metodo classico, che prevedeva l'utilizzo esclusivo di Comsol. Si tratta di un dato di partenza importante, che è indicativo del fatto che gli utenti sono stati in grado di comprendere in modo più efficace alcuni principi di meccanica delle strutture grazie alla simulazione in tempo reale. Una conferma è stata anche fornita "a voce" dagli utenti stessi, i quali erano invitati a esprimere impressioni, critiche e a fornire eventuali consigli di miglioramento.

Tutti gli studenti intervistati, dopo aver provato entrambe le modalità di risoluzione, hanno affermato di essere rimasti positivamente colpiti dal nuovo metodo e di aver capito qualcosa in più rispetto al metodo "classico".

9.3.3 Confronto tra risposte corrette dei due gruppi

Com'è già stato accennato, gli utenti sono stati suddivisi in due gruppi allo scopo di valutare in modo più efficace eventuali progressi nell'apprendimento. Di seguito sono riportati i risultati che riguardano le percentuali delle risposte corrette conseguite da ciascun gruppo nelle due prove:

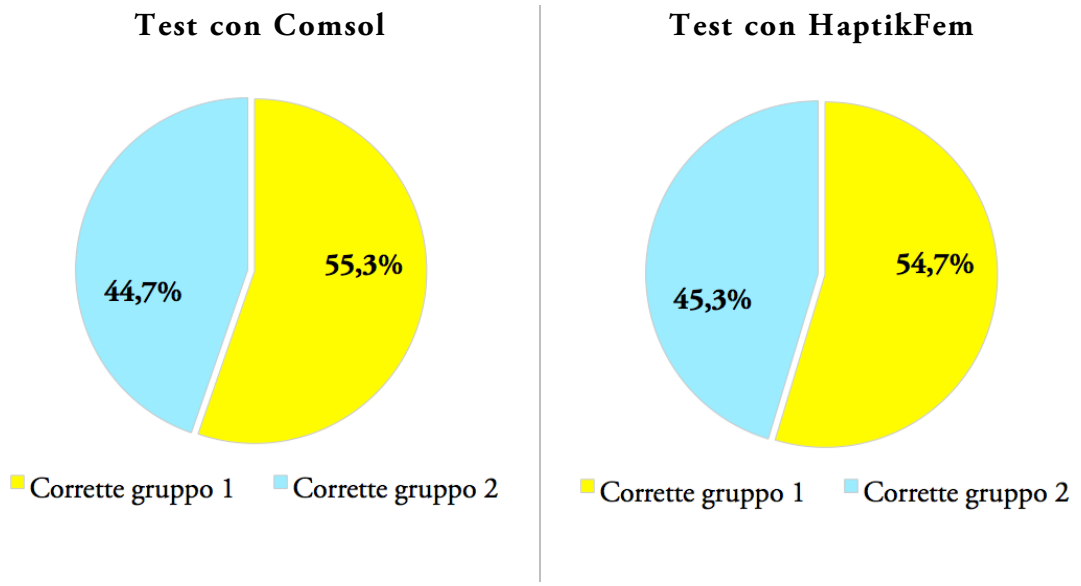


Figura 9.7: Confronto percentuale delle risposte corrette dei singoli gruppi nei test svolti

Il "Gruppo 1" corrisponde al gruppo di utenti (in giallo anche nelle Tabelle 9.3) che ha svolto prima il test con HaptikFem e in seguito con Comsol ed è quello che, come già precisato nel Paragrafo 9.2, auspicabilmente avrebbe dovuto conseguire risultati migliori

al fine di dimostrare un tasso di apprendimento positivo. Come si può notare, è proprio ciò che accade: il gruppo in questione si è dimostrato migliore in entrambi i test. Il punteggio più alto conseguito nella prova con HaptikFem è sintomo di una miglior comprensione dei problemi proposti, miglioramento che si riconferma anche nel successivo test con Comsol. Ciò potrebbe essere indice del fatto che ciò che è stato appreso nella prima prova è stato poi sfruttato anche nella seconda. Il "Gruppo 2" ha invece conseguito un risultato mediamente peggiore del 10% in entrambe le prove, il che potrebbe essere sintomo del fatto che eventuali perplessità nate dalla prova effettuata con Comsol siano state poi trasferite anche alla prova con HaptikFem.

9.3.4 Punteggi complessivi nelle singole domande

Dopo un'analisi piuttosto generale, si può passare a livelli di approfondimento maggiori, andando a sondare i risultati delle singole domande:

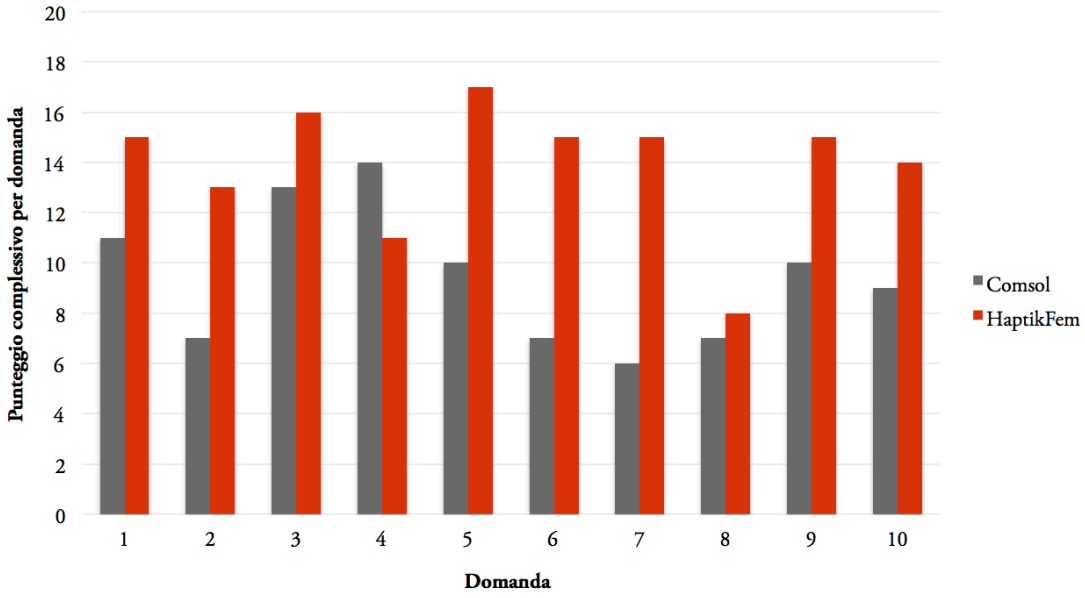


Figura 9.8: Istogramma dei punteggi complessivi per domanda ottenuti nei singoli test

Dalla tabella appare evidente il progresso fatto con l'applicazione in real-time rispetto al metodo di risoluzione con Comsol. Si nota in particolare:

- Un grande divario di punteggio, a favore dell'HaptikFem, nelle domande 2, 5, 6, 7. Andando a verificare a cosa si riferiscono, si possono ricavare gli aspetti che sono stati probabilmente più apprezzati. Le domande 2 e 7 si riferiscono alla comprensione della corretta deformata della struttura al variare di carico e vincoli rispettivamente. La domanda 5 verifica che si sia compreso il rapporto di linearità tra carico e deformazione. La domanda 6 cerca di stabilire se si è compresa la differenza tra le varie tipologie di vincolo.

- In due domande l'HaptikFem ha ottenuto un punteggio inferiore o uguale a quello di Comsol. Le domande 4 e 8 si occupano entrambe della capacità dell'utente di impostare un valore di carico e di verificare il corretto valore della deformata in punti specifici. Sotto questo punto di vista Comsol si rivela più efficace, poiché permette di impostare numericamente i valori desiderati, al contrario dell'HaptikFem, nella quale si deve impostare il corretto valore di carico utilizzando il mouse. Tenendo conto anche delle critiche mosse in tal senso da alcuni utenti, si prende quindi atto che questo è senza dubbio un punto da migliorare nelle future versioni dell'applicazione.

9.3.5 Punteggi dei singoli utenti

Allo stesso modo di quanto visto al paragrafo precedente, si può svolgere una analisi sui punteggi complessivi conseguiti da ciascun utente in ciascun test. Di seguito è riportato il grafico riassuntivo:

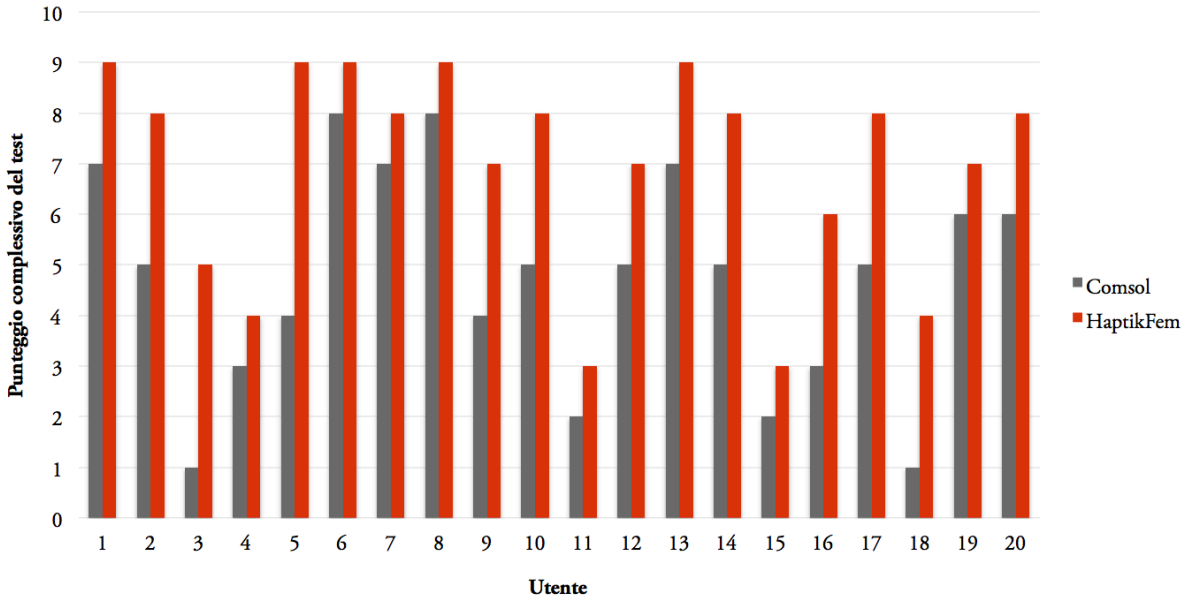


Figura 9.9: Istogramma dei punteggi totali ottenuti da ciascun utente nei singoli test

Dall'analisi dei punteggi è semplice verificare come tutti gli utenti abbiano conseguito un punteggio più alto utilizzando l'applicazione HaptikFem. Il progresso medio è stato del 22,5%, con un picco del 50% per l'utente 5. I risultati fanno sono dunque positivi, l'HaptikFem appare robusta sotto il punto di vista del grado di apprendimento e dell'usabilità, due dei punti fondamentali che ci si era prefissi di verificare al momento della stesura dei questionari.

9.3.6 Confronto dei punteggi in base all'esperienza

Si può infine stilare una statistica per confrontare i punteggi ottenuti al variare del grado di esperienza degli utenti, in modo analogo a quanto già fatto con i tempi di completamento dei questionari. Com'è ovvio che sia, ci si aspetta un punteggio mediamente superiore dagli utenti "esperti":

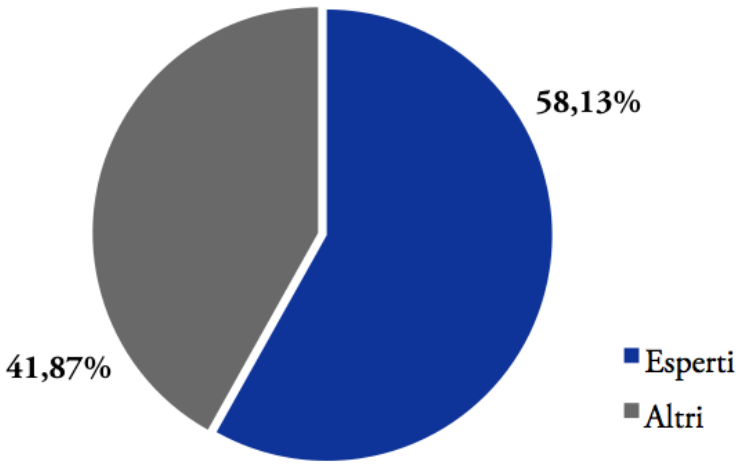


Figura 9.10: Percentuale di risposte corrette suddivise in base all'indirizzo di studio degli utenti

Com'era lecito aspettarsi, gli utenti provenienti facoltà ingegneristiche o che comunque avevano già dimestichezza con i software utilizzati e con gli argomenti trattati, hanno conseguito punteggi mediamente superiori degli altri. Più del 58% delle risposte corrette, infatti, è stata data da questi utenti.

9.4 Conclusioni sugli user test

I test sono senza dubbio uno strumento importante quando si vuole svolgere un'analisi sull'usabilità di un'applicazione o di un prodotto. Ignorarne i risultati vorrebbe dire perseverare negli errori e non andare incontro alle esigenze degli utenti. E' per questo motivo che in seguito alle indicazioni suggerite dagli stessi utenti è stato deciso di apportare alcuni cambiamenti a quella che era l'applicazione originaria. Una delle principali lamentele, che si è verificata fondata anche dal risultato oggettivo dei test, è stata fatta sull'eccessiva mancanza di controllo delle deformazioni o dei carichi imposti alla struttura. Al momento di questo test, l'utente modificava il carico o lo spostamento imposto semplicemente muovendo il mouse, con scarsa possibilità di controllo e diverse difficoltà nel fermare la simulazione al raggiungimento del valore di carico o di spostamento desiderato. Per questo motivo è stato introdotto uno *slider* laterale, che permette di avere maggior controllo (Figura 9.11).
Altre critiche riguardavano l'utilizzo della tastiera al posto del mouse nelle schermate di selezione dei modelli e delle altre opzioni. Ritenendo tuttavia che si tratti di critiche legate più all'abitudine (oggi si usa quasi esclusivamente il mouse nella maggior parte delle applicazioni, contrariamente al periodo precedente alla sua introduzione) che all'effettivo corretto funzionamento dell'applicazione, si è scelto, per il momento, di non apportare modifiche in tal senso, preferendo invece concentrarsi su aspetti più rilevanti.

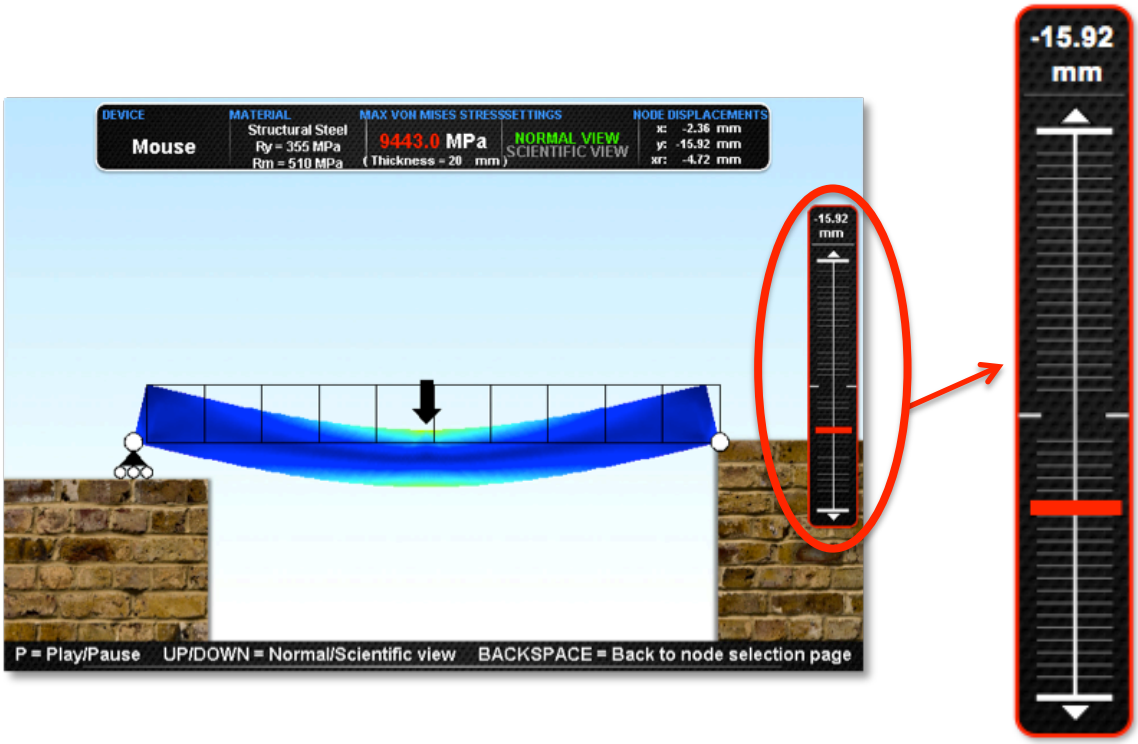


Figura 9.11: Slider laterale per la configurazione "Mouse" dell'HaptikFem introdotto dopo i test effettuati con gli utenti. Cliccando e trascinando in alto e in basso l'apposito cursore rosso, la struttura si deforma si conseguenza

Complessivamente il risultato dei test fa ben sperare, sia per le opinioni positive espresse dagli utenti, sia per i punteggi ottenuti, che mostrano un trend di apprendimento oggettivamente positivo con l'utilizzo dell'HaptikFem.

Capitolo 10: Conclusioni

L'obiettivo principale di questa tesi era realizzare un'applicazione a scopo didattico che riuscisse a svolgere simulazioni FEM in tempo reale, e che potesse essere utilizzata anche con dispositivi *haptic*. Ciò è stato fatto con l'intento di avvicinare queste tecnologie a un argomento, come quello della meccanica delle strutture, d'interesse ingegneristico. Nessun software in commercio possiede però un'interfaccia nativa per svolgere tutte queste operazioni, pertanto per raggiungere il risultato che ci si era proposto, è stato necessario trovare diverse soluzioni di compromesso, e il modo di far coesistere diversi software. La soluzione ottenuta, come si è visto, non è ottimale ma raggiunge lo scopo e, con un calcolatore professionale, consente oggi di eseguire analisi agli elementi finiti in tempo reale con un *frame-rate* ritenuto soddisfacente.

Alcuni problemi nascono dal momento in cui si utilizza il sistema *haptic*, che richiederebbe una frequenza di aggiornamento molto superiore ma per ora non raggiungibile. L'effetto indesiderato è che l'utente sente degli "scatti" durante la simulazione, ma anche in questo caso l'illusione di "toccare" una struttura reale è comunque realistica e apprezzabile. Altri problemi nascono, come già spiegato nel Paragrafo 6.7 della presente tesi, dalla compatibilità con la Haptik Library, strumento tuttavia indispensabile per far comunicare Matlab e il *Phantom*.

Nonostante ciò, i risultati ottenuti dopo la fase di test con gli utenti sono stati positivi e fanno ben sperare per eventuali sviluppi futuri. Tutti gli utenti hanno apprezzato la possibilità di scegliere una struttura, applicare vincoli e carichi ed eseguire simulazioni in un modo nuovo, più intuitivo e veloce. I dati confermano anche un effettivo miglioramento del grado di apprendimento, fattore importante per un'applicazione per uso didattico.

Molti sono i miglioramenti che tuttavia potrebbero essere apportati all'applicazione HaptikFem, primi tra tutti la risoluzione dei problemi di cui si è accennato.

L'interfaccia potrebbe essere strutturata in maniera più efficace, e consentire agli utenti di effettuare le scelte utilizzando il mouse al posto della tastiera, così come è stato richiesto dalla maggior parte di loro. Altre modifiche potrebbero essere inserite nella fase di simulazione, nella quale sarebbe ad esempio utile poter imporre un preciso valore numerico di forza o spostamento senza dover avvicinarsi alla quantità desiderata esclusivamente spostando il cursore con il mouse, o utilizzando il sistema *haptic*. Si ritiene inoltre che grandi vantaggi in termini di prestazioni potrebbero essere ottenuti con la compilazione del programma, operazione tentata più volte con il compilatore di Matlab ma non andata a buon fine. La compilazione consiste nella traduzione delle istruzioni scritte nel linguaggio di programmazione (chiamate "codice sorgente" e contenute nei file .m dell'HaptikFem), in istruzioni di un altro linguaggio, chiamato "codice oggetto". Questa operazione consentirebbe di utilizzare HaptikFem indipendentemente dai software installati sul computer, come una vera e propria applicazione a se stante, con conseguenti benefici dal punto di vista della semplicità d'installazione e dell'usabilità.

Si ritiene tuttavia che il vantaggio maggiore potrebbe essere ottenuto se in futuro Comsol implementasse nell'interfaccia "*Comsol with Matlab*" la possibilità di effettuare simulazioni in tempo reale e con l'utilizzo di sistemi *haptic*, cosa che, con questa tesi, è stato dimostrato essere già perfettamente possibile.

Appendice I: Questionari e disegni

Di seguito viene riportato il questionario e gli schemi sono stati forniti agli utenti nella fase dei test. I questionari sono due, uno per il test con Comsol, e uno per il test con l'applicazione HaptikFem. Le domande nei due test sono molto simili, allo scopo di mantenere lo stesso grado di difficoltà. Ciascun questionario contiene dieci domande: le prime cinque si concentrano sull'applicazione dei carichi, le rimanenti sulle condizioni di vincolo. Ogni parte del test è corredata di precise istruzioni per aiutare l'utente a orientarsi nella risoluzione degli esercizi proposti.

Questionario per il test con Comsol:

Real Time FEA

Comsol-FEM Approach

Use Comsol and the attachments to consider the following questions.
Ask for help just in case of extreme necessity.

General information

1. **Please specify your gender**
   Mark only one oval.
   - ◯ M
   - ◯ F
2. **Please specify your age:**
   .....
3. **What are you studying?**
   .....

PART 1 - LOADS

PART 1 DESCRIPTION:

Look at the model in Comsol.

It is a portal with a load in the point corresponding to "A6" point on the figure in the attachment and with a fixed/fixed constraint on edges "C1-C2" and "C3-C4".

Work with the model following the skills below. Pay attention to deformed shapes, stresses, loads etc, because when you're ready you have to CLOSE COMSOL and ask to several adimensional questions about them.

SKILLS for Part 1:

Load change:

If you want to change the load or its point of application press F5 and then:

- Select in the left tab the node number corresponding to the point on which there is a load, an in "Load" tab reset it to 0 N.

- Then select a point on which you want to apply a load, and set its value only for Fy (leave Fx=0 N). Press OK.

- Click on "=" icon on top to start the simulation.

Point displacement/stress evaluation:

Go to: POSTPROCESSING ⇒ POINT EVALUATION...

- Select a point in the left tab.

- Set "Total displacement" as Predefined displacement on the right if you want the displacement for that point; instead if you want to calculate the von Mises stress for that point, set "miss_ps" as Expression. Then press OK.

- The result is in the grey tab at the bottom of the main window.

When you think you're ready, CLOSE COMSOL and start the test below!

4. **1) Imagine a y-load down directed on point "A6". Where is the maximum von Mises stress? (4)**
   Mark only one oval.
   - ◯ point "C1"
   - ◯ point "C2"
   - ◯ point "B1"
   - ◯ point "A6"
   - ◯ point "B5"

---

5. **2) Imagine a y-load down directed on point "A4". Which is the correct deformation of the structure? (3)**
   Mark only one oval.
   - ◯ Figure 1
   - ◯ Figure 2
   - ◯ Figure 3
   - ◯ Figure 4
   - ◯ Figure 5

---

6. **3) Imagine to apply a y-load on point "A6". Then image to double it. In the last case the maximum von Mises stress become: (1)**
   Mark only one oval.
   - ◯ exactly double
   - ◯ about the same
   - ◯ about 2/3
   - ◯ it depends by the material
   - ◯ It's impossible to say

7. **4) Which is the magnitude order of y-displacement of point "A6" when there i a load of 100 N of magnitude order on it? (4)**
   Mark only one oval.
   - 0 mm
   - 1 mm
   - 0.1 mm
   - 0.01 mm
   - 0.001 mm

---

8. **5) To a double load on a point corresponds: (5)**
   Mark only one oval.
   - a deformation depending by the material
   - a deformation depending by the thickness
   - an equal deformation
   - a half deformation
   - a double deformation

PART 2 - CONSTRAINTS

PART 2 DESCRIPTION:

- RESET THE MODEL FIRST (ASK FOR HELP)

Look at the model in Comsol.

It is a portal with a load in the point corresponding to "A6" point on the figure in the attachment and with a fixed/fixed constraint on edges "C1-C2" and "C3-C4".

Work with the model following the skills below. Pay attention to deformed shapes, stresses, loads and in particular to constraints, because when you're ready you have to CLOSE COMSOL and ask to several adimensional questions about them.

SKILLS for Part 2:

Constraints change (for edges):

If you want to change the constraint or its edge of application press F7 and then:

- Select in the left tab the node number corresponding to the edge on which there is a constraint, and in "Constraint" tab reset Rx and Ry to 0 mm.

- Then select a edge on which you want to apply a constraint, and set Rx=0 to lock its x-displacements, Ry=0 to lock its y-displacements. Then press OK.

- Click on "=" icon on top to start the simulation.

Constraints change (for points):

If you want to change the constraint or its point of application press F5 and then:

- Select in the left tab the node number corresponding to the point on which there is a constraint, and in "Constraint" tab reset Rx and Ry to 0 mm.

- Then select a node on which you want to apply a constraint, and set Rx=0 to lock its x-displacements, Ry=0 to lock its y-displacements. Then press OK.

- Click on "=" icon on top to start the simulation.

When you think you're ready, CLOSE COMSOL and start the test below!

9. **6) Imagine that edges "C1-C2" and "C3-C4" have a fixed constraint. Compared to a fixed/roller constraint scheme respectively for edge "C1-C2" and point "H2", the maximum von Mises stress is: (2)**
   Mark only one oval.
   - ◯ lower
   - ◯ higher
   - ◯ about the same
   - ◯ it depends by the point of load application
   - ◯ it depends by the thickness of the structure

---

10. **7) Image that on point "H1" there is a hinge, and on point "H2" a roller. The correct deformed shape is: (5)**
    Mark only one oval.
    - ◯ Figure 2
    - ◯ Figure 3
    - ◯ Figure 4
    - ◯ Figure 5
    - ◯ None

---

11. **8) Which is the magnitude order of x-displacement of point "H2" when there is a roller on it and a fixed constraint on edge "C1-C2", with a load of 100 N of magnitude order on point "A6"? (2)**
    Mark only one oval.
    - ◯ 0 mm
    - ◯ 1 mm
    - ◯ 0.1 mm
    - ◯ 0.01 mm
    - ◯ 0.001 mm

12. **9) Imagine there is a y-load, down directed on point "A6". The maximum von Mises stress there will be for the constraint scheme (respectively on edges "C1-C2" and "C3-C4"): (1)**
Mark only one oval.

- ◯ fixed/fixed
- ◯ fixed/hinge
- ◯ fixed/roller
- ◯ hinge/hinge
- ◯ hinge/roller

---

13. **10) Imagine there is a y-load, up directed on point "A6". The maximum y-displacement of point "A6" there will be for the constraint scheme (respectively on edges "C1-C2" and "C3-C4"): (5)**
Mark only one oval.

- ◯ fixed/fixed
- ◯ fixed/hinge
- ◯ fixed/roller
- ◯ hinge/hinge
- ◯ hinge/roller

Questionario per il test con HaptikFem:

PART 1 - LOADS

PART 1 DESCRIPTION:

Look at the models in HaptikFem, in particular focus your attention on model 3 (PORTAL).

Work with the models following the skills below. Pay attention to deformed shapes, stresses, and in particular on LOADS and DISPLACEMENTS, because when you're ready you have to CLOSE MATLAB and ask to several adimensional questions about them.

SKILLS for Part 1:

Use the KEYBOARD to make your choose in every window of the application. If you choose "MOUSE" as device, you will control the deformation of the structure with up/down movements of the mouse. If you choose "PHANTOM" you will control the structure with the haptic device, and you will "feel" the structure and its responses with your hands.

Load/Displacement point change:

- If you want to change the point of application of the load or the point to displace, move the red arrow with left/right buttons on the keyboard, then press enter.

- To change the entity of the force or the displacement, simply move the mouse or the haptic device up and down, very SLOWLY!

Point displacement/stress evaluation:

- Look at the top of the simulation window: there are all the informations about the structure, from the material used to the displacement. Pay attention to these quantities, because they will be useful to ask to the questions.

When you think you're ready, CLOSE MATLAB and start the test below!

4. **1) Imagine a y-load down directed on point "A6". Where is the maximum von Mises stress? (4)**
   Mark only one oval.
   - point "C1"
   - point "C2"
   - point "B1"
   - point "A6"
   - point "B5"

---

5. **2) Imagine a y-load down directed on point "A4". Which is the correct deformation of the structure? (3)**
   Mark only one oval.
   - Figure 1
   - Figure 2
   - Figure 3
   - Figure 4
   - Figure 5

---

6. **3) Imagine to apply a y-load on point "A6". Then image to double it. In the last case the maximum von Mises stress become: (1)**
   Mark only one oval.
   - exactly double
   - about the same
   - about 2/3
   - it depends by the material
   - It's impossible to say

7. **4) Which is the magnitude order of y-displacement of point "A6" when there i a load of 100 N of magnitude order on it? (4)**
   Mark only one oval.
   - ◯ 0 mm
   - ◯ 1 mm
   - ◯ 0.1 mm
   - ◯ 0.01 mm
   - ◯ 0.001 mm

---

8. **5) To a double load on a point corresponds: (5)**
   Mark only one oval.
   - ◯ a deformation depending by the material
   - ◯ a deformation depending by the thickness
   - ◯ an equal deformation
   - ◯ a half deformation
   - ◯ a double deformation

PART 2 - CONSTRAINTS

PART 2 DESCRIPTION:

Now focus your attention on constraint schemes. Modify the constraints, in particular for model 3 (PORTAL).

Work with the models following the skills below. Pay attention to deformed shapes, stresses, and in particular on the effect of CONSTRAINTS, because when you're ready you have to CLOSE MATLAB and ask to several adimensional questions about them.

SKILLS for Part 2:

Constraints change:

If you want to change the constraint, you can go to the "Constraint" window and make your choose using up/down buttons on the keyboard.

When you think you're ready, CLOSE COMSOL and start the test below!

9. **1) Imagine that edges "C1-C2" and "C3-C4" have a fixed constraint. Compared to a fixed/roller constraint scheme respectively for edge "C1-C2" and point "H2", the maximum von Mises stress is: (2)**
   Mark only one oval.
   - ◯ lower
   - ◯ higher
   - ◯ about the same
   - ◯ it depends by the point of load application
   - ◯ it depends by the thickness of the structure

---

10. **2) Image that on point "H1" there is a hinge, and on point "H2" a roller. The correct deformed shape is: (5)**
    Mark only one oval.
    - ◯ Figure 2
    - ◯ Figure 3
    - ◯ Figure 4
    - ◯ Figure 5
    - ◯ None

---

11. **3) Which is the magnitude order of x-displacement of point "H2" when there is a roller on it and a fixed constraint on edge "C1-C2", with a load of 100 N of magnitude order on point "A6"? (2)**
    Mark only one oval.
    - ◯ 0 mm
    - ◯ 1 mm
    - ◯ 0.1 mm
    - ◯ 0.01 mm
    - ◯ 0.001 mm

12. **4) Imagine there is a y-load, down directed on point "A6". The maximum von Mises stress there will be for the constraint scheme (respectively on edges "C1-C2" and "C3-C4"): (1)**
Mark only one oval.

- ◯ fixed/fixed
- ◯ fixed/hinge
- ◯ fixed/roller
- ◯ hinge/hinge
- ◯ hinge/roller

---

13. **5) Imagine there is a y-load, up directed on point "A6". The maximum y-displacement of point "A6" there will be for the constraint scheme (respectively on edges "C1-C2" and "C3-C4"): (5)**
Mark only one oval.

- ◯ fixed/fixed
- ◯ fixed/hinge
- ◯ fixed/roller
- ◯ hinge/hinge
- ◯ hinge/roller

Disegni:

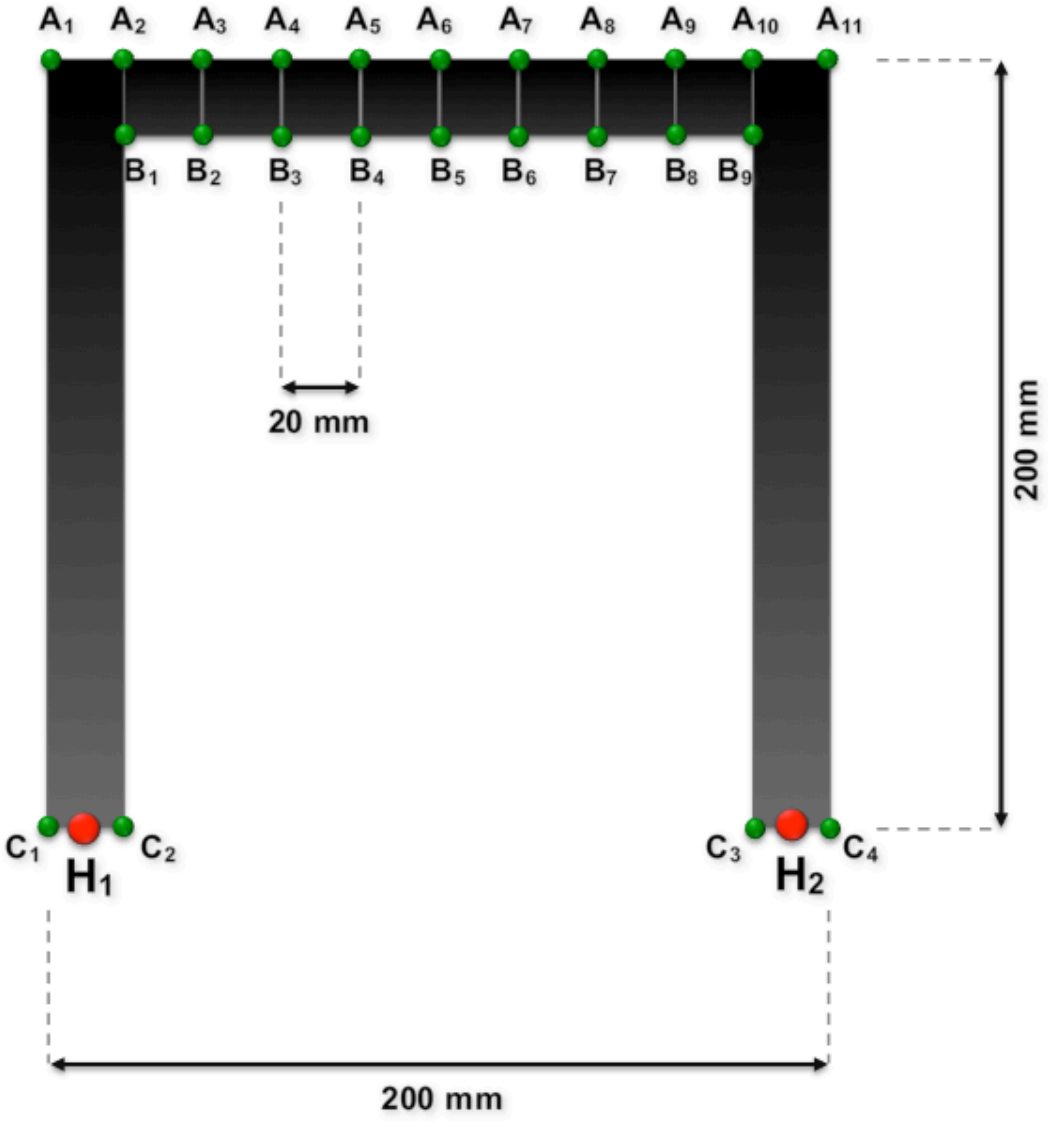

A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11
B1 B2 B3 B4 B5 B6 B7 B8 B9
20 mm
200 mm
C1 C2
H1
C3 C4
H2
200 mm

	Figure 1
	Figure 2
	Figure 3
	Figure 4
	Figure 5

Appendice II: Cartelle e sottocartelle

In questa breve appendice è descritta la struttura delle cartelle e sottocartelle dell'HaptikFem, con lo scopo di consentire agli utenti di orientarsi al loro interno in modo più veloce ed efficace.

Cartella principale

La cartella principale dell'applicazione si chiama "HAPTIKFEM", ed è la cartella di riferimento da impostare come "*Current Folder*" di Matlab per consentire il corretto avvio dell'applicazione. Per farlo, l'utente può selezionarla dal corrispondente menu a tendina presente nella parte superiore di Matlab:

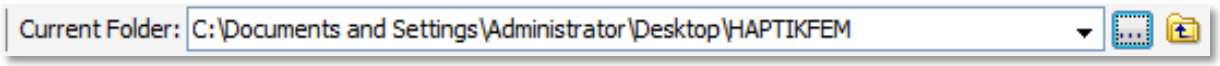


Figura II.1: Selezione della directory corretta in Matlab

Dopo aver svolto questa operazione, la cartella e tutte le sottocartelle presenti vengono caricate in Matlab, e sono visualizzabili nella finestra chiamata anch'essa "*Current Folder*" posta a sinistra, che indica nel dettaglio tutte le cartelle e i file presenti nella cartella principale:

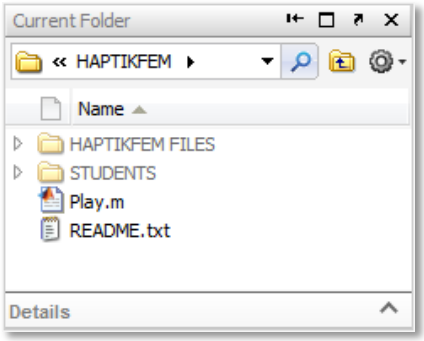


Figura II.2: Dettaglio delle cartelle e dei file presenti nella cartella principale

Come si può notare da Figura II.2, nella cartella principale sono presenti due sottocartelle ("HAPTIKFEM FILES" e "STUDENTS"), il file "Play.m" e un file di testo. Aprendo quest'ultimo si possono trovare una breve presentazione dell'applicazione e i principali requisiti di sistema:

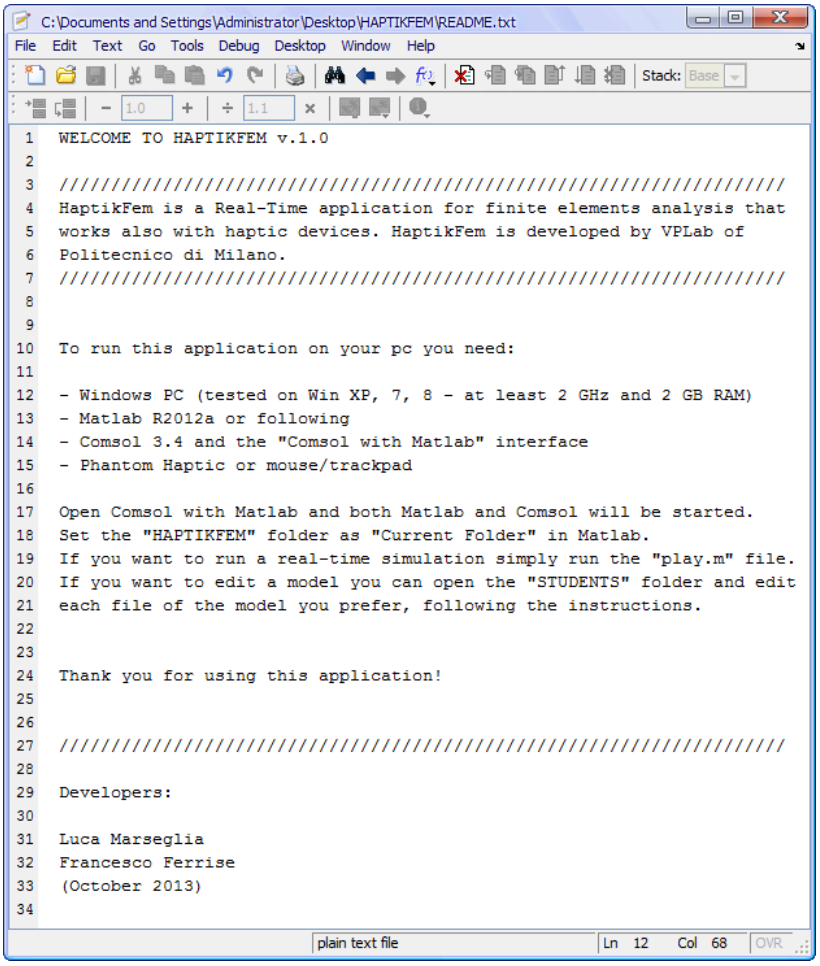


Figura II.3: Contenuto del file "README.txt"

Aprendo il file "Play.m", invece, l'utente può avviare direttamente l'applicazione HaptikFem. All'interno del file sono contenute istruzioni che servono all'inizializzazione del programma, come ad esempio l'acquisizione delle dimensioni dello schermo, e la sincronizzazione di tutti i file e delle sottocartelle.

Cartella "HAPTIKFEM FILES"

La cartella "HAPTIKFEM FILES" è il cuore dell'applicazione che lavora in *real-time*. Al suo interno è contenuta una lunga serie di sottocartelle, che si occupano della gestione dei menu e dei modelli, sia per la parte haptic che per quella classica:

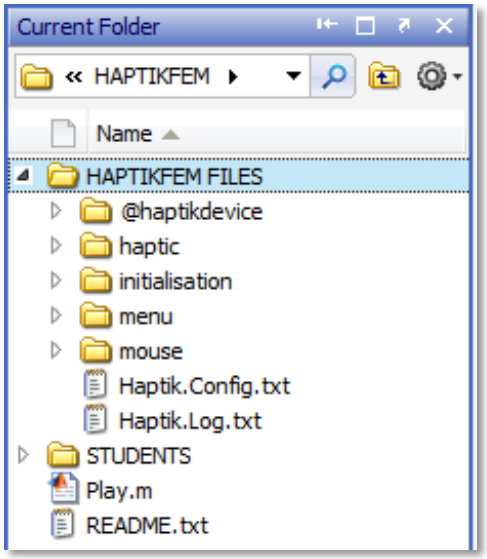


Figura II.4: Contenuto della cartella "HAPTIKFEM FILES"

Di seguito è riportato un elenco dei contenuti di ciascuna sottocartella:

- **@haptikdevice**: contiene i file necessari alla configurazione del dispositivo *haptic* (se presente) in modo che possa funzionare in correttamente.

- **haptic**: contiene i modelli utilizzati per la simulazione in tempo reale con il dispositivo *haptic*. I modelli sono suddivisi in cartelle denominate "hmodel".

- **initialisation**: come si può intuire contiene alcuni file di inizializzazione, dedicati in parte al corretto funzionamento del dispositivo haptic. Contiene anche il file .jpg utilizzato per la schermata di benvenuto e la funzione "stopall.m" che interviene quando l'utente decide di terminare l'esecuzione dell'applicazione chiudendo la finestra grafica (vedere il Paragrafo 6.6).

- **menu**: questa cartella contiene tutti i file necessari al corretto funzionamento dei menu, nonché i file .jpg di ciascuna schermata e file che registrano le scelte fatte dall'utente a ogni *step* del menu stesso.

- **mouse**: il contenuto è del tutto simile a quello della cartella "haptic", ma in questo caso i modelli contenuti sono ottimizzati per il funzionamento con il mouse o trackpad e sono inclusi in cartelle denominate "model".

- **Haptik.Config e Haptik.Log**: si tratta di file di sistema della "Haptik Library". Il primo è un file di configurazione attraverso il quale si può interfacciare il dispositivo haptic in uso con Matlab; il secondo è un file di log generato automaticamente e che tiene conto di tutte le operazioni e degli eventuali errori svolti con il dispositivo.

Cartella "STUDENTS"

La cartella "*students*" è stata creata appositamente per gli studenti che vogliano cimentarsi con l'analisi dei file e la loro modifica. Al suo interno sono presenti sei sottocartelle contenenti i primi 6 modelli con visualizzazione 2D già descritti nel Capitolo 8. I modelli in questione sono i più semplici da modificare e riproducono più strutture tipiche dei corsi di meccanica delle strutture. I file derivano direttamente da quelli dell'applicazione, ma sono stati semplificati e funzionano esclusivamente con mouse o trackpad, poiché normalmente la disponibilità di sistemi *haptic* è molto limitata. Sono ad esempio stati eliminati molti elementi grafici, e le informazioni riguardanti sforzi e deformazioni compaiono ancora in tempo reale ma sotto forma di semplice "titolo" della finestra grafica. Anche lo "slider" laterale è stato eliminato, e l'utente potrà modificare il valore del carico semplicemente con un "*click and drag*" sulla struttura.

Aprendo ad esempio la cartella relativa al modello 2, sono presenti i seguenti file:

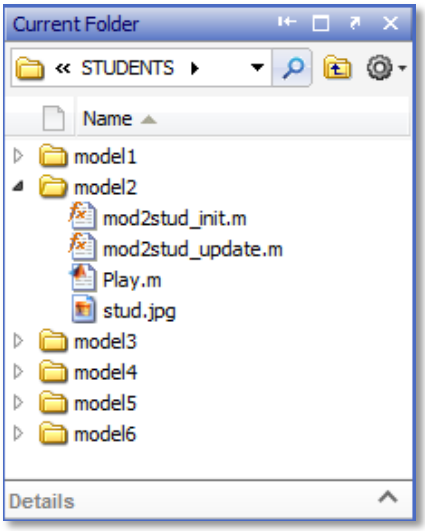


Figura II.5: Contenuto della sottocartella relativa al modello 2.

Di tutti i file contenuti in ciascuna sottocartella lo studente è invitato a modificare esclusivamente il file di inizializzazione, in questo caso "mod2stud_init.m", poiché in esso sono contenute tutte le istruzioni relative a geometria, carichi e vincoli. Aprendo tale file l'utente sarà guidato nella modifica delle varie sezioni:

```
%-----------------------------------------------------------------
% (EDIT) CONSTRAINTS:
% set   pnt.Hy={0,1,1,0}  and  pnt.Hx={0,1,1,0}  for  hinge/hinge
% set   pnt.Hy={0,1,1,0}  and  pnt.Hx={0,0,1,0}  for  hinge/roller

pnt.Hy = {0,1,1,0};
pnt.Hx = {0,0,1,0};
%-----------------------------------------------------------------
```

I campi che possono essere modificati sono indicati con la dicitura "EDIT", contrariamente a quelli che invece non devono assolutamente essere editati poiché si potrebbe compromettere il corretto funzionamento dell'applicazione, che sono introdotti dalla scritta "DO NOT EDIT". Il primo campo editabile, come si nota dalla porzione di codice riportata è quella che si occupa della definizione di eventuali costanti. L'utente è invitato a modificare il valore delle costanti già presenti, o a introdurne di nuove da utilizzare nella fase successiva, dedicata alla costruzione della struttura.

Si passa quindi alla fase di modifica della geometria. Il Modello 2 è quello di una semplice trave a doppio appoggio composta da una successione di geometrie quadrate. L'utente può quindi modificare la posizione, la dimensione o la rotazione di ciascuna geometria. Ovviamente le modifiche dovranno generare una nuova struttura caratterizzata da continuità e coerenza delle geometrie che la compongono.

```
% (EDIT) GEOMETRY:
% building of the structure. Edit only this quantities:
% "pos" is the position of each entity.
% "rot" is the rotation of each entity.

g1=square2('L','base','corner','pos',{'0','-L/2'},'rot','0','const',fem.const);
g2=square2('L','base','corner','pos',{'L','-L/2'},'rot','0','const',fem.const);
g3=square2('L','base','corner','pos',{'2*L','-L/2'},'rot','0','const',fem.const);
g4=square2('L','base','corner','pos',{'3*L','-L/2'},'rot','0','const',fem.const);
g5=square2('L','base','corner','pos',{'4*L','-L/2'},'rot','0','const',fem.const);
g6=square2('L','base','corner','pos',{'5*L','-L/2'},'rot','0','const',fem.const);
g7=square2('L','base','corner','pos',{'6*L','-L/2'},'rot','0','const',fem.const);
g8=square2('L','base','corner','pos',{'7*L','-L/2'},'rot','0','const',fem.const);
g9=square2('L','base','corner','pos',{'8*L','-L/2'},'rot','0','const',fem.const);
g10=square2('L','base','corner','pos',{'9*L','-L/2'},'rot','0','const',fem.const);
```

Lo *step* successivo è dedicato alla modifica dello schema di vincolo. Dalle istruzioni nella porzione di codice riportata sopra si può notare come sia molto semplice per l'utente imporre una cerniera al posto di un carrello semplicemente imponendo il valore "1" al posto dello "0" nella seconda posizione del vettore "Hx":

```
%--------------------------------------------------------------------------
% (EDIT) CONSTRAINTS:
% set   pnt.Hy={0,1,1,0}  and  pnt.Hx={0,1,1,0}  for  hinge/hinge
% set   pnt.Hy={0,1,1,0}  and  pnt.Hx={0,0,1,0}  for  hinge/roller

pnt.Hy = {0,1,1,0};
pnt.Hx = {0,0,1,0};
%--------------------------------------------------------------------------
```

L'utente è in seguito invitato a modificare il punto di applicazione del carico. Come si è già visto al Paragrafo 5.1, la disposizione di vincoli e carico è riassunta in un unico vettore, qui chiamato "pnt.ind". Semplicemente spostando il numero "4" in una posizione (pari) differente, l'utente è in grado di cambiarne il punto di applicazione sulla struttura.

```
%-------------------------------------------------------------------------
% (EDIT) NODE SELECTION:
% move the number "4" in a parity position in the pnt.ind vector to change
% the point of force/displacement application on the structure

pnt.ind = [2,1,1,1,1,1,1,1,1,1,1,4,1,1,1,1,1,1,1,1,3,1];
%-------------------------------------------------------------------------
```

Si passa poi a un'impostazione molto veloce, che stabilisce solo il metodo di controllo. L'utente può quindi decidere se effettuare una simulazione imponendo uno spostamento o una forza nel punto precedentemente selezionato:

```
%-------------------------------------------------------------------------
% (EDIT) CONTROL METHOD:
% set  control=1  for force control
% set  control=2  for displacement control

control=2;
%-------------------------------------------------------------------------
```

L'ultima porzione di codice liberamente modificabile dagli utenti è quella dedicata ai parametri del materiale e allo spessore della struttura. I parametri del materiale possono essere impostati a piacimento, esulando quindi da quelli preimpostati nell'applicazione. In alcuni modelli l'utente è in grado di modificare non solo lo spessore ma anche il diametro di eventuali fori.

```
%-------------------------------------------------------------------------
% (EDIT) MATERIAL AND THICKNESS:
% set Young modulus (E), density (rho) and thickness of the structure.

equ.E = 210000;
equ.rho = '7870 [Kg/m^3]';
equ.thickness = 20;
%-------------------------------------------------------------------------
```

Dopo aver modificato e salvato il file, l'utente può aprire il file "play.m" ed avviarlo. La prima esecuzione risulta di solito un po' più lenta delle successive, poiché Matlab deve inizializzare per la prima volta la geometria. Se l'utente non varia la geometria, il Modello 2 viene visualizzato come segue:

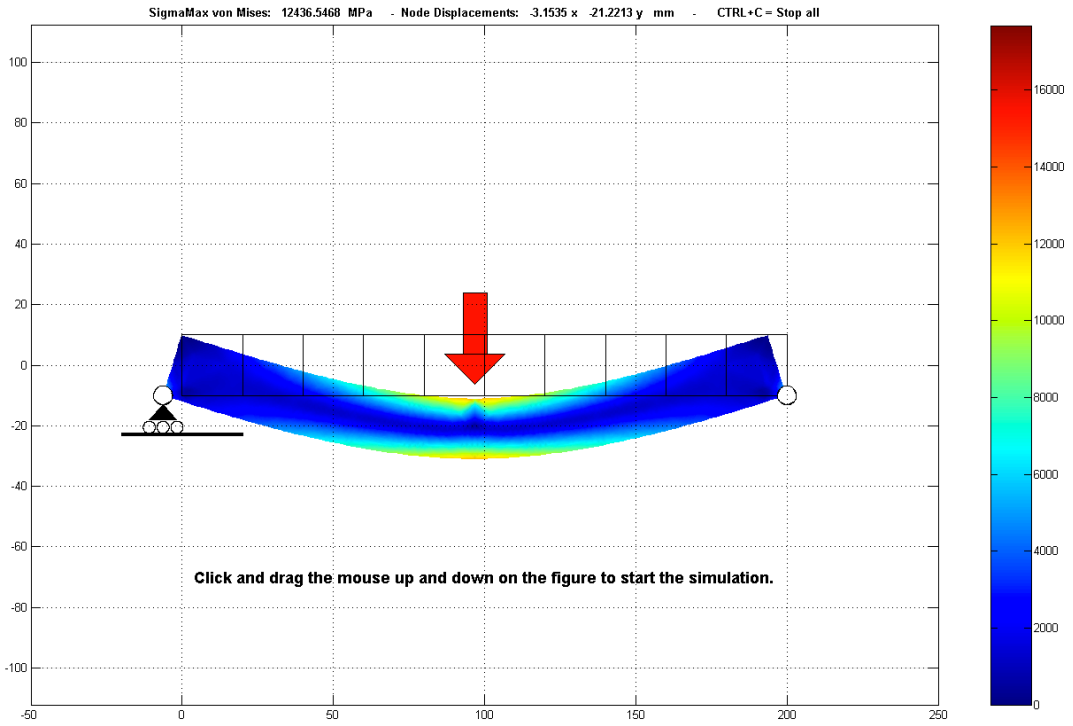


Figura II.6: Schermata di visualizzazione tipica dei file per gli studenti

Com'era già stato accennato all'inizio di questo paragrafo, la rappresentazione grafica per questi modelli è stata notevolmente semplificata rispetto a quella dell'applicazione completa. Ciò è stato fatto sia per evitare di confondere gli utenti con porzioni di codice dedicate a mere funzioni di ottimizzazione grafica, sia per velocizzare al massimo la fase di simulazione anche su computer non molto potenti.

Bibliografia

Articoli:

[1] M. Fiorentino, G. Monno, A. E. Uva, S. Cristiano (2009): "Distributed design review using tangible augmented technical drawings". Computer-Aided Design 42 (2010) pag. 364-372.

[2] Kevin T. McDonnell Hong Qin (2001): "FEM-based Subdivision Solids for Dynamic and Haptic Interaction". Proceedings of the sixth ACM symposium on Solid modeling and applications, pag. 312-313.

[3] Teeranoot Chanthasopeephan, Jaydev P. Desai†, and Alan C. W.: "3D and 2D Finite Element Analysis in Soft Tissue Cutting for Haptic Display*". Advanced Robotics, 2005. ICAR '05. Proceedings., 12th International Conference on.

[4] Andrew H. Gosline, Septimiu E. Salcudean, and Joseph Yan (2004): "Haptic Simulation of Linear Elastic Media with Fluid Inclusions". Journal of Haptics Research.

[5] Jeffrey Berkley, George Turkiyyah, Daniel Berg, Mark Ganter, Suzanne Weghorst (2004): "Real-Time Finite Element Modeling for Surgery Simulation: An Application to Virtual Suturing". IEEE Transactions on Visualizazion and Computer Graphics.

[6] Thomas H. Massie, J. K. Salisbury (1994): "The PHANTOM Haptic Interface: A Device for Probing Virtual Objects". Proceedings of the ASME Winter Annual Meeting, Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems.

Libri:

[7] Tom Tullis, Bill Albert: "Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics" - Morgan Kaufmann Publishers.

[8] Grunwald M.: "Human Haptic Perception - Basic and Application" - Springer Verlag Publishers

[9] Anton Weber, Schahram Dustar: "Haptic Systems Architecture Modeling". Springer Verlag KG Publishers

[10] P. Davoli, A. Bernasconi, M. Filippini, S. Foletti: "Comportamento Meccanico dei Materiali". McGrawHill, 2005

Siti internet:

[11] http://www.kaemart.it/

[12] http://www.sensable.com/

[13] http://www.openscenegraph.org/projects/osg

[14] http://www.vrlab.umu.se/research/osgHaptics/

[15] http://www.chai3d.org

[16] http://www.h3dapi.org

[17] http://sirslab.dii.unisi.it/papers/haptic/DePr-RAM08.pdf

[18] http://www.calculix.de

[19] http://www.3ds.com/products/simulia/portfolio/abaqus/latest-release/

[20] http://www.solidthinking.com

[21] http://www.comsol.it/products/4.3a/

[22] http://www.majentasolutions.com/

[23] http://blogs.mathworks.com/pick/2008/05/27/

[24] http://www.mathworks.it/support/solutions/en/data/1-2X10AT/

[25] http://graphics.cs.cmu.edu/projects/stvk/

[26] http://www.altair.com/

[27] http://www.ansys.com/it_it

[28] http://www.mscsoftware.com/product/msc-nastran

[29] http://www.code-aster.org/V2/spip.php?rubrique1

[30] http://www.freefem.org/ff++/

[31] http://www.mathworks.it/

[32] http://it.wikipedia.org/wiki/Real-time