POLITECNICO DI TORINO

FACOLTA' DI INGEGNERIA

Corso di Laurea Magistrale in Ingegneria Civile

Tesi di Laurea Magistrale

Software FEM per il monitoraggio strutturale di ponti a graticcio

Relatore

Prof. Gabriele Bertagnoli

Candidato

Luciano Gandolfo

Anno Accademico 2019/2020

A mio padre e a mia madre,

A mio fratello,

A chi ha creduto in me.

Indice

Abstract

Il monitoraggio strutturale, con particolare attenzione al settore delle infrastrutture, ha assunto un ruolo centrale negli ultimi anni. Nel presente lavoro di tesi è stato sviluppato ed applicato un programma scritto con il linguaggio di programmazione Python che permette di effettuare analisi statiche elastiche lineari di strutture intelaiate tridimensionali. Il modello numerico della struttura monitorata costituisce infatti uno degli elementi necessari per un sistema di monitoraggio. L'applicazione ha riguardato la modellazione, attraverso un modello a grigliato, di una campata dell'impalcato di un viadotto autostradale.

Il software sviluppato è pensato per poter essere implementato all'interno di una architettura complessa di monitoraggio strutturale formata da un elevato numero di sensori, nodi di controllo, trasmissione GSM e cloud data analysis. Esso può essere caricato sia sul cloud, che sui nodi (gateway) che controllano i sensori. Queste ultime apparecchiature sono elaboratori piuttosto semplici, tipo Raspberry PI 4 e necessitano quindi di un software leggero dedicato alla loro architettura.

La scelta del linguaggio di programmazione Python è stata effettuata in quanto si tratta di un linguaggio multi-paradigma che supporta la programmazione orientata agli oggetti, completamente open-source, caratterizzato da una sintassi chiara e molto immediato nel suo utilizzo. Queste caratteristiche hanno fatto di Python il protagonista di una enorme diffusione in tutto il mondo, e anche in Italia. Lo sviluppo del software si è svolto in collaborazione con D. Craig Brinck, ingegnere civile strutturista operante in Salt Lake City (Utah) negli USA e ideatore iniziale del codice PyNiteFEA (*Building.Code@outlook.com*).

La tesi è organizzata nel seguente modo:

Nel Capitolo 1 viene affrontato in generale il tema del monitoraggio strutturale, definendo il concetto di danno e lo sviluppo di un processo di monitoraggio; in più vengono descritti alcuni tra i sistemi di monitoraggio utilizzati nell'ambito delle infrastrutture civili.

Il Capitolo 2 analizza brevemente la programmazione orientata agli oggetti (Object Oriented Programming), con i suoi concetti base e i suoi vantaggi, introducendo la terminologia necessaria alla comprensione della libreria python utilizzata nel presente lavoro di tesi.

Il Capitolo 3 tratta la libreria python PyNiteFEA, spiegando la suddivisione interna in moduli e i sistemi di riferimento locale e globali utilizzati dal codice.

Il Capitolo 4 si pone l'obiettivo di fornire tutte le conoscenze necessarie per la costruzione di un qualsiasi modello ad elementi finiti tramite la libreria python PyNiteFEA. Vengono spiegate dettagliatamente tutte le funzionalità supportate dal codice e le istruzioni necessarie all'utente per il loro utilizzo.

Il Capitolo 5, infine, contiene un'applicazione della libreria python a un caso studio. Viene descritto il viadotto in esame e il sistema di monitoraggio su di esso installato. Con riferimento alle prove di carico effettuate sull'impalcato e sulla base delle misurazioni dei sensori installati, si è costruito e calibrato un modello elastico lineare. Verranno illustrate tutte le varie scelte di modellazione e la loro influenza sui risultati finali.

CAPITOLO 1
Monitoraggio strutturale

Il monitoraggio strutturale, noto nella letteratura scientifica come "Structural Health Monitoring" (SHM), è il processo di caratterizzazione delle strutture esistenti con il quale ci si propone di identificare alcune proprietà della struttura, in modo da avere informazioni sul livello prestazionale della stessa e con le quali poter calibrare modelli analitici per la valutazione dello stato della struttura e dell'evoluzione del suo comportamento nel tempo. Nell'ambito delle opere civili, quali per esempio edifici, ponti, viadotti e dighe, il monitoraggio strutturale ricopre un ruolo decisivo perché direttamente collegato alla sicurezza degli utenti. Il controllo di tali opere richiede infatti uno studio attento e dettagliato al fine di garantire condizioni di servizio compatibili con gli standard di sicurezza indicati dalle norme tecniche di settore.

L'installazione e la gestione di un sistema di monitoraggio strutturale necessita l'investimento di risorse economiche in misura non trascurabile. La scelta della strumentazione dipende dai possibili scenari di danno che si vogliono analizzare e dall'entità delle grandezze che si devono misurare per poter risalire a tale danno. Pertanto, nel caso in cui si utilizzi una strumentazione non adeguata, l'intervento di monitoraggio si può rivelare un'applicazione di poca o addirittura nulla utilità.

L'impiego di tecniche di monitoraggio avanzate, permette di intervenire in modo proattivo, cioè, percependo anticipatamente i potenziali problemi strutturali, le tendenze o i cambiamenti futuri, al fine di programmare in modo efficace un piano di manutenzione per ridurre i costi dello stesso e prevenire l'insorgere del danno. Diversamente, l'utilizzo di tecniche di monitoraggio meno avanzate consentirebbe di intervenire con attività di manutenzione solamente dopo il verificarsi del danno.

E' chiaro, che nell'ambito dell'ingegneria civile, ciò non rappresenta la soluzione migliore sia per i costi che devono essere sostenuti sia per ragioni legate ai rischi sull'incolumità delle persone.

I sistemi di monitoraggio strutturale possono essere realizzati con modalità molto differenti a seconda delle finalità e delle motivazioni tecniche. Negli ultimi anni si stanno sviluppando apparati moderni che sfruttano lo sviluppo di nuove tecniche numeriche di identificazione strutturale e l'adozione di sensori sempre più sofisticati. Nonostante le varie differenze, tutte le tipologie di sistema presentano alcune componenti comuni:

- Sensori
- Sistema di archivio dei dati
- Sistema di trasmissione dei dati (solitamente in rete)
- Algoritmo di pulitura dei dati
- Modello numerico o sperimentale per lo studio dei dati
- Sistema decisionale sulla base dei dati di output

Per i sistemi di monitoraggio più semplici, il maggior problema dal punto di vista pratico è stato spesso legato alla difficoltà di raccogliere i dati e inviarli a dispositivi remoti. Negli ultimi anni lo sviluppo di reti wireless locali e delle trasmissioni radio in generale ha in gran parte risolto tale difficoltà, perciò al giorno d'oggi i problemi maggiori riguardano le operazioni di pulitura e interpretazione dei dati raccolti.

1.1 Individuazione del danno

All'interno di un sistema di monitoraggio è necessario definire il concetto di danno, affinché il sistema sia in grado individuarlo e segnalarlo opportunamente. I moderni sistemi di monitoraggio si basano sull'utilizzo di sensori permanentemente installati sulla struttura, poco invasivi e automatizzati. Altrettanto automatico dovrà essere il processo di individuazione del danno e la successiva segnalazione, in base ad una condizione limite fissata in fase di progetto del sistema.

Il danno può considerarsi come il manifestarsi nella struttura di una variazione rispetto ad una condizione iniziale di perfetta salute dell'opera, perciò è bene installare i sistemi di monitoraggio su strutture sane, in modo che essi rimangano funzionanti e raccolgano dati

partendo da una condizione iniziale nota e priva di criticità. Nell'ambito strutturale, il danno può essere ricondotto ad un cambiamento:

- delle proprietà dei materiali (e.g. carbonatazione, corrosione);
- delle condizioni di vincolo (cedimenti vincolari, bloccaggio di gradi di libertà altrimenti liberi, interazione non desiderata con strutture adiacenti);
- della geometria del sistema strutturale (riduzione delle sezioni: e.g. espulsione del copriferro, corrosione delle armature, corrosione dei cavi di precompressione);
- delle presollecitazioni di progetto (variazione delle forze di precompressione);

Ad esempio, si può assumere che la struttura non lavori più in condizioni accettabili se la rigidezza scende al di sotto di un limite prefissato, causando:

- spostamenti nei punti monitorati superiori a un valore soglia;
- variazione delle frequenze principali di vibrazione;
- variazione delle forme modali;
- redistribuzioni di sforzo con conseguente variazione delle tensioni o delle forze nei punti monitorati.

1.2 Il processo di monitoraggio

L'attività di monitoraggio consiste nell'osservazione di una struttura nel tempo a intervalli regolari e la successiva analisi statistica e strutturale dei dati ottenuti, finalizzata a ricavare i parametri indicativi delle condizioni di "salute" della struttura.

In generale, il processo di monitoraggio può essere diviso in 4 fasi [1]:

- Valutazione operativa: la valutazione operativa riguarda la fattibilità tecnico-economica del sistema di monitoraggio che si vuole realizzare, a seconda del tipo di struttura analizzata e delle sue peculiarità. In funzione di ciò si definisce il processo di individuazione del danno più adatto, la tipologia di sensori, il numero e la loro posizione.
- Acquisizione e pulitura dei dati: i dati registrati dai sensori sono raccolti in centraline e inviati (per i sistemi moderni mediante rete wirless) a memorie hardware aventi dimensione in funzione del numero di sensori e della frequenza di acquisizione. I dati raccolti sono sottoposti a operazioni di pulitura ("data cleaning") in modo da distinguere le variazioni di misurazioni causate dal danno da quelle causate da rumori

ambientali. I dati ottenuti dopo il "data cleaning" conterranno informazioni strettamente legate allo stato di salute della struttura monitorata.

- Estrazione delle caratteristiche: è necessario ottenere delle caratteristiche o dei parametri fisici significativi a partire dai quali poter differenziare tra strutture danneggiate e non. Un metodo di estrazione delle caratteristiche per l'identificazione del danno è quello di applicare difetti di ingegneria, simili a quelli che ci si aspetta nella struttura in condizioni operative, a sistemi fittizi e sviluppare un'iniziale comprensione dei parametri sensibili al danno previsto. Il sistema difettoso può essere usato al fine di verificare che le misurazioni diagnostiche siano sufficientemente sensibili nella distinzione tra sistema danneggiato e sistema privo di danno. L'interpretazione dei dati può essere svolta, per esempio, con riferimento a esperienze passate, oppure tramite strumenti analitici sperimentalmente convalidati come modelli ad elementi finiti.
- Sviluppo di modelli statistici: lo sviluppo di modelli statistici si occupa dell'implementazione di algoritmi che permettono l'estrazione di caratteristiche per poter individuare e quantificare lo stato di danno della struttura.

1.3 Esempi di sistemi di monitoraggio

La necessità di monitorare le strutture esistenti ha fatto sì che negli ultimi anni la ricerca in questo settore si concentrasse sullo sviluppo dei sensori. Molte tecnologie oggi impiegate nel settore civile provengono da altri rami dell'ingegneria (meccanica, aeronautica, …), altri sensori invece sono studiati appositamente per le strutture civili. Alcuni esempi sono gli accelerometri, i sensori di emissioni acustiche, i sensori in fibra ottica, i dispositivi di geolocalizzazione (GPS – Global Positioning System) e molti altri. Di seguito vengono descritti alcune tipologie utilizzate nel settore civile.

- *GPS*

Il sistema di posizionamento globale (GPS) ha fornito nuove possibilità per la misura degli spostamenti nelle infrastrutture. La tecnica del posizionamento relativo sfrutta la misurazione dei tempi di ritorno dei segnali inviati dal satellite al ricevitore, garantisce misurazioni assolute e dirette degli spostamenti senza la maggior parte dei problemi intrinsecamente legati ai sistemi ottici. Il sistema necessita di una stazione master GPS installata in una posizione di riferimento nota vicino alla struttura monitorata e di un'antenna rover installata sulla struttura in corrispondenza dei punti che si vogliono monitorare. Il posizionamento relativo permette di

determinare il vettore "baseline" che collega la stazione master, di coordinate note, con la generica stazione rover. Le posizioni delle due antenne sono definite in un determinato sistema di coordinate geodetiche Note le coordinate della stazione master e calcolato il vettore "baseline", si ottengono le coordinate del punto monitorato.

In alternativa all'utilizzo di una stazione master, soprattutto qualora non fosse possibile collocarne una nelle immediate vicinanze della struttura da monitorare o per ridurre i costi del sistema di monitoraggio, si possono utilizzare algoritmi di processazione dei dati forniti da più antenne rover collocate sulla struttura oppure sfruttare il "sistema di riferimento virtuale" che fornisce, in funzione della località, le correzioni da applicare ai dati di posizione.

Nel caso in cui non vi siano effetti dinamici importanti (monitoraggio strutturale in campo statico) e adottando opportuni tempi di misura e frequenze di acquisizione, si riescono a ottenere misurazioni sufficientemente accurate, dell'ordine di alcuni millimetri. Ciò è dimostrato anche dall'esperienza condotta sui ponti Akashi-Kaikio e Tsing Ma da parte di Wong et al.

- *Interferometria radar terrestre*

Tra le tecnologie emergenti nell'ambito del monitoraggio strutturale, i sistemi di telerilevamento hanno avuto un ruolo predominante. Tra le tecniche di telerilevamento più efficaci per lo SHM figura l'Interferometria Radar Terrestre, la cui efficacia è pienamente comprovata da diversi studi e applicazioni. I risultati forniti da tale tecnologia sono del tutto paragonabili a quelli ottenuti mediante altre tecniche convenzionali di monitoraggio "da contatto" (ad esempio reti di accelerometri, inclinometri, estensimetri ecc. installati sulle strutture). Tale tecnica si presta bene a rapide analisi di parametri strutturali, anche in condizioni di emergenze, col vantaggio di abbattere i costi legati alle fasi progettuali e di installazione tipiche di altre tecniche convenzioni a contatto. In particolare, non essendo possibile discriminare oggetti ricadenti nella medesima cella di risoluzione in range (ubicati quindi alla stessa distanza dal radar), le strutture che meglio si prestano per un monitoraggio dinamico con tecnica ad interferometria radar, sono quelle caratterizzate da un prevalente sviluppo monodimensionale verticale (come torri, ciminiere, grattacieli, ecc.) od orizzontale (come ponti, viadotti, passerelle, ecc.) rispetto a strutture a prevalente estensione planimetrica sulle quali si potrebbe incorrere in problemi di ambiguità.

L'Interferometria Radar Terrestre permette di misurare contemporaneamente gli spostamenti di numerosi punti di strutture o di altri elementi antropici e naturali, con elevate frequenze di campionamento del dato. La misurazione è eseguita completamente in remoto sfruttando la natura riflettiva alle microonde degli elementi presenti nello scenario irradiato. Il calcolo degli spostamenti avviene attraverso il confronto delle informazioni di fase dell'onda elettromagnetica emessa e riflessa a differenti intervalli temporali (Figura 1.2).

Figura 1.1 – Strumentazione per interferometria radar terrestre

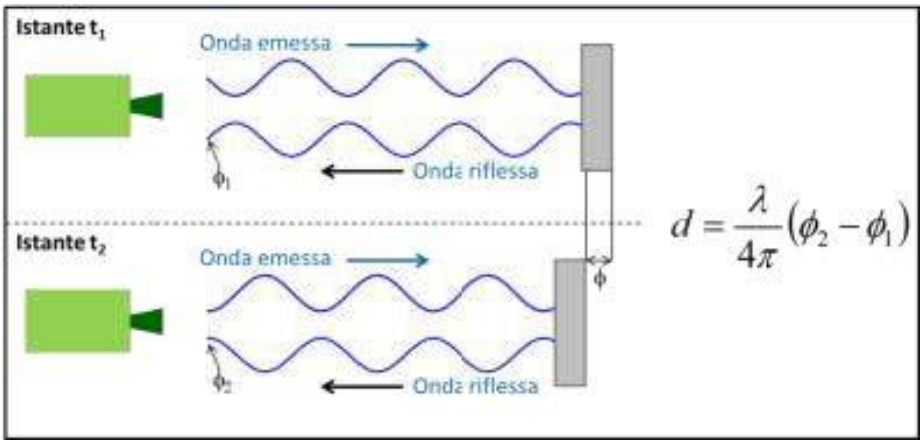


Figura 1.2 - Principio interferometrico per il calcolo degli spostamenti (d=spostamento misurato; λ=lunghezza d'onda; ϕ=misura della fase)

Le portate strumentali tipiche delle tecnologie attualmente in commercio vanno da alcune centinaia di metri fino ad alcuni km. In Tab. 1.1 si riportano alcuni parametri tecnici relativi ai principali sistemi di interferometria radar terrestre presenti in commercio.

Tab. 1.1 – Parametri tecnici generali dei sistemi di interferometria radar terrestre

Parametro	Valore
Massima distanza di acquisizione	Variabile da alcune centinaia di metri ad alcuni chilometri
Massima frequenza di campionamento (PRF)	Variabile da alcune centinaia di Hz fino ad alcune migliaia di Hz
Accuratezza	Fino al centesimo di mm (in condizioni di elevato rapporto segnale/rumore)
Risoluzione spaziale	Max. 75 cm (in range)

Per esempi applicativi di tale tecnica di monitoraggio si rimanda alle pubblicazioni di Beninati et. al [2] e Bongiovanni et. al [3] in cui vengono trattati rispettivamente il monitoraggio dinamico di un antico ponte in ferro e della Colonna Aureliana a Roma [4].

- *Accelerometri*

Gli accelerometri sono componenti chiave all'interno di un sistema di monitoraggio strutturale nel campo dell'ingegneria civile. Sono strumenti molto versatili ed efficienti nell'individuazione di fenomeni di danneggiamento a seguito di carichi sismici o all'azione del vento, e nel settore della sensoristica, possono essere considerati come sensori a basso costo. Attraverso l'analisi dei dati di accelerazione acquisiti è possibile risalire ai principali parametri modali della struttura (frequenze proprie, smorzamenti e forme modali) e conoscerne il comportamento dal punto di vista dinamico. Tali parametri dinamici costituiscono "l'impronta digitale" della struttura e la ripetizione delle prove a distanza di tempo consente l'identificazione di mutamenti non sempre individuabili con monitoraggi statici.

Un accelerometro è un dispositivo elettro-meccanico in grado di misurare l'accelerazione statica (gravità) o dinamica (movimenti o vibrazioni). I moderni accelerometri sono dei dispostivi con tecnologia MEMS ("Micro Electro-Mechanical System) e in genere, oltre al sistema di acquisizione dei dati contengono ulteriori sensori per la misurazione di temperatura e umidità. Gli accelerometri possono essere classificati a seconda del principio di funzionamento del sensore di posizione contenuto al suo interno, il quale ha il compito di rilevare lo spostamento della massa rispetto alla struttura fissa del dispositivo.

Gli accelerometri piezoelettrici (Figura 1.4) sfruttano come principio di rilevazione dello spostamento il segnale elettrico generato da un cristallo piezoelettrico quando questo è soggetto ad una forza di compressione. In presenza di un'accelerazione la massa comprime il cristallo il quale genera un segnale elettrico proporzionale alla compressione. L'accelerometro

piezoelettrico offre molteplici caratteristiche vantaggiose come il fatto di essere robusto e affidabile, con caratteristiche stabili anche per lunghi periodi, e di essere in grado di resistere a sollecitazioni di shock molto elevate. Di contro, non è in grado di misurare accelerazioni quasi statiche; in questi casi infatti, se la compressione sul cristallo permane, le cariche elettriche raggiungeranno in breve tempo una configurazione di equilibrio e il segnale decadrà rapidamente.

Gli accelerometri capacitivi invece (Figura 1.5), sfruttano la variazione di capacità elettrica di un condensatore al variare della distanza tra le due sue armature. Nel caso dell'accelerometro, la struttura fissa del dispositivo e la massa di prova costituiscono le armature del condensatore (Figura 1.3).

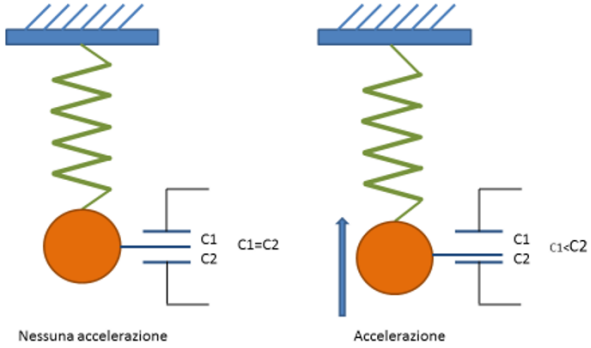


Figura 1.3 – Schematizzazione di un accelerometro capacitivo

Nello specifico, negli ultimi anni, gli accelerometri capacitivi vengono realizzati con tecnologia MEMS. Di seguito un esempio di accelerometro piezoelettrico e di accelerometro capacitivo con tecnologia MEMS.

Figura 1.4 - Accelerometro piezoelettrico

Figura 1.5 – Accelerometro capacitivo MEMS

- *Sensori a fibra ottica*

I sensori a fibra ottica sono rilevatori che, sfruttando i principi di ottica ondulatoria, adottano per le misurazioni non più le proprietà della corrente elettrica bensì quelle della luce. Ciò

garantisce l'insensibilità di tali sensori ai campi magnetici rispetto ai convenzionali sensori elettrici. I sistemi di monitoraggio con sensori a fibre ottiche presentano dei vantaggi importanti rispetto ai metodi di misura più convenzionali, come ad esempio, le loro dimensioni ridotte, la vasta gamma di parametri misurabili, l'insensibilità alla corrosione e una assoluta immunità da disturbi elettromagnetici.

Il sensore FBG ("Fiber Bragg Grating") è un sensore basato sulla tecnologia in fibra ottica ed è detto 'intrinseco': non necessita di alcuna altra fonte di energia per funzionare, se non la luce che si propaga lungo la fibra ottica stessa (non serve alimentazione elettrica presso il punto di misura); non necessita di alcun altro tipo di collegamento per la propagazione del segnale, se non la fibra ottica stessa. Il sensore FBG funziona 'in riflessione': la stessa fibra ottica che 'alimenta' il sensore, ne trasporta indietro anche il segnale. Nella parte più interna della fibra, nel core, viene realizzata una variazione periodica dell'indice di rifrazione che costituisce il reticolo di Bragg. Ciò significa che quando una luce proveniente da una sorgente viene introdotta nella fibra ottica, solo quella con una larghezza spettrale molto stretta, centrata sulla lunghezza d'onda di Bragg, verrà riflessa indietro dalla griglia. La luce restante continuerà il suo percorso nella fibra ottica fino alla successiva griglia di Bragg senza subire alcuna perdita. In tutto ciò, la lunghezza d'onda di Bragg dipende dalla contrazione/espansione della fibra, quest'ultima dovuta alle deformazioni della struttura monitorata. Dalla variazione nel tempo della lunghezza d'onda registrata è possibile risalire all'accelerazione; similmente si potrebbe lavorare in termini di spostamenti, pressioni o forze, anziché sulla base delle deformazioni.

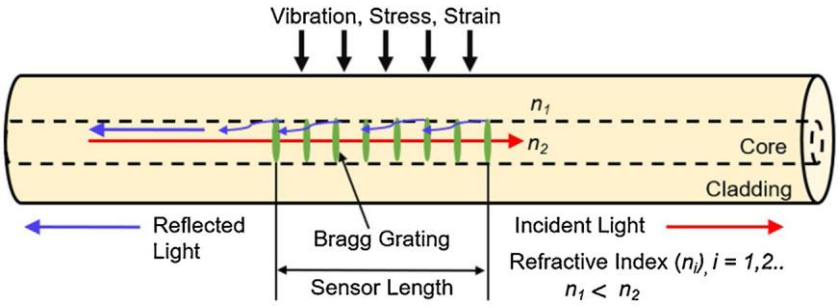


Figura 1.6 – Schematizzazione del reticolo di Bragg

Un esempio di sensori basati sulla fibra ottica è l'accelerometro con sensore in fibra ottica FBG (Figura 1.7). La risposta di questo accelerometro si basa sulla deformazione ε della fibra imposto dallo spostamento di massa inerziale [5].

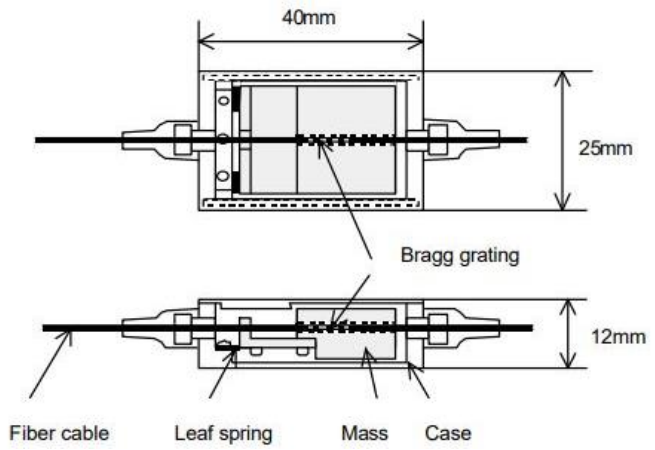


Figura 1.7 – Vista in piante e laterale di un accelerometro con sensore FBG

La struttura dell'accelerometro, schematizzata in (Figura 1.8), è costituita da una massa inerziale, supportata da una trave a sbalzo a forma di L, collegata alla base della struttura da una molla a balestra in acciaio ("leaf spring") e da un elemento FBG. Se esposta a un'accelerazione esterna, la massa inerziale si sposta nella direzione verticale, imponendo una conseguente contrazione/espansione della fibra ottica; da questa deformazione si risale all'accelerazione [5].

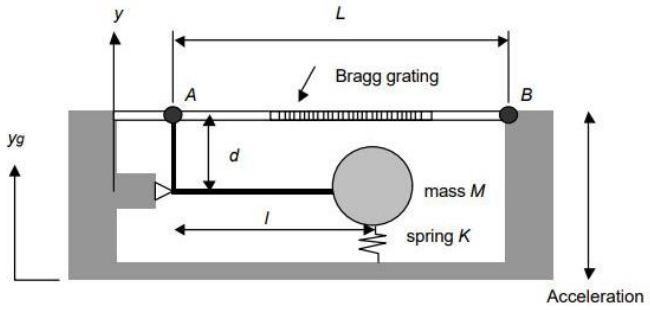


Figura 1.8 – Modello meccanico equivalente di un accelerometro con fibra ottica FBG

Ulteriore esempio di questa tecnologia, è l'Optical Fibre Strain Monitoring System (Sistema di monitoraggio della deformazione con sensori a fibre ottiche) o semplicemente Smart Fibre System (Sistema di fibre intelligenti). Il dispositivo è costituito da un insieme ordinato di sensori a fibre ottiche annegabili in strutture di calcestruzzo (ponti, edifici, ecc), materiali compositi o superfici di rivestimento di strutture in acciaio o calcestruzzo; esso consente di rilevare dall'interno il carico di deformazione con un procedimento assimilabile a quello del sistema nervoso umano (Figura 1.9).

Figura 1.9 – Sensori a fibra ottica fissati alle barre delle armature all'interno degli elementi strutturali

Nell'ambito delle infrastrutture civile come i ponti, un sistema di monitoraggio con tecnologia in fibra ottica è stato applicato al ponte Shaba An'ning River Bridge facente parte della linea ferroviaria Chengdu–Kunming in Cina [6]. Il ponte è composto da tre campate (52+88+52) m ed è realizzato in calcestruzzo armato precompresso con elementi scatolari gettati in opera. Il sistema di monitoraggio è stato installato per monitorare la struttura durante l'intero processo di costruzione degli elementi strutturali a sbalzo. In particolare, utilizzando gli inclinometri con tecnologia in fibra ottica, sono stati ricavati gli spostamenti della struttura durante le fasi costruttive.

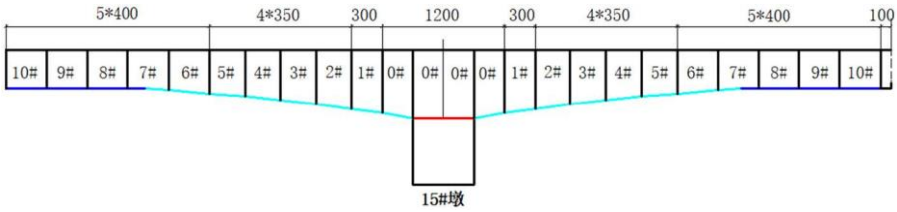


Figura 1.10 – Numerazione dei segmenti

Figura 1.11 – Layout del sistema di monitoraggio

Con riferimento alla fase costruttiva in cui viene introdotta la pretensione fino al blocco #2, si riportano a titolo di esempio gli spostamenti registrati per i blocchi già realizzati. I risultati in Figura 1.12 confermano spostamenti verso l'alto per tutti i segmenti già costruiti ed interessati dall'introduzione della pretensione.

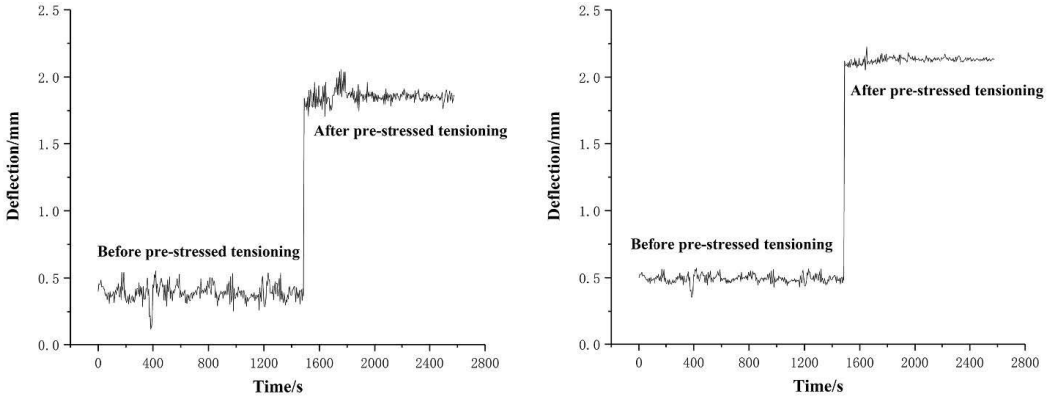


Figura 1.12 – Spostamenti durante l'introduzione della pretensione nel blocco #2

Un'altra applicazione riguardante i ponti è riportata nel lavoro di Dongtao Hu et al. [7]. In questo caso i sensori FBG sono stati impiegati per il monitoraggio dei cavi da sospensione del Tongwamen Bridge.

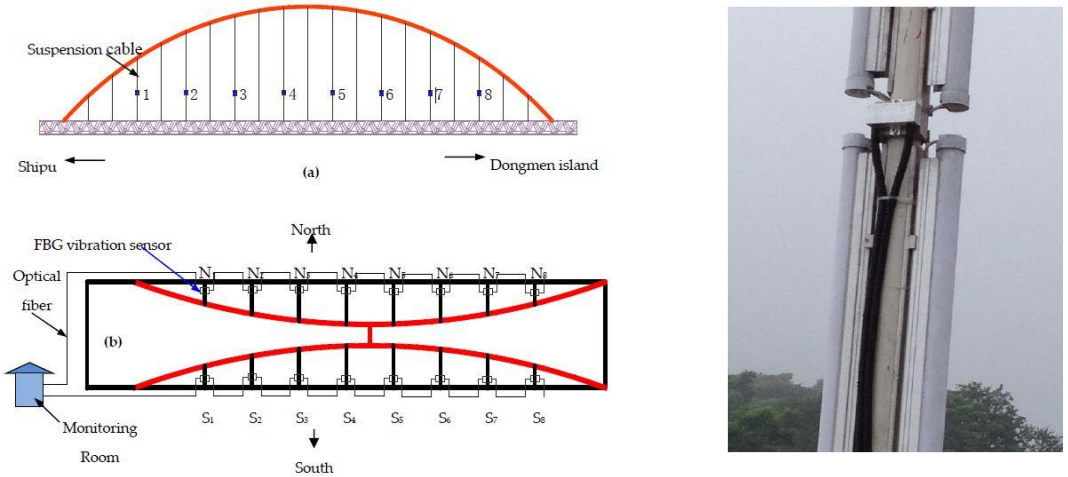


Figura 1.13 – Posizionamento sensori di vibrazione FBG

I sensori FBG sono stati utilizzati per l'individuazione della frequenza di vibrazione dei cavi monitorati, e in accordo con la teoria della vibrazione delle corde, nota il valore della n-esima frequenza, è stato possibile calcolare la forza assiale agente. Il valore di tale forza è direttamente collegato allo stato di salute del generico cavo.

- *Inclinometri*

In generale, il clinometro o inclinometro (detto anche tiltmetro) è lo strumento utilizzato per la misura di angoli di inclinazione (in inglese, "tilt"), di pendenza, di elevazione o di depressione rispetto all'orizzontale.

- *Inclinometri a livella torica*

L'inclinometro a livella torica è uno strumento ottico meccanico che permette la misurazione di piccole variazioni di inclinazione. La struttura interna dello strumento è costituita da un meccanismo ottico/meccanico il quale garantisce una precisione elevatissima di difficile conseguimento con gli altri strumenti di tipo elettrico e automatico. Ulteriore garanzia della bontà del sistema è rappresentata dalla possibilità di eseguire misure dirette e coniugate, consentendo così l'eliminazione di qualsiasi errore sistematico.

Figura 1.14 – Inclinometro a livella torica

Il clinometro a livella torica viene così denominato per la livella che porta al suo interno e le cui rotazioni sono comandate da un sistema di amplificazione meccanica con leve e vite micrometrica Le parti principali, in particolare le piaste di sostegno, sono stabilizzate con ripetuti cicli termici; questo assicura allo strumento una eccezionale stabilità nel tempo. Il clinometro riportato in Figura 1.14, per esempio, ha un campo di misura di ±30' e una precisione di 2" (≃0.01 mrad).

- *Inclinometri MEMS*

L'elemento centrale dell'inclinometro con tecnologia MEMS è un accelerometro il quale misura la forza di reazione vincolare con cui un supporto fisso agisce su una massa di prova per mantenerla legata a sé nel movimento. Dal valore della reazione vincolare si ottiene l'accelerazione, a partire dalla quale si calcolano gli angoli di inclinazione. A tal proposito, si

immagini un accelerometro mono-assiale posto su un piano orizzontale, cioè sul piano perpendicolare alla direzione del vettore rappresentante la forza di gravità. In questa configurazione, l'accelerometro misura un'accelerazione pari a 1g sull'unico asse di misura disponibile. Se l'accelerometro viene ruotato di un certo angolo α rispetto al piano dell'orizzonte, esso misurerà la componente del vettore accelerazione parallela all'asse su cui il sensore effettua misurazione, indicato in Figura 1.15 con A.

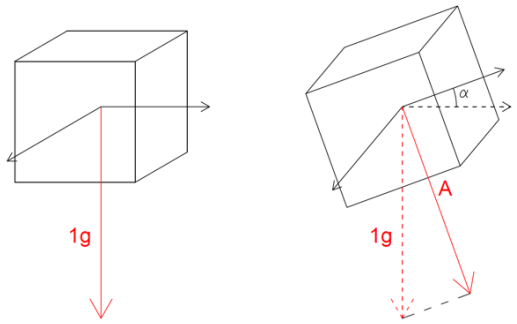


Figura 1.15 – Legame tra accelerazione misurata e inclinazione

Considerando il legame trigonometrico tra i due vettori, si può affermare che:

$$A = 1g \cdot \sin(\alpha)$$

Invertendo tale formula, si ottiene facilmente il valore dell'angolo α:

$$\alpha = \arcsin\left(\frac{A}{1g}\right)$$

Oggigiorno in commercio è disponibile una gran varietà di sensori inclinometrici con tecnologia MEMS. Possono essere analogici o digitali e il sensore può essere monoassiale o biassiale, misurando cosi rispettivamente angoli di inclinazione in una o due direzioni perpendicolari tra loro. La strumentazione è in genere disponibile in un contenitore scatolare per il posizionamento in piano o su parete verticale. Esistono sensori inclinometrici appositamente creati per applicazioni statiche, e altri destinati ad applicazioni dinamiche come per esempio il monitoraggio di fenomeni vibratori. Si riportano di seguito alcuni di questi sensori presenti in commercio:

Figura 1.16 – Bean Air HI-INC (precisione di ±5 x 10^{-2}°)

Figura 1.17 – IFM Electronic JD1121 (precisione di ≤ ±1 x 10^{-2}°)

Un'altra tipologia di inclinometri usata nell'ambito del monitoraggio strutturale è quella degli inclinometri MEMS a barra (Figura 1.18), i quali possono essere impiegati singolarmente per monitorare l'inclinazione relativa tra due punti, oppure possono essere installati in sequenza per controllare, ad esempio, i cedimenti differenziali di una struttura. Nel caso degli inclinometri a barra MEMS, il funzionamento si basa su un sensore posizionato su una barra di alluminio avente una determinata base di misura ("gauge length"), indicata con L in Figura 1.19, in generale pari a 1, 2 o 3 metri. Essa rappresenta la distanza tra i due punti di riferimento tra cui si vuole misurare l'inclinazione.

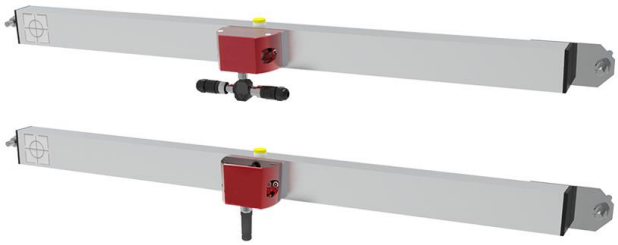


Figura 1.18 – Inclinometri MEMS a barra

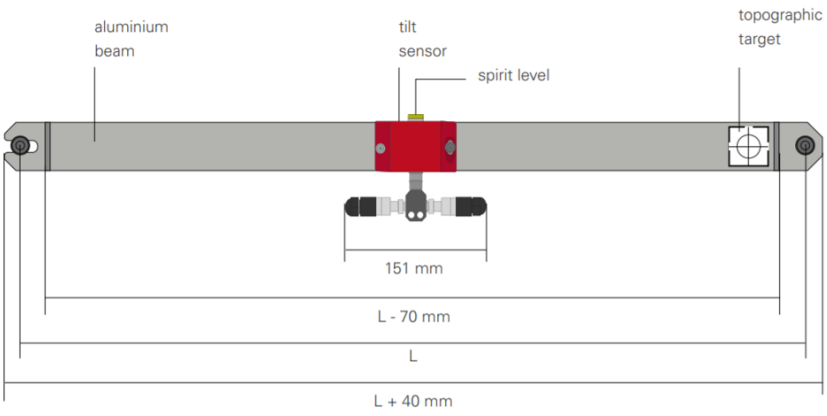


Figura 1.19 - Elementi e dimensioni dell'inclinometro a barra

La precisione degli inclinometri riportati in figura varia, a seconda dal range di misura del dello strumento (±2.5°, ±5°, ±10°), da ±0.008° a ±0.020°; alcuni raggiungono anche precisioni maggiori.

1.4 Monitoraggio strutturale con inclinometri

Gli inclinometri sono stati ampiamente utilizzati in applicazioni industriali nel settore automobilistico, aerospaziale ed elettronico. I primi esempi di inclinometri usati nel campo dell'ingegneria civile risalgono ad applicazioni geotecniche, settore nel quale i movimenti monitorati hanno ordini di grandezza maggiori rispetto a quelli che si verificano in strutture civili come i ponti. Tuttavia, nell'ultimo decennio, le prestazioni e l'accuratezza di questi dispositivi sono state notevolmente migliorate, permettendo così il loro utilizzo anche per il monitoraggio strutturale di infrastrutture civili. Nell'articolo di F. Huseynov et al. [8], è riportata una tabella contenente le principali specifiche tecniche di alcuni inclinometri disponibili in commercio.

Tab. 1.2 – Specifiche tecniche degli inclinometri disponibili in commercio

Model	Manufacturer	Country of Origin	Measurement Range	Precision in degrees	Resolution in degrees	Sampling rate
HI-INC	Bean Air	Germany	±15°	$\pm 5 \times 10^{-2}$°	1×10^{-3}°	100 Hz
DNS	MP SENSOR	Germany	±85°	$\pm 3 \times 10^{-2}$°	3×10^{-3}°	100 Hz
JDI 100	Jewell Instruments	U.S.A	±1°	$\pm 4 \times 10^{-3}$°	1×10^{-4}°	125 Hz
JN2101	IFM Electronic	Germany	±45°	$\pm 1 \times 10^{-2}$°	1×10^{-3}°	20 Hz
ACA2200	RION	Japan	±0.5°	$\pm 3 \times 10^{-3}$°	1×10^{-4}°	20 Hz
ZERO-TRONIC	WYLER AG	Switzerland	±0.5°	$\pm 3.5 \times 10^{-4}$°	1×10^{-4}°	10 Hz
T935	Sherborne Sensors	U.K	±1°	$\pm 4 \times 10^{-4}$°	6×10^{-5}°	10 Hz

Le informazioni tecniche riportate in tabella dimostrano la possibilità di misurare valori di rotazioni molto piccoli inerenti alle strutture dei ponti, con una precisione fino a 3,5 x 10^{-4} gradi usando sensori all'avanguardia.

Diversi sono i casi in cui gli inclinometri sono stati utilizzati in infrastrutture civili come i ponti per comprendere meglio il loro complesso comportamento strutturale durante la vita di servizio ma anche durante le fasi costruttive. Nel lavoro di Glišic´ et al. [9], per esempio, viene riportato il progetto di monitoraggio di un ponte in calcestruzzo armato post teso durante la sua costruzione, l'inserimento della post tensione e nel primo anno di vita, usando inclinometri ed estensimetri. Il lavoro di monitoraggio ha permesso di verificare l'effetto della post tensione e le performance complessiva della struttura [8].

Uno dei vantaggi di questa tecnica di monitoraggio è che essa risulta poco influenzata dalle condizioni ambientali in quanto la strumentazione è installata direttamente sull'impalcato. In questo modo, le misurazioni effettuate non risentono della subsidenza delle pile e della deformazione degli appoggi. Diversa è invece la situazione per altri sensori adoperati per il monitoraggio della medesima tipologia di opere. Le stazioni totali, per esempio, sono ampiamente usate per il controllo delle deformazioni di infrastrutture civili, ma il loro impiego

necessita punti stabili da cui fare le osservazioni. I sensori fotoelettrici (fotocellule) sono in grado di misurare continuamente le deformazioni statiche o dinamiche della struttura, ma di contro sono fortemente influenzati dalle condizioni ambientali esterne (pioggia, nebbia, ecc…) oltre alla necessità di punti di osservazione stabili. La necessità infatti di adeguati punti di osservazione, molto spesso esclude a priori l'utilizzo di un sistema di monitoraggio a favore di un altro; basti pensare a un ponte che attraversa un fiume o il mare.

La tecnica di monitoraggio basata sui dati inclinometrici permette di ottenere l'andamento temporale degli angoli di inclinazione dei punti monitorati e anche la deflessione dell'implacato Attualmente tale tecnica di monitoraggio è largamente applicata, con ottimi risultati in termini di precisione, a ponti autostradali a travata caratterizzati da modeste luci. Per ponti strallati, sospesi, ponti ad arco, e in generale ponti con grandi luci, il monitoraggio della configurazione deformata è più complesso e ancora oggetto di studi e ricerche [10].

L'attuale andamento della ricerca nell'ambito del monitoraggio strutturale mostra un allontanamento dai metodi tradizionali adottati in passato, a favore di sistemi di monitoraggio altamente tecnologici e wireless. Il forte interesse verso questa direzione è spinto dalle caratteristiche tecniche in continua evoluzione dei sensori utilizzati. Secondo quanto pubblicato in [11], i sensori più appropriati al fine del monitoraggio strutturale si sono dimostrati essere gli accelerometri e gli inclinometri. I primi consentono la misura dei cambiamenti causati da danni strutturali nel comportamento dinamico della struttura, e in alcuni casi anche la localizzazione del danno stesso. Gli inclinometri invece permettono di risalire alla deformazione degli elementi strutturali mediante la misura degli angoli di rotazione. Ad oggi, gli "smart sensors" sono considerati gli strumenti migliori per il monitoraggio strutturale. Con il termine "smart sensors" (sensore intelligente) si fa riferimento a un tipo di sensore che ha al suo interno circuiti elettrici capaci, oltre che di rilevare una grandezza di tipo fisico, chimico o elettrico, anche di elaborare le informazioni e di trasmetterle all'esterno sotto forma di segnale digitale. [11].

Un esempio di moderno sistema di monitoraggio basato sui dati inclinometrici è quello installato sul Xinguang Bridge in Guangzhou (Figura 1.20) [12]. Si tratta di un ponte ad arco con tre campate continue della lunghezza di 177+428+177 m in cui si vuole monitorare la deformazione degli arghi metallici in seguito al traffico veicolare (Figura 1.21).

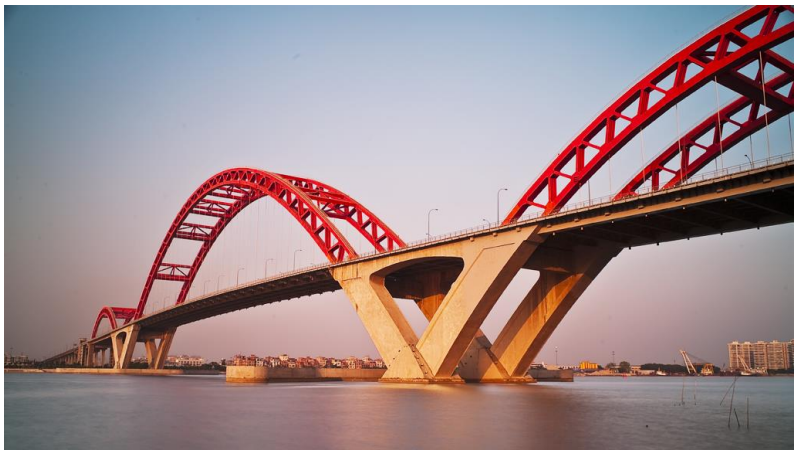


Figura 1.20 – Vista del Xinguang Bridge

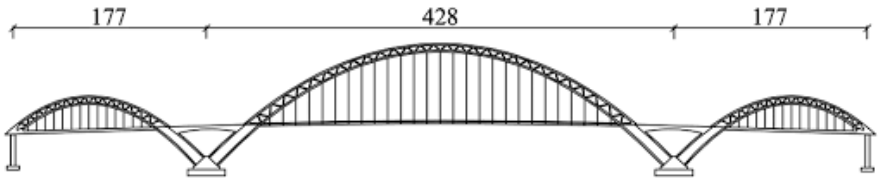


Figura 1.21 – Campate del Xinguang Bridge

Ogni punto di misura consiste in una base metallica e da un inclinometro, come mostrato in sensore (Figura 1.22). La base è imbullonata all'arco e il a sua volta è imbullonato alla base in posizione orizzontale.

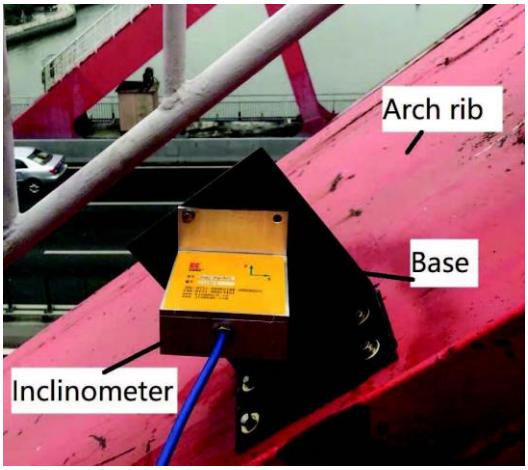


Figura 1.22 – Punto di misura

Il monitoraggio strutturale dell'opera analizzata si basa sui valori di angolo misurati dagli inclinometri, dai quali è possibile ricavare i valori di deflessione e confrontarli con i risultati del modello numerico. Il sistema è composto da una rete di inclinometri posizionati in diversi punti della struttura. I dati registrati vengono raccolti da un sistema di acquisizione costituito dai cavi e dal modulo di trasmissione. I dati raccolti vengono poi trasmessi, tramite reti

wireless, ad un sistema di trattamento e monitoraggio dei dati composto da un software per l'acquisizione e da un ulteriore software per l'analisi dei dati ricevuti. Entrambi i softwares sono installati ed operano su una piattaforma cloud.

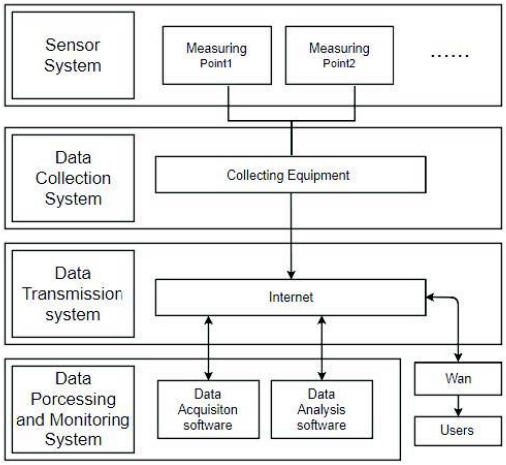


Figura 1.23 – Organizzazione generale del sistema di monitoraggio

Il sistema di elaborazione dei dati, ultimo blocco dello schema in Figura 1.23, effettua una serie di test statistici e delle operazioni finalizzate alla pulizia del segnale proveniente dalla struttura

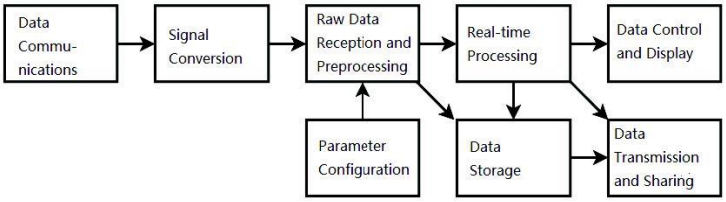


Figura 1.24 – Sistema di elaborazione dei dati

Tale architettura di monitoraggio strutturale permette all'utente il monitoraggio dell'arco in qualsiasi momento utilizzando un collegamento internet.

CAPITOLO 2
Python: cenni di programmazione orientata agli oggetti

La programmazione orientata agli oggetti (Object Oriented Programming) è uno stile fondamentale di programmazione composto da un insieme di strumenti concettuali forniti da un linguaggio di programmazione.

Nella pratica industriale dello sviluppo del software esistono molteplici paradigmi di programmazione con i relativi linguaggi utilizzabili. In questa tesi è stato utilizzato il linguaggio di programmazione Python, linguaggio multi-paradigma che supporta la programmazione orientata agli oggetti, la programmazione strutturale e molte caratteristiche della programmazione funzionale e riflessione.

Ideato da Guido van Rossum e rilasciato pubblicamente per la prima volta nel 1991, Python è stato protagonista di un'enorme diffusione in tutto il mondo. E' completamente gratuito e può essere liberamente modificato e ridistribuito secondo le regole di una licenza pienamente open-source. L'esser pseudointerpretato rende Python un linguaggio portabile; una volta scritto un codice sorgente, esso può essere interpretato ed eseguito sulla gran parte dei sistemi operativi utilizzati (Windows, Linux/Unix, Mac OS X, OS/2) sui quali deve essere installata la versione corretta dell'interprete [13].

La programmazione orientata agli oggetti è basata sul concetto di oggetti, ovvero strutture dati definite in una zona circoscritta del codice sorgente chiamata classe e in grado di interagire gli uni con gli altri. In fase di esecuzione del codice, le classi vengono invocate per generare oggetti software che avranno le stesse proprietà (attributi) e capacità (metodi) definite all'interno della rispettiva classe.

Un linguaggio di programmazione ad oggetti è definito tale quando permette di implementare tre meccanismi mediante la sintassi nativa del linguaggio:

- *Incapsulamento*: gli attributi e i metodi di un oggetto sono accessibili ai metodi dell'oggetto stesso ma non sono visibili ai client. Per utilizzare un oggetto, infatti, l'utente deve solo comprendere l'interfaccia, ovvero il sottoinsieme di metodi che appartengono all'oggetto utilizzato. In tal modo vengono date delle funzionalità all'utente nascondendo però i dettagli relativi all'implementazione

- *Ereditarietà*: è possibile definire una nuova classe come versione modificata di una classe già esistente; la nuova classe erediterà il comportamento (attributi e metodi) della classe base a cui sarà possibile aggiungere delle funzionalità proprie.

- *Polimorfismo:* indica il fatto che lo stesso codice eseguibile può essere utilizzato con istanze di classi diverse, aventi una classe madre comune. In altri termini, consente di invocare operazioni su un oggetto, pur non essendo nota a tempo di compilazione la classe a cui fa riferimento l'oggetto stesso. In questo modo, la versione del metodo da eseguire viene scelta sulla base del tipo di oggetto effettivamente contenuto in una variabile a tempo di esecuzione, anziché a tempo di compilazione.

Rispetto alla programmazione tradizionale, la programmazione orientata agli oggetti offre numerosi vantaggi in termini di:

- facile gestione e manutenzione di progetti di grande dimensione;
- modularità e riuso grazie all'organizzazione del codice sotto forma di classi;
- estendibilità, in quanto il meccanismo di polimorfismo rende minime le modifiche necessarie quando si vuole estendere il codice con l'aggiunta di nuove funzioni.

2.1 Classi e oggetti

La classe è la rappresentazione astratta di una categoria di oggetti implementata in un software. Essa rappresenta un "modello generico" a partire dal quale è possibile creare degli oggetti. La creazione di un oggetto a partire da una classe prende il nome di istanziazione e avviene invocando la classe stessa. L'oggetto creato è chiamato anche istanza della classe.

La classe definisce al proprio interno lo stato e il comportamento di qualsiasi oggetto da lei derivato. In particolare, lo stato rappresenta le caratteristiche dell'oggetto ed è definito mediante i cosiddetti attributi (detti anche variabili di istanza), il comportamento invece rappresenta le operazioni eseguibili sull'oggetto ed è definito da blocchi di codice chiamati metodi.

La creazione di una classe si effettua mediante il comando *class* seguito dal nome che si vuole assegnare alla classe:

```
class nomeclasse():
```

Creata la classe, è possibile assegnare ad essa i vari attributi (variabili di istanza) tramite un costruttore. In informatica, specialmente nella programmazione orientata agli oggetti, i costruttori indicano dei metodi associati alle classi che hanno lo scopo di inizializzare le variabili di istanza. Nelle classi di Python viene usato il metodo costruttore *__init__()*. Il primo parametro che il metodo riceve in input è sempre l'indicativo dell'oggetto che verrà creato (l'istanza della classe) ed è chiamato per convenzione *self*. Il termine self rappresenta quindi una referenza a ciascun oggetto creato dalla classe, e in questo modo il metodo __init__ inizializza e attiva le varie proprietà a ciascuna istanza. Per tale motivo, gli attributi di una classe sono chiamati anche variabili di istanza.

Esempio:

```
class Persona ( ):
def __init__(self, Nome, Cognome):
        self.Nome = Nome
    self.Cognome = Cognome
```

2.2 Metodi di istanza e attributi

I metodi di istanza sono delle funzioni interne alla classe e definiscono il comportamento della classe e degli oggetti da essa istanziati. Trattandosi di metodi, essi vengono creati utilizzando la seguente sintassi:

```
def metodo (self, argomento1, argomento2, … ):
        blocco istruzioni
    return
```

Il metodo è definito mediante il comando *def* seguito dal nome del metodo. Gli argomenti della funzione invece sono indicati tra parentesi dopo il nome, separati tra loro da una virgola. Anche per i metodi di istanza, il primo parametro che viene inserito è il termine *self*; il quale permette di creare un collegamento diretto con ciascuna istanza della classe e di utilizzare il metodo su ciascun oggetto creato.

La funzione prende in input i vari argomenti passati durante la chiamata del metodo, elabora il blocco di istruzioni indentato al suo interno, poi eventualmente restituisce il risultato dell'elaborazione tramite l'istruzione *return*. Nel linguaggio python, l'istruzione *return* viene inserita dentro la definizione di una funzione. Essa permette di trasferire i dati di ritorno, ossia quelli dalla funzione allo script che la invoca. Infatti, i due ambienti di lavoro sono diversi e per metterli in comunicazione occorre utilizzare l'istruzione *return*. Tuttavia, l'utilizzo di tale comando non è obbligatorio; infatti, in mancanza di tale istruzione la funzione termina l'elaborazione dopo l'ultima riga di codice indentato e non restituisce nulla allo script.

Dopo aver creato un oggetto si possono usare o modificare i valori degli attributi ed eseguire delle operazioni richiamandone i metodi. Ogni oggetto è memorizzato mediante un riferimento ("nome oggetto"), ovvero una variabile. Attraverso tale riferimento si possono richiamare i suoi attributi e metodi utilizzando la notazione puntata.

Si può fare riferimento ad un attributo con:

```
nomeoggetto.attributo
```

Si può richiamare un metodo con:

```
nomeoggetto.metodo(argomento1, argomento2, …)
```

Nel caso del metodo, è necessario specificare dentro parentesi tonda i parametri necessari definiti durante la fase di creazione del metodo.

CAPITOLO 3
PyNiteFEA

PyNiteFEA è una libreria per il linguaggio di programmazione Python che permette di effettuare analisi statiche elastiche lineari di telai 3D. E' un pacchetto open-source disponibile su PyPI (Python Package Index) [14] la più grande e ufficiale repository di pacchetti Python. Inoltre, è in corso un continuo aggiornamento grazie alla collaborazione di diversi utenti attraverso la piattaforma di sviluppo software GitHub [15].

Il pacchetto PyNiteFEA, realizzato secondo la programmazione ad oggetti, è costituito da diversi files python chiamati moduli e contenuti nella directory "PyNite" generata dopo l'installazione del pacchetto. Tali moduli contengono la definizione di classi e metodi e sono in grado di interagire tra loro. Alcuni di questi moduli definiscono al loro interno una classe con in relativi attributi e metodi; altri invece contengono solamente la definizione di metodi.

L'utilizzo del pacchetto da parte dell'utente consiste nella scrittura di istruzioni nel linguaggio Python che permettono di invocare le classi e i metodi definiti nei vari moduli. A sua volta, i metodi necessitano l'inserimento di argomenti indispensabili al corretto funzionamento del codice. Tutto ciò rappresenta l'input del programma e permette all'utente la costruzione del generico modello agli elementi finiti. Segue poi l'analisi del modello e infine la valutazione dei risultati. Si ottengono così gli spostamenti nodali, le caratteristiche della sollecitazione, le deformate di ogni elemento e le reazioni vincolari. Tutto ciò costituisce l'output del programma, che insieme agli input, sarà spiegato dettagliatamente nel capitolo 3.

Lo schema rappresentato in Figura 3.1 mostra l'organizzazione dei vari moduli che compongono la libreria PyNiteFEA. Ciascuno di essi verrà trattato nel paragrafo successivo.

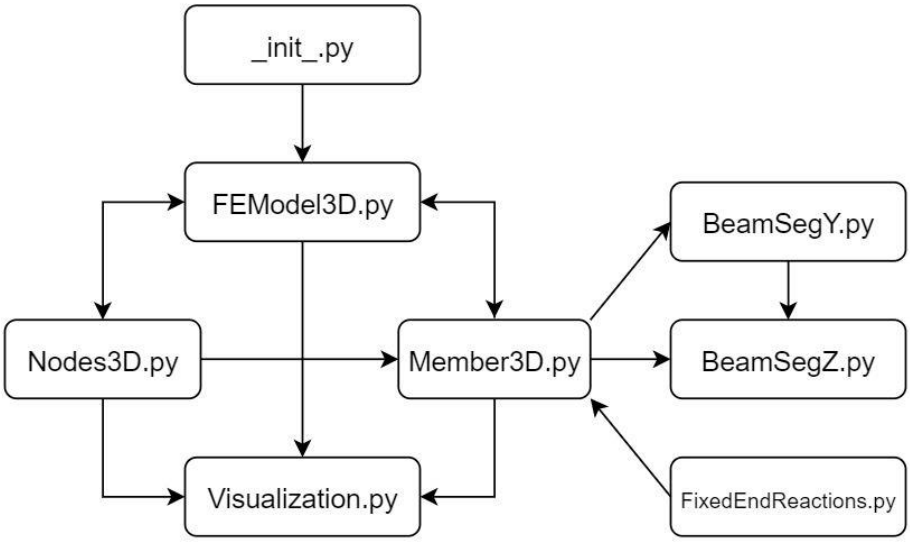


Figura 3.1 - PyNiteFEA

3.1 Moduli

Si elencano di seguito i moduli presenti e la loro funzione all'interno del pacchetto PyNiteFEA:

- __init__.py

Il modulo __init__.py rappresenta un file speciale indispensabile affinché la directory contenente i moduli sia considerata dall'interprete python come un pacchetto. In assenza di tale file, Python vede ciascuno dei moduli contenuti nel pacchetto come file separati e indipendenti e l'importazione del generico modulo fallisce. Il modulo __init__.py è eseguito automaticamente ogni volta che il pacchetto o i moduli all'interno vengono importati. Esso può essere privo di contenuto, ma in questo caso contiene le seguenti istruzioni di inizializzazione:

```
from PyNite.FEModel3D import FEModel3D
```

Tali istruzioni permettono di importare dal modulo FEModel3D.py contenuto nel pacchetto PyNite la classe FEModel3D. Per una maggiore comprensione questa riga di codice può essere letta anche nel seguente modo (in questo caso il modulo e la classe hanno lo stesso nome):

```
from Package.Module import Class
```

L'esecuzione di queste istruzioni permette all'utente di importare direttamente la classe FEModel3D invece di richiamarla ogni volta con la notazione Module.Class.

- FEModel3D.py

Il modulo FEModel3D.py definisce una classe rappresentante un modello 3D agli elementi finiti. Al suo interno sono contenuti diversi metodi necessari per la costruzione del modello, per la risoluzione del problema statico e per la lettura e visualizzazione grafica dei risultati. Si elencano di seguito tutti i metodi:

- *AddNode*: crea un nodo
- *AddAuxNode*: crea un nodo ausiliario
- *AddMember*: crea un elemento trave
- *RemoveNode*: elimina un nodo
- *RemoveMember*: elimina un elemento trave
- *DefineSupport*: definisce le condizioni di vincolo di un nodo
- *DefineReleases*: definisce il rilascio di specifici gradi di liberta
- *AddNodeDisplacement*: definisce uno spostamento nodale imposto
- *AddNodeLoad*: definisce un carico nodale
- *AddMemberPtLoad*: definisce un carico concentrato in campata
- *AddMemberDistLoad*: definisce un carico distribuito trapezoidale in campata
- *ClearLoads*: elimina tutti i carichi del modello
- *K*: assembla la matrice di rigidezza del sistema espressa nel sistema di riferimento globale
- *FER*: assembla il vettore dei carichi di incastro perfetto del sistema espresso nel sistema di riferimento globale
- *P*: assembla il vettore dei carichi nodali del sistema espresso nel sistema di riferimento globale
- *Analyze*: risolve il sistema di equazioni ottenendo gli spostamenti nodali incogniti
- *D*: restituisce il vettore degli spostamenti nodali calcolati espressi nel sistema di riferimento globale
- *GetNod*e: seleziona un nodo
- *GetAuxNode*: seleziona un nodo ausiliario
- *GetMember*: seleziona un elemento trave

- Member3D.py

Il modulo Member3D.py definisce una classe rappresentate il generico elemento trave 3D appartenente al modello agli elementi finiti. Al suo interno sono definiti diversi metodi necessari al calcolo della matrice di rigidezza, delle caratteristiche di sollecitazione, degli spostamenti e per la lettura e visualizzazione grafica dei risultati. Si elencano di seguito tutti i metodi:

- *L*: calcola la lunghezza dell'elemento trave
- *k*: costruisce la matrice di rigidezza dell'elemento trave espressa nel sistema di riferimento locale
- *fer*: costruisce il vettore dei carichi di incastro perfetto dell'elemento trave espresso nel sistema di riferimento locale
- *f*: calcola il vettore delle forze di estremità dell'elemento trave espresso nel sistema di riferimento locale
- *d*: restituisce il vettore degli spostamenti nodali dell'elemento trave (spostamenti e rotazioni) espresso nel sistema di riferimento locale
- *D*: restituisce il vettore degli spostamenti nodali dell'elemento trave (spostamenti e rotazioni) espresso nel sistema di riferimento globale
- *T*: costruisce la matrice di trasformazione
- *K*: calcola la matrice di rigidezza dell'elemento trave ruotata nel sistema di riferimento globale
- *F*: calcola il vettore delle forze di estremità dell'elemento trave espresso nel sistema di riferimento globale
- *FER:* calcola il vettore dei carichi di incastro perfetto dell'elemento trave ruotato nel sistema di riferimento globale
- *Axial, MaxAxial, MinAxial:* calcolo della sollecitazione di sforzo normale
- *Shear, MaxShear, MinShear:* calcolo della sollecitazione di taglio
- *Moment, MaxMoment, MinMoment:* calcolo della sollecitazione di momento flettente
- *Torsion, MaxTorsion, MinTorsion:* calcolo della sollecitazione di momento torcente
- *Deflection, MaxDeflection, MinDeflection:* calcolo degli spostamenti trasversali
- *PlotShear, PlotMoment, PlotAxial, PlotTorsion, PlotDeflection:* costruzione dei diagrammi delle caratteristiche della sollecitazione e delle deformate

- *RelativeDeflection:* calcolo della deformata trasversale relativa
- *PlotRelativeDeflection:* costruzione diagramma delle deformate trasversali relative
- *SegmentMember:* divide l'elemento trave in tratti matematicamente continui per ogni direzione del sistema di riferimento locale

- Nodes3D.py

Il modulo Node3D.py definisce una classe rappresentate il generico nodo 3D appartenente al modello agli elementi finiti.

- BeamSegZ.py e BeamSegY.py

I moduli BeamSegZ.py e BeamSegY.py definiscono una classe rappresentante il segmento di trave matematicamente continuo ("beam segment") e diversi metodi necessari per il calcolo delle caratteristiche di sollecitazione e degli spostamenti all'interno del segmento di trave.

Per segmento di trave matematicamente continuo si intende il tratto di elemento compreso tra due discontinuità. Per maggiore chiarezza si riporta di seguito un esempio (Figura 3.2); dato l'elemento di trave raffigurato e definiti i vincoli e i carichi applicati, i "beam segments" sono i tratti 1, 2, 3 e 4.

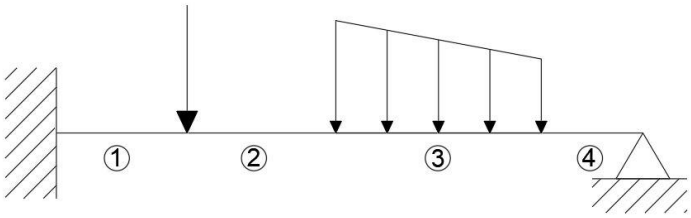


Figura 3.2 - Beam segments

Il modulo BeamSegZ.py definisce una classe con le formulazioni relative al comportamento assiale, torsionale e flessionale attorno l'asse z locale dell'elemento. In seguito alla convenzione adottata per le sollecitazioni interne, le formulazioni matematiche del comportamento flessionale attorno l'asse y locale si differenziano in termini di segni; per tale motivo esse sono implementate in un modulo distinto, chiamato appunto BeamSegY.py. Tale modulo sfrutta il principio dell'ereditarietà, definendo una sottoclasse a partire dalla classe creata nel modulo BeamSegZ.py dalla quale vengono ereditate tutte le proprietà e i metodi. Dopodiché, vengono

modificati solo quei metodi influenzati dalla convenzione. Per maggiori dettagli sulla convenzione positiva si rimanda al paragrafo 4.5.1.

Si elencano di seguito tutti i metodi:

- *Shear, MaxShear, MinShear:* calcolo della sollecitazione di taglio
- *Moment, MaxMoment, MinMoment:* calcolo della sollecitazione di momento flettente
- *Axial, MaxAxial, MinAxial:* calcolo della sollecitazione di sforzo normale
- *Torsion, MaxTorsion, MinTorsion:* calcolo della sollecitazione di momento torcente
- *Deflection:* calcolo degli spostamenti trasversali
- *Slope:* calcolo della rotazione

- FixedEndReactions.py

Il modulo FixedEndReactions.py definisce i metodi per il calcolo del vettore dei carichi di incastro perfetto espresso nel sistema di riferimento locale dell'elemento e per ciascuna delle condizioni di carico applicate. Si elencano i metodi per le varie tipologie di carico:

- *FER_PtLoad:* forza trasversale concentrata
- *FER_LinLoad:* carico trasversale distribuito trapezoidale
- *FER_AxialPtLoad:* forza assiale concentrata
- *FER_AxialLinLoad:* carico assiale distribuito trapezoidale
- *FER_Moment:* momento flettente concentrato
- *FER_Torque:* momento torcente concentrato

- Visualization.py

Il modulo Visualization.py definisce delle classi rappresentanti i nodi e gli elementi trave in configurazione deformata e non, convertiti in entità visibili tridimensionalmente in un ambiente virtuale navigabile. Queste classi vengono usate dai seguenti metodi per la visualizzazione del modello e della deformata:

- *RenderModel*: visualizzazione geometrica del modello agli elementi finiti
- *DeformedShape*: visualizzazione della deformata

Il funzionamento del pacchetto PyNiteFEA dipende anche da altre librerie Python ("dependencies") utilizzate all'interno dei vari moduli. Esse vengono scaricate e installate automaticamente durante l'installazione del pacchetto PyNiteFEA:

- Numpy: per i calcoli matriciali;
- Matplotlib: per plottare i diagrammi dei vari elementi;
- Vtk: per la visualizzazione 3D

3.2 Installazione

L'installazione di una libreria si effettua tramite pip ("Pip Installs Packages").

Pip rappresenta il gestore di pacchetti ("package manager") ufficiale di Python ed è uno strumento che permette di cercare, scaricare, installare, aggiornare e rimuovere packages che si trovano su PyPI [14]. Di fatto esso costituisce un modulo della libreria standard di Python, e in quanto tale, viene installato direttamente con l'installazione del linguaggio di programmazione. Ciò avviene nel caso di versioni di Python superiori alla 3.4 o alla 2.7.9, mentre per le altre versioni pip va installato separatamente tramite un apposito script [16].

Pip viene invocato tramite la shell standard del sistema operativo (cmd.exe in Windows) e utilizzando gli appositi comandi. In presenza di connessione, pip cercherà su PyPI [14] la versione più aggiornata del pacchetto desiderato, scaricherà eventuali dependencies richieste dal package e installerà tutto. Si elencano di seguito alcuni dei comandi più comuni:

Cercare	→	`pip search <package name>`
Installazione	→	`pip install <package name>`
Disinstallazione	→	`pip uninstall <package name>`
Aggiornamento	→	`pip install --upgrade <package name>`

In assenza di connessione alla rete internet non è possibile utilizzare pip per cercare il pacchetto desiderato su PyPI [14]; in questo caso è necessario scaricarlo preliminarmente tramite browser e installarlo localmente tramite il package manager pip.

Per ulteriori dettagli si rimanda alla consultazione online della documentazione ufficiale di pip [16].

3.3 Unità di misura

All'interno del pacchetto PyNiteFEA non è implementato un determinato sistema di unità di misura. Spetta all'utente scegliere delle unità di misura e rimanere coerente con esso nell'introduzione di grandezze fisiche derivate.

Esempio:

$$\text{lunghezza [mm], forza[N]} \rightarrow \text{tensione [N/mm}^2\text{]}$$

3.4 Sistemi di riferimento

In ogni struttura si possono identificare due diversi sistemi di coordinate per descrivere la posizione dei nodi, le direzioni dei carichi membranali e nodali e la direzione degli spostamenti:

- Sistema di coordinate globale
- Sistema di coordinate locali

Nella fase di analisi si passa dal sistema locale a quello globale tramite una operazione matriciale.

3.4.1 Sistema di riferimento globale

Il sistema di coordinate globali *X, Y* e *Z* (Figura 3.3) è un sistema di coordinate tridimensionale, ortogonale e destrorso.

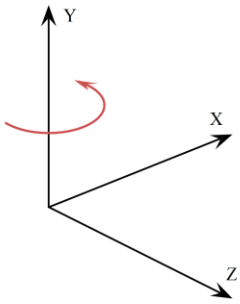


Figura 3.3 - Sistema di riferimento globale

La localizzazione all'interno del sistema di coordinate globali viene specificata tramite le variabili X, Y e Z.

3.4.2 Sistema di riferimento locale dell'elemento trave

Il sistema di riferimento locale dell'elemento trave (Figura 3.4) è anch'esso un sistema di coordinate tridimensionale, ortogonale e destrorso. I tre assi, denominati *x, y* e *z* sono mutuamente ortogonali e soddisfano la regola della mano destra. L'asse locale *x* coincide sempre con l'asse longitudinale dell'elemento avente la direzione positiva diretta dall'estremità i all'estremità j. Gli altri due assi *y* e *z* giacciono nel piano perpendicolare all'elemento e il loro orientamento è specificato dall'utente.

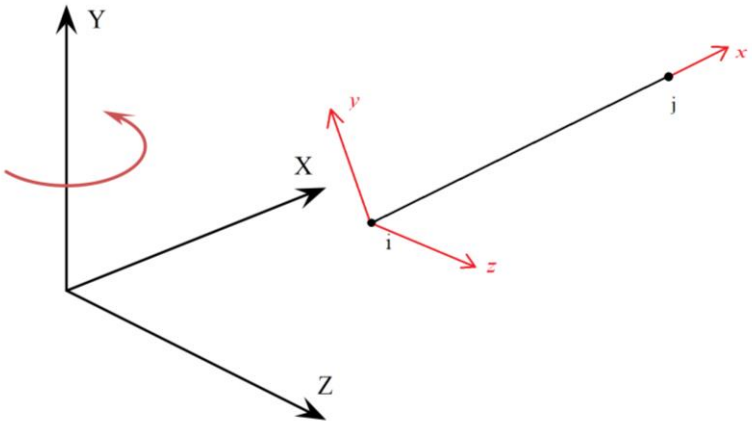


Figura 3.4 - Sistema di riferimento locale dell'elemento trave

L'orientamento del sistema di coordinate locali per il generico elemento avviene tramite l'utilizzo di un nodo ausiliario inserito opportunamente dall'utente durante la fase di costruzione del modello. Tale nodo fornisce all'utente la possibilità di scegliere l'orientamento dell'asse locale *z*. Fissati gli assi locali *x* e *z*, l'algoritmo definisce automaticamente il rimanente asse *y* in modo tale da ottenere un sistema di riferimento destrorso.

Affinché il nodo ausiliario inserito sia efficace, esso non deve essere allineato con l'elemento. Inoltre, il vettore che collega il nodo i e il nodo ausiliario deve essere ortogonale all'asse *x* locale dell'elemento (Figura 3.5).

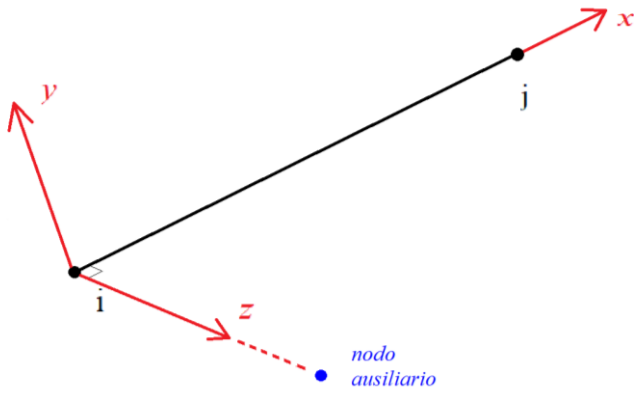


Figura 3.5 - Esempio di nodo ausiliario correttamente posizionato

Nel caso in cui, in seguito a un errato posizionamento del nodo ausiliario tale condizione non è rispettata, l'algoritmo interviene garantendo la condizioni di ortogonalità tra gli assi *x* e *z* locali; per maggiori dettagli si rimanda al calcolo della matrice di trasformazione (§ 3.4.3)

3.4.3 Matrice di trasformazione

La matrice di trasformazione [T] è una matrice quadrata di dimensione 12x12 e contiene i coseni direttori di ciascuno degli assi locali *x, y* e *z* rispetto agli assi globali X, Y e Z. Essa permette il passaggio dal sistema di riferimento locale al sistema di riferimento globale (e viceversa)

I coseni direttori dell'asse locale *x* vengono calcolati tramite le coordinate globali del nodo iniziale e finale del generico elemento:

$$x = [\lambda_{xx}, \lambda_{xy}, \lambda_{xz}]$$

$$\lambda_{xx} = cos\phi_{xx} = \frac{(X_j - X_i)}{L_{ij}} \qquad \lambda_{xy} = cos\phi_{xy} = \frac{(Y_j - Y_i)}{L_{ij}} \qquad \lambda_{xz} = cos\phi_{xz} = \frac{(Z_j - Z_i)}{L_{ij}}$$

dove L_{ij} è la lunghezza dell'elemento. Così facendo l'asse locale *x* sarà sempre allineato con l'elemento.

I coseni direttori dell'asse locale *z* sono determinati usando le coordinate globali del nodo i e del nodo ausiliario p:

$$z = [\lambda_{zx}, \lambda_{zy}, \lambda_{zz}]$$

$$\lambda_{zx} = cos\phi_{zx} = \frac{(X_p - X_i)}{L_{ip}} \qquad \lambda_{zy} = cos\phi_{zy} = \frac{(Y_p - Y_i)}{L_{ip}} \qquad \lambda_{zz} = cos\phi_{zz} = \frac{(Z_p - Z_i)}{L_{ip}}$$

dove L_{ip} è la lunghezza del segmento compreso tra il nodo iniziale *i* dell'elemento e il nodo ausiliario p. Infine, i coseni direttori dell'asse locale *y* si ottengono calcolando il vettore y come prodotto vettoriale tra le direzioni degli assi locali *z* e *x* e dividendo poi ciascuna componente del nuovo vettore per la sua lunghezza:

$$\vec{y} = \vec{z} \times \vec{x}$$

$$\lambda_{yx} = cos\phi_{yx} = \frac{y_x}{\|y\|} \qquad \lambda_{yy} = cos\phi_{yy} = \frac{y_y}{\|y\|} \qquad \lambda_{yz} = cos\phi_{yz} = \frac{y_z}{\|y\|}$$

Quando il posizionamento del nodo ausiliario genera un vettore che collega il nodo i e il nodo ausiliario p non ortogonale all'asse locale *x* (Figura 3.6), vengono eseguite ulteriori operazioni. Esse vanno a forzare la condizione di ortogonalità richiesta calcolando un nuovo vettore *z'* come prodotto vettoriale tra l'asse locale *x* e l'asse locale *y* calcolato precedentemente. Così facendo il nuovo asse *z'* sarà sicuramente ortogonale all'asse locale *x* in quanto derivante da un prodotto vettoriale.

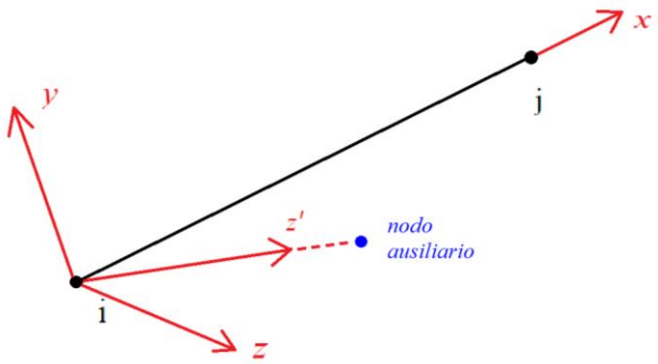


Figura 3.6 - Nodo ausiliario mal posizionato

Si calcolano poi i coseni direttori del nuovo asse *z'* dividendo ciascuna componente del nuovo vettore per la sua lunghezza:

$$\vec{z'} = \vec{x} \times \vec{y}$$

$$\lambda_{zx} = cos\phi_{zx} = \frac{z'_x}{\|z'\|} \qquad \lambda_{zy} = cos\phi_{zy} = \frac{z'_y}{\|z'\|} \qquad \lambda_{zz} = cos\phi_{zz} = \frac{z'_z}{\|z'\|}$$

Nel caso di corretto posizionamento del nodo ausiliario gli assi z e z' saranno coincidenti e anche i rispettivi coseni direttori.

Calcolati i coseni direttori dei vari assi, è possibile costruire la matrice di trasformazione [T] del generico elemento trave di dimensione 12x12:

$$[T] = \begin{bmatrix} \lambda_{xx} & \lambda_{xy} & \lambda_{xz} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \lambda_{yx} & \lambda_{yy} & \lambda_{yz} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \lambda_{zx} & \lambda_{zy} & \lambda_{zz} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \lambda_{xx} & \lambda_{xy} & \lambda_{xz} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \lambda_{yx} & \lambda_{yy} & \lambda_{yz} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \lambda_{zx} & \lambda_{zy} & \lambda_{zz} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \lambda_{xx} & \lambda_{xy} & \lambda_{xz} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \lambda_{yx} & \lambda_{yy} & \lambda_{yz} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \lambda_{zx} & \lambda_{zy} & \lambda_{zz} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda_{xx} & \lambda_{xy} & \lambda_{xz} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda_{yx} & \lambda_{yy} & \lambda_{yz} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda_{zx} & \lambda_{zy} & \lambda_{zz} \end{bmatrix}$$

CAPITOLO 4

Costruzione di un nuovo modello

La costruzione di un nuovo modello avviene tramite la scrittura di un programma, ovvero una sequenza di istruzioni python inserite dall'utente. Per creare un nuovo progetto occorre avviare la shell di Python (IDLE) e creare un nuovo file (Figura 4.1):

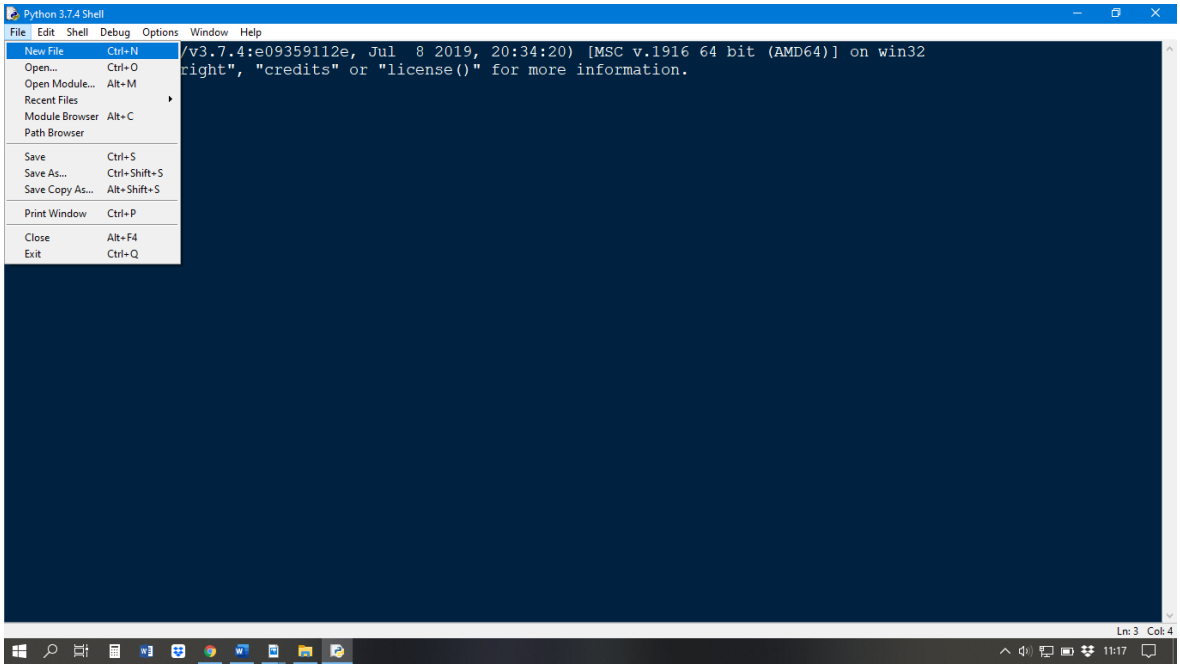


Figura 4.1 - Nuovo modello Python

4.1 Importazione moduli

Per l'utilizzo di PyNiteFEA è necessario importare all'interno del nuovo file python due dei moduli che compongono il pacchetto:

- FEModel3D.py
- Visualization.py

Il modulo FEModel3D.py permette di costruire il modello della struttura ed è in grado di comunicare con tutti agli altri moduli che costituiscono il pacchetto PyNiteFEA. Il modulo

Visualization.py invece permette la visualizzazione 3D navigabile del modello. Nel caso in cui non si voglia usufruire della visualizzazione 3D, si può prescindere dall'importazione del modulo Visualization.py.

L'importazione si effettua con le seguenti righe di codice:

```
from PyNite import FEModel3D
from PyNite import Visualization
```

Con queste istruzioni, dal pacchetto PyNite installato precedentemente, verranno importati i moduli specificati, sarà possibile utilizzare le classi definite nei moduli e richiamare i metodi su qualsiasi oggetto definito.

4.2 Costruzione del modello

La costruzione del modello di un generico telaio 3D necessita la definizione di nodi, vincoli, elementi e proprietà della sezione. Tutto ciò è reso possibile dai metodi appartenenti al modulo FEModel3D.py, all'interno del quale è definita la classe FEModel3D.

In accordo con il paradigma della programmazione orientata agli oggetti, è necessario definire all'inizio del progetto una istanza della classe FEModel3D dalla quale sarà possibile richiamare i vari metodi durante le fasi di costruzione del modello. Il nome da assegnare all'istanza è arbitrario ed è una scelta dell'utente:

```
nomemodello = FEModel3D()
```

Si noti che tale riga di comando è sufficiente all'istanziazione della classe FEModel3D in seguito alle istruzioni salvate nel modulo __init__.py illustrato all'inizio di questo capitolo.

Definita l'istanza, il generico metodo viene richiamato tramite la notazione puntata, ovvero scrivendo il nome dell'istanza e il nome del metodo separati da un punto e specificando dentro parentesi gli attributi necessari (vale anche per qualsiasi altra classe diversa da FEModel3D):

```
nomemodello.metodo ( attributo1, attributo2, … )
```

Gli attributi da specificare si differenziano a seconda del metodo richiamato e sono illustrati dettagliatamente nei paragrafi successivi.

4.2.1 Nodi

Il nodo viene creato tramite la funzione *AddNode* definita all'interno della classe FEModel3D e richiamata nel programma con la seguente sintassi:

```
nomemodello.AddNode (Name, X, Y, Z)
```

Il nodo fa riferimento al sistema di coordinate globali (§3.4.1). Esso è usato per definire i gradi di libertà, i vincoli esterni, i carichi sul nodo e per interpretare gli output

Tale funzione richiede l'inserimento dei seguenti input:

- Name: carattere alfanumerico che rappresenta il nome del nodo definito dall'utente inserito come dato stringa.
- X,Y,Z: coordinate globali del nodo.

Esempio:

```
Beam.AddNode("N1", 0, 0, 0)
Beam.AddNode("N2", 10, 0, 0)
```

In accordo con la definizione della classe Node3D, il nodo creato avrà come attributi gli spostamenti nodali, le reazioni vincolari, e le condizioni di vincolo. Gli spostamenti nodali e le reazioni vincolari sono inizializzati a zero nel momento della creazione del nodo e dopo la fase di analisi del modello assumono i valori derivanti dal calcolo. Le condizioni di vincolo invece sono definite tramite delle variabili booleane inizializzate al valore False. Di seguito la nomenclatura utilizzata:

- spostamenti nodali

  DX, DY, DZ, RX, RY, RZ
- reazioni vincolari

  RxnFX, RxnFY, RxnFZ, RxnMX, RxnMY, RxnMZ
- Vincoli

  SupportDX, SupportDY, SupportDZ, SupportRX, SupportRY, SupportRZ

E' possibile eliminare un nodo del modello tramite la funzione *RemoveNode* definita anch'essa all'interno della classe FEModel3D e richiamata nel programma con la seguente sintassi:

```
nomemodello.RemoveNode (Name)
```

La funzione richiede in input il nome del nodo inserito come dato stringa e insieme ad esso vengono eliminati anche i carichi nodali e gli elementi trave ad esso collegati.

Esempio:

```
Beam.RemoveNode("N1")
```

4.2.2 Nodi ausiliari

Il nodo ausiliario viene creato tramite la funzione *AddauxNode* definita all'interno della classe *FEModel3D* e richiamata nel programma con la seguente sintassi:

```
nomemodello.AddAuxNode (Name, X, Y, Z)
```

I nodi ausiliari, in quanto tali, vengono memorizzati in un vettore distinto da quello contenente in nodi del modello strutturale; così facendo essi non sono presi in considerazione nell'analisi statica.

Tale funzione richiede l'inserimento dei seguenti input:

- Name: carattere alfanumerico che rappresenta il nome del nodo definito dall'utente inserito come dato stringa.
- X,Y,Z: coordinate globali del nodo.

Esempio:

```
Beam.AddAuxNode("aN1", 0, 0, 10)
```

L'inserimento del nodo ausiliario deve essere effettuato in modo tale che il nodo sia efficace all'orientamento del sistema di riferimento locale. Si rimanda a quanto spiegato al paragrafo 3.4.2.

4.2.3 Elementi trave

L'elemento trave viene modellato considerando l'asse longitudinale e viene creato tramite la funzione *AddMember*, definita all'interno della classe FEModel3D e richiamata nel programma con la seguente sintassi:

```
nomemodello.AddMember (Name, iNode, jNode, E, G, Iy, Iz, J, A,
                       auxNode)
```

Tale funzione richiede l'inserimento dei seguenti input:

- Name: carattere alfanumerico che rappresenta il nome dell'elemento definito dall'utente inserito come dato stringa;
- iNode: nodo iniziale, inserito come dato stringa
- jNode: nodo finale, inserito come dato stringa
- E: modulo di elasticità longitudinale
- G: modulo di elasticità tangenziale
- Iy: momento di inerzia attorno all'asse locale y
- Iz: momento di inerzia attorno all'asse locale z
- J: momento di inerzia torsionale
- A: area sezione trasversale
- auxNode: nodo ausiliare usato per fissare il sistema di riferimento locale

Le proprietà meccaniche e geometriche vengono inserite dall'utente coerentemente al sistema di misure adottato. Esse possono essere introdotte dentro le parentesi al momento dell'utilizzo della funzione *AddMember* oppure possono essere definite precedentemente tramite la definizione di variabili a cui viene assegnato il corrispondente valore numerico.

Esempio:

```
Beam.AddMember("M1", "N1", "N2", E, G, Iy, Iz, J, A)
```

Inoltre, è possibile rilasciare specifici gradi di libertà alle estremità del generico elemento mediante la funzione *DefineReleases*, definita all'interno della classe FEModel3D e richiamata dal programma con la seguente sintassi:

```
nomemodello.DefineReleases (Member, Dxi, Dyi, Dzi, Rxi, Ryi, Rzi,
                            Dxj, Dyj, Dzj, Rxj, Ryj, Rzj)
```

Nel caso generale di problema tridimensionale, è possibile svincolare un totale di 6 gradi di libertà per nodo. Di seguito sono elencati gli input necessari alla funzione:

- Member: carattere alfanumerico inserito come dato stringa.
- 12 rilasci definiti nel sistema di riferimento locale per mezzo delle variabili booleane (True/False), in cui True= grado di libertà rilasciato e False= grado di libertà non rilasciato.

E' possibile eliminare un elemento trave del modello tramite la funzione *RemoveMember* definita anch'essa all'interno della classe FEModel3D e richiamata nel programma con la seguente sintassi:

```
nomemodello.RemoveMember (Name)
```

La funzione richiede in input il nome dell'elemento inserito come dato stringa e insieme ad esso vengono eliminati anche i carichi applicati.

Esempio:

```
Beam.RemoveMember("M1")
```

4.2.4 Vincoli e cedimenti vincolari

Le condizioni di vincolo ed eventuali cedimenti vincolari vengono assegnate ai nodi precedentemente creati tramite la funzione *DefineSupport* definita all'interno della classe FEModel3D e richiamata nel programma con la seguente sintassi:

```
nomemodello.DefineSupport (Node, SupportDX, SupportDY, SupportDZ,
                SupportRX, SupportRY, SupportRZ)
```

Nel caso generale di problema tridimensionale, la funzione richiede l'inserimento di 6 condizioni di vincolo (3 traslazioni e 3 rotazioni) aventi la nomenclatura riportata dentro parentesi ed espresse nel sistema di riferimento globale XYZ. Essi rappresentano degli attributi del nodo che si vuole vincolare (4.2.1). Si noti che non c'è distinzione tra il sistema di riferimento usato per il nodo e quello usato per la definizione dei vincoli.

Di seguito sono elencati gli input necessari:

- Node: carattere alfanumerico inserito come dato stringa.
- 6 condizioni di vincolo espresse nel seguente modo:

1) nel caso di vincolo rigido, le condizioni di vincolo sono definite per mezzo delle variabili booleane (True/False), in cui True=grado di libertà vincolato e False=grado di libertà non vincolato
2) nel caso di cedimento vincolare si inserisce il valore del cedimento espresso nel sistema di riferimento globale XYZ.

In assenza di tale funzione il nodo sarà di default totalmente svincolato. Combinando opportunamente gli input (True, False e cedimenti) è possibile modellare diverse condizioni di vincolo. Si riportano alcuni esempi.

Esempio:

1. Incastro	`Beam.DefineSupport("N1", True, True, True, True, True, True)`
2. Incastro cedevole	`Beam.DefineSupport("N1", True, deltaY, True, True, True, True)`
3. Cerniera sferica	`Beam.DefineSupport("N1", True, True, True, False, False, False)`
4. Carrello in direzione X	`Beam.DefineSupport("N1", False, True, True, False, False, False)`

4.3 Carichi

4.3.1 Carichi nodali

Il metodo carico nodale permette di applicare forze e momenti concentrati ai vari nodi del modello. L'applicazione del carico avviene tramite la funzione *AddNodeLoad* definita all'interno della classe FEModel3D e richiamata dal programma con la seguente sintassi:

```
nomemodello.AddNodeLoad (Node, Direction, P)
```

Essa richiede l'inserimento dei seguenti input:

- Node: carattere alfanumerico inserito come dato stringa e rappresentante il nodo caricato;
- Direction: la direzione di applicazione dei carichi viene specificata rispetto al sistema di riferimento globale e viene definita sotto forma di carattere alfanumerico inserito

come dato stringa. Per le forze si utilizza “FX”, “FY”, “FZ” e per i momenti “MX”, “MY”, “MZ”. Si notino le lettere maiuscole X, Y e Z utilizzate per esprimere le direzioni rispetto al sistema di riferimento globale.

- P: valore del carico.

Convenzione positiva:

Le forze sono positive se hanno direzione concorde con gli assi XYZ del sistema di riferimento globale. I momenti sono positivi se i vettori momento hanno direzione concorde con gli assi XYZ del sistema di riferimento globale (Figura 4.2 - Convenzione positiva dei carichi nodaliFigura 4.2).

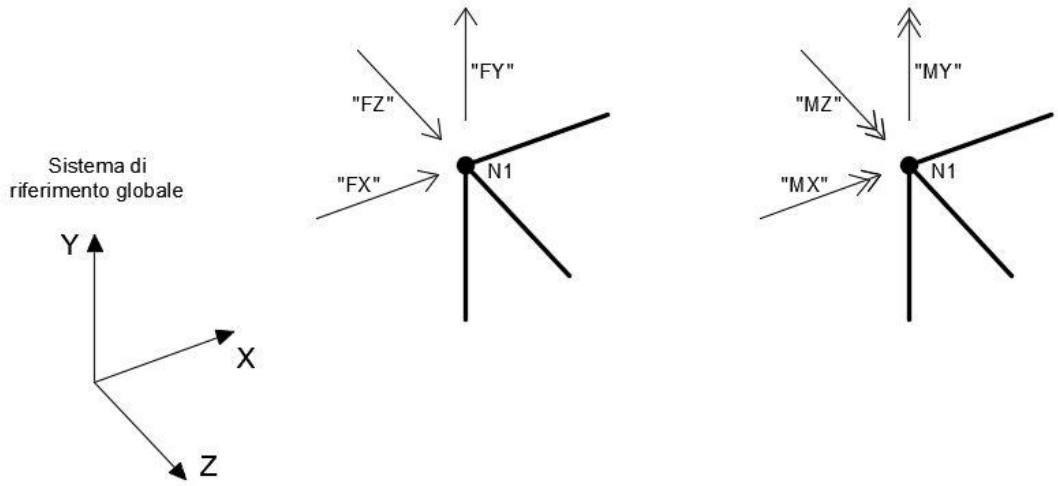


Figura 4.2 - Convenzione positiva dei carichi nodali

4.3.2 Carichi applicati sugli elementi

4.3.2.1 Carico concentrato in campata

Il metodo carico di campata concentrato permette di applicare forze e momenti concentrati in punti arbitrari dell’elemento. L’inserimento del carico avviene tramite la funzione *AddMemberPtLoad* definita all’interno della classe FEModel3D e richiamata dal programma con la seguente sintassi:

```
nomemodello.AddMemberPtLoad (Member, Direction, P, x)
```

Essa richiede l’inserimento dei seguenti input:

- Member: carattere alfanumerico inserito come dato stringa e rappresentante l’elemento trave caricato;

- Direction: la direzione di applicazione dei carichi viene specificata rispetto al sistema di riferimento locale e viene definita sottoforma di carattere alfanumerico inserito come dato stringa. Per le forze si utilizza “Fx”, “Fy”, “Fz” e per i momenti “Mx”, “My”, “Mz”. Si notino le lettere minuscole x, y e z utilizzate per esprimere le direzioni rispetto al sistema di riferimento locale.
- P: valore del carico;
- x: posizione del carico concentrato rispetto all’asse x del sistema di riferimento locale dell’elemento.

Convenzione positiva:

Le forze concentrate sono positive se concordi con gli assi x, y e z del sistema di riferimento locale dell’elemento. I momenti concentrati sono positivi se i vettori momento hanno direzione concorde con gli assi x, y e z del sistema di riferimento locale dell’elemento (

Figura 4.3).

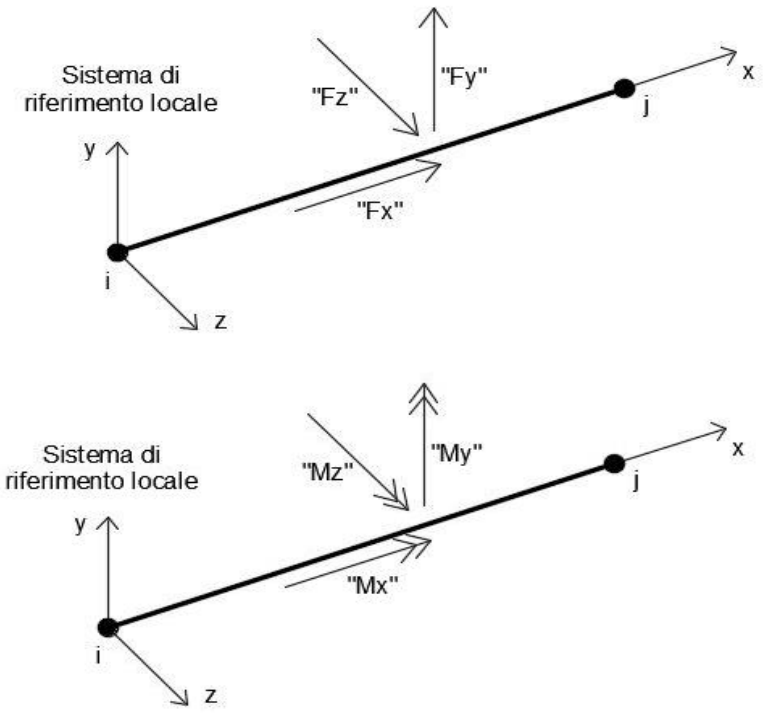


Figura 4.3 - Convenzione positiva dei carichi concentrati in campata

4.3.2.2 Carico distribuito trapezoidale in campata

Il metodo carico distribuito trapezoidale permette di applicare un carico linearmente distribuito sull'elemento. L'inserimento del carico avviene tramite la funzione *AddMemberDistLoad* definita all'interno della classe FEModel3D e richiamata dal programma con la seguente sintassi:

```
nomemodello.AddMemberDistLoad (Member, Direction, w1, w2, x1, x2)
```

Essa richiede l'inserimento dei seguenti input:

- Member: carattere alfanumerico inserito come dato stringa rappresentante l'elemento trave caricato;
- Direction: la direzione di applicazione dei carichi viene specificata rispetto al sistema di riferimento locale e viene definita sottoforma di carattere alfanumerico inserito come dato stringa. Per le forze si utilizza "Fx", "Fy", "Fz". Si notino le lettere minuscole x, y utilizzate per esprimere le direzioni rispetto al sistema di riferimento locale x,y e z.
- w1: valore iniziale del carico
- w2: valore finale del carico
- x1: inizio del carico distribuito espresso rispetto all'asse x del sistema di riferimento locale dell'elemento.
- x2: fine del carico distribuito espresso rispetto al sistema di riferimento locale dell'elemento.

In assenza di indicazioni su x1 e x2, il carico distribuito sarà applicato su tutto lo sviluppo dell'elemento.

Convenzione positiva:

I carichi distribuiti sono positivi se concordi con gli assi x, y e z del sistema di riferimento locale dell'elemento (Figura 4.4).

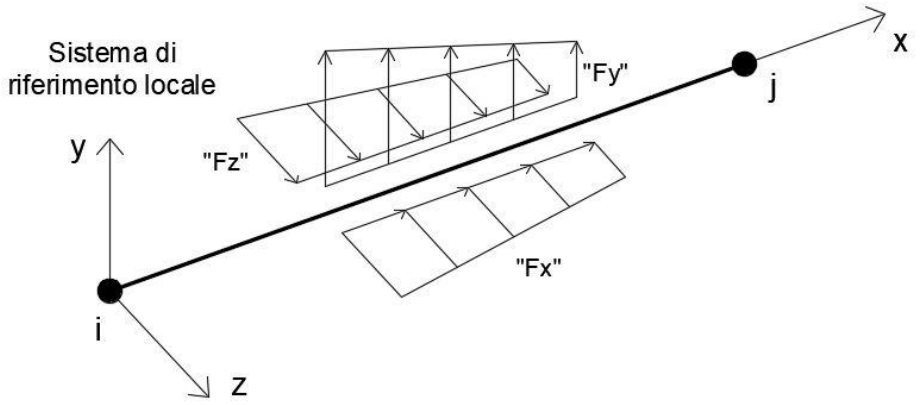


Figura 4.4 - Convenzione positiva dei carichi distribuiti

4.3.3 Spostamento nodale imposto

E' possibile applicare uno spostamento nodale imposto ad un nodo non vincolato tramite la funzione *AddNodeDisplacement* definita all'interno della classe FEModel3D e richiamata dal programma con la seguente sintassi:

```
nomemodello.AddNodeDisplacement (Node, Direction, value)
```

In generale, per spostamento nodale, si intende spostamento o rotazione. La funzione richiede l'inserimento dei seguenti input:

- Node: carattere alfanumerico inserito come dato stringa e rappresentante il nodo a cui applicare lo spostamento;
- Direction: la direzione di applicazione dello spostamento viene specificata rispetto al sistema di riferimento globale e viene definita sotto forma di carattere alfanumerico inserito come dato stringa. Per gli spostamenti si utilizza "DX", "DY", "DZ" e per le rotazioni "RX", "RY", "RZ".
- value: valore dello spostamento o rotazione.

4.4 Analisi

La risoluzione dello schema strutturale avviene mediante la funzione *Analyze* definita all'interno del modulo FEModel3D.py e richiamata dal programma con la seguente sintassi:

```
nomemodello.Analyze()
```

L'analisi strutturale si basa sul metodo degli spostamenti, secondo il quale, vengono assunte come incognite gli spostamenti e le rotazioni dei vari nodi. L'equazione risolvente è espressa nel sistema di riferimento globale ed è la seguente:

$$\{D\} = [K]^{-1}(\{P\} - \{FER\})$$

- $\{D\}$ = vettore degli spostamenti nodali
- $[K]$ = matrice di rigidezza totale
- $\{P\}$ = vettore dei carichi nodali
- $\{FER\}$ = vettore di incastro perfetto ("Fixed End Reaction")

Le operazioni necessarie per la costruzione della matrice $[K]$ e dei vettori $\{P\}$ e $\{FER\}$ non sono implementate dall'utente nella fase di costruzione del modello ma vengono effettuate direttamente dal codice PyNiteFEA, tramite metodi di istanza definiti all'interno della classe FEModel3D, nel momento in cui viene richiamata la funzione *Analyze.* In particolare, la matrice $[K]$ e il vettore $\{FER\}$ sono calcolati inizialmente nel sistema di riferimento locale del generico elemento trave ("frame"), dopodiché sono ruotati nel sistema di riferimento globale e infine vengono assemblati passando alla dimensione totale del problema. Il vettore $\{P\}$ invece richiede solamente l'operazione di assemblaggio.

La risoluzione del sistema di equazioni sfrutta il metodo della condensazione statica, il quale permette di ridurre la dimensione delle matrici e dei vettori interessati dal calcolo, focalizzandosi solamente sui gradi di libertà incogniti.

L'equazione di equilibrio globale è espressa come:

$$[K]\{D\} = \{P\} - \{FER\}$$

Tale equazione può essere partizionata nel seguente modo:

$$\begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \begin{Bmatrix} D_1 \\ D_2 \end{Bmatrix} = \begin{Bmatrix} P_1 \\ P_2 \end{Bmatrix} - \begin{Bmatrix} FER_1 \\ FER_2 \end{Bmatrix}$$

dove:

- D_1, P_2: spostamenti nodali e forze nodali incognite. Gli spostamenti incogniti sono riferiti ai gradi di libertà non vincolati, invece le forze nodali rappresentano le reazioni vincolari incognite;

- D_2, P_1: spostamenti nodali e forze nodali note. Gli spostamenti sono noti o perché sono nulli in seguito alle condizioni di vincolo o perché imposti diversi da zero dall'utente. Le forze nodali invece rappresentano i carichi nodali applicati al modello.
- K_{11}, K_{12}, K_{21}, K_{22}: partizioni della matrice di rigidezza K;
- FER_1, FER_2: partizioni del vettore dei carichi di incastro perfetto.

Sviluppando i prodotti matriciali si ottiene:

$$[K_{11}]\{D_1\} + [K_{12}]\{D_2\} = \{P_1\} - \{FER_1\} \quad (1)$$

$$[K_{21}]\{D_1\} + [K_{22}]\{D_2\} = \{P_2\} - \{FER_2\} \quad (2)$$

Dall'equazione (1) è possibile adesso ricavare gli spostamenti nodali incogniti:

$$\{D_1\} = [K_{11}]^{-1}(\{P_1\} - \{FER_1\} - [K_{12}]\{D_2\})$$

Calcolati gli spostamenti nodali incogniti, si ricompone il vettore $\{D\}$ che conterrà adesso tutti gli spostamenti nodali del sistema.

Successivamente si calcolano le reazioni vincolari, le quali, insieme agli spostamenti e alle rotazioni, vengono assegnati come attributi dei nodi a cui si riferiscono. In questo modo, in qualsiasi momento successivo all'analisi, è possibile interrogare ciascun nodo ottenendo il valore degli spostamenti e delle reazioni vincolari.

La funzione *Analyze* è in grado di riconoscere la labilità dello schema strutturale. Prima della risoluzione delle equazioni, viene effettuato un controllo sulla singolarità della matrice di rigidezza $[K_{11}]$. Se il rango è inferiore alla dimensione della matrice stessa, allora la matrice non è invertibile e l'analisi viene interrotta.

Si sottolinea che, l'algoritmo sviluppato per il partizionamento delle matrici e dei vettori interessati dal calcolo, può essere soggetto a sviluppi futuri al fine di ottimizzarne i tempi di esecuzione.

4.4.1 Vettore dei carichi nodali

Il vettore dei carichi nodali $\{P\}$ viene costruito nel modulo FEModel3D.py assemblando tutti i carichi nodali definiti durante la fase di costruzione del modello (§4.3.1). I carichi nodali sono

espressi sin da subito nel sistema di riferimento globale, quindi per la costruzione del vettore $\{P\}$ non è necessaria l'operazione di rotazione.

4.4.2 Matrice di rigidezza locale

La matrice di rigidezza locale [k] del generico elemento trave ("frame") ha dimensione 12x12 (**Errore. L'origine riferimento non è stata trovata.**). Definite le proprietà geometriche e m eccaniche dell'elemento (§4.2.3), essa ha la seguente forma:

Digitare l'equazione qui.

$$\begin{bmatrix}
\frac{EA}{l} & 0 & 0 & 0 & 0 & 0 & -\frac{EA}{l} & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{12EJ_z}{l^3} & 0 & 0 & 0 & \frac{6EJ_z}{l^2} & 0 & -\frac{12EJ_z}{l^3} & 0 & 0 & 0 & \frac{6EJ_z}{l^2} \\
0 & 0 & \frac{12EJ_y}{l^3} & 0 & -\frac{6EJ_y}{l^2} & 0 & 0 & 0 & -\frac{12EJ_y}{l^3} & 0 & -\frac{6EJ_y}{l^2} & 0 \\
0 & 0 & 0 & \frac{GJ_x}{l} & 0 & 0 & 0 & 0 & 0 & -\frac{GJ_x}{l} & 0 & 0 \\
0 & 0 & -\frac{6EJ_y}{l^2} & 0 & \frac{4EJ_y}{l} & 0 & 0 & 0 & \frac{6EJ_y}{l^2} & 0 & \frac{2EJ_y}{l} & 0 \\
0 & \frac{6EJ_z}{l^2} & 0 & 0 & 0 & \frac{4EJ_z}{l} & 0 & -\frac{6EJ_z}{l^2} & 0 & 0 & 0 & \frac{2EJ_z}{l} \\
-\frac{EA}{l} & 0 & 0 & 0 & 0 & 0 & \frac{EA}{l} & 0 & 0 & 0 & 0 & 0 \\
0 & -\frac{12EJ_z}{l^3} & 0 & 0 & 0 & -\frac{6EJ_z}{l^2} & 0 & \frac{12EJ_z}{l^3} & 0 & 0 & 0 & -\frac{6EJ_z}{l^2} \\
0 & 0 & -\frac{12EJ_y}{l^3} & 0 & \frac{6EJ_y}{l^2} & 0 & 0 & 0 & \frac{12EJ_y}{l^3} & 0 & \frac{6EJ_y}{l^2} & 0 \\
0 & 0 & 0 & -\frac{GJ_x}{l} & 0 & 0 & 0 & 0 & 0 & \frac{GJ_x}{l} & 0 & 0 \\
0 & 0 & -\frac{6EJ_y}{l^2} & 0 & \frac{2EJ_y}{l} & 0 & 0 & 0 & \frac{6EJ_y}{l^2} & 0 & \frac{4EJ_y}{l} & 0 \\
0 & \frac{6EJ_z}{l^2} & 0 & 0 & 0 & \frac{2EJ_z}{l^2} & 0 & -\frac{6EJ_z}{l^2} & 0 & 0 & 0 & \frac{4EJ_z}{l}
\end{bmatrix}$$

Figura 4.5 - Matrice di rigidezza locale elemento trave

4.4.3 Vettore dei carichi di incastro perfetto

Il vettore di incastro perfetto $\{fer\}$ ("Fixed End Reaction vectors") del generico elemento trave ("frame") ha dimensione 12x1:

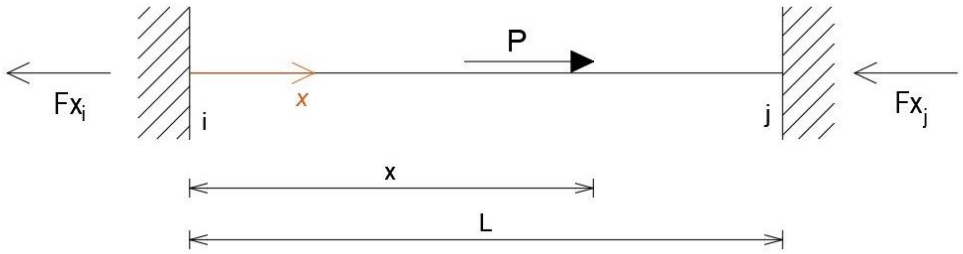
$$\{fer\}^T = \{Fxi\,,\quad Fyi,\quad Fyi,\quad Mx_i,\quad My_i,\quad Mz_i,\quad Fx_j,\quad Fy_j,\quad Fz_j,\quad Mx_j,\quad My_j,\quad Mz_j\}$$

Esso viene definito nel modulo Member3D.py come somma dei vettori di incastro perfetto dovuti ai diversi carichi applicati all'elemento. A sua volta, i singoli vettori di incastro perfetto sono calcolati nel modulo FixedEndReaction.py, all'interno del quale vengono analizzate diverse tipologie di carico. Tale calcolo avviene tramite la definizione di apposite funzioni elencate all'inizio del capitolo 2. Tutto ciò viene svolto in automatico e non necessita l'implementazione di apposite istruzioni da parte dell'utente.

Si elencano di seguito le espressioni delle reazioni di incastro perfetto per le diverse condizioni di carico.

- Carico concentrato in campata
  - Carico assiale direzione x

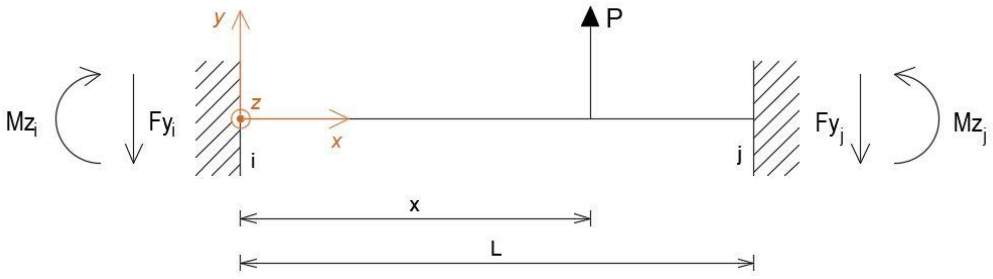
$$Fx_i = -P\frac{(L-x)}{L} \qquad Fx_j = -P\frac{x}{L}$$



  - Carico trasversale direzione y

$$Fy_i = -\frac{P(L-x)^2(L+2x)}{L^3} \qquad Fy_j = -\frac{Px^2[L+2(L-x)]}{L^3}$$

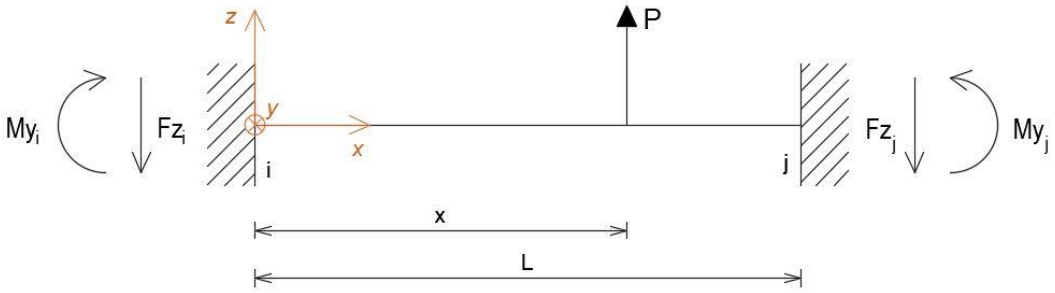
$$Mz_j = +\frac{Px^2(L-x)}{L^2} \qquad Mz_j = +\frac{Px^2(L-x)}{L^2}$$



  - Carico trasversale direzione z

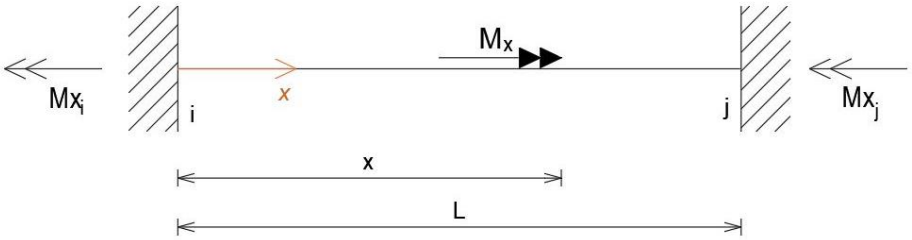
$$Fz_i = -P\frac{(L-x)^2(L+2x)}{L^3} \qquad Fz_j = -P\frac{x^2[L+2(L-x)]}{L^3}$$

$$My_i = P\frac{x(L-x)^2}{L^2} \qquad My_j = -P\frac{x^2(L-x)}{L^2}$$



- Momento concentrato intorno all'asse x (torcente)

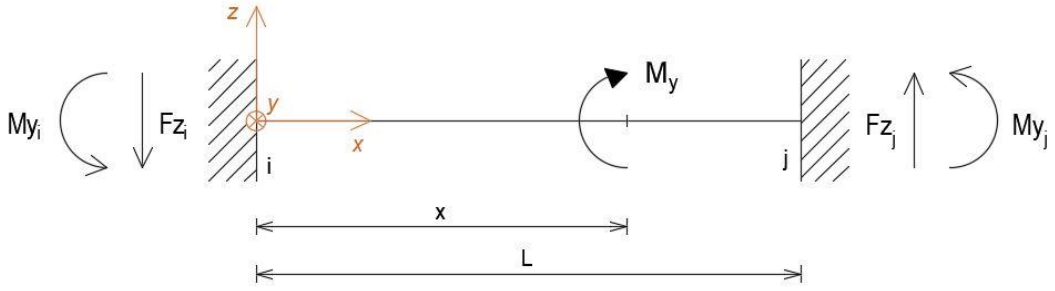
$$Mx_i = -M\frac{(L-x)}{L} \qquad Mx_j = -M\frac{x}{L}$$



- Momento concentrato intorno a asse y (flettente)

$$Fz_i = -6M\frac{x(L-x)}{L^3} \qquad Fz_j = M\frac{x(L-x)}{L^3}$$

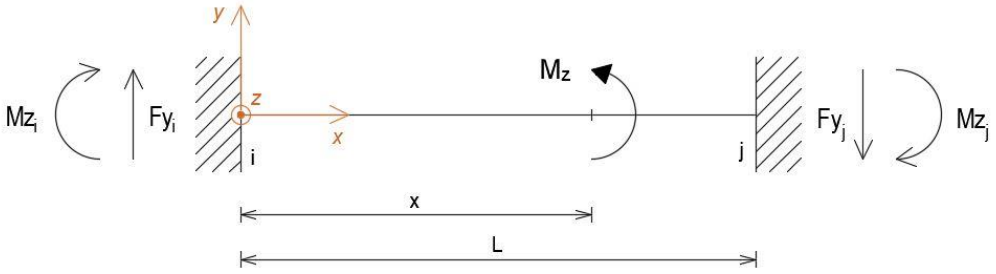
$$My_i = M\frac{(L-x)[2x-(L-x)]}{L^2} \qquad My_j = M\frac{x[2(L-x)-x]}{L^2}$$



- Momento concentrato intorno a asse z (flettente)

$$Fy_i = 6M\frac{x(L-x)}{L^3} \qquad Fy_j = -M\frac{x(L-x)}{L^3}$$

$$Mz_i = M\frac{(L-x)[2x-(L-x)]}{L^2} \qquad Mz_j = M\frac{x[2(L-x)-x]}{L^2}$$



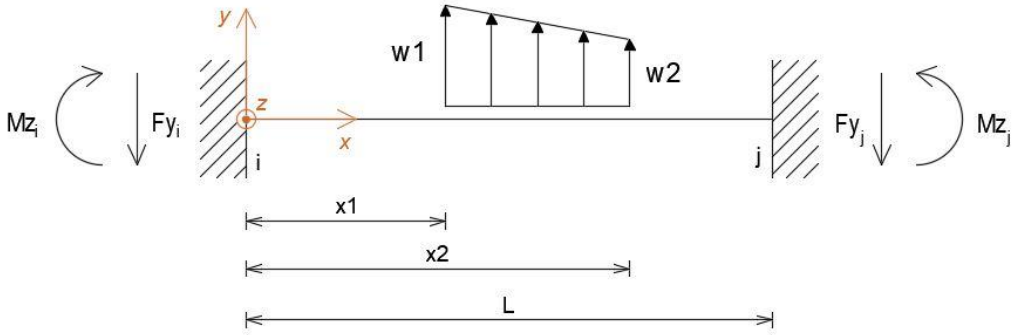
- Carico distribuito trapezoidale in campata
  - Carico trasversale direzione y

$$Fy_i = (x_1-x_2)(10L^3w_1 + 10L^3w_2 - 15Lw_1x_1^2 - 10Lw_1x_1x_2 - 5Lw_1x_2^2 - 5Lw_2x_1^2 - 10Lw_2x_1x_2 - 15Lw_2x_2^2 + 8w_1x_1^3 + 6w_1x_1^2x_2 + 4w_1x_1x_2^2 + 2w_1x_2^3 + 2w_2x_1^3 + 4w_2x_1^2x_2 + 6w_2x_1x_2^2 + 8w_2x_2^3)/(20L^3)$$

$$Mz_i = (x_1-x_2)(20L^2w_1x_1 + 10L^2w_1x_2 + 10L^2w_2x_1 + 20L^2w_2x_2 - 30Lw_1x_1^2 - 20Lw_1x_1x_2 - 10Lw_1x_2^2 - 10Lw_2x_1^2 - 20Lw_2x_1x_2 - 30Lw_2x_2^2 + 12w_1x_1^3 + 9w_1x_1^2x_2 + 6w_1x_1x_2^2 + 3w_1x_2^3 + 3w_2x_1^3 + 6w_2x_1^2x_2 + 9w_2x_1x_2^2 + 12w_2x_2^3)/(60L^2)$$

$$Fy_j = -(x_1-x_2)(-15Lw_1x_1^2 - 10Lw_1x_1x_2 - 5Lw_1x_2^2 - 5Lw_2x_1^2 - 10Lw_2x_1x_2 - 15Lw_2x_2^2 + 8w_1x_1^3 + 6w_1x_1^2x_2 + 4w_1x_1x_2^2 + 2w_1x_2^3 + 2w_2x_1^3 + 4w_2x_1^2x_2 + 6w_2x_1x_2^2 + 8w_2x_2^3)/(20L^3)$$

$$Mz_j = (x_1-x_2)(-15Lw_1x_1^2 - 10Lw_1x_1x_2 - 5Lw_1x_2^2 - 5Lw_2x_1^2 - 10Lw_2x_1x_2 - 15Lw_2x_2^2 + 12w_1x_1^3 + 9w_1x_1^2x_2 + 6w_1x_1x_2^2 + 3w_1x_2^3 + 3w_2x_1^3 + 6w_2x_1^2x_2 + 9w_2x_1x_2^2 + 12w_2x_2^3)/(60L^2)$$



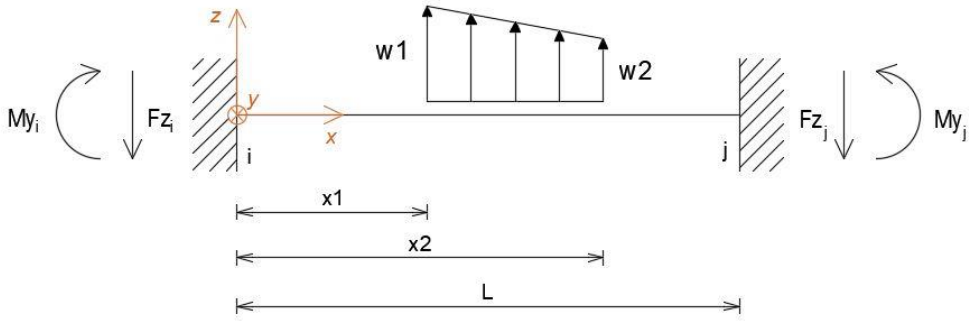
- Carico trasversale direzione z

$$Fz_i = (x_1 - x_2)(10L^3w_1 + 10L^3w_2 - 15Lw_1x_1^2 - 10Lw_1x_1x_2 - 5Lw_1x_2^2 - 5Lw_2x_1^2 - 10Lw_2x_1x_2 - 15Lw_2x_2^2 + 8w_1x_1^3 + 6w_1x_1^2x_2 + 4w_1x_1x_2^2 + 2w_1x_2^3 + 2w_2x_1^3 + 4w_2x_1^2x_2 + 6w_2x_1x_2^2 + 8w_2x_2^3)/(20L^3)$$

$$My_i = -(x_1 - x_2)(20L^2w_1x_1 + 10L^2w_1x_2 + 10L^2w_2x_1 + 20L^2w_2x_2 - 30Lw_1x_1^2 - 20Lw_1x_1x_2 - 10Lw_1x_2^2 - 10Lw_2x_1^2 - 20Lw_2x_1x_2 - 30Lw_2x_2^2 + 12w_1x_1^3 + 9w_1x_1^2x_2 + 6w_1x_1x_2^2 + 3w_1x_2^3 + 3w_2x_1^3 + 6w_2x_1^2x_2 + 9w_2x_1x_2^2 + 12w_2x_2^3)/(60L^2)$$

$$Fz_j = -(x_1 - x_2)(-15Lw_1x_1^2 - 10Lw_1x_1x_2 - 5Lw_1x_2^2 - 5Lw_2x_1^2 - 10Lw_2x_1x_2 - 15Lw_2x_2^2 + 8w_1x_1^3 + 6w_1x_1^2x_2 + 4w_1x_1x_2^2 + 2w_1x_2^3 + 2w_2x_1^3 + 4w_2x_1^2x_2 + 6w_2x_1x_2^2 + 8w_2x_2^3)/(20L^3)$$

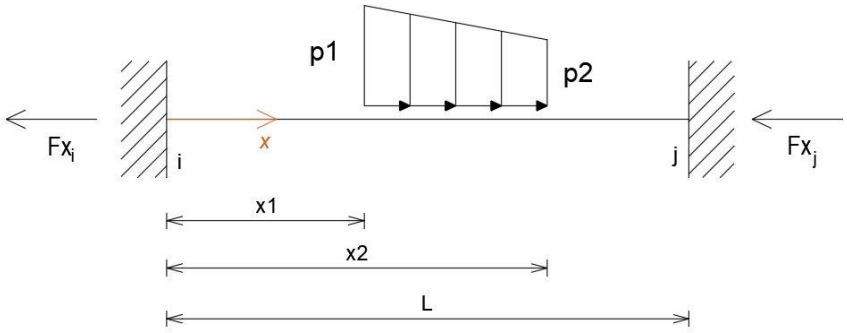
$$My_j = -(x_1 - x_2)(-15Lw_1x_1^2 - 10Lw_1x_1x_2 - 5Lw_1x_2^2 - 5Lw_2x_1^2 - 10Lw_2x_1x_2 - 15Lw_2x_2^2 + 12w_1x_1^3 + 9w_1x_1^2x_2 + 6w_1x_1x_2^2 + 3w_1x_2^3 + 3w_2x_1^3 + 6w_2x_1^2x_2 + 9w_2x_1x_2^2 + 12w_2x_2^3)/(60L^2)$$



- Carico assiale direzione x

$$F_{x_i} = \frac{1}{6L}(x_1 - x_2)(3Lp_1 + 3Lp_2 - 2p_1x_1 - p_1x_2 - p_2x_1 - 2p_2x_2)$$

$$F_{x_j} = \frac{1}{6L}(x_1 - x_2)(2p_1x_1 + p_1x_2 + p_2x_1 + 2p_2x_2)$$



4.4.4 Vettore degli spostamenti nodali

Il vettore degli spostamenti nodali $\{D\}$ ha dimensione 3nx1 (n=numero totale di nodi) e contiene gli spostamenti dei nodi ottenuti con la risoluzione del modello agli elementi finiti. Tale vettore, inizialmente vuoto, viene riempito durante l'esecuzione della funzione *Analyze()* dopo la risoluzione del sistema di equazioni.

4.5 Risultati

Terminata la fase di risoluzione del modello agli elementi finiti, i risultati possono essere richiesti e visualizzati dall'utente.

Per ottenere i risultati del generico nodo o elemento trave, è necessario preliminarmente selezionare l'oggetto desiderato per mezzo di appositi metodi definiti all'interno della classe FEModel3D; essi si differenziano per i nodi e per gli elementi trave.

Il nodo viene selezionato mediante la funzione *GetNode*. Il nome del nodo, inserito come stringa, costituisce l'argomento della funzione:

```
nomemodello.GetNode (Name)
```

L'elemento trave viene selezionato mediante la funzione *GetMember*. Il nome dell'elemento, inserito come stringa, costituisce l'argomento della funzione:

```
nomemodello.GetMember (Name)
```

I metodi sopracitati operano nel medesimo modo; il nome dell'entità da selezionare viene confrontato con i nomi di tutte le entità inserite in fase di costruzione del modello e, una volta ottenuta la corrispondenza, la funzione restituisce allo script chiamante l'oggetto voluto. Queste funzioni permettono di passare dalla classe FEModel3D a cui appartiene il modello FEM, alla classe Member3D o Node3D a cui appartengono rispettivamente gli elementi trave e i nodi. A questo punto, l'oggetto selezionato, essendo istanza di una classe, permette di richiamare tramite opportune istruzioni gli attributi e i metodi desiderati.

4.5.1 Sollecitazioni, spostamenti e relativi diagrammi dell'elemento trave

Per ciascun elemento trave è possibile calcolare le sollecitazioni di sforzo normale, taglio, momento e gli spostamenti assiali e trasversali. Il calcolo è effettuato da metodi la cui esecuzione è subordinata a ulteriori funzioni definite all'interno delle classi BeamSegment; tali funzioni analizzano tutti i segmenti matematicamente continui in cui è suddiviso il generico elemento trave. I valori numerici calcolati vengono visualizzati a schermo mediante la funzione print() di Python, il cui argomento è costituito dalle istruzioni necessarie per calcolare il risultato desiderato e sarà spiegato dettagliatamente nei paragrafi successivi. La rappresentazione grafica dei diagrammi 2D è resa possibile dalla libreria Matplotlib e da tutte le funzioni ad essa collegata. Si riportano di seguito le convenzioni positive per le caratteristiche della sollecitazione e per gli spostamenti:

1) In Figura 4.6 è rappresentato un concio di trave; si consideri il sistema di riferimento locale x, y e z dell'elemento trave. Si definisce sezione positiva la faccia del concio che si trova nella direzione positiva dell'asse locale considerato. Si definisce sezione negativa la faccia del concio che trova nella direzione negativa dell'asse locale considerato.

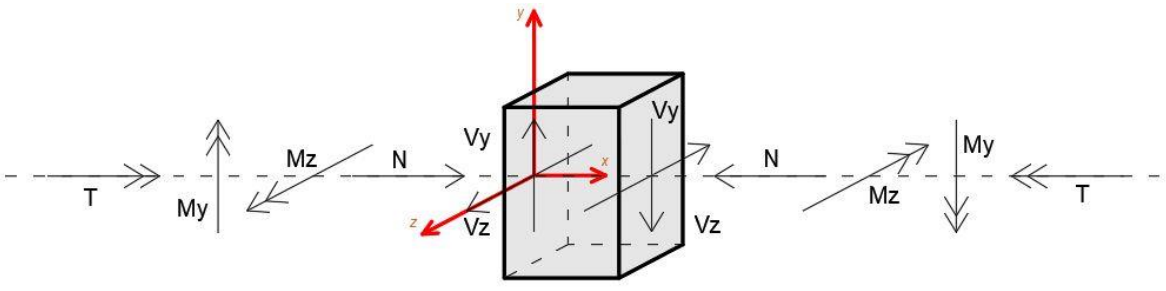


Figura 4.6 - Convenzione positiva delle caratteristiche della sollecitazione

- Lo sforzo normale N si considera positivo se di compressione;

- Il taglio V_y si considera positivo se sulla sezione positiva è in direzione discorde con il verso positivo dell'asse locale y;
- Il taglio V_z si considera positivo se sulla sezione positiva è in direzione discorde con il verso positivo dell'asse locale z;
- Il momento flettente M_y si considera positivo se tende le fibre della faccia negativa;
- Il momento flettente M_z si considera positivo se tende le fibre della faccia positiva;
- Il momento torcente T si considera positivo se sulla sezione positiva è in direzione discorde con il verso positivo dell'asse locale x;

2) Gli spostamenti assiali e trasversali sono positivi se concordi con il verso positivo dell'asse locale considerato.

Per tutte le caratteristiche della sollecitazione e per gli spostamenti, sono a disposizione dell'utente metodi specifici che consentono di:

- Calcolare il valore in un punto generico dell'elemento
- Calcolare il valore massimo
- Calcolare il valore minimo
- Plottare il grafico

Si riportano di seguito le istruzioni necessarie.

4.5.1.1 Sforzo normale

Lo sforzo normale in un punto generico dell'elemento è calcolato attraverso la funzione *Axial;* il massimo e il minimo sforzo normale sono calcolati rispettivamente tramite le funzioni *MaxAxial* e *MinAxial*; il grafico è ottenuto tramite la funzione *PlotAxial*. I metodi sono richiamati dal programma con le seguenti sintassi:

```
print(nomemodello.GetMember(Name).Axial(x))
print(nomemodello.GetMember(Name).MaxAxial())
print(nomemodello.GetMember(Name).MinAxial())
nomemodello.GetMember(Name).PlotAxial()
```

L'unico input necessario è il seguente:

- x: posizione lungo l'asse locale x dell'elemento.

4.5.1.2 Taglio

Il taglio in un punto generico dell'elemento è calcolato attraverso la funzione *Shear;* il massimo e il minimo taglio sono calcolati rispettivamente tramite le funzioni *MaxShear* e *MinShear;* il grafico è ottenuto tramite la funzione *PlotShear*. I metodi sono richiamati dal programma con le seguenti sintassi:

```
print(nomemodello.GetMember(Name).Shear(Direction, x))
print(nomemodello.GetMember(Name).MaxShear (Direction))
print(nomemodello.GetMember(Name).MinShear (Direction))
nomemodello.GetMember(Name).PlotShear(Direction)
```

Le funzioni richiedono l'inserimento dei seguenti input:

- Direction: la direzione in cui calcolare il taglio viene specificata rispetto alle direzioni y e z del sistema di riferimento locale. Essa viene definita sottoforma di carattere alfanumerico inserito come dato stringa, scegliendo tra "Fy" e"Fz".
- x: posizione lungo l'asse locale x dell'elemento.

4.5.1.3 Momento

Il momento in un punto generico dell'elemento è calcolato attraverso la funzione *Moment;* il massimo e il minimo momento sono calcolati rispettivamente tramite le funzioni *MaxMoment* e *MinMoment;* il grafico è ottenuto tramite la funzione *PlotMoment*. I metodi sono richiamati dal programma con le seguenti sintassi:

```
print(nomemodello.GetMember(Name).Moment(Direction, x))
print(nomemodello.GetMember(Name).MaxMoment (Direction))
print(nomemodello.GetMember(Name).MinMoment (Direction))
nomemodello.GetMember(Name).PlotMoment(Direction)
```

Le funzioni richiedono l'inserimento dei seguenti input:

- Direction: la direzione in cui calcolare il momento flettente viene specificata rispetto alle direzioni y e z del sistema di riferimento locale. Essa viene definita sottoforma di carattere alfanumerico inserito come dato stringa, scegliendo tra "Mx", "My" e "Mz".
- x: posizione lungo l'asse locale x dell'elemento.

4.5.1.4 Deformata

Gli spostamenti assiali e/o trasversali in un punto generico dell'elemento sono calcolati attraverso la funzione *Deflection;* il massimo e il minimo spostamento sono calcolati rispettivamente tramite le funzioni *MaxDeflection* e *MinDeflection;* il grafico è ottenuto tramite la funzione *PlotDeflection*. I metodi sono richiamati dal programma con le seguenti sintassi:

```
print(nomemodello.GetMember(Name).Deflection(Direction,x))
print(nomemodello.GetMember(Name).MaxDeflection(Direction))
print(nomemodello.GetMember(Name).MinDeflection(Direction))
nomemodello.GetMember(Name).PlotDeflection(Direction)
```

Le funzioni richiedono l'inserimento dei seguenti input:

- Direction: direzione in cui calcolare lo spostamento. Viene specificata rispetto alle direzioni x, y e z del sistema di riferimento locale e viene definita sottoforma di carattere alfanumerico inserito come dato stringa. Si utilizza "dx" per gli spostamenti assiali, "dy" e "dz" per gli spostamenti trasversali.
- x: posizione lungo l'asse locale x dell'elemento.

E' possibile calcolare e plottare anche la deformata relativa del generico elemento trave. Questa funzione risulta particolarmente utile nell'analisi di grandi strutture composte da un numero elevato di elementi e con grandi spostamenti nodali. In questi casi, il diagramma degli spostamenti trasversali ottenuto con la funzione PlotDeflection() è caratterizzato da spostamenti nodali molto più grandi degli spostamenti trasversali dell'elemento con una conseguente difficile interpretazione del grafico.

La deformata relativa è ottenuta a partire dalla deformata calcolata con la funzione Deflection depurata dal moto rigido, quest'ultimo calcolato con gli spostamenti nodali dell'elemento trave (Figura 4.7).

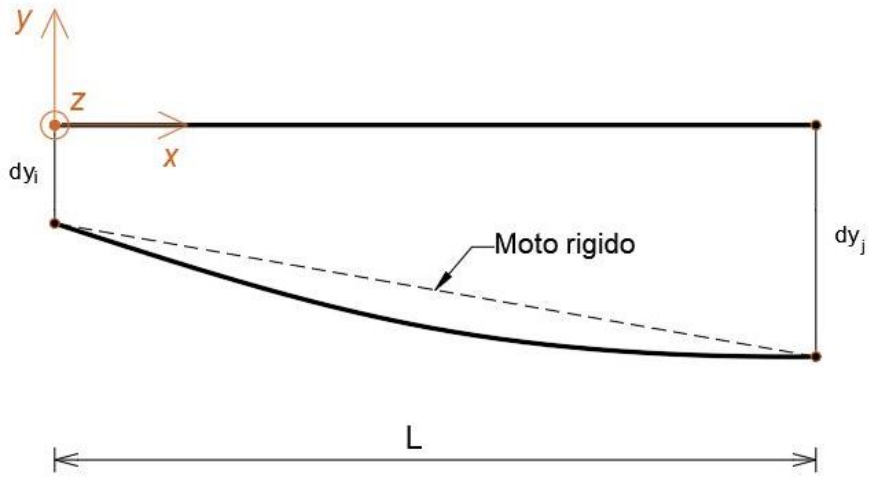


Figura 4.7 - Esempio di deformata relativa e moto rigido nel piano yx locale

La deformata relativa in un punto generico dell'elemento è calcolata attraverso la funzione *RelativeDeflection;* il grafico è ottenuto tramite la funzione *PlotRelativeDeflection*. I metodi sono richiamati dal programma e visualizzati con le seguenti sintassi:

```
print(nomemodello.GetMember(Name).RelativeDeflection(Direction,x))
nomemodello.GetMember(Name).PlotRelativeDeflection(Direction)
```

Le funzioni richiedono l'inserimento dei seguenti input:

- Direction: direzione in cui calcolare la deformata relativa. Viene specificata rispetto alle direzioni y e z del sistema di riferimento locale e viene definita sottoforma di carattere alfanumerico inserito come dato stringa scegliendo tra "dy" e "dz".
- x: posizione lungo l'asse locale x dell'elemento.

4.5.1.5 Torsione

La torsione in un punto generico dell'elemento è calcolata attraverso la funzione *Torsion;* la massima e la minima torsione sono calcolate rispettivamente tramite le funzioni *MaxTorsion* e *MinTorsion;* il grafico è ottenuto tramite la funzione *PlotTorsion*. I metodi sono richiamati dal programma con le seguenti sintassi:

```
print(nomemodello.GetMember(Name).Torsion(x))
print(nomemodello.GetMember(Name).MaxTorsion ())
print(nomemodello.GetMember(Name).MinTorsion ())
nomemodello.GetMember(Name).PlotTorsion()
```

Le funzioni richiedono l'inserimento dei seguenti input:

- x: posizione lungo l'asse locale x dell'elemento.

4.5.1.6 Diagrammi

Il generico diagramma viene costruito attraverso i metodi sopraesposti (*PlotAxial, PlotShear, PlotMoment, PlotDeflection, PlotRelativeDeflection*). All'interno di ciascuno di questi metodi, l'elemento viene discretizzato in un numero finito di punti, il cui valore non può essere scelto dall'utente in fase di scrittura dello script ma può essere modificato andando a cambiare il valore di default definito nel modulo Member3D.py. Quindi, per ogni punto, nota l'ascissa, viene calcolato il valore della grandezza da diagrammare. La visualizzazione grafica 2D avviene tramite la libreria Matplotlib. In particolare, viene usato il modulo pyplot che consiste in una raccolta di funzioni che consentono di utilizzare in interattivo le funzionalità della libreria Matplotlib.

Tramite apposite istruzioni viene generata la plotting window, si scelgono i valori da diagrammare, il nome degli assi e il titolo del grafico. Si mostra di seguito un grafico a titolo di esempio (Figura 4.8):

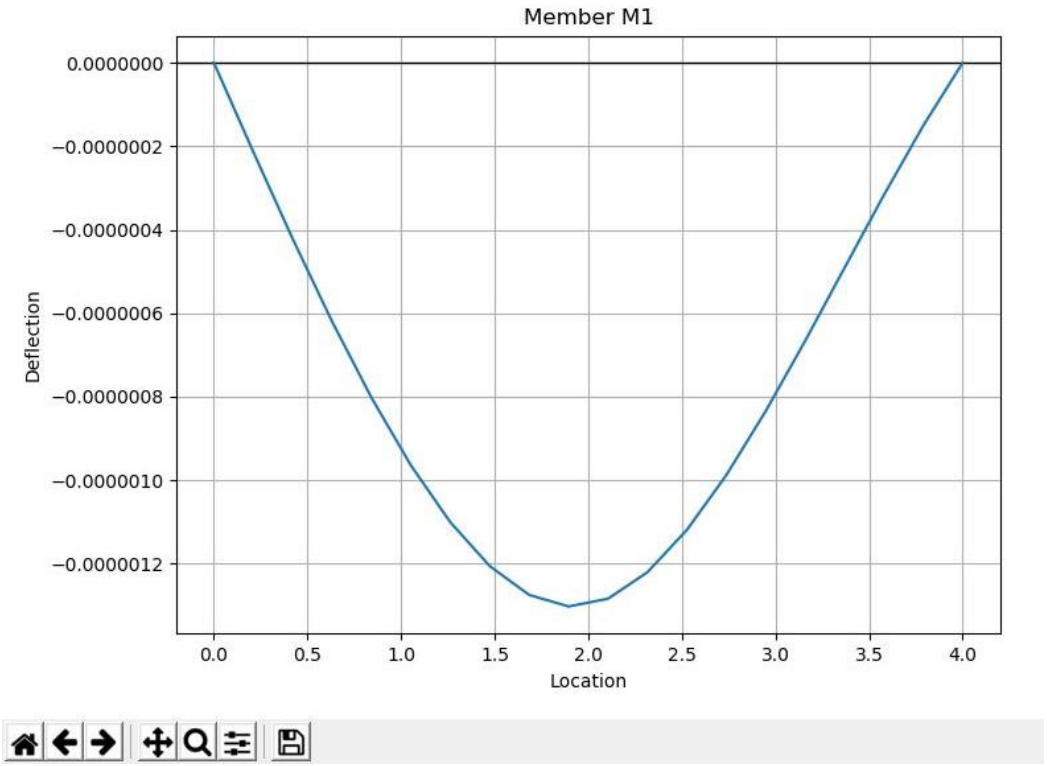


Figura 4.8 - Esempio grafico – plotting window

Nella parte inferiore della plotting window è presente una barra degli strumenti che permettono la navigazione interattiva nel grafico.

4.5.2 Spostamenti nodali

Gli spostamenti del generico nodo (4.2.1) sono espressi nel sistema di riferimento globale (3.4.1) e vengono richiamati e visualizzati a schermo con la seguente sintassi:

```
print(nomemodello.GetNode(Name).DX)
print(nomemodello.GetNode(Name).DY)
print(nomemodello.GetNode(Name).DZ)
print(nomemodello.GetNode(Name).RX)
print(nomemodello.GetNode(Name).RY)
print(nomemodello.GetNode(Name).RZ)
```

In maniera simile è possibile visualizzare direttamente un vettore (12x1) contenente gli spostamenti nodali (4.4.4). Essendo esso definito nella classe FEModel3D, viene richiamato e visualizzato a schermo con la seguente sintassi;

```
print(nomemodello.D())
```

4.5.3 Reazioni vincolari

Le reazioni vincolari (4.2.1) sono espresse nel sistema di riferimento globale (3.4.1) e vengono richiamate e visualizzate a schermo con la seguente sintassi:

```
print(nomemodello.GetNode(Name).RxnFX)
print(nomemodello.GetNode(Name).RxnFY)
print(nomemodello.GetNode(Name).RxnFZ)
print(nomemodello.GetNode(Name).RxnMX)
print(nomemodello.GetNode(Name).RxnMY)
print(nomemodello.GetNode(Name).RxnMZ)
```

4.6 Visualizzazione 3D

La visualizzazione 3D del modello avviene mediante la libreria python vtk la quale permette di generare entità geometriche visualizzabili tridimensionalmente. In particolare, PyNiteFEA permette la visualizzazione 3D della geometria del modello e la visualizzazione 3D della deformata. Si illustrano di seguito le due

La visualizzazione geometrica del modello avviene tramite la funzione *RenderModel* definita all'interno del modulo Visualization.py e richiamata dal programma con la seguente sintassi:

```
Visualization.RenderModel (model, textHeight)
```

Essa richiede l'inserimento dei seguenti input:

- model: nome del modello che si vuole visualizzare, ovvero il nome dell'istanza della classe FEModel3D definita al paragrafo 4.2;
- textHeight: fattore di scala scelto dall'utente per la dimensione del testo e per la simbologia grafica.

I nodi sono rappresentati con delle entità geometriche diverse a seconda del grado di vincolo invece gli elementi trave mediante dei segmenti di retta. I nodi ausiliari sono rappresentati con delle sfere e hanno una colorazione diversa rispetto agli altri nodi. Per ciascuna entità viene visualizzato il nome assegnato durante la fase di costruzione del modello (Figura 4.9). I colori delle entità geometriche e dello sfondo possono essere impostati dall'utente modificando il codice all'interno del modulo Visualization.py. Per maggiore chiarezza si riportano di seguito delle immagini.

Esempio:

Grigliato tridimensionale costruito nel piano XY e caricato con una forza nodale applicata al nodo N12 in direzione contraria all'asse Z

```
Visualization.RenderModel (Frame, 0.3)
```

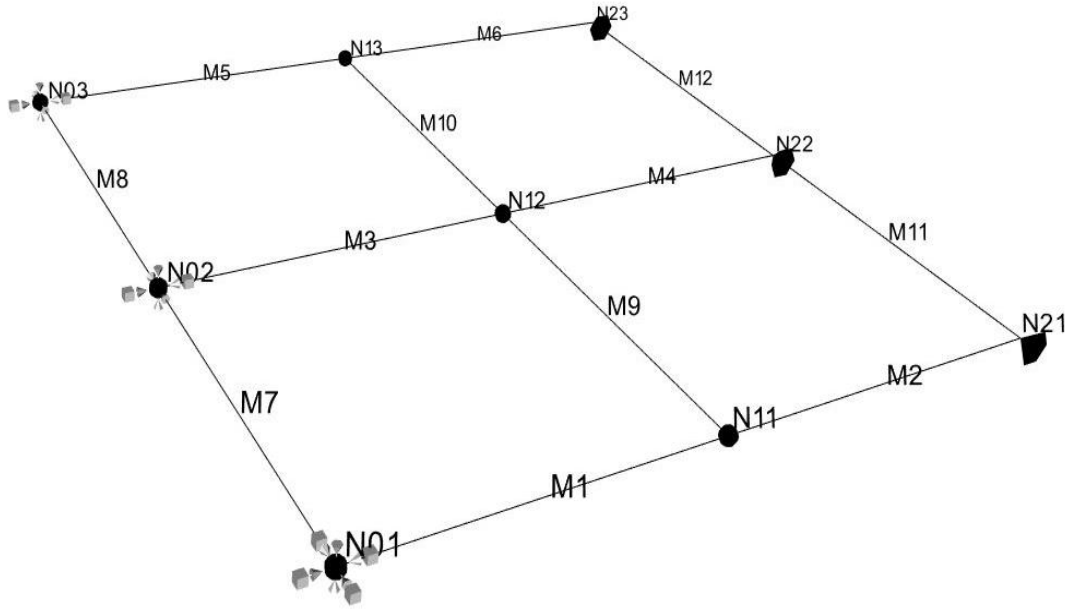


Figura 4.9 - Visualizzazione 3D della geometria del modello

Oltre alla visualizzazione geometrica è possibile visualizzare anche la deformata spaziale del modello. Ciò è possibile tramite la funzione *DeformedShape* definita anch'essa all'interno del modulo Visualization.py e richiamata dal programma con la seguente sintassi:

```
Visualization.DeformedShape (model, scalefactor, textHeight)
```

Essa richiede l'inserimento degli stessi input necessari alla funzione *RenderModel* più un ulteriore fattore di scala:

- scalefactor: fattore di scala scelto dall'utente per l'amplificazione della deformata

Tale funzione calcola la configurazione deformata dei nodi aggiungendo alle coordinate iniziali gli spostamenti ottenuti dalla risoluzione del modello. La configurazione deformata degli elementi trave invece viene calcolata con gli stessi metodi esposti al paragrafo 4.5.1.5. Il tutto è convertito in entità geometriche tridimensionali tramite la libreria vtk.

Esempio:

Grigliato tridimensionale costruito nel piano XY e caricato con una forza nodale applicata nodo N12 in direzione contraria all'asse Z

```
Visualization.DeformedShape (Frame, 5e4, 0.3)
```

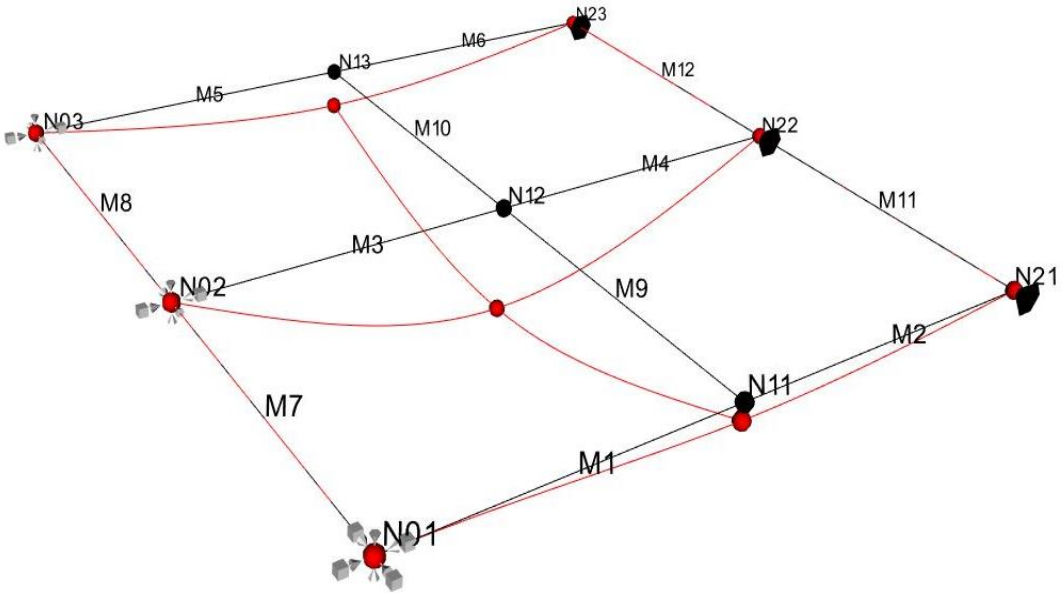


Figura 4.10 - Visualizzazione 3D della configurazione deformata

CAPITOLO 5
Applicazione al caso studio

In questo capitolo si descrive l'utilizzo del codice PyNiteFEA per la modellazione di un impalcato a graticcio di un viadotto autostradale sul quale è stato anche installato un sistema di monitoraggio strutturale. L'obiettivo prefissato è quello di calibrare il modello sulla base delle misurazioni fatte dai sensori durante una prova di carico. In particolare, si farà riferimento alle rotazioni delle travi longitudinali misurate da inclinometri a barra.

5.1 Ponti a graticcio: generalità

I ponti a graticcio sono strutture composte da più travi longitudinali le quali possono essere a parete piena o di tipo reticolare. Le travi sono appoggiate alle sottostrutture (pile o spalle) e sono sollecitate prevalentemente da azioni taglianti a da momenti flettenti. Dal punto di vista dei materiali, i ponti stradali e ferroviari sono generalmente realizzati con travi prefabbricate in calcestruzzo armato precompresso (Figura 5.1), oppure in acciaio con soletta in calcestruzzo collaborante (Figura 5.2). In genere vengono impiegato per luci medio piccole (fino 40-50 m), preferendo ponti a cassone per luci maggiori.

L'impalcato a graticcio è costituito dalle travi longitudinali affiancate fra loro e collegate puntualmente da elementi trasversali detti traversi, aventi la funzione di irrigidimento. Le travi principali, insieme ai traversi, formano la struttura portante a graticcio, generalmente a maglie rettangolari. Per luci modeste (fino circa a 40 m) si utilizzano travi prefabbricate in c.a.p.; per luci maggiori, risultano più vantaggiose economicamente le travi saldate a doppio T in acciaio.

Negli ultimi decenni il costo della mano d'opera richiesta e la crescente offerta di prodotti prefabbricati, ha ridotto di molto l'impiego di travi in cemento armato gettato in opera.

Figura 5.1 – Impalcato in calcestruzzo armato precompresso

Figura 5.2 - Impalcato con travi in acciaio e soletta in calcestruzzo collaborante

5.2 Descrizione del caso studio

Il caso studio analizzato riguarda un viadotto autostradale italiano degli anni ’70. Si tratta di un ponte a travata con impalcati a graticcio in calcestruzzo armato precompresso, costituito da campate di 45 m di luce.

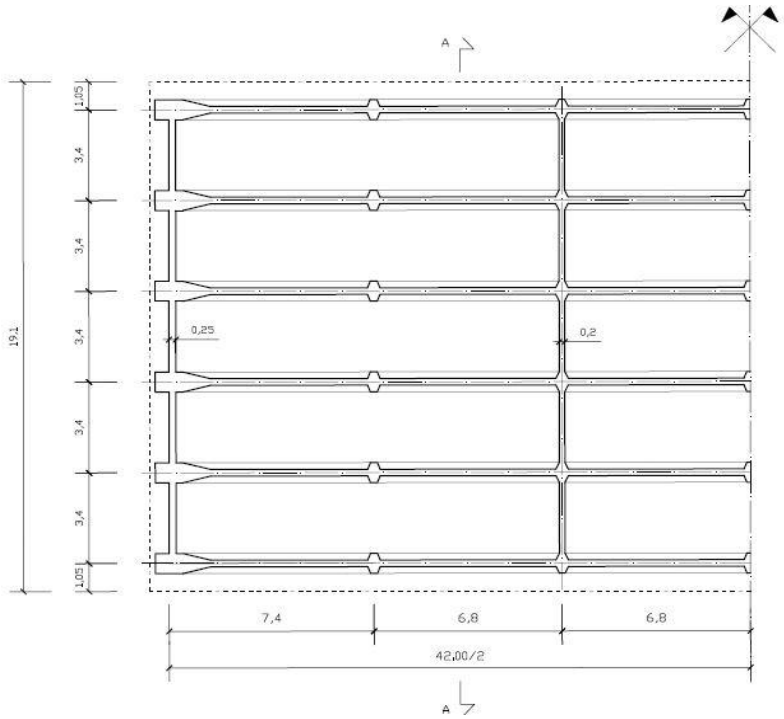


Figura 5.3 – Planimetria dell’impalcato

L’impalcato (Figura 5.3), largo complessivamente 19.10 m, è realizzato per mezzo di sei travi longitudinali in calcestruzzo armato precompresso con sezione a “I”, solidarizzate da una soletta gettata in opera dello spessore di 20 cm e da quattro traversi. L’impalcato è in semplice appoggio sulle pile con una distanza longitudinale tra gli appoggi pari a 42 m.

Le travi longitudinali sono posizionate con un interasse di 3.40 m e hanno tutte la medesima sezione trasversale di altezza 2.60 m (Figura 5.4). Esse presentano dei ringrossi in corrispondenza degli appoggi, dei traversi di campata e nei tratti intermedi tra un traverso e un altro.

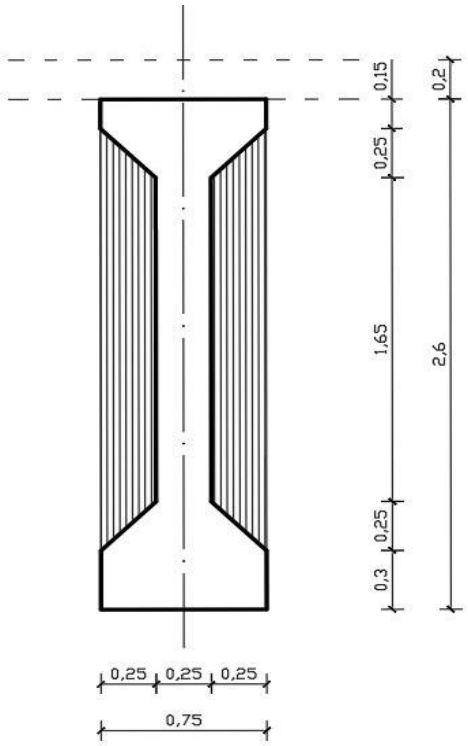


Figura 5.4 – Sezione trasversale trave longitudinale

I traversi, due di testata e due di campata (Figura 5.5 e Figura 5.6), hanno sezione rettangolare di altezza 2.30 m e spessore rispettivamente di 25 cm e 20 cm.

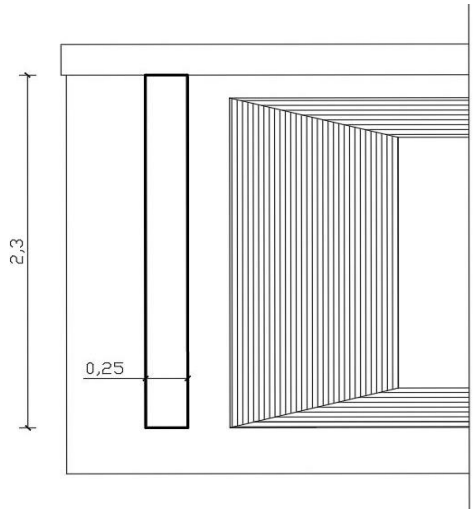


Figura 5.5 - Sezione trasversale traverso di testata

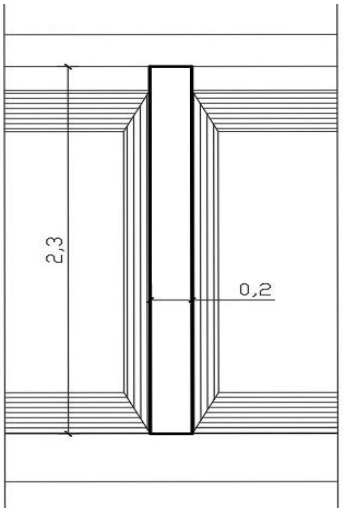


Figura 5.6 - Sezione trasversale traverso di campata

La sezione stradale è realizzata a "schiena d'asino" mediante un opportuno posizionamento delle travi longitudinali (Figura 5.7). In particolare, le due travi centrali si trovano alla stessa quota. Spostandosi verso i bordi esterni dell'impalcato, ogni trave è posizionata ad una quota più bassa di 10 cm rispetto alla precedente. In questo modo, la trave di bordo sarà 20 cm più in basso rispetto alle due travi centrali.

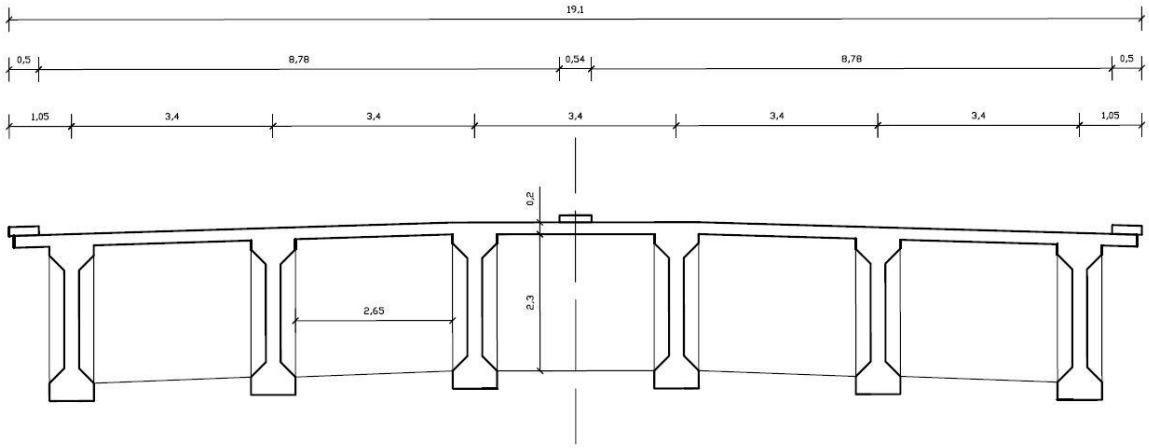


Figura 5.7 . Sezione trasversale A-A dell'impalcato

5.3 Sistema di monitoraggio installato

Il viadotto dispone di un sistema di monitoraggio installato su alcune campate, composto da clinometri a barra di lunghezza pari a 2 m (§1.3) ed estensimetri a corda vibrante. Tali sensori sono distribuiti tra le varie travi longitudinali del generico impalcato, in particolare, nella campata oggetto di studio, sono installati solo sulle quattro travi longitudinali centrali. Si riporta di seguito la disposizione dei sensori presenti sulla campata oggetto di studio; i clinometri sono rappresentati in verde mentre gli estensimetri in rosso.

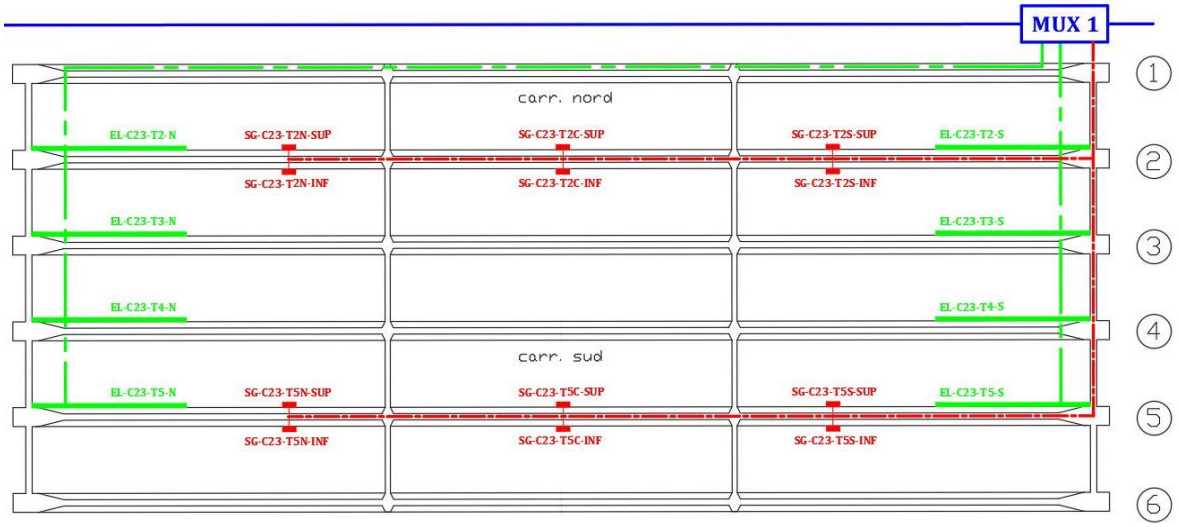


Figura 5.8 – Sensori del sistema di monitoraggio

Per ciascuna delle travi longitudinali è installato un clinometro a barra sul lato nord della campata e uno sul lato sud. Gli estensimetri invece sono installati sulle travi longitudinali adiacenti alle travi longitudinali di bordo, indicate in Figura 5.8 con i numeri 2 e 5.

Ogni clinometro viene indicato con un codice identificativo del tipo EL-C23-T2-N, in cui il primo gruppo di lettere specifica il tipo di strumento (EL = clinometro, SG = estensimetro), il secondo e il terzo gruppo determinano rispettivamente il numero della campata e il numero della trave, mentre il quarto e ultimo gruppo specifica il lato della campata su cui è posizionato (nord o sud).

5.4 Prove di carico

Sono state eseguite due prove di carico sull'impalcato al fine di tarare la strumentazione installata. Entrambe le prove di carico sono state suddivise in due diverse fasi (fase 1 e fase 2), aventi come scopo quello di generare i massimi effetti sull'impalcato: la fase 1 prevede l'introduzione di una porzione del carico totale che verrà poi completato nella successiva fase 2. Inoltre, in entrambe le prove, l'eccentricità del carico riferita all'asse dei veicoli rispetto al centro dell'impalcato è pari a 6,80 m.

5.4.1 Prova di carico con mezzi leggeri

La prima prova è stata effettuata con mezzi leggeri (Figura 5.9), impiegando 8 autovetture di peso pari a 1.6 ton ciascuna. La fase 1 ha previsto l'introduzione di quattro autovetture allineate in colonna, disposte sulla corsia più esterna dell'impalcato in corrispondenza della mezzeria e in posizione spalla contro spalla (posto di guida in posizioni opposte). Nella fase 2, sono state posizionate le restanti quattro autovetture, sempre sulla corsia più esterna, accodandosi ai mezzi posizionati nella fase 1 e col medesimo verso del mezzo precedente. Terminata la prima fase di carico, si effettuano le letture su ognuno degli undici punti di stazione, a distanza di 10 minuti l'una dall'altra, per un totale di 30 minuti. Al termine della seconda fase di carico, le letture sono ripetute in maniera analoga. Infine, si procede con la fase di scarico dell'impalcato, che si sviluppa analogamente alla fase di carico. Al termine di questo vengono effettuate letture per 60 minuti.

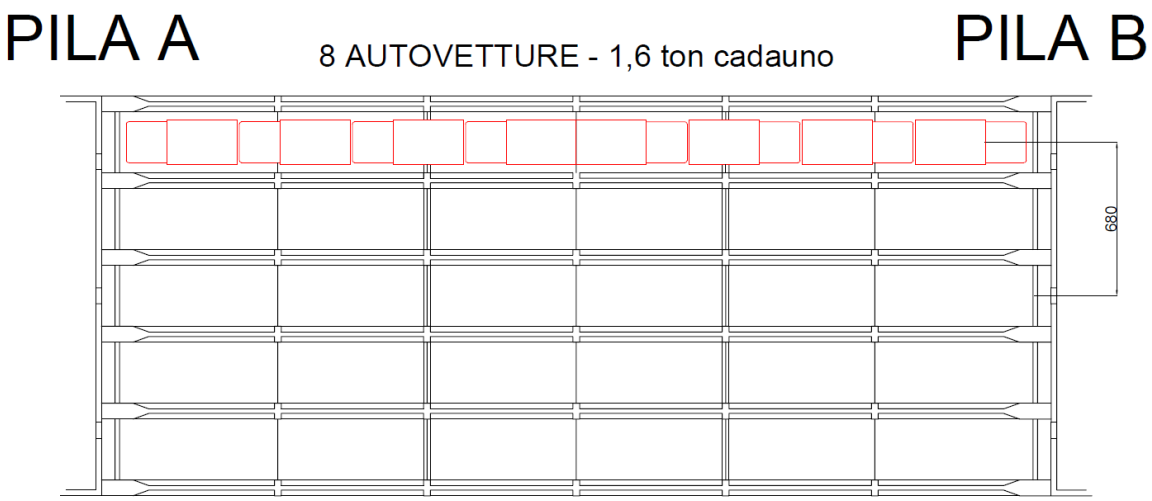


Figura 5.9 - Disposizione delle autovetture per prova di carico con mezzi leggeri

5.4.2 Prova di carico con mezzi pesanti

La seconda prova invece è stata effettuata con mezzi pesanti (Figura 5.10), impiegando 4 automezzi di peso pari a 34 ton ciascuno. Tale prova si è svolta secondo le medesime modalità e procedure previste per la prova con mezzi leggeri, solamente che questa volta nella fase 1 vengono posizionati solo 2 veicoli pesanti in posizione spalla contro spalla, che diventano quattro nella fase 2.

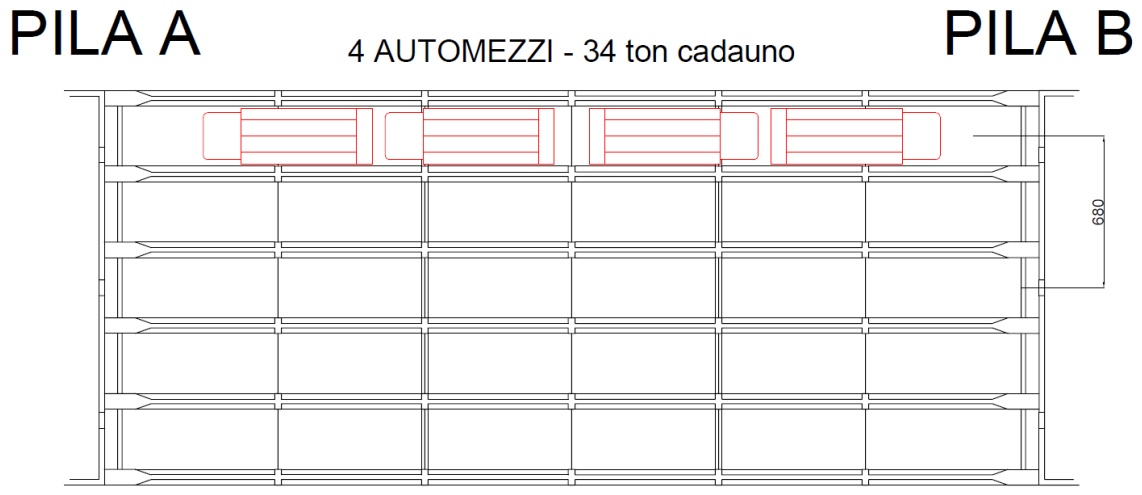


Figura 5.10 - Disposizione degli automezzi per prova di carico con mezzi pesanti

5.4.3 Misurazioni

Durante le prove di carico sono state registrate le misurazioni effettuate dagli inclinometri e dagli estensimetri, la temperatura e sono stati monitorati anche gli spostamenti verticali di undici punti di controllo.

Le rotazioni degli inclinometri a barra sono riportate nel grafico in appendice A3; esso riporta l'evoluzione temporale delle rotazioni evidenziando le singole fasi della prova di carico. Si è scelto come verso positivo della rotazione quello mostrato in Figura 5.11, avendo assunto negative le rotazioni che comportano un'inflessione verso il basso dell'impalcato.

Figura 5.11 – Verso positivo delle rotazioni

Nella tabella seguente (Tab. 5.1) si riportano le rotazioni massime registrate da ciascun inclinometro durante la fase 2 della prova con mezzi pesanti. Queste rappresentano le rotazioni massime che le travi dell'impalcato hanno subito durante l'intera prova di carico e verranno prese come riferimento nella fase di modellazione successiva. Per la posizione e il codice identificativo degli inclinometri si rimanda alla Figura 5.8.

Tab. 5.1 - Rotazioni massime registrate

Trave longitudinale	**Inclinometri**	**Rotazioni misurate [mrad]**
T2	EL-C23-T2-N	-0,53
	EL-C23-T2-S	-0,52
T3	EL-C23-T3-N	-0,36
	EL-C23-T3-S	-0,32
T4	EL-C23-T4-N	-0,15
	EL-C23-T4-S	-0,2
T5	EL-C23-T5-N	≃0
	EL-C23-T5-S	≃0

Gli spostamenti verticali degli undici punti di controllo sono stati monitorati attraverso una livellazione topografica di precisione dall'estradosso dell'impalcato (Figura 5.12). I punti osservati sono organizzati in due file di cinque punti ciascuna lungo i bordi dell'impalcato e da un punto isolato al centro dell'impalcato stesso. I cinque punti sul bordo sono divisi in due punti in corrispondenza degli appoggi (x=0, x=L), due punti nei quarti della campata (x=L/4, x=3L/4), ed uno in mezzeria (x=L/2).

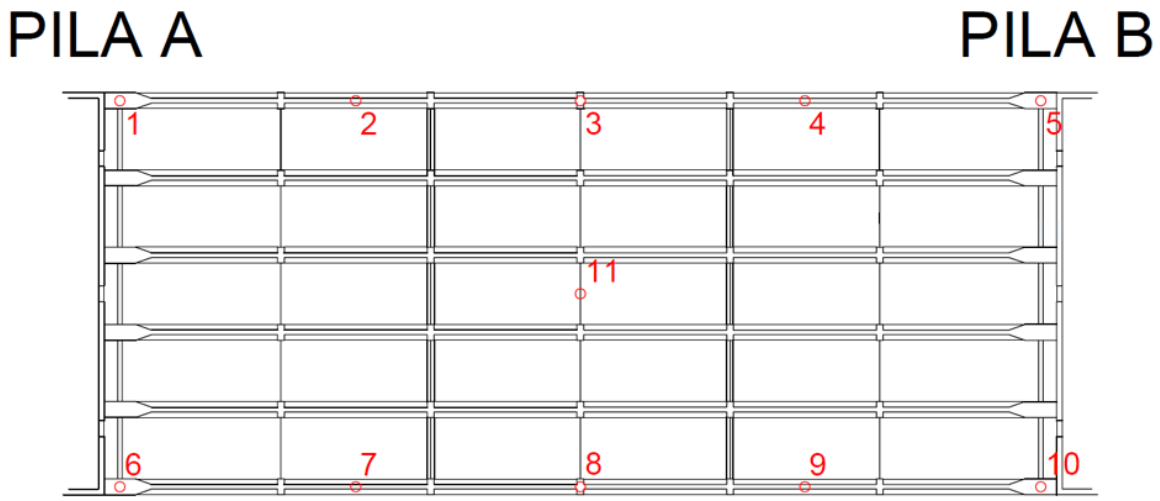


Figura 5.12 – Punti di controllo per il monitoraggio degli spostamenti verticali

I dati registrati dagli estensimetri e gli spostamenti verticali dei punti non sono stati analizzati in quanto non disponibili.

5.5 Modellazione dell'impalcato

L'impalcato oggetto di studio è stato modellato come un grigliato tridimensionale costruito mediante la libreria python PyNiteFEA analizzata nei capitoli 3 e 4. La modellazione è stata sviluppata partendo da un modello semplificato e aumentando di volta in volta il livello di dettaglio fino ad arrivare al modello definitivo. La validazione di quest'ultimo è stata effettuata facendo riferimento alle massime rotazioni misurate dagli inclinometri durante le prove di carico e riportate in Tab. 5.1. Il modello si compone di una serie di elementi longitudinali e trasversali interconnessi rigidamente e rappresentanti le travi longitudinali, i traversi di testata e di campata e la soletta. Tutti gli elementi del grigliato sono stati modellati utilizzando elementi trave monodimensionali e attribuendo ad essi le relative proprietà.

Il sistema di riferimento globale XYZ è stato definito nel seguente modo (Figura 5.13):

- X: direzione longitudinale dell'impalcato;
- Y: direzione trasversale dell'impalcato;
- Z: direzione verticale di simmetria

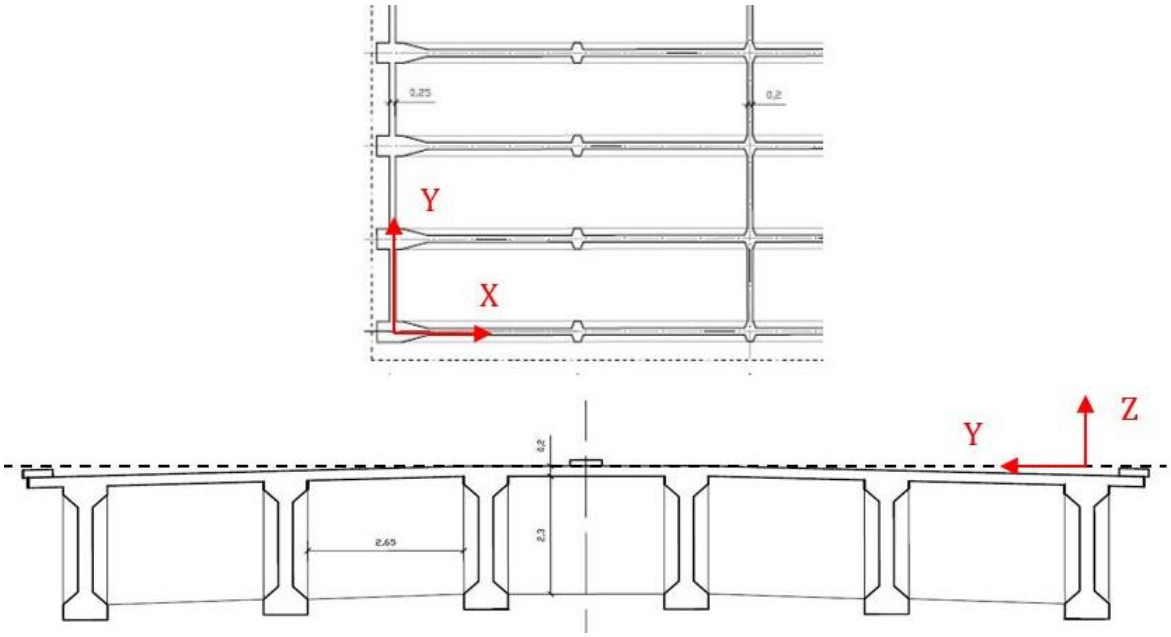


Figura 5.13 - Sistema di riferimento globale

Nei paragrafi successivi, si farà riferimento al lato nord e al lato sud dell'impalcato e alla seguente numerazione delle travi longitudinali e trasversali (Figura 5.14):

Figura 5.14 – Numerazione travi longitudinali e trasversali

5.5.1 Materiali

Le proprietà dei materiali sono state ottenute mediante indagini in situ con il prelievo di carote cilindriche di calcestruzzo e provini di armatura metallica per le armature lente e per i trefoli di pretensione. Nessun campione è stato prelevato per i cavi di post tensione. Dai risultati di laboratorio sono state effettuale le operazioni di media, ottenendo così le proprietà in situ dei

materiali indagati. Questi sono stati utilizzati per il calcolo di altri parametri necessari per la fase di calcolo e modellazione (ad esempio il modulo elastico E del calcestruzzo) con riferimento alle formulazioni riportate in Tabella 3.1 dell'EN1992-1-2 [17].

Tab. 5.2 - Proprietà calcestruzzo trave prefabbricata

Descrizione	Simbolo	Valore	Unità
Resistenza a compressione cilindrica ottenuta da prove in sito	$f_{cm,\ situ}$	57	MPa
Fattore di confidenza	FC	1	-
Resistenza a compressione media usata nei calcoli ($f_{cm,situ}$/FC)	$f_{cm}(t)$	57	MPa
Modulo elastico	E_{cm} (t)	37026	MPa

Tab. 5.3 – Proprietà calcestruzzo soletta

Descrizione	Simbolo	Valore	Unità
Resistenza a compressione cilindrica ottenuta da prove in sito	$f_{cm,\ situ}$	49	MPa
Fattore di confidenza	FC	1	-
Resistenza a compressione media usata nei calcoli ($f_{cm,situ}$/FC)	$f_{cm}(t)$	49	MPa
Modulo elastico	E_{cm} (t)	35336	MPa

Tab. 5.4 – Proprietà armatura lenta

Descrizione	Simbolo	Valore	Unità
Tensione di snervamento da prove in situ	$f_{ym,\ situ}$	417	MPa
Modulo elastico	E_s	200000	MPa

Tab. 5.5 – Proprietà trefoli pretesi

Descrizione	Simbolo	Valore	Unità
Tensione di rottura da prove in sito	$f_{ptm,\ situ}$	1818	MPa
Modulo elastico	$E_{p,tr}$	195000	MPa

Tab. 5.6 - Proprietà cavi post tesi

Descrizione	Simbolo	Valore	Unità
Tensione di rottura da prove in sito	$f_{ptm,\ situ}$	nd	MPa
Modulo elastico	$E_{p,ten}$	195000	MPa

In riferimento al calcestruzzo armato, noto il modulo di elasticità E del calcestruzzo e fissato il coefficiente di Poisson ν=0.2, si è ricavato il modulo di elasticità tangenziale G con la relazione:

$$G = \frac{E}{2(1 + \nu)}$$

5.5.2 Condizioni di vincolo

Ciascuna delle travi longitudinali è vincolata alle estremità con delle condizioni di vincolo che si differenziano per la direzione longitudinale e trasversale.

Figura 5.15 - Appoggi

I nodi del modello sono stati vincolati così come si presentano nella realtà. Il grado di vincolo per ogni nodo è riportato in Tab. 5.7 con riferimento alla Figura 5.15:

Tab. 5.7 – Condizioni di vincolo (True=bloccato, False=libero)

Nodo	DX	DY	DZ	RX	RY	RZ
V11	True	False	True	False	False	False
V21	True	False	True	False	False	False
V31	True	True	True	False	False	False
V41	True	False	True	False	False	False
V51	True	False	True	False	False	False
V61	True	False	True	False	False	False

Nodo	DX	DY	DZ	RX	RY	RZ
V14	False	False	True	False	False	False
V24	False	False	True	False	False	False
V34	False	True	True	False	False	False
V44	False	False	True	False	False	False
V54	False	False	True	False	False	False
V64	False	False	True	False	False	False

5.5.3 Carichi applicati

La validazione del modello è stata effettuata facendo riferimento alle massime rotazioni relative alla fase 2 della prova di carico con mezzi pesanti (§5.4.2). A tal fine, non è stato necessario applicare al modello né il peso proprio dell'impalcato e né l'effetto della precompressione, ma è stato sufficiente considerare i carichi della prova, cioè 4 autocarri di 34 ton ciascuno disposti sulla corsia più esterna dell'impalcato in corrispondenza della mezzeria. Il peso totale del singolo autocarro, modellato tramite forze concentrate, è ripartito in 8 ton sull'asse anteriore e 13 ton sui due assi posteriori. Inoltre, al fine di disporre i mezzi di prova nella medesima configurazione descritta al paragrafo 5.4.2, si è assunta una sagoma del mezzo illustrata in Figura 5.16.

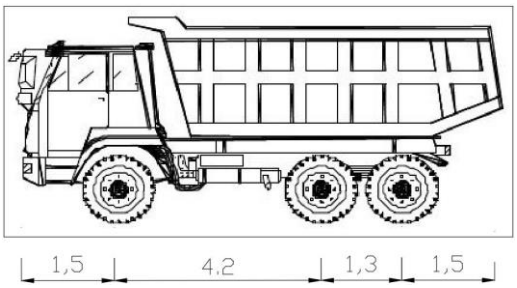


Figura 5.16 – Sagoma autocarro

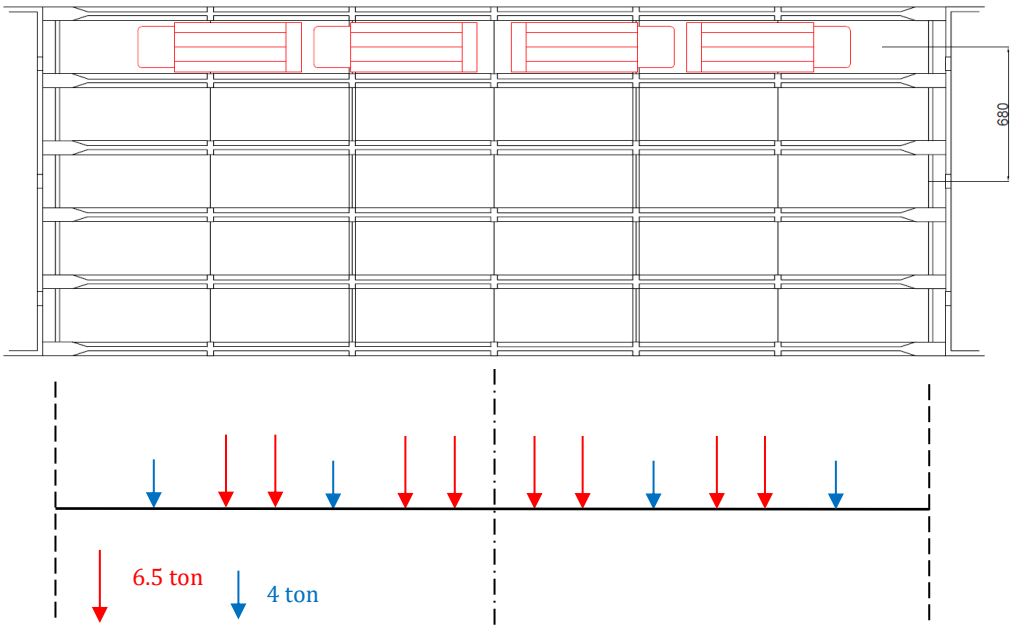


Figura 5.17 – Schema di carico trave longitudinale 5 e 6

Il carico trasmesso dagli autocarri è stato ripartito alle travi longitudinali 5 e 6.

5.5.4 Rotazioni calcolate dal modello

Per ottenere i valori di rotazione confrontabili con quelli misurati dal sistema di monitoraggio, è stato necessario definire apposite righe di codice. Gli inclinometri a barra, infatti, misurano l'inclinazione relativa tra il punto iniziale e finale della barra metallica. Da fonti fotografiche, si evince che l'inclinometro presenta il punto iniziale molto vicino all'appoggio della trave e il punto finale invece, distante 2 m dal precedente. Nel modello dell'implacato si può considerare il punto iniziale coincidente con il primo nodo della trave longitudinale e il nodo finale ricadente all'interno di un certo elemento trave (Figura 5.18).

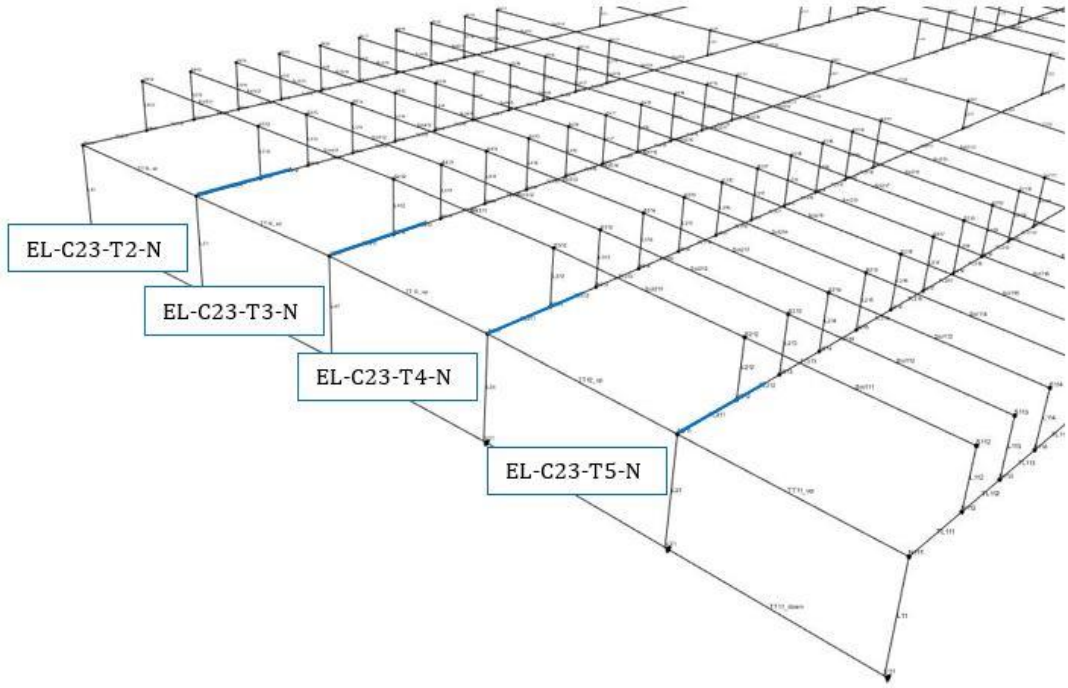


Figura 5.18 - Posizione degli inclinometri lato nord nel modello

In definitiva, la rotazione relativa tra il punto iniziale e il punto finale è stata calcolata come rapporto tra la differenza degli spostamenti verticali dei due punti e la lunghezza dell'inclinometro stesso.

5.5.5 Costruzione modello

Il modello iniziale consiste in un grigliato piano (Figura 5.19) costruito sulla base di varie ipotesi semplificative. Le travi longitudinali, i traversi e la soletta vengono modellati mediante elementi trave e per quanto riguarda i materiali si trascura la differenza tra il modulo elastico della trave prefabbricata (37026 MPa) e quello della soletta (35336 MPa). Pertanto, per tutte le sezioni trasversali del modello verrà utilizzato il modulo elastico del cls della trave prefabbricata; si adottano di conseguenza i seguenti moduli elastici:

E = 37026 MPa

G = 15428 MPa

Le proprietà geometriche e inerziali degli elementi fanno riferimento a sezioni interamente in cls, non considerando quindi la differenza di proprietà dei calcestruzzi e la presenza dell'armatura lenta e da precompressione. Inoltre, le condizioni di vincolo sono applicate ai nodi estremi degli elementi che rappresentano gli assi baricentrici delle travi longitudinali. Nell'immagine seguente è possibile osservare i nodi, gli elementi trave e i nodi ausiliari utilizzati per la costruzione del modello stesso.

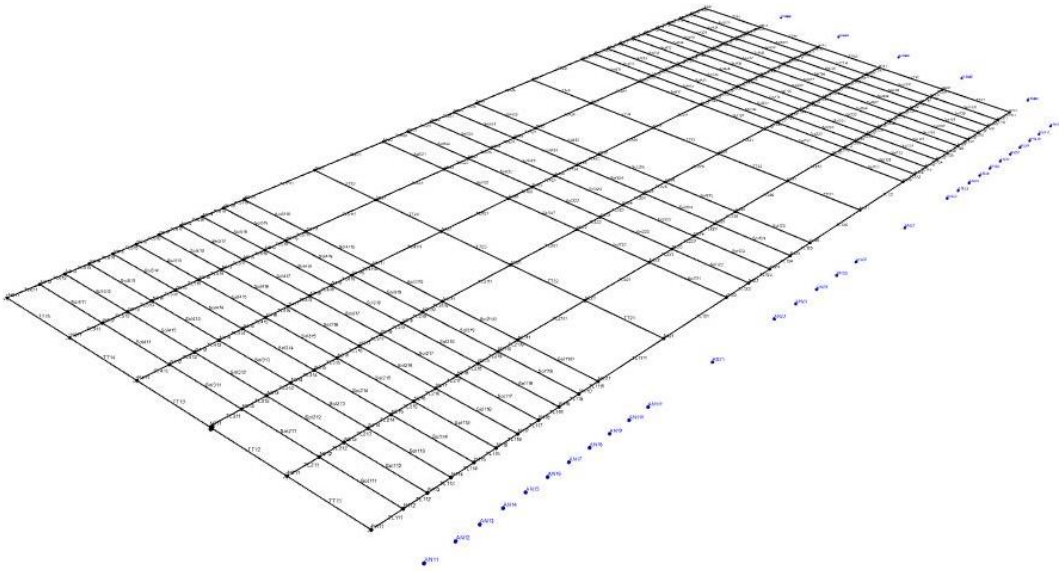


Figura 5.19 – Modello iniziale semplificato

La presenza della soletta necessita qualche considerazione in più riguardo la sezione resistente delle travi longitudinali e trasversali. Essa, infatti, rappresenta un elemento strutturale bidimensionale e come tale dovrebbe essere modellato al fine di considerare correttamente il suo contributo nel comportamento globale dell'impalcato. Il concetto di larghezza di soletta efficace (o larghezza collaborante), permette di semplificare notevolmente il modello di calcolo evitando l'utilizzo di elementi bidimensionali. La norma 1992-1-1:2004 fornisce le relazioni per il calcolo della larghezza efficace nel paragrafo 5.3.2.1 [17]:

$$b_{eff} = \sum b_{eff,i} + b_w \leq b$$

$$b_{eff,i} = 0,2 b_i + 0,1 l_0 \leq 0,2 l_0$$

$$b_{eff,i} \leq b_i$$

l_0 = distanza tra i punti di momento nullo

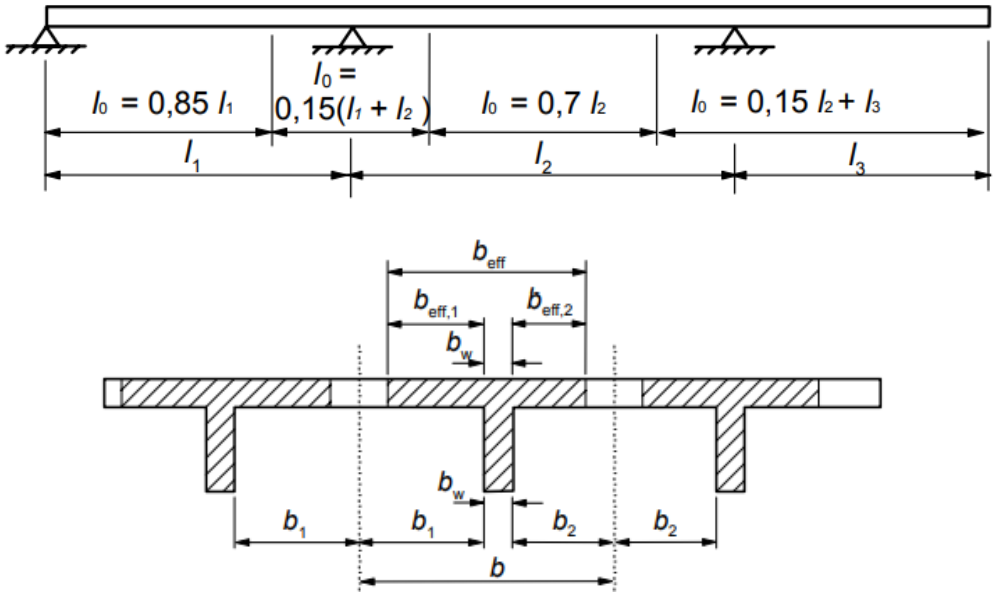


Figura 5.20 - Larghezza di soletta collaborante (EC2)

Si sono determinate così le larghezze di soletta collaborante per tutti i casi, ottenendo le sezioni trasversali delle travi. Le inerzie flessionali I_y e I_z sono state calcolate con le relazioni della geometria delle aree. L'inerzia torsionale I_t è stata calcolata con la formulazione seguente relativa alle sezioni rettangolari [18]:

$$I_t = \frac{b \cdot a^3}{\beta} \qquad \beta \cong \left(1 + \frac{1}{n^2}\right)\left[3.56 - 0.56\left(\frac{n-1}{n+1}\right)^2\right]$$

con *a* lato maggiore, *b* lato minore ed *n=b/a*. Per le sezioni a "T", essendo la flangia costituita dalla soletta che nella realtà è un elemento bidimensionale continuo, l'inerzia torsionale è data dal contributo dell'anima più il 50% di quello della flangia. Si assume quindi che:

$$I_t = I_{t,web} + \frac{1}{2} I_{t,flange}$$

Si riportano di seguito le sezioni trasversali e le relative proprietà.

1) Travi longitudinali di bordo

Per le travi longitudinali di bordo è stata considerata una lunghezza l_0 pari alla distanza tra gli appoggi dell'impalcato, ovvero 42 m, ottenendo una larghezza di soletta collaborante di a 2.75 m.

Tab. 5.8 – Proprietà travi longitudinali di bordo

A [m²]	**I_y [m⁴]**	**I_z [m⁴]**	**I_t [m⁴]**
1.550	0.4086	1.567	0.0343

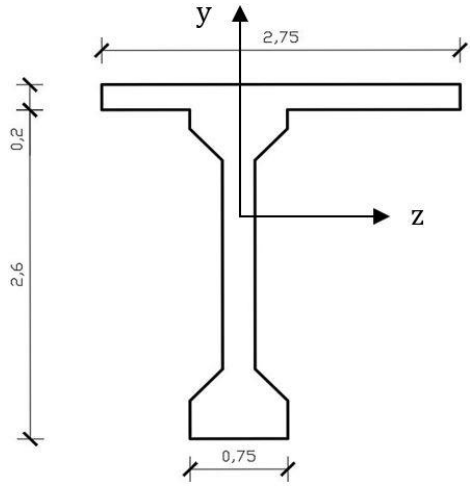


Figura 5.21 – Travi longitudinali di bordo (TL1 e TL6)

2) Travi longitudinali interne

Per le travi longitudinali interne è stata usata una lunghezza l_0 pari alla distanza tra gli appoggi dell'impalcato, ovvero 42 m, ottenendo una larghezza di soletta collaborante di a 3.40 m (uguale all'interasse tra le travi longitudinali).

Tab. 5.9 – Proprietà travi longitudinali interne

A [m²]	**I_y [m⁴]**	**I_z [m⁴]**	**I_t [m⁴]**
1.680	0.680	1.676	0.0352

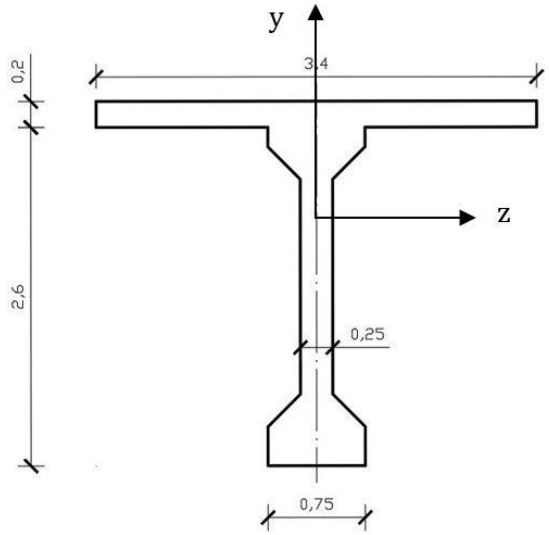


Figura 5.22 - Travi longitudinali interne (TL2, TL3, TL4, TL5)

3) Traversi di campata

Per i traversi di campata è stata considerata una lunghezza l_0 pari a 17 m pari a quella che si avrebbe se fossero vincolati in semplice appoggio sulle travi longitudinali esterne, ottenendo una larghezza di soletta collaborante di 6.33 m. In realtà il calcolo della lunghezza l_0 dei traversi presenta notevoli difficoltà poiché essi risultano vincolati su appoggi elastici forniti dalle travi longitudinali. Si ritiene che la stima effettuata fornisca un valore di larghezza collaborante prossimo o sottostimato rispetto a quello reale.

Tab. 5.10 – Proprietà traversi di campata

A [m²]	**I_y [m⁴]**	**I_z [m⁴]**	**I_t [m⁴]**
1.726	4.230	0.734	0.138

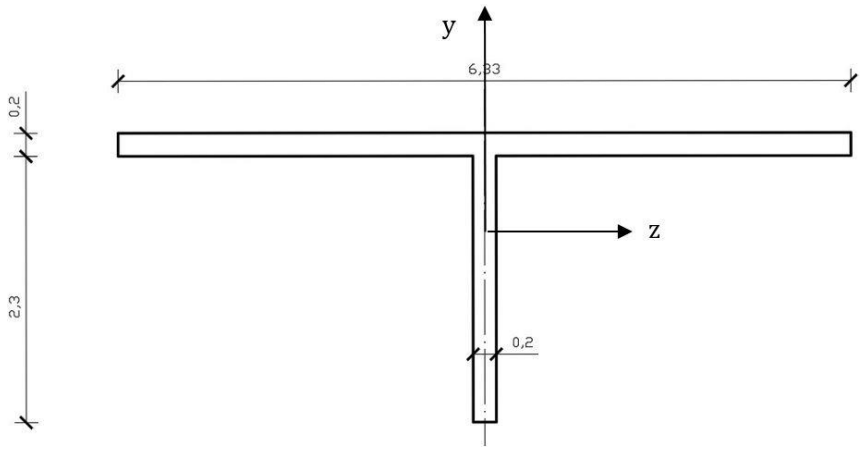


Figura 5.23 – Traversi di campata (TT2 e TT3)

5) Traversi di testata

Infine, per i traversi di testata è stata considerata una lunghezza l_0 pari a 3.4 m, in quanto risentono del vincolo diretto sugli appoggi, ottenendo una larghezza di soletta collaborante di 1.40 m.

Tab. 5.11 – Proprietà traversi di testata

A [m²]	**I_y [m⁴]**	**I_z [m⁴]**	**I_t [m⁴]**
0.855	0.050	0.548	0.126

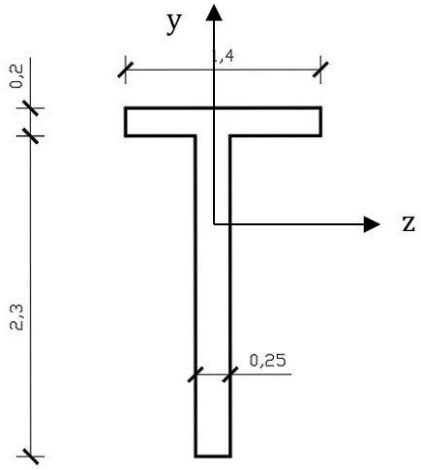


Figura 5.24 – Traversi di testata (TT1 e TT4)

6) Soletta

La soletta, di spessore 0.20 m, è stata modellata mediante elementi trave trasversali collegati alle travi longitudinali. È importante sottolineare che tali elementi sono stati introdotti per modellare la distribuzione trasversale dei carichi verticali. Per un'analisi della soletta si rimanda all'utilizzo di un modello locale in grado di rappresentare correttamente il comportamento in caso di carichi concentrati. La sezione di questi elementi è stata determinata andando a discretizzare in un numero finito di parti i tratti di campata compresi tra un traverso e il successivo, sottraendo le larghezze di soletta collaborante che competono ai traversi stessi. In questo modo, la larghezza di soletta collaborante facente parte della sezione trasversale dei traversi non viene computata due volte. Il tratto compreso tra i traversi TT1-TT2 e TT3-TT4 è stato diviso in 10 parti, ottenendo elementi di soletta di larghezza 1.01 m (Figura 5.25).

Tab. 5.12 – Proprietà soletta 1

A [m²]	**I_y [m⁴]**	**I_z [m⁴]**	**I_t [m⁴]**
0.203	0.017	0.0007	2.36E-3

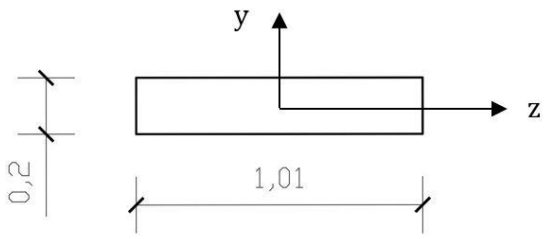


Figura 5.25 – Soletta 1

Il tratto compreso tra i traversi TT2-TT3 è stato invece diviso in 5 parti, ottenendo elementi di soletta di larghezza 1.45 m (Figura 5.26).

Tab. 5.13 – Proprietà soletta 2

A [m²]	**I_y [m⁴]**	**I_z [m⁴]**	**I_t [m⁴]**
0.291	0.051	0.001	3.53E-3

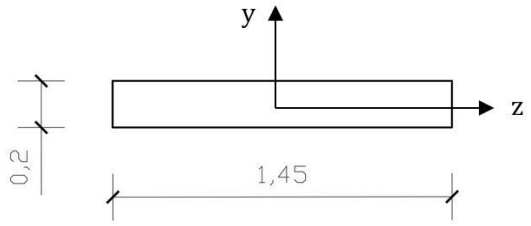


Figura 5.26 – Soletta 2

Tutti gli elementi dell'impalcato sono stati modellati con riferimento all'asse baricentrico. In particolare, nel primo modello, sono stati trascurati gli offset verticali tra i vari assi baricentrici derivanti dalla differenza delle sezioni trasversali. In questo modo, si è realizzato un grigliato interamente giacente sul piano XY del sistema di riferimento globale. Definite le sezioni trasversali, esse sono state assegnate ai rispettivi elementi trave del modello, definendo per ciascuno di essi l'orientamento del sistema di riferimento locale (§3.4.2). Si riportano per completezza gli orientamenti scelti:

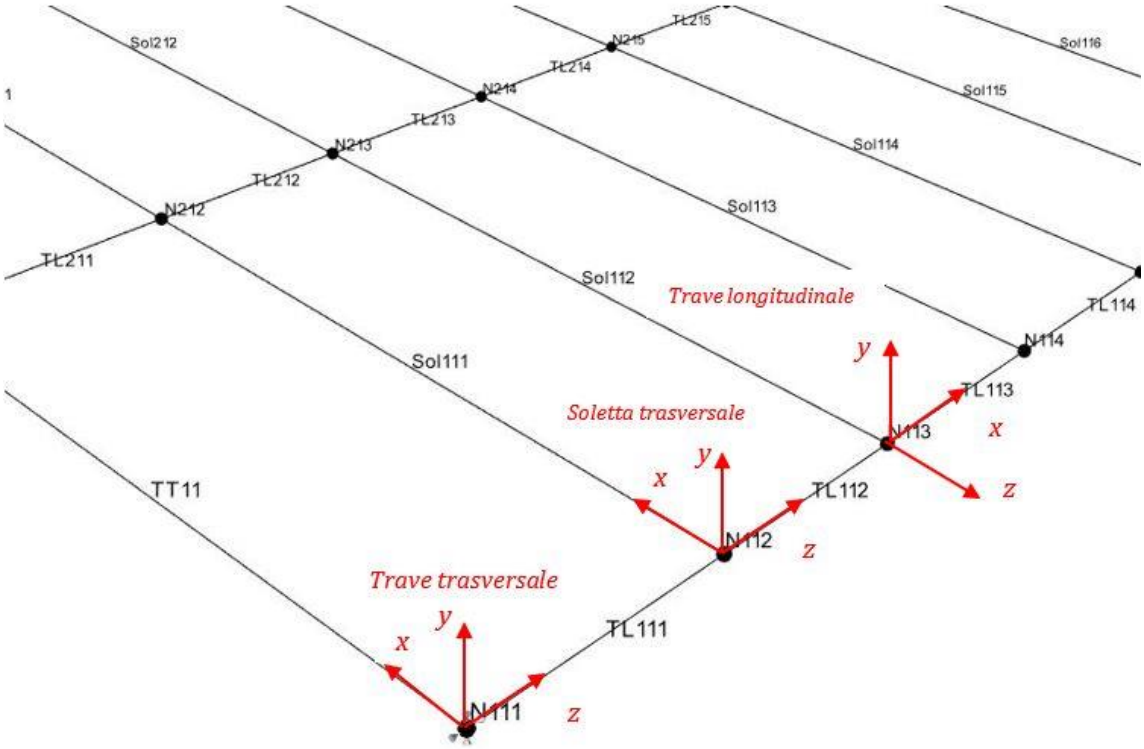


Figura 5.27 – Sistemi di riferimento locali

I risultati della prima analisi hanno dimostrato un'eccessiva deformabilità del modello iniziale con valori di rotazione maggiori rispetto ai valori registrati.

Per ogni trave longitudinale monitorata, si sono calcolate le variazioni percentuali tra le rotazioni ottenute dal modello e quelle misurate, rapportando questa differenza alle rotazioni misurate per la trave longitudinale 2 (-0.53 mrad nel lato Nord, -0.52 mrad nel lato Sud). I risultati sono riportati in Tab. 5.14 e Tab. 5.15.

Tab. 5.14 – Confronto delle rotazioni lato nord

Inclinometri	Misura [mrad]	Modello [mrad]	Variazione %
EL-C23-T2-N	-0.53	-0.66	25
EL-C23-T3-N	-0.36	-0.42	11
EL-C23-T4-N	-0.15	-0.20	9
EL-C23-T5-N	≃0	≃0	-

Tab. 5.15 - Confronto delle rotazioni lato sud

Inclinometri	Misura [mrad]	Modello [mrad]	Variazione %
EL-C23-T2-S	-0.52	-0.66	27
EL-C23-T3-S	-0.32	-0.42	19
EL-C23-T4-S	-0.20	-0.20	0
EL-C23-T5-S	≃0	≃0	-

Sulla base di questi primi risultati ottenuti, si è proceduto con un miglioramento del modello.

5.5.5.1 Introduzione di link rigidi per simulare offset degli assi baricentrici

I link rigidi sono stati inseriti per modellare la differenza di quota tra gli assi baricentrici degli elementi trave, traversi e soletta. Essi hanno il compito di garantire la trasmissione delle sollecitazioni tra gli elementi a cui sono collegati. Sono stati modellati mediante elementi travi con una elevata rigidezza. Questa viene conferita assegnando loro moduli di elasticità E ed G mille volte più grandi rispetto ai valori del calcestruzzo impiegato e sezione trasversale con caratteristiche unitarie:

Tab. 5.16 – Proprietà link rigidi

A [m²]	**I_y [m⁴]**	**I_z [m⁴]**	**I_t [m⁴]**
1	1	1	1

Si è visto che le proprietà della sezione trasversale, in termini di area e inerzia, influenzano in maniera irrilevante i risultati finali a fronte degli elevati moduli di rigidezza assegnati.

In termini di modellazione sono state fatte le seguenti ipotesi semplificative:

- Si trascura l'offset tra le travi longitudinali interne e quelle di bordo;
- Si trascura l'offset tra le travi longitudinali interne e i traversi di testata;

Di conseguenze, sono stati modellati gli offset tra le travi longitudinali e la soletta trasversale, e quelli tra le travi longitudinali e i traversi di campata (). L'introduzione di questi elementi conferisce maggiore rigidezza al modello.

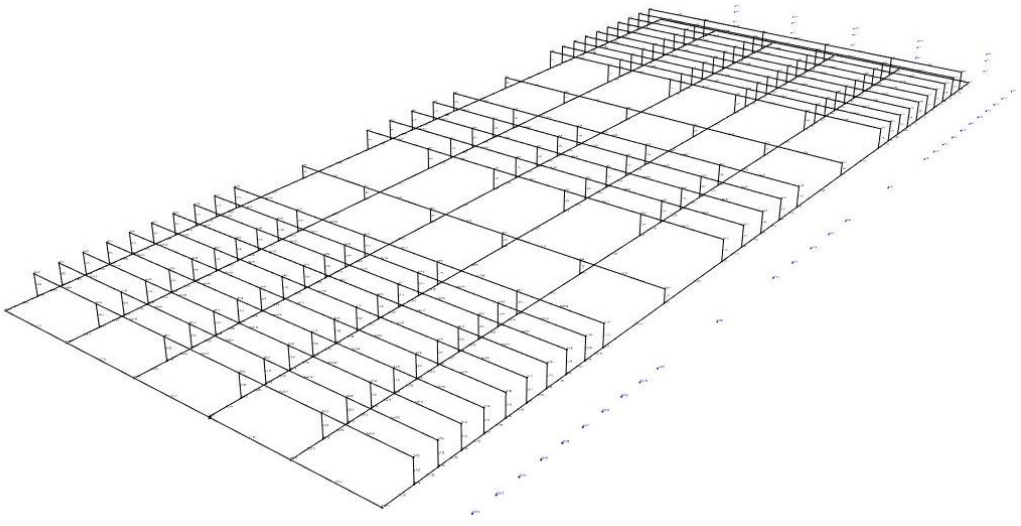


Figura 5.28 – Introduzione link rigidi

Per questi elementi non è stato definito un determinato sistema di riferimento locale in quanto non necessario né per l'applicazione di carichi, né per la gestione delle proprietà inerziali. L'introduzione di questi link link sostanzialmente non cambia la risposta del modello, aumentandone in modo trascurabile la rigidezza. In termini numerici, si ha una leggera diminuzione delle rotazioni, la quale appare più evidente per la trave longitudinale 2 e meno per le restanti travi monitorate (variazione delle cifre decimali successive).

Tab. 5.17 – Confronto delle rotazioni lato nord

Inclinometri	Misura [mrad]	Modello [mrad]	Variazione %
EL-C23-T2-N	-0.53	-0.65	23
EL-C23-T3-N	-0.36	-0.42	11
EL-C23-T4-N	-0.15	-0.20	9
EL-C23-T5-N	≃0	≃0	-

Tab. 5.18 - Confronto delle rotazioni lato sud

Inclinometri	Misura [mrad]	Modello [mrad]	Variazione %
EL-C23-T2-S	-0.52	-0.65	25
EL-C23-T3-S	-0.32	-0.42	19
EL-C23-T4-S	-0.20	-0.20	0
EL-C23-T5-S	≃0	≃0	-

Sulla base di questi risultati ottenuti, si è proceduto con un ulteriore miglioramento del modello.

5.5.5.2 Introduzione dei valori di aree e inerzie omogeneizzate per le sezioni

La generica sezione trasversale delle travi è composta da materiali con proprietà differenti (calcestruzzi e armature metalliche). Per tenere conto di ciò, le proprietà delle sezioni trasversali sono state calcolate mediante un'operazione di omogeneizzazione in cui si è fatto riferimento ad una sezione fittizia costituita dal solo calcestruzzo della trave prefabbricata. Per fare ciò, tutti i contributi in termini di area e inerzie della soletta e delle armature metalliche sono stato moltiplicate per il rispettivo coefficiente di omogeneizzazione n_i dato dal rapporto dei moduli elastici. L'operazione di omogeneizzazione ha riguardato le sezioni trasversali delle travi longitudinali (interne e di bordo) e dei traversi di campata. Per i traversi di testata non è stato possibile calcolare le grandezze omogeneizzata a causa della mancanza di informazioni dettagliate sull'armatura, invece per gli elementi di soletta trasversale si è valutato tale calcolo di poca influenza sui risultati finali del modello. Inoltre, con riferimento agli assi y e z locali del generico elemento trave, essendo prevalente il comportamento flessionale attorno all'asse z, si

è deciso di non calcolare l'inerzia I_y omogeneizzata; ciò richiederebbe un calcolo più oneroso che tenga conto della posizione trasversale delle varie armature.

Si riportano di seguito i coefficienti di omogeneizzazione rispetto al calcestruzzo della trave prefabbricata (E=37026 MPa):

- Coefficiente di omogeneizzazione soletta:

$$n_{sol} = 0.95$$

- Coefficiente di omogeneizzazione armatura lenta:

$$n_{str} = 5.40$$

- Coefficiente di omogeneizzazione trefoli pretesi:

$$n_{str} = 5.27$$

- Coefficiente di omogeneizzazione cavi post tesi:

$$n_{str} = 5.27$$

Di seguito invece le nuove proprietà delle sezioni trasversali in seguito all'omogeneizzazione:

Tab. 5.19 – Proprietà omogeneizzate trave longitudinale di bordo

	A [m²]	**I_y [m⁴]**	**I_z [m⁴]**	**I_t [m⁴]**
Proprietà non omogeneizzate	1.550	0.4086	1.567	0.0343
Proprietà omogeneizzate	1.580	0.4086	1.680	0.0342

Tab. 5.20 – Proprietà omogeneizzate travi longitudinali interne

	A [m²]	**I_y [m⁴]**	**I_z [m⁴]**	**I_t [m⁴]**
Proprietà non omogeneizzate	1.680	0.680	1.676	0.0352
Proprietà omogeneizzate	1.710	0.680	1.790	0.0350

Tab. 5.21 – Proprietà omogeneizzate traversi di campata

	A [m²]	**I_y [m⁴]**	**I_z [m⁴]**	**I_t [m⁴]**
Proprietà non omogeneizzate	1.726	4.230	0.734	0.138
Proprietà omogeneizzate	1.775	4.230	0.771	0.134

Confrontando le nuove proprietà con quelle utilizzate nel modello iniziale, si può notare un aumento dei valori e ciò si traduce in un ulteriore piccolo aumento di rigidezza dell'impalcato modellato. Le rotazioni delle travi longitudinali ottenute dall'analisi confermano quanto appena detto.

Tab. 5.22 – Confronto delle rotazioni lato nord

Inclinometri	Misura [mrad]	Modello [mrad]	Variazione %
EL-C23-T2-N	-0.53	-0.61	15
EL-C23-T3-N	-0.36	-0.39	6
EL-C23-T4-N	-0.15	-0.19	8
EL-C23-T5-N	≃0	≃0	-

Tab. 5.23 - Confronto delle rotazioni lato sud

Inclinometri	Misura [mrad]	Modello [mrad]	Variazione %
EL-C23-T2-S	-0.52	-0.61	17
EL-C23-T3-S	-0.32	-0.39	13
EL-C23-T4-S	-0.20	-0.19	-2
EL-C23-T5-S	≃0	≃0	-

5.5.5.3 Introduzione dell'inclinazione trasversale dell'impalcato

La sezione stradale a "schiena d'asino", realizzata mediante un opportuno posizionamento delle travi longitudinali, è stata modellata applicando delle traslazioni verticali agli assi baricentrici delle travi longitudinali.Le due travi centrali si trovano alla stessa quota, invece, spostandosi verso il bordo dell'impalcato, ogni trave è posizionata ad una quota più bassa di 10 cm rispetto alla precedente. In questo modo, la trave di bordo sarà 20 cm più in basso rispetto alle due travi centrali. Si ottiene il seguente modello (Figura 5.29); si noti che per motivi grafici è stata accentuata l'inclinazione della sezione stradale.

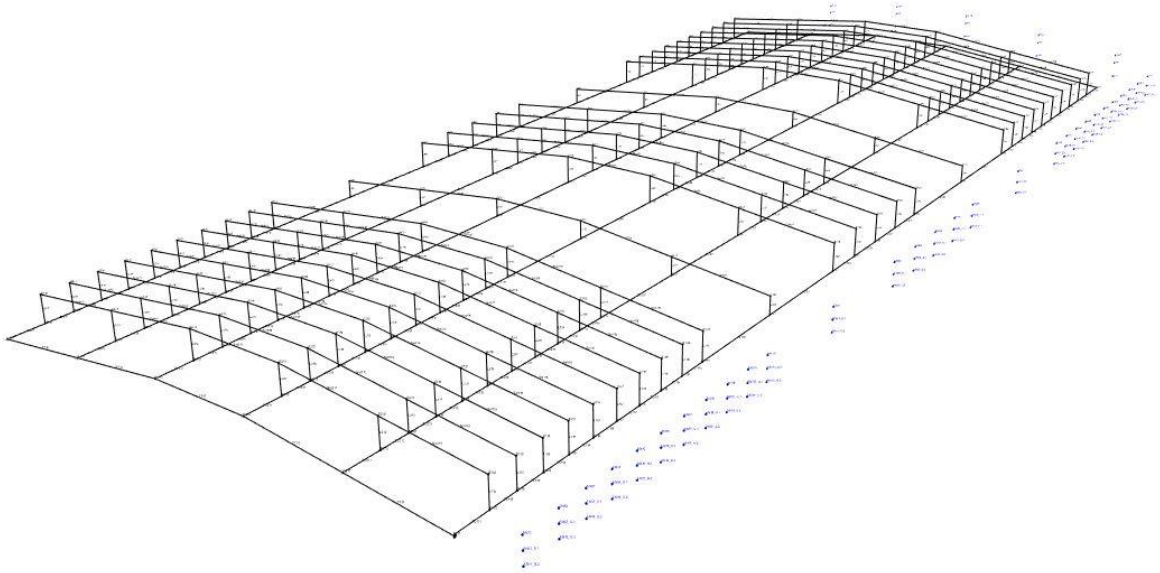


Figura 5.29 – Modellazione sezione stradale "a schiena d'asino"

Questa modifica al modello non comporta variazioni apprezzabili dei valori di rotazione che rimangono sostanzialmente uguali a quelli riportati nelle tabelle precedenti (Tab. 5.22 e Tab. 5.23).

5.5.5.4 Introduzione di link verticali di collegamento tra asse appoggi e asse travi

I vincoli considerati fino ad ora sono stati applicati in corrispondenza del baricentro delle travi longitudinali. Nella realtà, i dispositivi di vincolo sono posizionati ad una quota più bassa sul piano di appoggio della pila. Nella costruzione del modello si è tenuto conto di questo aspetto inserendo dei link di collegamento tra il punto effettivo di appoggio e l'asse baricentrico della trave longitudinale (Figura 5.30).Figura 5.30 - Link di collegamento per gli appoggi

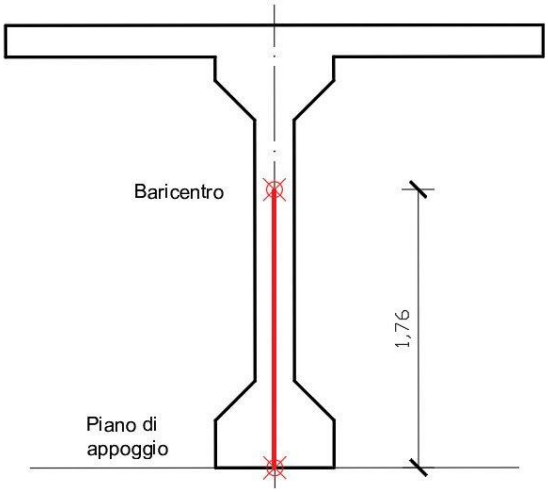


Figura 5.30 - Link di collegamento per gli appoggi

I link di collegamento hanno una lunghezza di 1.76 m. Tale lunghezza è stata determinata facendo riferimento alla posizione del baricentro della trave longitudinale calcolata tenendo contro dell'omogeneizzazione della sezione. La sezione trasversale e le relative proprietà di questi elementi di collegamento sono state oggetto di diverse analisi. Si è visto che l'utilizzo di link rigidi, così come fatto nel caso della soletta trasversale, causava un comportamento globale dell'impalcato non realistico. Questo perché nella realtà questi elementi di collegamento sono caratterizzati da rigidezze diverse nelle due direzioni in quanto di fatto modellano una porzione di anima della trave longitudinale. Si è scelto allora di assumere una sezione trasversale di lunghezza unitaria e larghezza pari allo spessore dell'anima della trave longitudinale (0.20 m).

Tab. 5.24 – Proprietà link di collegamento per gli appoggi

A [m²]	**I_y [m⁴]**	**I_z [m⁴]**	**I_t [m⁴]**
0.2	0.00066	0.017	0.0023

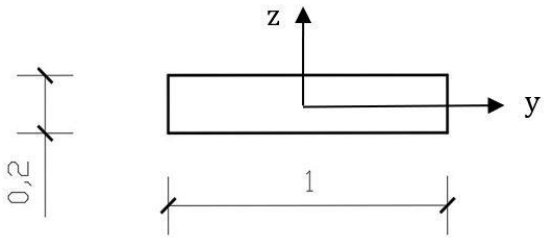


Figura 5.31 – Sezione dei link di collegamento per gli appoggi

Si riporta di seguito uno stralcio del modello e il sistema di riferimento locale scelto:

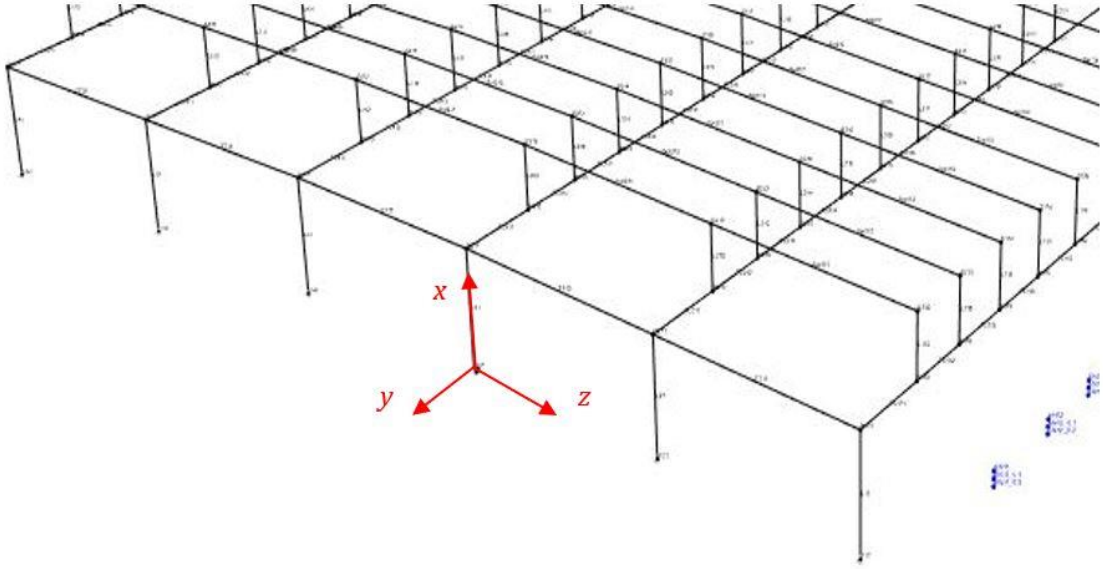


Figura 5.32 – Introduzione link di collegamento per gli appoggi

L'introduzione di questi link di collegamento comporta una leggera diminuzione delle rotazioni.

Tab. 5.25 – Confronto delle rotazioni lato nord

Inclinometri	Misura [mrad]	Modello [mrad]	Variazione %
EL-C23-T2-N	-0.53	-0.60	13
EL-C23-T3-N	-0.36	-0.39	6
EL-C23-T4-N	-0.15	-0.19	8
EL-C23-T5-N	≃0	≃0	-

Tab. 5.26 - Confronto delle rotazioni lato sud

Inclinometri	Misura [mrad]	Modello [mrad]	Variazione %
EL-C23-T2-S	-0.52	-0.60	15
EL-C23-T3-S	-0.32	-0.39	13
EL-C23-T4-S	-0.20	-0.19	-2
EL-C23-T5-S	≃0	≃0	-

5.5.5.5 Adeguamento dei traversi di testata alla posizione reale degli appoggi

I traversi di testata sono stati modellati inizialmente come semplici elementi di collegamento tra le travi longitudinali. In realtà, il traverso esercita un'azione di collegamento che si risente anche nei nodi di appoggio traslati verso il basso. Per tale motivo, i traversi di testata sono stati modellati mediante due elementi trave distinti che collegano sia i baricentri delle travi longitudinali, sia i nodi vincolati. La sezione del traverso di testata, riportata precedentemente in Figura 5.24 è stata divisa in due sezioni distinte dividendo la sezione originaria a metà altezza dell'anima. Per ogni porzione sono state calcolate le proprietà geometriche e inerziali.

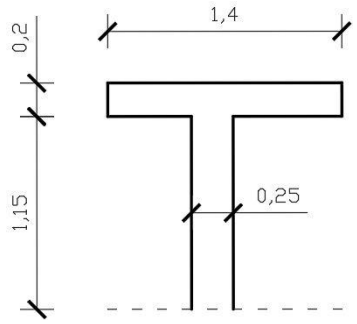


Figura 5.33 – Sezione trasversale superiore

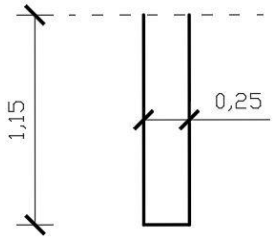


Figura 5.34 – Sezione trasversale inferiore

Questa modifica comporta un erroneo calcolo della rigidezza flessionale del traverso, ma distribuisce meglio la rigidezza assiale dello stesso collegando sia i nodi vincolati ad altezza pila sia i nodi baricentrici delle travi longitudinali. Tale miglioramento è ritenuto superiore all'errore commesso sulla rigidezza flessionale.

Si riportano di seguito le proprietà assegnate alle sezioni trasversali (Tab. 5.27) e il modello ottenuto (Figura 5.35):

Tab. 5.27 – Proprietà sezioni usate per modellare il traverso di testata

	A [m²]	**I_y [m⁴]**	**I_z [m⁴]**	**I_t [m⁴]**
Sezione trasversale superiore	0.568	0.048	0.097	0.0067
Sezione trasversale inferiore	0.288	0.0015	0.032	0.0052

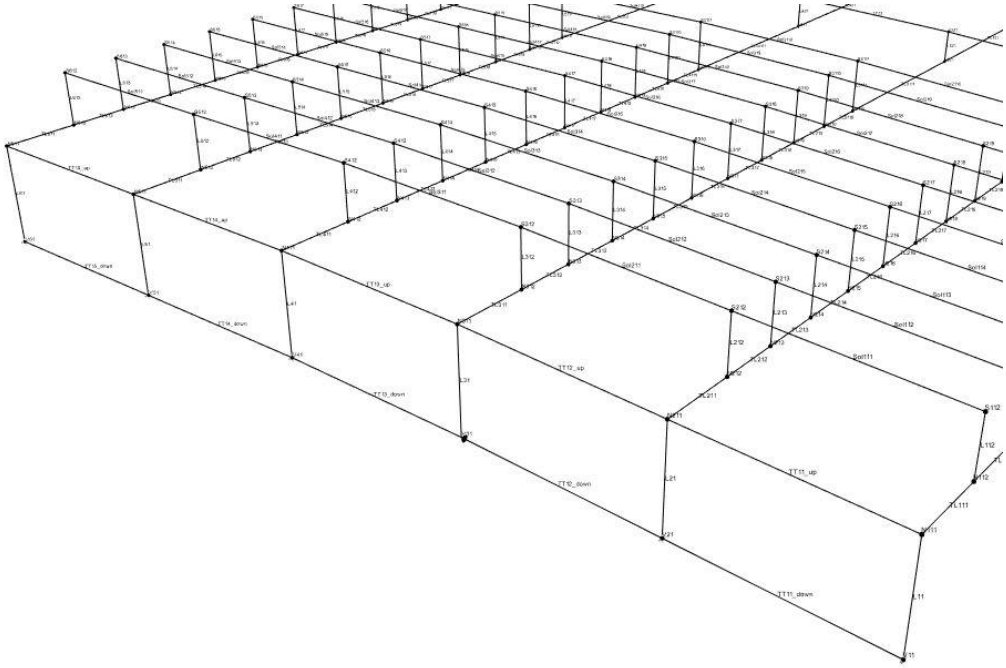


Figura 5.35 – Modellazione dei traversi di testata

Anche questa scelta di modellazione porta complessivamente ad un aumento di rigidezza del modello, con rotazioni che si avvicinano sempre di più ai valori target.

Tab. 5.28 – Confronto delle rotazioni lato nord

Inclinometri	Misura [mrad]	Modello [mrad]	Variazione %
EL-C23-T2-N	-0.53	-0.57	8
EL-C23-T3-N	-0.36	-0.38	4
EL-C23-T4-N	-0.15	-0.20	9
EL-C23-T5-N	≃0	≃0	-

Tab. 5.29 - Confronto delle rotazioni lato sud

Inclinometri	Misura [mrad]	Modello [mrad]	Variazione %
EL-C23-T2-S	-0.52	-0.59	13
EL-C23-T3-S	-0.32	-0.38	12
EL-C23-T4-S	-0.20	-0.20	0
EL-C23-T5-S	≃0	≃0	-

5.5.5.6 Introduzione delle forze di attrito negli appoggi

Infine, è stata modellata la forza di attrito longitudinale che si sviluppa all'interfaccia tra le travi longitudinali e gli appoggi presenti nel lato sud dell'impalcato i quali permettono gli spostamenti longitudinali. A tal fine, sono state calcolate le reazioni vincolari dovuti ai carichi permanenti (peso proprio, pavimentazione e barriere) effettuando una preventiva analisi dei carichi:

Tab. 5.30 – Analisi dei carichi permanenti

Elemento	**Peso [kN]**
Travi longitudinali e trasversali	6986
Soletta	4120
Cordoli	232
Pavimentazione	2472
Barriere	260
TOTALE	*14068*

Il peso totale dell'impalcato è stato dunque diviso per il numero totale degli appoggi (12) ottenendo una reazione verticale dovuta ai carichi permanenti di 1172 kN. Oltre ai carichi permanenti, è stato considerato il peso degli autocarri presenti durante la prova di carico; la risoluzione del modello con i carichi applicati ha fornito il valore delle reazioni cercate. Riguardo il coefficiente di attrito, tenendo conto del naturale attrito presente nei vincoli e considerando l'età dell'infrastruttura, si è assunto un coefficiente μ=5%. Si riportano in tabella i due contributi in termini di reazioni vincolari e la forza di attrito derivante da essi (Tab. 5.31):

Tab. 5.31 - Forze di attrito

Vincoli lato sud	**$R_{autocarri}$ [kN]**	**$R_{carichi\ permanenti}$ [kN]**	**coeff**	**Massima forza di attrito [kN]**
V14	-46	1172	0,05	56
V24	5.7	1172	0,05	59
V34	65	1172	0,05	62
V44	130	1172	0,05	65
V54	228	1172	0,05	70
V64	298	1172	0,05	73

Le forze di attrito calcolate sono state applicate ai nodi vincolati come forze orizzontali in direzione longitudinale. Il verso è stato definito dopo aver analizzato gli spostamenti longitudinali dei nodi ottenuti dal modello senza l'attrito (Tab. 5.32). Essendo tali spostamenti tutti positivi, le forze di attrito sono state applicate in direzione longitudinale negativa. Per

confronto, si riportano sin da adesso in Tab. 5.32 gli spostamenti longitudinali calcolati col modello finale in seguito all'inserimento delle forze di attrito,

Tab. 5.32 – Spostamenti longitudinali vincoli lato sud

Vincoli lato sud	Spostamento longitudinale senza attrito[mm]	Spostamento longitudinale con attrito[mm]
V14	0.18	-0.33
V24	0.50	-0.02
V34	0.85	0.32
V44	1.24	0.70
V54	1.70	1.12
V64	2.11	1.52

Riassumendo, il modello finale risulta composto dalle travi longitudinali, dai traversi, dalla soletta trasversale con i link rigidi, e dai vincoli opportunamente posizionati tramite i link di collegamento. E' modellata anche la pendenza trasversale dell'impalcato e l'attrito in corrispondenza dei dispositivi di vincolo. In tutti ciò, si fa riferimento alle proprietà omogeneizzate delle sezioni trasversali. I risultati del modello finale sono trattati nel paragrafo successivo.

5.5.6 Risultati del modello finale

Tab. 5.33 – Reazioni vincolari lato Nord

Vincoli lato nord	Reazione verticale [kN]	Reazione longitudinale [kN]	Reazione trasversale [kN]
V11	-94	-108	0
V21	3	-50	0
V31	56	13	-102
V41	136	86	0
V51	231	179	0
V61	349	266	0

Tab. 5.34 – Reazioni vincolari lato Sud

Vincoli lato sud	Reazione verticale [kN]	Reazione longitudinale [kN]	Reazione trasversale [kN]
V14	-47	0	0
V24	5	0	0
V34	65	0	102
V44	129	0	0
V54	228	0	0
V64	299	0	0

Tab. 5.35 – Spostamenti longitudinali vincolai lato Nord

Vincoli lato nord	Spostamento longitudinale [mm]
V11	0
V21	0
V31	0
V41	0
V51	0
V61	0

Tab. 5.36 – Spostamenti longitudinali vincoli lato Sud

Vincoli lato sud	Spostamento longitudinale [mm]
V14	-0.33
V24	-0.02
V34	0.32
V44	0.70
V54	1.12
V64	1.52

Le reazioni verticali variano da un nodo ad un altro con l'andamento tipico della ripartizione trasversale alla Courbon. Le reazioni trasversali sono coerenti con l'assenza di carichi trasversali e con le condizioni di vincolo applicate. Nulla di anomalo nemmeno per le reazioni e per gli spostamenti longitudinali. Di seguito la deformata dell'impalcato (Figura 5.36):

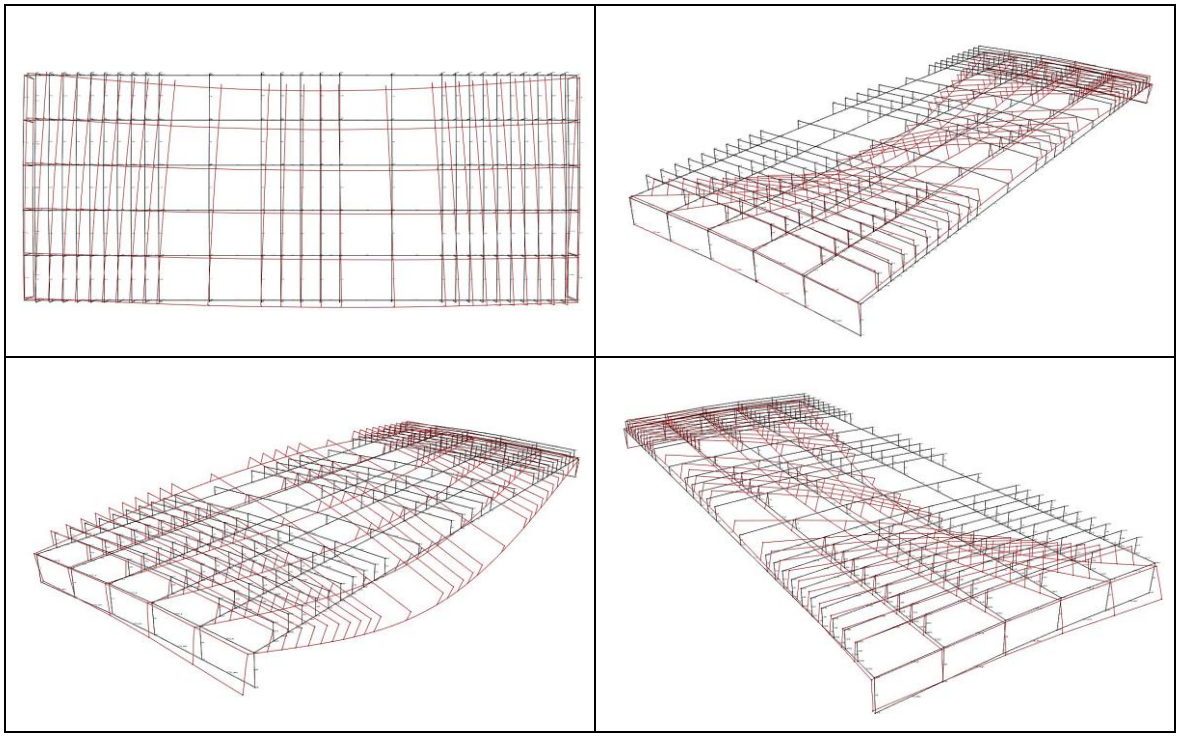


Figura 5.36 - Deformata impalcato

Si riporta infine il confronto tra le rotazioni degli inclinometri calcolate dal modello python e quelle misurate dal sistema di monitoraggio durante la prova di carico:

Tab. 5.37 – Confronto finale delle rotazioni lato nord

Inclinometri	Misura [mrad]	Modello [mrad]	Variazione %
EL-C23-T2-N	-0.53	-0.54	2
EL-C23-T3-N	-0.36	-0.34	-4
EL-C23-T4-N	-0.15	-0.17	4
EL-C23-T5-N	≃0	≃0	-

Tab. 5.38 - Confronto finale delle rotazioni lato sud

Inclinometri	Misura [mrad]	Modello [mrad]	Variazione %
EL-C23-T2-S	-0.52	-0.55	6
EL-C23-T3-S	-0.32	-0.35	6
EL-C23-T4-S	-0.20	-0.16	-8
EL-C23-T5-S	≃0	≃0	-

Conclusioni

Lo scopo del presente elaborato di tesi è stato quello di sviluppare un software FEM in Python, in grado di effettuare analisi elastiche lineari di strutture intelaiate tridimensionali. Il modello numerico della struttura monitorata costituisce infatti uno degli elementi necessari per un sistema di monitoraggio.

L'utilizzo del software sviluppato consiste nella scrittura di una sequenza di istruzioni che permettono la costruzione del modello, l'analisi e la valutazione dei risultati finali. Il software risulta molto leggero e richiede tempi di esecuzione dell'ordine di alcuni secondi; in generale, durante le fasi di sviluppo, si è visto che il tempo di elaborazione dipende dalla complessità del modello da costruire e dalla quantità di output richiesti dall'utente. Il software si presta ad essere caricato sia sul cloud, che sui nodi (gateway) che controllano i sensori. Queste ultime apparecchiature sono elaboratori piuttosto semplici che necessitano quindi di un software leggero dedicato alla loro architettura.

L'idea alla base del presente lavoro di tesi e di quelli futuri che seguiranno, è stata quella di realizzare un sistema di monitoraggio strutturale per ponti a graticcio in cui l'analisi dei dati registrati e la modellazione della struttura avviene mediante codici Python. In dettaglio, i dati registrati dai sensori vengono processati da codici appositamente creati per la pulizia e l'analisi dei dati. Questi sono poi confrontati con i valori teorici derivanti dalle analisi elastiche lineari del modello numerico. Una volta fissati i valori soglia (per esempio, allerta e allarme) e le relative tolleranze di accettazione, è possibile ottenere un monitoraggio in continuo. Tutto ciò può essere effettuato su piattaforma cloud permettendo all'utente il controllo dello stato di salute della struttura in tempo reale utilizzando un collegamento internet. Le ricerche bibliografiche hanno confermato la presenza di sistemi di monitoraggio con architetture analoghe o molto simili a quella appena descritta.

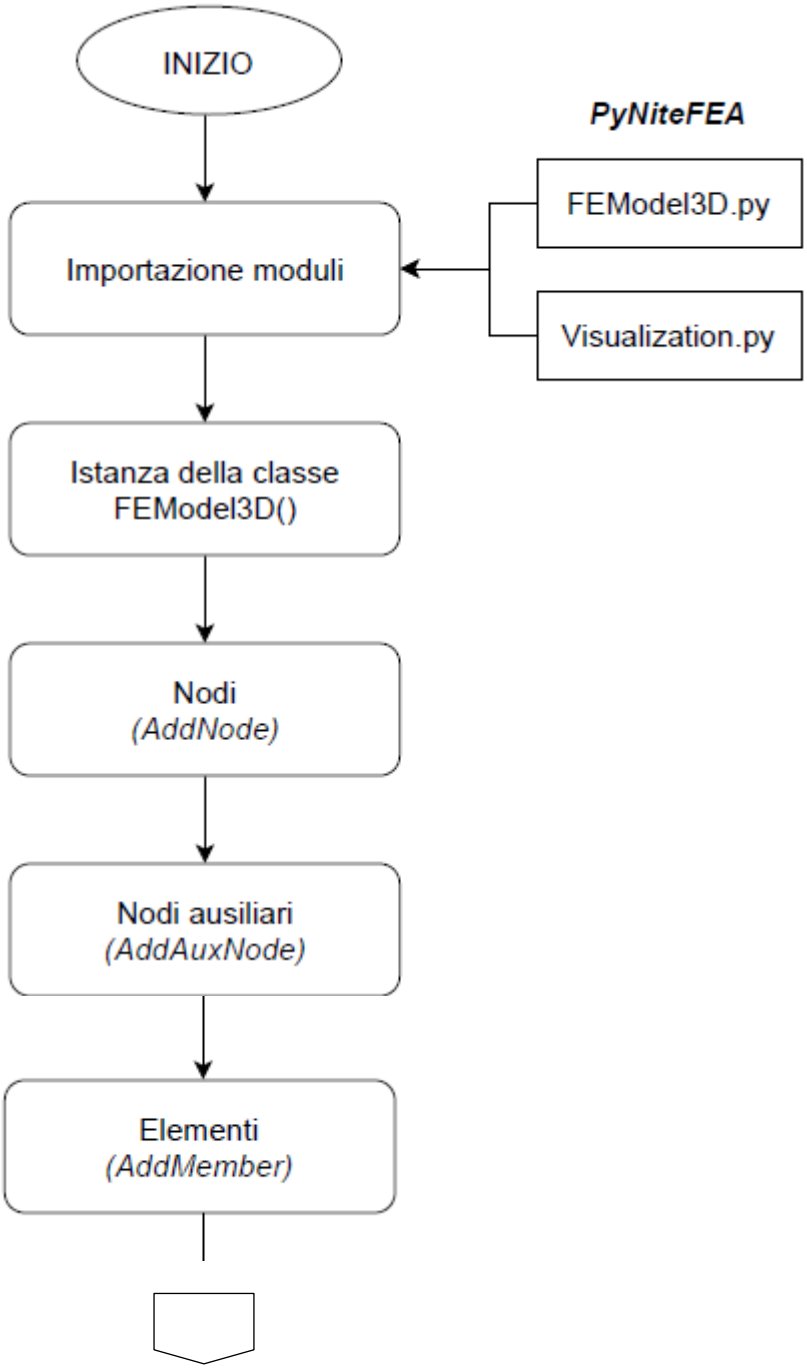
Il software FEM è stato testato attraverso la modellazione di un impalcato a graticcio in calcestruzzo armato precompresso. Adottando ragionevoli ipotesi semplificative, il software si è mostrato all'altezza del problema, permettendo un'ottima modellazione della struttura con riferimento ai dati registrati durante delle prove di carico.

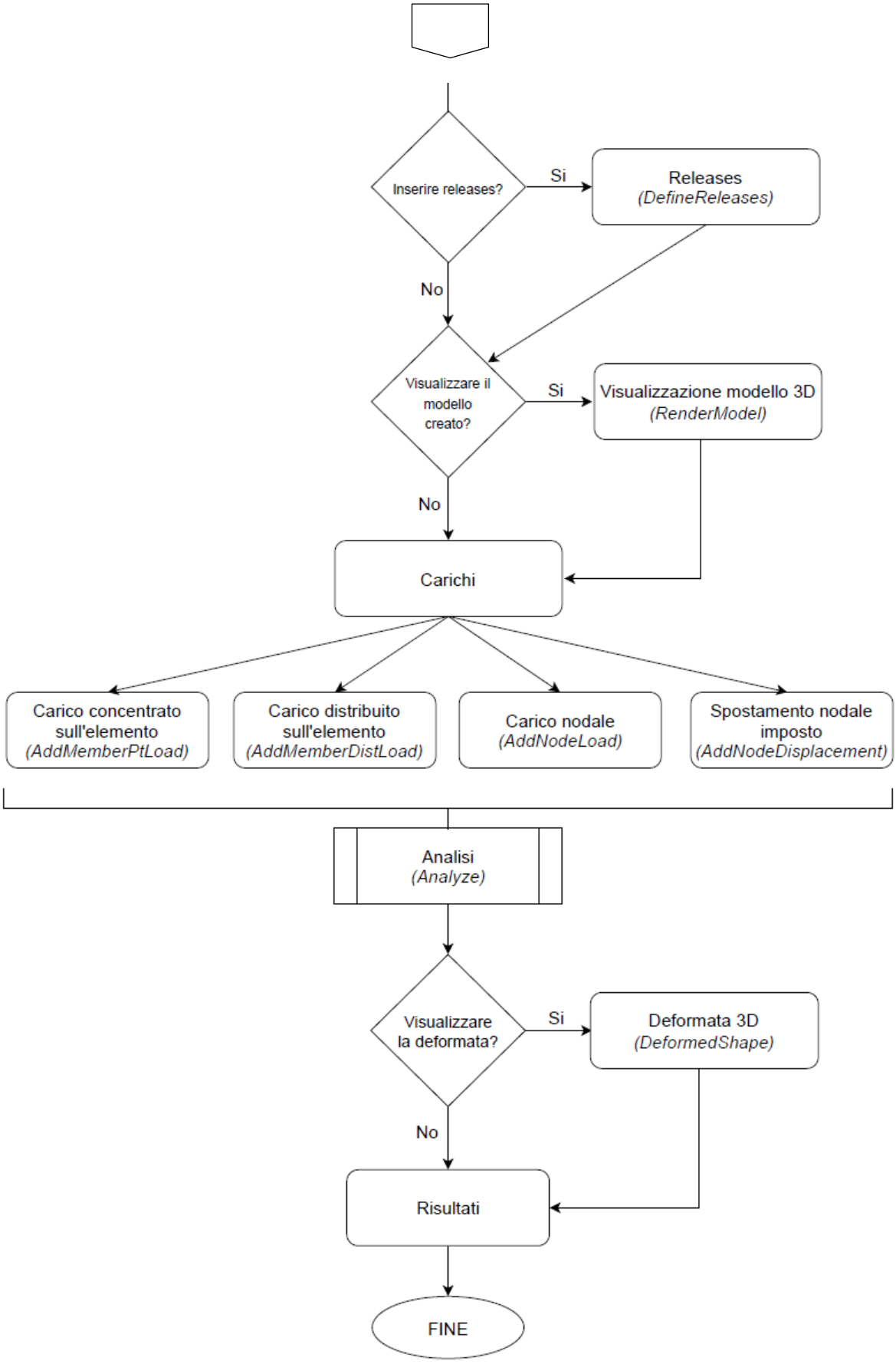
Il confronto tra i risultati del modello numerico e i dati di monitoraggio ha confermato la validità del software per lo scopo prefissato e per il suo ruolo all'interno di una complessa architettura di monitoraggio strutturale.

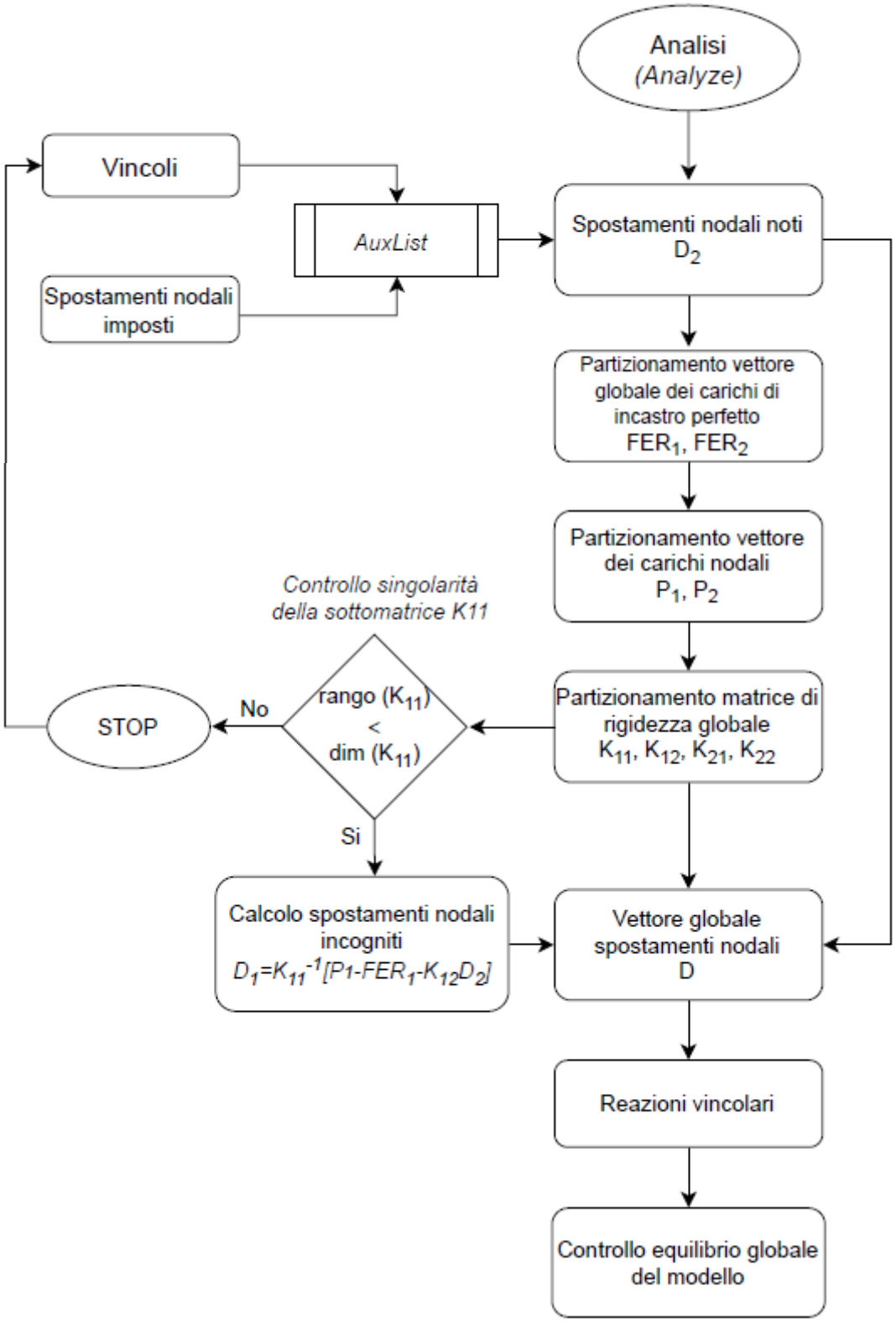
Tale lavoro, pertanto, rappresenta una base di partenza per ulteriori sviluppi futuri del software PyNiteFEA. Quest'ultimo, nato da poco, è in costante sviluppo grazie alla collaborazione di diversi utenti attraverso la piattaforma di sviluppo software GitHub. Per qualsiasi collaborazione è sufficiente entrare a far parte della community e proporre nuove funzionalità.

Appendice

A1 Flow chart




Inserire releases?
Si
Releases
(DefineReleases)
No
Visualizzare il modello creato?
Si
Visualizzazione modello 3D
(RenderModel)
No
Carichi
Carico concentrato sull'elemento
(AddMemberPtLoad)
Carico distribuito sull'elemento
(AddMemberDistLoad)
Carico nodale
(AddNodeLoad)
Spostamento nodale imposto
(AddNodeDisplacement)
Analisi
(Analyze)
Visualizzare la deformata?
Si
Deformata 3D
(DeformedShape)
No
Risultati
FINE


Analisi
(Analyze)
Vincoli
AuxList
Spostamenti nodali imposti
Spostamenti nodali noti
D_2
Partizionamento vettore globale dei carichi di incastro perfetto
FER_1, FER_2
Partizionamento vettore dei carichi nodali
P_1, P_2
Partizionamento matrice di rigidezza globale
K_{11}, K_{12}, K_{21}, K_{22}
Controllo singolarità della sottomatrice K11
rango (K_{11}) < dim (K_{11})
No
STOP
Si
Calcolo spostamenti nodali incogniti
$D_1=K_{11}^{-1}[P1-FER_1-K_{12}D_2]$
Vettore globale spostamenti nodali
D
Reazioni vincolari
Controllo equilibrio globale del modello

A2 Codice per la costruzione del modello

```
# Units: N, m

from PyNite import FEModel3D
from PyNite import Visualization

# Nuovo modello agli elementi finiti
Girder = FEModel3D()

    ###############################################################################
                                GEOMETRIA IMPALCATO
    ###############################################################################

N_travi_long = 6
N_travi_trasv = 4

# SVILUPPO LONGITUDINALE
# Porzioni di campata
L1= 14.075    #m                    #TT1-TT2
L2= 13.6      #m                    #TT2-TT3
L3= 14.075    #m                    #TT3-TT4
L_impalcato=L1+L2+L3 #m  #Dall'asse dei traversi di testata
L_ext = 0.625 #m           #Da asse traverso di testata a filo esterno soletta

# Larghezze solette collaboranti delle travi trasversali
Soletta_collaborante_TT1 = 1.4 #m
Soletta_collaborante_TT2 = 6.33 #m
Soletta_collaborante_TT3 = 6.33 #m
Soletta_collaborante_TT4 = 1.4 #m

# SVILUPPO TRASVERSALE
int_travi_long = 3.4 #m

#LINK RIGIDI
linkrigido_long_sol = 0.94 #m
linkrigido_long_trav = 0.605 #m
linkrigido_long_vincolo = 1.76 #m

#Trascuro le differenze di baricentro in termini di quota tra la trave
#longitudinale con la trave longitudinale di bordo e con le travi #trasversali TT1
e TT2

# Calcolo i tratti di soletta che si trovano in ogni porzione di campata. # Poi
divido ognuna di esse in un numero intero di parti

a = round(L1 + L_ext - Soletta_collaborante_TT1 - Soletta_collaborante_TT2/2, 3)
# campo TTI-TT2 e TT3-TT4
b = round(L2 - Soletta_collaborante_TT2/2 - Soletta_collaborante_TT3/2, 3)
        # campo TT2-TT3

div_a_c = 10 #n° elementi di soletta in cui divere i tratti a e c
div_b = 5          #n° elementi di soletta in cui divere il tratto b

    ###############################################################################
                                    MATERIALI
    ###############################################################################

E=3.7026e10               #N/m^2
v = 0.2                   #Poisson
G=E/(2*(1+v))        #N/m^2

E_rigido = E*1000    #N/m^2
G_rigido = G*1000    #N/m^2
```

```
    ##########################################################################
                                NODI TRAVE LONGITUDINALE
    ##########################################################################

### Nodi

# Abbassamenti delle travi longitudinali rispetto alla z=0 posizionata in
corrispondenza delle travi longitudinali 3 e 4
zTL1=-0.2 #m
zTL2=-0.1 #m
zTL3=-0 #m
zTL4=-0 #m
zTL5=-0.1 #m
zTL6=-0.2 #m

zTL= [zTL1, zTL2, zTL3, zTL4, zTL5, zTL6]

for j in range(N_travi_long):

      #Campata TT1-TT2
      Girder.AddNode("N"+str(j+1)+"11", 0, (int_travi_long*j),zTL[j])
      for i in range(div_a_c):
            if i == 0:
                  Girder.AddNode("N"+str(j+1)+"12",
((a/div_a_c)/2)+Soletta_collaborante_TT1-L_ext, (int_travi_long*j),zTL[j])
            else:
                  Girder.AddNode("N"+str(j+1)+"1"+str(i+2),
(((a/div_a_c)/2)+Soletta_collaborante_TT1-L_ext)+(a/div_a_c)*i, int_travi_long*j,
zTL[j])

      #Campata TT2-TT3
      Girder.AddNode("N"+str(j+1)+"21", L1, (int_travi_long*j),zTL[j])
      for i in range(div_b):
            if i == 0:
                  Girder.AddNode("N"+str(j+1)+"22",
L1+(Soletta_collaborante_TT2/2)+((b/div_b)/2), (int_travi_long*j),zTL[j])
            else:
                  Girder.AddNode("N"+str(j+1)+"2"+str(i+2),
L1+(Soletta_collaborante_TT2/2)+((b/div_b)/2)+(b/div_b)*i, int_travi_long*j,
zTL[j])

      #Campata TT3-TT4
      Girder.AddNode("N"+str(j+1)+"31", L1+L2, (int_travi_long*j),zTL[j])
      for i in range(div_a_c):
            if i == 0:
                  Girder.AddNode("N"+str(j+1)+"32",
L1+L2+(Soletta_collaborante_TT3/2)+((a/div_a_c)/2), (int_travi_long*j),zTL[j])
            else:
                  Girder.AddNode("N"+str(j+1)+"3"+str(i+2),
L1+L2+(Soletta_collaborante_TT3/2)+((a/div_a_c)/2)+(a/div_a_c)*i,
int_travi_long*j, zTL[j])

      Girder.AddNode("N"+str(j+1)+"41", L_impalcato, (int_travi_long*j),zTL[j])


### Nodi ausiliari a -0.2 m (per travi longitudinali 1 e 6)
dy=-2
Girder.AddAuxNode("AN11_0.2", 0, dy, zTL1)
Girder.AddAuxNode("AN21_0.2", L1, dy, zTL1)
Girder.AddAuxNode("AN31_0.2", L1+L2, dy, zTL1)
Girder.AddAuxNode("AN41_0.2", L_impalcato, dy, zTL1)

#Campata TT1-TT2
for i in range(div_a_c):
      if i == 0:

      Girder.AddAuxNode("AN"+"12_0.2",((a/div_a_c)/2)+Soletta_collaborante_TT1-
L_ext, dy, zTL1)
```

```
        else:

        Girder.AddAuxNode("AN"+"1"+str(i+2)+"_0.2",(Soletta_collaborante_TT1-
L_ext)+((a/div_a_c)/2)+((a/div_a_c))*i,dy,zTL1)

#Campata TT2-TT3
for i in range(div_b):
        if i == 0:

        Girder.AddAuxNode("AN"+"22_0.2",L1+(Soletta_collaborante_TT2/2)+((b/div_b)
/2), dy, zTL1)
        else:
                Girder.AddAuxNode("AN"+"2"+str(i+2)+"_0.2",
L1+(Soletta_collaborante_TT2/2)+((b/div_b)/2)+(b/div_b)*i, dy, zTL1)

#Campata TT3-TT4
for i in range(div_a_c):
        if i == 0:

        Girder.AddAuxNode("AN"+"32_0.2",L1+L2+(Soletta_collaborante_TT3/2)+((a/div
_a_c)/2), dy, zTL1)
        else:
                Girder.AddAuxNode("AN"+"3"+str(i+2)+"_0.2",
L1+L2+(Soletta_collaborante_TT3/2)+((a/div_a_c)/2)+(a/div_a_c)*i, dy, zTL1)


### Nodi ausiliari a -0.1 m (per travi longitudinali 2 e 5)
Girder.AddAuxNode("AN11_0.1", 0, dy, zTL2)
Girder.AddAuxNode("AN21_0.1", L1, dy, zTL2)
Girder.AddAuxNode("AN31_0.1", L1+L2, dy, zTL2)
Girder.AddAuxNode("AN41_0.1", L_impalcato, dy, zTL2)

#Campata TT1-TT2
for i in range(div_a_c):
        if i == 0:

        Girder.AddAuxNode("AN"+"12_0.1",((a/div_a_c)/2)+Soletta_collaborante_TT1-
L_ext, dy, zTL2)
        else:

        Girder.AddAuxNode("AN"+"1"+str(i+2)+"_0.1",(Soletta_collaborante_TT1-
L_ext)+((a/div_a_c)/2)+((a/div_a_c))*i,dy,zTL2)

#Campata TT2-TT3
for i in range(div_b):
        if i == 0:

        Girder.AddAuxNode("AN"+"22_0.1",L1+(Soletta_collaborante_TT2/2)+((b/div_b)
/2), dy, zTL2)
        else:
                Girder.AddAuxNode("AN"+"2"+str(i+2)+"_0.1",
L1+(Soletta_collaborante_TT2/2)+((b/div_b)/2)+(b/div_b)*i, dy, zTL2)

#Campata TT3-TT4
for i in range(div_a_c):
        if i == 0:

        Girder.AddAuxNode("AN"+"32_0.1",L1+L2+(Soletta_collaborante_TT3/2)+((a/div
_a_c)/2), dy, zTL2)
        else:
                Girder.AddAuxNode("AN"+"3"+str(i+2)+"_0.1",
L1+L2+(Soletta_collaborante_TT3/2)+((a/div_a_c)/2)+(a/div_a_c)*i, dy, zTL2)


### Nodi ausiliari a 0 m (per travi longitudinali 3 e 4)
Girder.AddAuxNode("AN11", 0, dy, zTL3)
Girder.AddAuxNode("AN21", L1, dy, zTL3)
Girder.AddAuxNode("AN31", L1+L2, dy, zTL3)
```

```
Girder.AddAuxNode("AN41", L_impalcato, dy, zTL3)

#Campata TT1-TT2
for i in range(div_a_c):
      if i == 0:

      Girder.AddAuxNode("AN"+"12",((a/div_a_c)/2)+Soletta_collaborante_TT1-
L_ext, dy, zTL3)
      else:
            Girder.AddAuxNode("AN"+"1"+str(i+2),(Soletta_collaborante_TT1-
L_ext)+((a/div_a_c)/2)+((a/div_a_c))*i,dy,zTL3)

#Campata TT2-TT3
for i in range(div_b):
      if i == 0:

      Girder.AddAuxNode("AN"+"22",L1+(Soletta_collaborante_TT2/2)+((b/div_b)/2),
dy, zTL3)
      else:
            Girder.AddAuxNode("AN"+"2"+str(i+2),
L1+(Soletta_collaborante_TT2/2)+((b/div_b)/2)+(b/div_b)*i, dy, zTL3)

#Campata TT3-TT4
for i in range(div_a_c):
      if i == 0:

      Girder.AddAuxNode("AN"+"32",L1+L2+(Soletta_collaborante_TT3/2)+((a/div_a_c
)/2), dy, zTL3)
      else:
            Girder.AddAuxNode("AN"+"3"+str(i+2),
L1+L2+(Soletta_collaborante_TT3/2)+((a/div_a_c)/2)+(a/div_a_c)*i, dy, zTL3)


    ##########################################################################
                          TRAVI LONGITUDINALI ESTERNE
    ##########################################################################

#Proprietà sezione omogeneizzata
Iy_long_ext=0.4086                                      #m^4
Iz_long_ext=1.68                                        #m^4
J_long_ext= 0.0342                                      #m^4
A_long_ext=1.58                                         #m^2

# Elementi TL1
#Campata TT1-TT2
for i in range(div_a_c + 1):
      if i == div_a_c:
            Girder.AddMember("TL1"+"1"+str(i+1), "N1"+"1"+str(i+1), "N121", E,
G, Iy_long_ext, Iz_long_ext, J_long_ext, A_long_ext, "AN1"+str(i+1)+"_0.2")
      else:
            Girder.AddMember("TL1"+"1"+str(i+1), "N1"+"1"+str(i+1),
"N1"+"1"+str(i+2), E, G, Iy_long_ext, Iz_long_ext, J_long_ext, A_long_ext,
"AN1"+str(i+1)+"_0.2")
#Campata TT2-TT3
for i in range(div_b + 1):
      if i == div_b:
            Girder.AddMember("TL1"+"2"+str(i+1), "N1"+"2"+str(i+1), "N131", E,
G, Iy_long_ext, Iz_long_ext, J_long_ext, A_long_ext, "AN2"+str(i+1)+"_0.2")
      else:
            Girder.AddMember("TL1"+"2"+str(i+1), "N1"+"2"+str(i+1),
"N1"+"2"+str(i+2), E, G, Iy_long_ext, Iz_long_ext, J_long_ext, A_long_ext,
"AN2"+str(i+1)+"_0.2")
#Campata TT3-TT4
for i in range(div_a_c + 1):
      if i == div_a_c:
            Girder.AddMember("TL1"+"3"+str(i+1), "N1"+"3"+str(i+1), "N141", E,
G, Iy_long_ext, Iz_long_ext, J_long_ext, A_long_ext, "AN3"+str(i+1)+"_0.2")
      else:
```

```
            Girder.AddMember("TL1"+"3"+str(i+1), "N1"+"3"+str(i+1),
"N1"+"3"+str(i+2), E, G, Iy_long_ext, Iz_long_ext, J_long_ext, A_long_ext,
"AN3"+str(i+1)+"_0.2")

# Elementi TL6
#Campata TT1-TT2
for i in range(div_a_c + 1):
      if i == div_a_c:
            Girder.AddMember("TL6"+"1"+str(i+1), "N6"+"1"+str(i+1), "N621", E,
G, Iy_long_ext, Iz_long_ext, J_long_ext, A_long_ext, "AN1"+str(i+1)+"_0.2")
      else:
            Girder.AddMember("TL6"+"1"+str(i+1), "N6"+"1"+str(i+1),
"N6"+"1"+str(i+2), E, G, Iy_long_ext, Iz_long_ext, J_long_ext, A_long_ext,
"AN1"+str(i+1)+"_0.2")
#Campata TT2-TT3
for i in range(div_b + 1):
      if i == div_b:
            Girder.AddMember("TL6"+"2"+str(i+1), "N6"+"2"+str(i+1), "N631", E,
G, Iy_long_ext, Iz_long_ext, J_long_ext, A_long_ext, "AN2"+str(i+1)+"_0.2")
      else:
            Girder.AddMember("TL6"+"2"+str(i+1), "N6"+"2"+str(i+1),
"N6"+"2"+str(i+2), E, G, Iy_long_ext, Iz_long_ext, J_long_ext, A_long_ext,
"AN2"+str(i+1)+"_0.2")
#Campata TT3-TT4
for i in range(div_a_c + 1):
      if i == div_a_c:
            Girder.AddMember("TL6"+"3"+str(i+1), "N6"+"3"+str(i+1), "N641", E,
G, Iy_long_ext, Iz_long_ext, J_long_ext, A_long_ext, "AN3"+str(i+1)+"_0.2")
      else:
            Girder.AddMember("TL6"+"3"+str(i+1), "N6"+"3"+str(i+1),
"N6"+"3"+str(i+2), E, G, Iy_long_ext, Iz_long_ext, J_long_ext, A_long_ext,
"AN3"+str(i+1)+"_0.2")


    ##########################################################################
                          TRAVI LONGITUDINALI INTERNE
    ##########################################################################

#Proprietà sezione con omogeneizzazione
Iy_long= 0.6795                         #m^4
Iz_long= 1.79                           #m^4
J_long= 0.0350                          #m^4
A_long= 1.71                      #m^2


# TRAVE LONGITUDINALE 2
#Campata TT1-TT2
for i in range(div_a_c + 1):
      if i == div_a_c:
            Girder.AddMember("TL21"+str(i+1), "N21"+str(i+1), "N221", E, G,
Iy_long, Iz_long, J_long, A_long, "AN1"+str(i+1)+"_0.1")
      else:
            Girder.AddMember("TL21"+str(i+1), "N21"+str(i+1), "N21"+str(i+2), E,
G, Iy_long, Iz_long, J_long, A_long, "AN1"+str(i+1)+"_0.1")

#Campata TT2-TT3
for i in range(div_b + 1):
      if i == div_b:
            Girder.AddMember("TL22"+str(i+1), "N22"+str(i+1), "N231", E, G,
Iy_long, Iz_long, J_long, A_long, "AN2"+str(i+1)+"_0.1")
      else:
            Girder.AddMember("TL22"+str(i+1), "N22"+str(i+1), "N22"+str(i+2), E,
G, Iy_long, Iz_long, J_long, A_long, "AN2"+str(i+1)+"_0.1")
#Campata TT3-TT4
for i in range(div_a_c + 1):
      if i == div_a_c:
            Girder.AddMember("TL23"+str(i+1), "N23"+str(i+1), "N241", E, G,
Iy_long, Iz_long, J_long, A_long, "AN3"+str(i+1)+"_0.1")
```

```
        else:
                Girder.AddMember("TL23"+str(i+1), "N23"+str(i+1), "N23"+str(i+2), E,
G, Iy_long, Iz_long, J_long, A_long, "AN3"+str(i+1)+"_0.1")

# TRAVE LONGITUDINALE 3
#Campata TT1-TT2
for i in range(div_a_c + 1):
        if i == div_a_c:
                Girder.AddMember("TL31"+str(i+1), "N31"+str(i+1), "N321", E, G,
Iy_long, Iz_long, J_long, A_long, "AN1"+str(i+1))
        else:
                Girder.AddMember("TL31"+str(i+1), "N31"+str(i+1), "N31"+str(i+2), E,
G, Iy_long, Iz_long, J_long, A_long, "AN1"+str(i+1))

#Campata TT2-TT3
for i in range(div_b + 1):
        if i == div_b:
                Girder.AddMember("TL32"+str(i+1), "N32"+str(i+1), "N331", E, G,
Iy_long, Iz_long, J_long, A_long, "AN2"+str(i+1))
        else:
                Girder.AddMember("TL32"+str(i+1), "N32"+str(i+1), "N32"+str(i+2), E,
G, Iy_long, Iz_long, J_long, A_long, "AN2"+str(i+1))
#Campata TT3-TT4
for i in range(div_a_c + 1):
        if i == div_a_c:
                Girder.AddMember("TL33"+str(i+1), "N33"+str(i+1), "N341", E, G,
Iy_long, Iz_long, J_long, A_long, "AN3"+str(i+1))
        else:
                Girder.AddMember("TL33"+str(i+1), "N33"+str(i+1), "N33"+str(i+2), E,
G, Iy_long, Iz_long, J_long, A_long, "AN3"+str(i+1))

# TRAVE LONGITUDINALE 4
#Campata TT1-TT2
for i in range(div_a_c + 1):
        if i == div_a_c:
                Girder.AddMember("TL341"+str(i+1), "N41"+str(i+1), "N421", E, G,
Iy_long, Iz_long, J_long, A_long, "AN1"+str(i+1))
        else:
                Girder.AddMember("TL41"+str(i+1), "N41"+str(i+1), "N41"+str(i+2), E,
G, Iy_long, Iz_long, J_long, A_long, "AN1"+str(i+1))

#Campata TT2-TT3
for i in range(div_b + 1):
        if i == div_b:
                Girder.AddMember("TL42"+str(i+1), "N42"+str(i+1), "N431", E, G,
Iy_long, Iz_long, J_long, A_long, "AN2"+str(i+1))
        else:
                Girder.AddMember("TL42"+str(i+1), "N42"+str(i+1), "N42"+str(i+2), E,
G, Iy_long, Iz_long, J_long, A_long, "AN2"+str(i+1))
#Campata TT3-TT4
for i in range(div_a_c + 1):
        if i == div_a_c:
                Girder.AddMember("TL43"+str(i+1), "N43"+str(i+1), "N441", E, G,
Iy_long, Iz_long, J_long, A_long, "AN3"+str(i+1))
        else:
                Girder.AddMember("TL43"+str(i+1), "N43"+str(i+1), "N43"+str(i+2), E,
G, Iy_long, Iz_long, J_long, A_long, "AN3"+str(i+1))

# TRAVE LONGITUDINALE 5
#Campata TT1-TT2
for i in range(div_a_c + 1):
        if i == div_a_c:
                Girder.AddMember("TL51"+str(i+1), "N51"+str(i+1), "N521", E, G,
Iy_long, Iz_long, J_long, A_long, "AN1"+str(i+1)+"_0.1")
        else:
                Girder.AddMember("TL51"+str(i+1), "N51"+str(i+1), "N51"+str(i+2), E,
G, Iy_long, Iz_long, J_long, A_long, "AN1"+str(i+1)+"_0.1")
```

```
#Campata TT2-TT3
for i in range(div_b + 1):
      if i == div_b:
            Girder.AddMember("TL52"+str(i+1), "N52"+str(i+1), "N531", E, G,
Iy_long, Iz_long, J_long, A_long, "AN2"+str(i+1)+"_0.1")
      else:
            Girder.AddMember("TL52"+str(i+1), "N52"+str(i+1), "N52"+str(i+2), E,
G, Iy_long, Iz_long, J_long, A_long, "AN2"+str(i+1)+"_0.1")
#Campata TT3-TT4
for i in range(div_a_c + 1):
      if i == div_a_c:
            Girder.AddMember("TL53"+str(i+1), "N53"+str(i+1), "N541", E, G,
Iy_long, Iz_long, J_long, A_long, "AN3"+str(i+1)+"_0.1")
      else:
            Girder.AddMember("TL53"+str(i+1), "N53"+str(i+1), "N53"+str(i+2), E,
G, Iy_long, Iz_long, J_long, A_long, "AN3"+str(i+1)+"_0.1")


    ##########################################################################
                                    VINCOLI
    ##########################################################################

#Proprietà link
Iy_link = 0.00066 #m^4
Iz_link = 0.017 #m^4
J_link = 0.0023 #m^4
A_link = 0.2 #m^2

# Nodi
for j in range(N_travi_long):
      Girder.AddNode("V"+str(j+1)+"1", 0, (int_travi_long*j),zTL[j]-
linkrigido_long_vincolo)
      Girder.AddNode("V"+str(j+1)+"4", L_impalcato, (int_travi_long*j),zTL[j]-
linkrigido_long_vincolo)


#Nodi ausiliari per i link (ANv1, ANv2, ANv3, ANv4, ANv5, ANv6)  --> "v" =
vincolo


#Per travi longitudinali 3 e 4
Girder.AddAuxNode("ANv1",0, dy, zTL3-linkrigido_long_vincolo)
Girder.AddAuxNode("ANv4",L_impalcato, dy, zTL3-linkrigido_long_vincolo)
#Per travi longitudinali 2 e 5
Girder.AddAuxNode("ANv1_0.1", 0, dy, zTL2-linkrigido_long_vincolo)
Girder.AddAuxNode("ANv4_0.1", L_impalcato, dy, zTL2-linkrigido_long_vincolo)
#Per travi longitudinali 1 e 6
Girder.AddAuxNode("ANv1_0.2",  0, dy, zTL1-linkrigido_long_vincolo)
Girder.AddAuxNode("ANv4_0.2",  L_impalcato, dy, zTL1-linkrigido_long_vincolo)

# Link
# Fila vincoli nord
Girder.AddMember("L11","V11","N111", E, G, Iy_link, Iz_link, J_link, A_link,
"ANv1_0.2")
Girder.AddMember("L21","V21","N211", E, G, Iy_link, Iz_link, J_link, A_link,
"ANv1_0.1")
Girder.AddMember("L31","V31","N311", E, G, Iy_link, Iz_link, J_link, A_link,
"ANv1")
Girder.AddMember("L41","V41","N411", E, G, Iy_link, Iz_link, J_link, A_link,
"ANv1")
Girder.AddMember("L51","V51","N511", E, G, Iy_link, Iz_link, J_link, A_link,
"ANv1_0.1")
Girder.AddMember("L61","V61","N611", E, G, Iy_link, Iz_link, J_link, A_link,
"ANv1_0.2")
# Fila vincoli sud
Girder.AddMember("L14","V14","N141", E, G, Iy_link, Iz_link, J_link, A_link,
"ANv4_0.2")
```

```
Girder.AddMember("L24","V24","N241", E, G, Iy_link, Iz_link, J_link, A_link,
"ANv4_0.1")
Girder.AddMember("L34","V34","N341", E, G, Iy_link, Iz_link, J_link, A_link,
"ANv4")
Girder.AddMember("L44","V44","N441", E, G, Iy_link, Iz_link, J_link, A_link,
"ANv4")
Girder.AddMember("L54","V54","N541", E, G, Iy_link, Iz_link, J_link, A_link,
"ANv4_0.1")
Girder.AddMember("L64","V64","N641", E, G, Iy_link, Iz_link, J_link, A_link,
"ANv4_0.2")

# Condizioni di vincolo
Girder.DefineSupport("V11",True, False, True, False, False, False)
Girder.DefineSupport("V21",True, False, True, False, False, False)
Girder.DefineSupport("V31",True, True, True, False, False, False)
Girder.DefineSupport("V41",True, False, True, False, False, False)
Girder.DefineSupport("V51",True, False, True, False, False, False)
Girder.DefineSupport("V61",True, False, True, False, False, False)

Girder.DefineSupport("V14",False, False, True, False, False, False)
Girder.DefineSupport("V24",False, False, True, False, False, False)
Girder.DefineSupport("V34",False, True, True, False, False, False)
Girder.DefineSupport("V44",False, False, True, False, False, False)
Girder.DefineSupport("V54",False, False, True, False, False, False)
Girder.DefineSupport("V64",False, False, True, False, False, False)



    ##########################################################################
                                LINK RIGIDI
    ##########################################################################

#Link tra trave longitudinale e soletta

#Nodi
for j in range(N_travi_long):
      #Campata TT1-TT2
      for i in range (div_a_c):
            if i == 0:
                  Girder.AddNode("S"+str(j+1)+"12",
((a/div_a_c)/2)+Soletta_collaborante_TT1-L_ext,
(int_travi_long*j),linkrigido_long_sol+zTL[j])
            else:
                  Girder.AddNode("S"+str(j+1)+"1"+str(i+2),
(((a/div_a_c)/2)+Soletta_collaborante_TT1-L_ext)+(a/div_a_c)*i, int_travi_long*j,
linkrigido_long_sol+zTL[j])

      #Campata TT2-TT3
      Girder.AddNode("S"+str(j+1)+"21", L1,
(int_travi_long*j),linkrigido_long_sol+zTL[j])
      for i in range (div_b):
            if i == 0:
                  Girder.AddNode("S"+str(j+1)+"22",
L1+(Soletta_collaborante_TT2/2)+((b/div_b)/2), (int_travi_long*j),
linkrigido_long_sol+zTL[j])
            else:
                  Girder.AddNode("S"+str(j+1)+"2"+str(i+2),
L1+(Soletta_collaborante_TT2/2)+((b/div_b)/2)+(b/div_b)*i, int_travi_long*j,
linkrigido_long_sol+zTL[j])

      #Campata TT3-TT4
      Girder.AddNode("S"+str(j+1)+"31", L1+L2,
(int_travi_long*j),linkrigido_long_sol+zTL[j])
      for i in range (div_a_c):
            if i == 0:
```

```
                    Girder.AddNode("S"+str(j+1)+"32",
L1+L2+(Soletta_collaborante_TT3/2)+((a/div_a_c)/2), (int_travi_long*j),
linkrigido_long_sol+zTL[j])
              else:
                    Girder.AddNode("S"+str(j+1)+"3"+str(i+2),
L1+L2+(Soletta_collaborante_TT3/2)+((a/div_a_c)/2)+(a/div_a_c)*i,
int_travi_long*j, linkrigido_long_sol+zTL[j])

# Elementi infinitamente rigidi

for j in range(N_travi_long):
       for i in range(div_a_c+1):
              if i == 0:
                    pass
              else:

       Girder.AddMember("L"+str(j+1)+"1"+str(i+1),"N"+str(j+1)+"1"+str(i+1),"S"+s
tr(j+1)+"1"+str(i+1),E_rigido,G_rigido,1,1,1,1)

       for i in range(div_b+1):
              if i == 0:
                    pass
              else:
       Girder.AddMember("L"+str(j+1)+"2"+str(i+1),"N"+str(j+1)+"2"+str(i+1),"S"+s
tr(j+1)+"2"+str(i+1),E_rigido,G_rigido,1,1,1,1)

       for i in range(div_a_c+1):
              if i == 0:
                    pass
              else:

       Girder.AddMember("L"+str(j+1)+"3"+str(i+1),"N"+str(j+1)+"3"+str(i+1),"S"+s
tr(j+1)+"3"+str(i+1),E_rigido,G_rigido,1,1,1,1)


# Elimino i link creati in corrispondenza dei traversi TT2 e TT3

for j in range(N_travi_long):
       Girder.RemoveNode("S"+str(j+1)+"21")
       Girder.RemoveNode("S"+str(j+1)+"31")

#Link tra trave longitudinale e trave trasversale

#Nodi

for j in range(N_travi_long):
       Girder.AddNode("S"+str(j+1)+"21", L1, int_travi_long*j,
linkrigido_long_trav+zTL[j])
       Girder.AddNode("S"+str(j+1)+"31", L1+L2, int_travi_long*j,
linkrigido_long_trav+zTL[j])

# Elementi infinitamente rigidi

for j in range(N_travi_long):
       Girder.AddMember("L"+str(j+1)+"21","N"+str(j+1)+"21", "S"+str(j+1)+"21",
E_rigido,G_rigido,1,1,1,1)
       Girder.AddMember("L"+str(j+1)+"31","N"+str(j+1)+"31", "S"+str(j+1)+"31",
E_rigido,G_rigido,1,1,1,1)


    ##########################################################################
                          TRAVI TRASVERSALI ESTERNE
    ##########################################################################

### UP

#Proprietà sezione
Iy_trasv_ext_up= 0.048     #m^4
```

```
Iz_trasv_ext_up= 0.0972    #m^4
J_trasv_ext_up= 0.0067     #m^4
A_trasv_ext_up= 0.5675     #m^2


# Nodi ausiliari (ANte1, ANte2, ANte3, ANte4, ANte5)
dx = L_impalcato + 2

for j in range(N_travi_long):
      if j == N_travi_long-1:
            break
      else:
            Girder.AddAuxNode("ANte"+str(j+1)+"_up",dx, (int_travi_long*j),
zTL[j])
# Elementi
for i in range(N_travi_trasv):
      if i > 0 and i < N_travi_trasv-1:
            pass
      else:
            for j in range(N_travi_long):
                  if j == N_travi_long-1:
                        break
                  else:

      Girder.AddMember("TT"+str(i+1)+str(j+1)+"_up","N"+str(j+1)+str(i+1)+"1","N
"+str(j+2)+str(i+1)+"1", E, G, Iy_trasv_ext_up, Iz_trasv_ext_up, J_trasv_ext_up,
A_trasv_ext_up, "ANte"+str(j+1)+"_up")

### DOWN

#Proprietà sezione
Iy_trasv_ext_down= 0.0015  #m^4
Iz_trasv_ext_down= 0.0317  #m^4
J_trasv_ext_down= 0.0052   #m^4
A_trasv_ext_down= 0.2875   #m^2


# Nodi ausiliari (ANte1, ANte2, ANte3, ANte4, ANte5)  "trasversali esterne -->
te"
dx = L_impalcato + 2

for j in range(N_travi_long):
      if j == N_travi_long-1:
            break
      else:
            Girder.AddAuxNode("ANte"+str(j+1)+"_down",dx, (int_travi_long*j),
zTL[j]-linkrigido_long_vincolo)

# Elementi
for i in range(N_travi_trasv):
      if i > 0 and i < N_travi_trasv-1:
            pass
      else:
            for j in range(N_travi_long):
                  if j == N_travi_long-1:
                        break
                  else:

      Girder.AddMember("TT"+str(i+1)+str(j+1)+"_down","V"+str(j+1)+str(i+1),"V"+
str(j+2)+str(i+1), E, G, Iy_trasv_ext_down, Iz_trasv_ext_down, J_trasv_ext_down,
A_trasv_ext_down, "ANte"+str(j+1)+"_down")


    ##########################################################################
                          TRAVI TRASVERSALI INTERNE
    ##########################################################################

#Proprietà sezione con omogeneizzazione
```

```
Iy_trasv= 4.2288                                                    #m^4
Iz_trasv= 0.771                                                     #m^4
J_trasv= 0.0134                                                     #m^4
A_trasv= 1.775                                                      #m^2

# Nodi ausiliari (ANti1, ANti2, ANti3, ANti4, ANti5)
dx = L_impalcato + 2
for j in range(N_travi_long):
      if j == N_travi_long-1:
            break
      else:
            Girder.AddAuxNode("ANti"+str(j+1),dx,
(int_travi_long*j),linkrigido_long_trav+ zTL[j])

# Elementi

for i in range(N_travi_trasv):
      if i == 0 or i == N_travi_trasv-1:
            pass
      else:
            for j in range(N_travi_long):
                  if j == N_travi_long-1:
                        break
                  else:

      Girder.AddMember("TT"+str(i+1)+str(j+1),"S"+str(j+1)+str(i+1)+"1","S"+str(
j+2)+str(i+1)+"1", E, G, Iy_trasv, Iz_trasv, J_trasv, A_trasv, "ANti"+str(j+1))


    ##########################################################################
SOLETTA ###################################################################################

# Nodi ausiliari (ANsol1, ANsol2, ANsol3, ANsol4, ANsol5)

dx = L_impalcato + 2

for j in range(N_travi_long):
      if j == N_travi_long-1:
            break
      else:
            Girder.AddAuxNode("ANsol"+str(j+1),dx,
(3.40*j),linkrigido_long_sol+zTL[j])


#SOLETTA campo TT1-TT2 e TT3-TT4

#Proprietà sezione non omogeneizzata
Iy_sol1= 0.0174                         #m^4
Iz_sol1= 0.0007                  #m^4
J_sol1= 2.367e-3                 #m^4
A_sol1= 0.2027                          #m^2

# Nodi ausiliari (ANsol1, ANsol2, ANsol3, ANsol4, ANsol5)

# Elementi
for i in range(N_travi_trasv):
      if i == 1 or i == N_travi_trasv-1:
            pass
      else:
            for j in range(N_travi_long):
                  if j == N_travi_long-1:
                        break
                  else:
                        for i in range(div_a_c+1):
                              if i == 0:
                                    pass
                              else:
                                    #Campata TT1-TT2
```

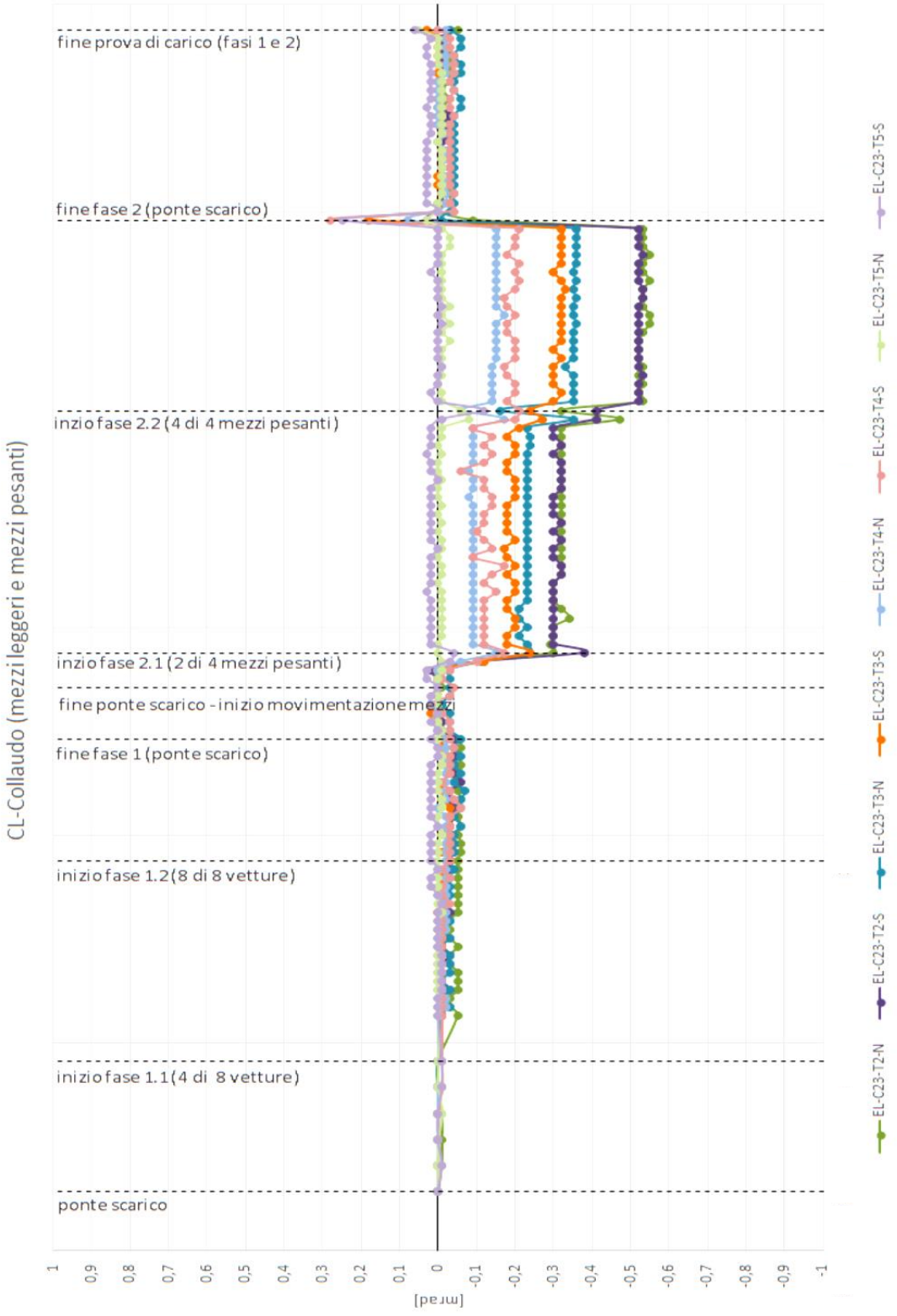
```
      Girder.AddMember("Sol"+str(j+1)+"1"+str(i),"S"+str(j+1)+"1"+str(i+1),"S"+s
tr(j+2)+"1"+str(i+1), E, G, Iy_sol1, Iz_sol1, J_sol1, A_sol1,"ANsol"+str(j+1))
                                        #Campata TT3-TT4

      Girder.AddMember("Sol"+str(j+1)+"3"+str(i),"S"+str(j+1)+"3"+str(i+1),"S"+s
tr(j+2)+"3"+str(i+1), E, G, Iy_sol1, Iz_sol1, J_sol1, A_sol1,"ANsol"+str(j+1))



#SOLETTA campo TT2-TT3

#Proprietà sezione non omogeneizzata
Iy_sol2= 0.0513                         #m^4
Iz_sol2= 0.0010                  #m^4
J_sol2= 3.54e-3                         #m^4
A_sol2= 0.2908                          #m^2

# Elementi

for j in range(N_travi_long):
      if j == N_travi_long-1:
            break
      else:
            for i in range(div_b+1):
                  if i == 0:
                        pass
                  else:
                        #Campata TT2-TT3

      Girder.AddMember("Sol"+str(j+1)+"2"+str(i),"S"+str(j+1)+"2"+str(i+1),"S"+s
tr(j+2)+"2"+str(i+1), E, G, Iy_sol2, Iz_sol2, J_sol2, A_sol2,"ANsol"+str(j+1))


    ##########################################################################
                        PROVA DI CARICO CON MEZZI PESANTI
    ##########################################################################

# AUTOCARRO 3 assi 4 ton

# Asse anteriore -> 8 ton         --> 80 kN    --> 40 kN  per ogni trave
# Asse posteriore 1 -> 13     --> 130 kN   --> 65 kN  per ogni trave
# Asse posteriore 2 -> 13 ton     --> 130 kN   --> 65 kN  per ogni trave

# TL6 (trave esterna) - 4 autocarri
Girder.AddMemberPtLoad("TL616", "Fy",-40*1000,0.0392)
Girder.AddMemberPtLoad("TL6110","Fy",-65*1000,0.1852)
Girder.AddMemberPtLoad("TL6111","Fy",-65*1000,0.4717)
Girder.AddMemberPtLoad("TL6111","Fy",-40*1000,3.4717)
Girder.AddMemberPtLoad("TL622", "Fy",-65*1000,0.108)
Girder.AddMemberPtLoad("TL622", "Fy",-65*1000,1.408)

Girder.AddMemberPtLoad("TL625","Fy",-65*1000,0.046)
Girder.AddMemberPtLoad("TL625","Fy",-65*1000,1.346)
Girder.AddMemberPtLoad("TL631","Fy",-40*1000,0.2)

Girder.AddMemberPtLoad("TL631","Fy",-65*1000,3.2)
Girder.AddMemberPtLoad("TL632","Fy",-65*1000,0.8282)
Girder.AddMemberPtLoad("TL636","Fy",-40*1000,0.9742)

# TL5 (trave esterna) - 4 autocarri
Girder.AddMemberPtLoad("TL516", "Fy",-40*1000,0.0392)
Girder.AddMemberPtLoad("TL5110","Fy",-65*1000,0.1852)
Girder.AddMemberPtLoad("TL5111","Fy",-65*1000,0.4717)

Girder.AddMemberPtLoad("TL5111","Fy",-40*1000,3.4717)
Girder.AddMemberPtLoad("TL522", "Fy",-65*1000,0.108)
Girder.AddMemberPtLoad("TL522", "Fy",-65*1000,1.408)
```

```
Girder.AddMemberPtLoad("TL525","Fy",-65*1000,0.046)
Girder.AddMemberPtLoad("TL525","Fy",-65*1000,1.346)
Girder.AddMemberPtLoad("TL531","Fy",-40*1000,0.2)

Girder.AddMemberPtLoad("TL531","Fy",-65*1000,3.2)
Girder.AddMemberPtLoad("TL532","Fy",-65*1000,0.8282)
Girder.AddMemberPtLoad("TL536","Fy",-40*1000,0.9742)

    ##########################################################################
                                ATTRITO
    ##########################################################################

#Forze orizzontali derivanti dalla risoluzione del modello senza attrito

H_V14 = 56284.47185918281    #N
H_V24 = 58885.63705416343  #N
H_V34 = 61876.943219617235 #N
H_V44 = 65055.604967787105 #N
H_V54 = 70009.21113141545  #N
H_V64 = 73480.58151664004  #N

#Applicazione forze longitudinali

Girder.AddNodeLoad("V14", "FX",-H_V14)
Girder.AddNodeLoad("V24", "FX",-H_V24)
Girder.AddNodeLoad("V34", "FX",-H_V34)
Girder.AddNodeLoad("V44", "FX",-H_V44)
Girder.AddNodeLoad("V54", "FX",-H_V54)
Girder.AddNodeLoad("V64", "FX",-H_V64)

    ##########################################################################
ANALISI ###################################################################################

Visualization.RenderModel(Girder, 0.05)
Girder.Analyze()
Visualization.DeformedShape(Girder,500, 0.05)

    ##########################################################################
                        REAZIONI E SPOSTAMENTI NODALI
    ##########################################################################

print("REAZIONI VNCOLARI VERTICALI")
for i in range(N_travi_long):
      print("RZ_V"+str(i+1)+"1",Girder.GetNode("V"+str(i+1)+"1").RxnFZ)
print("\n")
for i in range(N_travi_long):
      print("RZ_V"+str(i+1)+"4",Girder.GetNode("V"+str(i+1)+"4").RxnFZ)

print("\n")
print("REAZIONI VNCOLARI TRASVERSALI")
for i in range(N_travi_long):
      print("RY_V"+str(i+1)+"1",Girder.GetNode("V"+str(i+1)+"1").RxnFY)
print("\n")
for i in range(N_travi_long):
      print("RY_V"+str(i+1)+"4",Girder.GetNode("V"+str(i+1)+"4").RxnFY)

print("\n")
print("REAZIONI VNCOLARI LONGITUDINALI ")
for i in range(N_travi_long):
      print("RX_V"+str(i+1)+"1",Girder.GetNode("V"+str(i+1)+"1").RxnFX)
print("\n")
for i in range(N_travi_long):
      print("RX_V"+str(i+1)+"4",Girder.GetNode("V"+str(i+1)+"4").RxnFX)

print("\n")
print("CONTROLLO SPOSTAMENTI LONGITUDINALI")
```

```
for i in range(N_travi_long):
       print("DX_V"+str(i+1)+"1",Girder.GetNode("V"+str(i+1)+"1").DX)
print("\n")
for i in range(N_travi_long):
       print("DX_V"+str(i+1)+"4",Girder.GetNode("V"+str(i+1)+"4").DX)

print("\n")
print("CONTROLLO SPOSTAMENTI TRASVERSALI")

for i in range(N_travi_long):
       print("DY_V"+str(i+1)+"1",Girder.GetNode("V"+str(i+1)+"1").DY)
print("\n")
for i in range(N_travi_long):
       print("DY_V"+str(i+1)+"4",Girder.GetNode("V"+str(i+1)+"4").DY)

print("\n")
print("CONTROLLO SPOSTAMENTI VERTICALI")

for i in range(N_travi_long):
       print("DZ_V"+str(i+1)+"1",Girder.GetNode("V"+str(i+1)+"1").DZ)
print("\n")
for i in range(N_travi_long):
       print("DZ_V"+str(i+1)+"4",Girder.GetNode("V"+str(i+1)+"4").DZ)

    ##########################################################################
                          CALCOLO ROTAZIONI INCLINOMETRI
    ##########################################################################

L_inclinometro = 2 #m

#Inclinometri lato NORD
x1=0.7182 #m

#Abbassamenti estremità iniziale incliometri
D_inclinometro_TL2_N_start = Girder.GetNode("N211").DZ
D_inclinometro_TL3_N_start = Girder.GetNode("N311").DZ
D_inclinometro_TL4_N_start = Girder.GetNode("N411").DZ
D_inclinometro_TL5_N_start = Girder.GetNode("N511").DZ

# Abbassamenti estremità finale inclinometri
D_inclinometro_TL2_N_end = Girder.GetMember("TL212").Deflection("dy",x1)
D_inclinometro_TL3_N_end = Girder.GetMember("TL312").Deflection("dy",x1)
D_inclinometro_TL4_N_end = Girder.GetMember("TL412").Deflection("dy",x1)
D_inclinometro_TL5_N_end = Girder.GetMember("TL512").Deflection("dy",x1)



# Calcolo rotazioni inclinometri nord (mrad)
Rot_inclinometro_TL2_N = ((D_inclinometro_TL2_N_end -
D_inclinometro_TL2_N_start)/L_inclinometro)*1000
Rot_inclinometro_TL3_N = ((D_inclinometro_TL3_N_end -
D_inclinometro_TL3_N_start)/L_inclinometro)*1000
Rot_inclinometro_TL4_N = ((D_inclinometro_TL4_N_end -
D_inclinometro_TL4_N_start)/L_inclinometro)*1000
Rot_inclinometro_TL5_N = ((D_inclinometro_TL5_N_end -
D_inclinometro_TL5_N_start)/L_inclinometro)*1000


#Inclinometri lato SUD
x2=0.2953 #m

#Abbassamenti estremità iniziale incliometri
D_inclinometro_TL2_S_start = Girder.GetNode("N241").DZ
D_inclinometro_TL3_S_start = Girder.GetNode("N341").DZ
D_inclinometro_TL4_S_start = Girder.GetNode("N441").DZ
D_inclinometro_TL5_S_start = Girder.GetNode("N541").DZ

# Abbassamenti estremità finale inclinometri
```

```
D_inclinometro_TL2_S_end = Girder.GetMember("TL2310").Deflection("dy",x2)
D_inclinometro_TL3_S_end = Girder.GetMember("TL3310").Deflection("dy",x2)
D_inclinometro_TL4_S_end = Girder.GetMember("TL4310").Deflection("dy",x2)
D_inclinometro_TL5_S_end = Girder.GetMember("TL5310").Deflection("dy",x2)

# Calcolo rotazioni inclinometri sud (mrad)
Rot_inclinometro_TL2_S = ((D_inclinometro_TL2_S_end -
D_inclinometro_TL2_S_start)/ L_inclinometro)*1000
Rot_inclinometro_TL3_S = ((D_inclinometro_TL3_S_end -
D_inclinometro_TL3_S_start)/ L_inclinometro)*1000
Rot_inclinometro_TL4_S = ((D_inclinometro_TL4_S_end -
D_inclinometro_TL4_S_start)/ L_inclinometro)*1000
Rot_inclinometro_TL5_S = ((D_inclinometro_TL5_S_end -
D_inclinometro_TL5_S_start)/ L_inclinometro)*1000


# Creo file .txt di salvataggio
Rotazioni_inclinometri=open("Rotazioni inclinometri.txt", "w")
Rotazioni_inclinometri.write("EL-C23-T5-N = "+str(Rot_inclinometro_TL2_N)+ " mrad
"+ "\n")
Rotazioni_inclinometri.write("EL-C23-T4-N = "+str(Rot_inclinometro_TL3_N)+ " mrad
"+ "\n")
Rotazioni_inclinometri.write("EL-C23-T3-N = "+str(Rot_inclinometro_TL4_N)+ " mrad
"+ "\n")
Rotazioni_inclinometri.write("EL-C23-T2-N = "+str(Rot_inclinometro_TL5_N)+ " mrad
"+ "\n")
Rotazioni_inclinometri.write("\n")
Rotazioni_inclinometri.write("EL-C23-T5-S = "+str(Rot_inclinometro_TL2_S)+ " mrad
"+ "\n")
Rotazioni_inclinometri.write("EL-C23-T4-S = "+str(Rot_inclinometro_TL3_S)+ " mrad
"+ "\n")
Rotazioni_inclinometri.write("EL-C23-T3-S = "+str(Rot_inclinometro_TL4_S)+ " mrad
"+ "\n")
Rotazioni_inclinometri.write("EL-C23-T2-S = "+str(Rot_inclinometro_TL5_S)+ " mrad
"+ "\n")
```

A3 Risultati delle prove di carico



Bibliografia e Sitografia

[1] C. Farrar, «A Statistical Pattern Recognition Paradigm for Vibration-Based Structural».

[2] «Beninati L., Brunetti A., Caruso C., Mazzanti P., 2015.,» *Structural Health Characterization of an Old Riveted Iron Bridge By Remote Sensing Techniques,* Proceedings of the 7th International Conference on Structural Health Monitoring of Intelligent Infrastructure (Turin, Italy, 1-3 July 2015).

[3] «Bongiovanni G., Brunetti A., Clemente P., Conti C., Mazzanti P., Verrubbi V., 2015.,» *Dynamic characterization of tower structures by means of interferometry measurements.,* Proceedings of the 7th International Conference on Structural Health Monitoring of Intelligent Infrastructure (Turin, Italy, 1-3 July 2015).

[4] «Mazzanti Paolo - Earth Sciences Department, Sapienza University of Rome, NHAZCA S.r.l, Brunetti Alessandro - NHAZCA S.r.l., Spin-off di Università degli Studi di Roma "La Sapienza",» *Il monitoraggio dinamico delle strutture e delle infrastrutture con interferometria radar terrestre,* 25/07/2015.

[5] A. Mita, «Fiber Bragg Grating-Based Acceleration Sensors for Civil and Building Structures,» in *International Workshop on Present and Future in Health Monitoring*, Bauhaus-University Weimar, Germany, 2000.

[6] «Congrui Zhang, Yongxiang Ge, Zhongchun Hu, Ke Zhou,Gaofeng Ren, Xiaodong Wang, Research on deflection monitoring for long span cantilever bridge based on optical fiber sensing,» *Elsevier,* 2019.

[7] «Dongtao Hu, Yongxing Guo, Xianfeng Chen, and Congrui Zhang - Cable Force Health Monitoring of Tongwamen Bridge Based on Fiber Bragg Grating,» *Applied Sciences,* 2017.

[8] «F. Huseynov, C. Kim, E.J. OBrien, J.M.W. Brownjohn, D. Hester, K.C Chang "Bridge damage detection using rotation measurements –Experimental validation",» *Elsevier,* 2019.

[9] «B. Glišic´ , D. Posenato, D. Inaudi et al. "Structural health monitoring method for curved concrete bridge box girders.",» *Tomizuka, M., (ed). International.*

[10] «Riqing Lan et al. "Reconstitution of Static Deflections of Suspension Bridge Based on Inclinometer Data,» *IOP Conference Series: Earth and Environmental Science,* 2019.

[11] D. M. Y. P. Sergey Konovalov, «"High-precision smart system on accelerometers and inclinometers for Structural Health Monitoring: development and applications",» *Bauman Moscow State Technical University.*

[12] «JinHui Ye, MingDa Huang, YuFeng Xu, ZhaoFeng Li, "Design of Deformation Monitoring System of Arch Rib of Steel Arch Bridge Based on Inclination",» *IEEE,* The 31th Chinese Control and Decision Conference (2019 CCDC).

[13] «Python,» [Online]. Available: https://www.python.it/about/. [Consultato il giorno 13 Febbraio 2020].

[14] «Python Package Index,» [Online]. Available: https://pypi.org/. [Consultato il giorno 6 Dicembre 2019].

[15] «GitHub,» [Online]. Available: https://github.com/. [Consultato il giorno 6 Dicembre 2019].

[16] «pip - The Python Package Installer,» [Online]. Available: https://pip.pypa.io/en/stable/. [Consultato il giorno 02 Gennaio 2020].

[17] Eurocode 2: Design of concrete structures - Part 1-1: General rules and rules for buildings.

[18] G. Bertagnoli, Dispense del corso di "Bridge Design", Politecnico di Torino - DISEG, A.A. 2018/2019.

Ringraziamenti

Alla fine di questo lungo e faticoso percorso voglio dedicare alcune righe a tutti coloro che mi hanno accompagnato e sostenuto in questi anni, ognuno a modo suo.

Primi tra tutti ringrazio mio padre e mia madre, che con i loro grandi sacrifici mi hanno permesso di raggiungere questo grande traguardo. E Gianpaolo, fratello piccolo ma grande allo stesso tempo.

Ringrazio il professore Gabriele Bertagnoli per avermi seguito in questo lavoro di tesi. Senza dubbio, è stato uno dei migliori professori incontrati in questa esperienza universitaria. La ringrazio per la pazienza, la disponibilità e l'interesse mostrato in tutto il periodo di lavoro insieme.

Ringrazio gli amici colleghi universitari con i quali ho trascorso periodi intensi, giornate interminabili e piene di stress. Dicevamo sempre che un giorno ce l'avremmo fatta e finalmente il grande giorno è arrivato.

Ringrazio gli amici di Villa Claretta con i quali ho trascorso uno dei periodi più belli di questa vita universitaria. Non dimenticherò mai le tante serate e i bei momenti trascorsi insieme. E' stato bello conoscervi.

Ringrazio chiunque altro mi è stato accanto in questi anni, chi ha sempre creduto in me nonostante tutto, chiunque abbia contribuito con un sorriso, con un gesto o con una semplice parola a rendere più leggero questo cammino, anche chi ha deciso di andarsene.

Infine, un pensiero a nonna Rosa, presente in prima fila alla laurea triennale e oggi sempre presente ma molto lontano da tutti noi. Sono sicuro che anche adesso sarai fiera di tuo nipote Luciano.

Grazie!