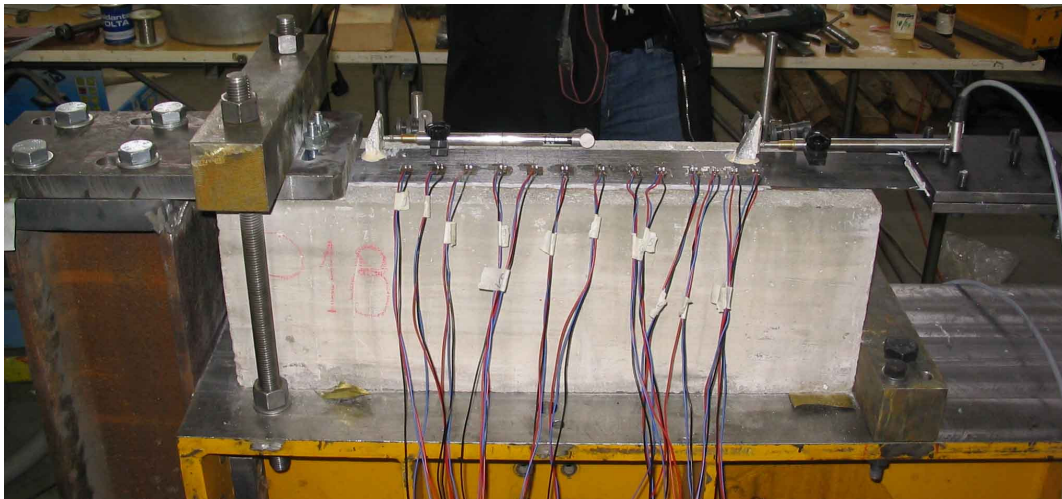


UNIVERSITÀ DEGLI STUDI DI BOLOGNA FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA
INGEGNERIA CIVILE (LS)

METODI NUMERICI PER L'INGEGNERIA CIVILE

SIMULAZIONI NUMERICHE DI PROVE DI TAGLIO DIRETTA TRA CALCESTRUZZO ED ELEMENTI FIBRO-RINFORZATI



STUDENTI:

*CHIODI DAVIDE (0000233721)
LEROSE SALVINO (0000233726)*

PROFESSORI:

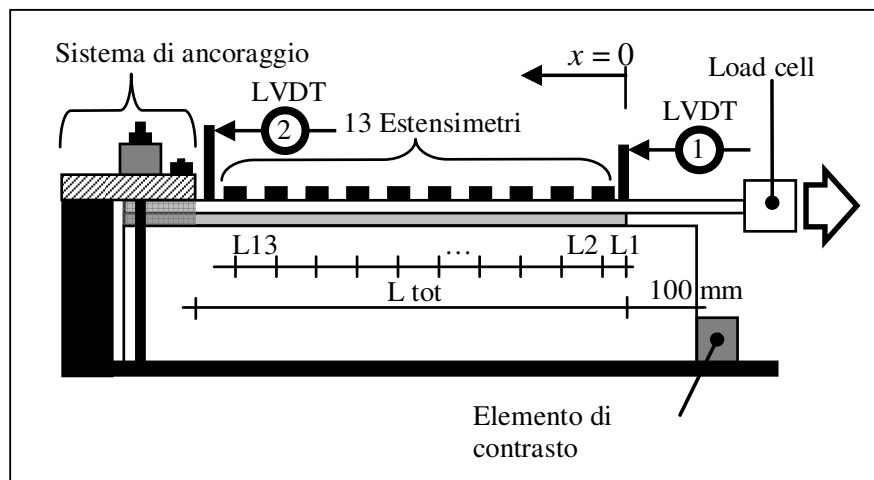
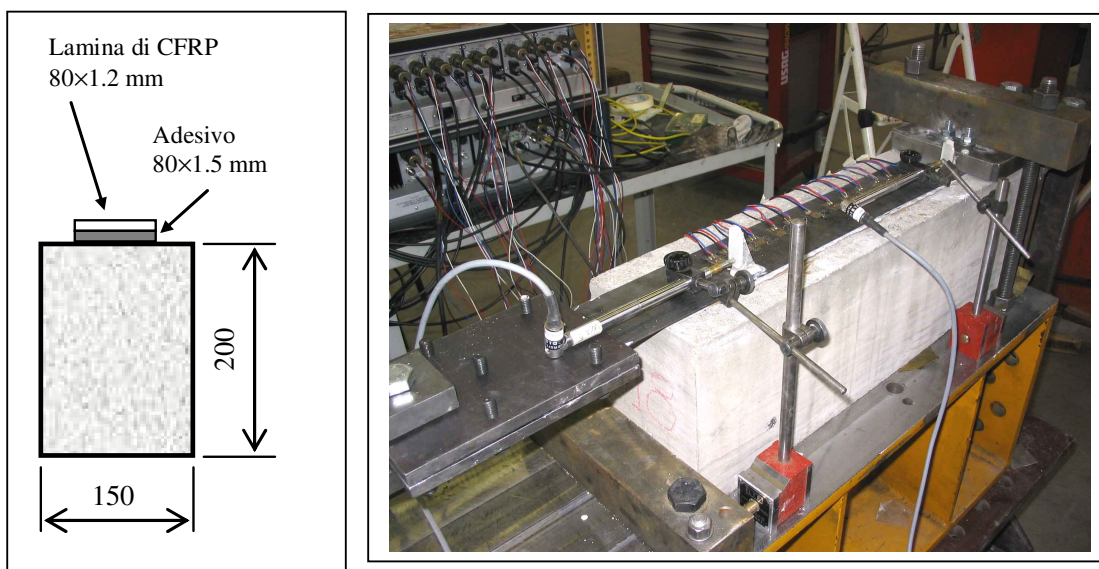
*F. UBERTINI
F. SGALLARI*

1. Descrizione della prova

Il nostro lavoro descrive una simulazione numerica di una prova di taglio diretto su un campione di calcestruzzo sul quale è stata incollata una lamina di materiale composito fibro-rinforzato (FRP).

La prova consiste nell'applicare progressivamente un carico ad un'estremità libera della placca sino ad ottenere il completo distacco di quest'ultima dal blocco di calcestruzzo (fenomeno della delaminazione).

Nelle figure vengono illustrate la geometria e le apparecchiature utilizzate per l'esecuzione delle prove.



Le resistenze dei materiali utilizzati e le loro caratteristiche meccaniche vengono in seguito riportate:

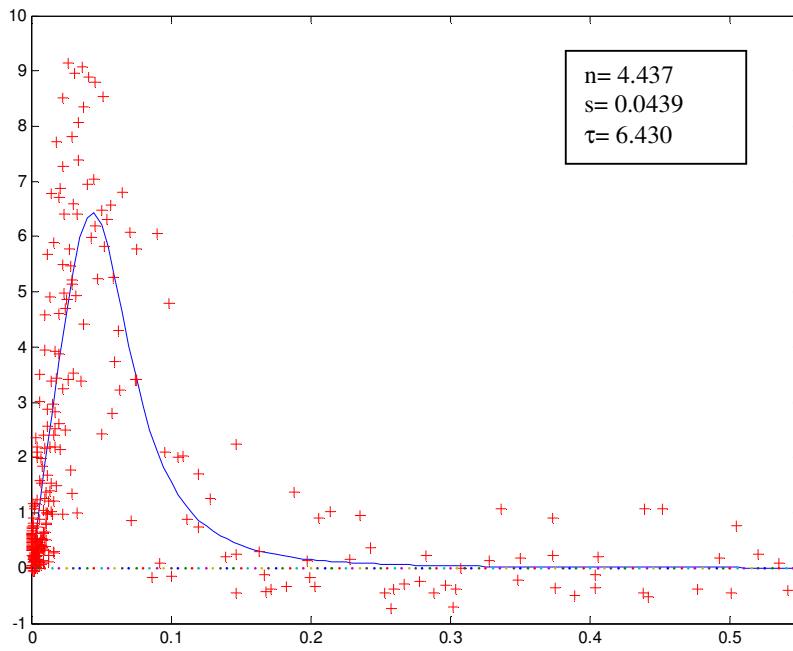
- Calcestruzzo
 - $f_{cm} = 52.7 \text{ MPa}$
 - $f_{ctm} = 3.81 \text{ MPa}$
 - $E_{cm} = 30700 \text{ MPa}$
 - $\nu = 0.227$
- Lamina FRP
 - Resistenza minima valutata = 2200 MPa
 - $E = 165000 \text{ MPa}$
- Adesivo
 - Resistenza a compressione = 95 MPa
 - $E = 12800 \text{ MPa}$

Mediante la realizzazione delle prove sperimentali è stato possibile definire una legge d'interfaccia tra i due materiali; la forma della legge d'interpolazione adottata è analoga a quella proposta da Popovics per il legame costitutivo del calcestruzzo:

$$k_p = \frac{\bar{\tau}}{s} \frac{n}{(n-1) + (s_p / \bar{s})^n} \quad (1)$$

I tre parametri della legge di interfaccia presenti nella formula d'interpolazione sono stati ottenuti con una procedura ai minimi quadrati a partire dai dati sperimentali, utilizzando il valore dell'energia di frattura come vincolo nel processo di minimizzazione dello scarto tra risultati sperimentali e previsti dalla legge di interfaccia.

$$\min_{\tau, s, n} \sum_{i=1}^m \left(\tau_i \left(\tau_{\max}, \bar{s}, n \right) - \tau_{sper,i} \right)^2 \quad (2)$$

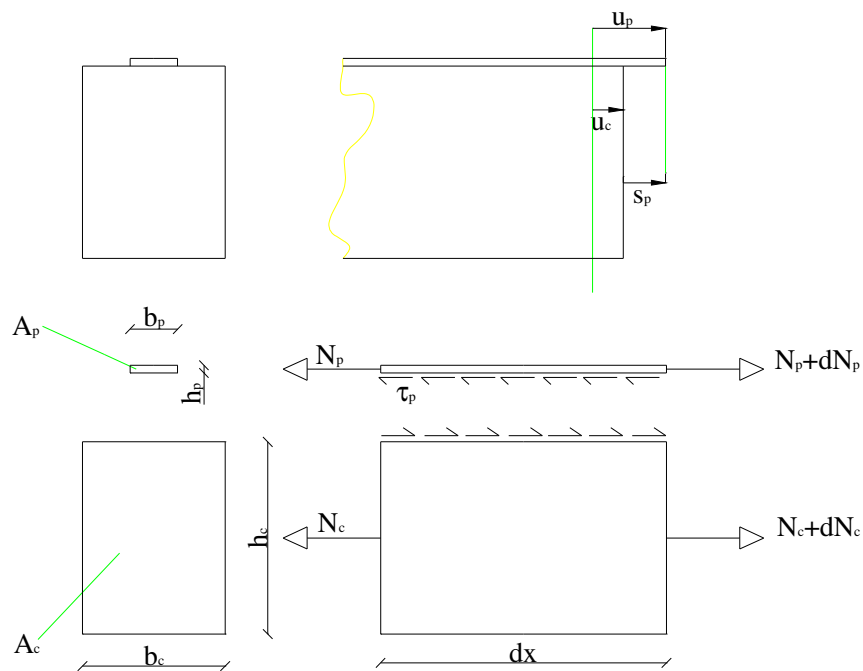


2. Modello numerico proposto

Il modello cinematico presentato per lo studio del fenomeno della delaminazione viene sviluppato dall'assunzione che sia la placca che il calcestruzzo siano soggette solo a deformazioni assiali, ove quindi l'intera deformabilità viene considerata concentrata a livello dell'interfaccia. Gli spostamenti assiali e le forze applicate relative alle sezioni di calcestruzzo e FRP sono indicate rispettivamente con u_c , N_c , u_p , N_p .

La deformabilità flessionale viene trascurata a causa della modesta rigidezza delle placche di FRP rispetto a quella del calcestruzzo

La notazione adottata per gli spostamenti e le tensioni viene illustrata in figura.



3. Equazioni governanti del modello

Le equazioni governanti del modello sono le equazioni di equilibrio, le leggi costitutive e le condizioni di compatibilità interna.

Riferendosi ad un concio di dimensione infinitesima le equazioni governanti sono:

$$\left\{ \begin{array}{l} \frac{du_p}{dx} = \frac{I}{E_p \cdot A_p} \cdot N_p \\ \frac{dN_p}{dx} = b_p \cdot \tau_p \\ \frac{du_c}{dx} = \frac{I}{E_c \cdot A_c} \cdot N_c \\ \frac{dN_c}{dx} = -2 \cdot b_p \cdot \tau_p \end{array} \right. \quad (3)$$

dove:

A ed E rappresentano l'area e il modulo di Young;

τ_p è la tensione tangenziale di interfaccia calcestruzzo-placca;

b_p è la larghezza della placca.

Si assume che le sezioni trasversali dei due materiali rimangano piane anche dopo la deformazione, per cui è possibile riferirsi solamente alle deformazioni assiali del piano medio, indicate con ϵ_c ed ϵ_p , trascurando così sia la deformazione flessionale e di taglio.

Lo scorrimento tangenziale, facendo riferimento all'interfaccia, viene definito come differenza tra scorrimento della placca e quello del calcestruzzo:

$$s_p = u_p - u_c$$

Secondo la legge di interfaccia della formulazione di Popovics si può scrivere la seguente formula, che rappresenta l'equazione generale all'interfaccia placca-calcestruzzo:

$$\tau_p = k_p(s_p) \cdot s_p \quad (4)$$

dove:

s_p rappresenta lo scorrimento placca-calcestruzzo

$k_p(s_p)$ rappresenta la rigidità secante pari a:

$$k_p = \frac{\bar{\tau}}{s} \frac{n}{(n-1) + (s_p / \bar{s})^n} \quad (5)$$

Sostituendo l'equazione generale dell'interfaccia (4) all'interno delle equazioni governanti (3) è possibile formulare il sistema di equazioni differenziali del primo ordine, che descrivono il problema in ambito elastico lineare:

$$\begin{cases} \frac{du_p}{dx} = \frac{I}{E_p \cdot A_p} \cdot N_p \\ \frac{dN_p}{dx} = k_p \cdot (u_p - u_c) \\ \frac{du_c}{dx} = \frac{I}{E_c \cdot A_c} \cdot N_c \\ \frac{dN_c}{dx} = 2 \cdot k_p \cdot (u_c - u_p) \end{cases} \quad (6)$$

dove:

$$K_p = \frac{b_p}{A_p} k_p \quad (7)$$

rappresenta la rigidità secante della legge d'interfaccia.

La rappresentazione in forma matriciale del sistema di equazioni differenziali può essere scritta come:

$$\frac{dy(x)}{dx} = A(y, x) \cdot y(x) \quad \text{con } 0 \leq x \leq L \quad (8)$$

dove:

L è la lunghezza di ancoraggio della placca,

$y^T = \{N_p, N_c, u_p, u_c\}$ è il vettore delle funzioni incognite,

A è la matrice non lineare dei coefficienti definita come:

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & K_p & -K_p \\ 0 & 0 & -K_p & K_p \\ \frac{1}{E_p \cdot A_p} & 0 & 0 & 0 \\ 0 & \frac{1}{E_c \cdot A_c} & 0 & 0 \end{bmatrix} \quad (9)$$

4. Risoluzione numerica

4.1 Risoluzione tramite il metodo delle differenze finite

Il metodo alle differenze finite è una tecnica di analisi numerica per ottenere una soluzione approssimata di una equazione differenziale, questa si ottiene approssimando il dominio continuo V , sul quale è definita la funzione incognita dell'equazione stessa, a una serie di punti discreti del dominio stesso. In questo modo, invece di ottenere una funzione continua in V , la soluzione dell'equazione assume il valore di questi punti isolati.

La riduzione dell'equazione differenziale che governa il fenomeno e delle relative condizioni al contorno del dominio V nelle equazioni del dominio discreto, si ottiene con un metodo fisico matematico.

Fisicamente si costruisce un modello a parametri concentrati e per quest'ultimo si scrivono le equazioni, mentre matematicamente la formulazione continua è ridotta a discreta sostituendo alle derivate dell'equazione differenziale delle differenze finite.

Il problema da cui si parte è del tipo:

$$y' = f(x; y) \quad a \leq x \leq b \quad (10)$$

$$y(a) = \alpha \quad (11)$$

Il metodo più semplice per risolverlo è il cosiddetto metodo delle differenze in avanti o metodo di Eulero, usato nel nostro caso per risolvere il problema considerato, esso consiste nel fissare all'interno dell'intervallo di definizione $[a, b]$ una serie di $N+1$ punti equidistanti, tali per cui:

$$x_j = x_0 + j \cdot h \quad (12)$$

dove:

$$j = 1, 2, \dots, N$$

h passo di discretizzazione

$$h = \frac{b-a}{N} \quad (13)$$

sostituendo in (10) il rapporto incrementale alla derivata nel punto j

$$y' = \frac{y_{j+1} - y_j}{h} \quad (14)$$

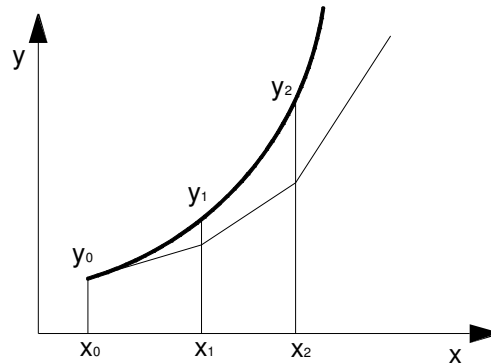
si ottiene che

$$y_{j+1} = y_j + h \cdot f(x; y_j) \quad (15)$$

e la condizione al contorno (11) diventa

$$y_0 = a \quad (16)$$

Il metodo si presta anche ad un'interpretazione geometrica illustrata in figura:



Il punto $(x_{j+1}; y_{j+1})$ ottenuto applicando la formula di Eulero vista sopra, è individuato dalla retta passante per il punto precedente $(x_j; y_j)$ e parallela alla tangente, calcolata nel medesimo punto $(x_j; y_j)$, della curva rappresentante la soluzione esatta.

Si passa quindi da un'equazione differenziale definita su un continuo ad un sistema algebrico lineare di tante equazioni quanti sono i punti (nodi) in cui è stato approssimato il continuo stesso, e con altrettante incognite, le quali sono i valori che la funzione y assume nei medesimi punti (nodi).

Il grado di approssimazione della soluzione dipende senz'altro dal passo h che è stato scelto al momento della discretizzazione del dominio, più h è piccolo e più la soluzione risulta essere precisa, ma è anche maggiore l'onere computazionale richiesto per la soluzione del problema.

Il sistema di equazioni differenziali riguardante il nostro problema è governato da un vettore delle incognite del tipo:

$$y^T = \{\sigma_p, \sigma_c, u_p, u_c\} \quad (17)$$

ossia considerando indipendenti le forze assiali e gli spostamenti di calcestruzzo e placca.

Le condizioni al contorno che andiamo ad imporre sono:

$$\begin{cases} u_p(x=0) = 0 \\ u_c(x=0) = 0 \\ \sigma_c(x=L) = 0 \\ u_p(x=L) = \Delta L \end{cases} \quad (18)$$

dove l'ultima condizione rappresenta lo spostamento imposto all'estremo libero della placca.

Si possono anche scrivere nella forma:

$$[B_a]\{y_0\} + [B_b]\{y_L\} = \{\alpha\} \quad (19)$$

dove

$$B_a = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad B_b = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \alpha = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ \Delta L \end{Bmatrix}$$

In forma compatta il problema differenziale si scriverà dunque nella forma:

$$\begin{cases} \{y'(x)\} = [A(y(x)), x] \cdot \{y(x)\} \\ [B_a]\{y_0\} + [B_b]\{y_L\} = \{\alpha\} \end{cases} \quad 0 \leq x \leq L \quad (20)$$

Un problema di questo tipo, con condizioni al contorno definite ai limiti del dominio di integrazione, prende il nome di "problema ai limiti".

Per il generico nodo interno alla mesh ($0 < j < J$), il vettore delle derivate delle incognite $\{y\}$ è sostituito dall'approssimazione $\frac{\{y\}_{j+1} - \{y\}_j}{h}$ fornita dalla differenza finita centrata in $x_{j+1/2}$.

Successivamente si utilizza la formula dei trapezi per trasformare il sistema di equazioni differenziali (20) in un sistema di equazioni algebriche

$$\begin{cases} \{y\}_{j+1} = \{y\}_j + \frac{h}{2} ([A(\{y\}_{j+1}, x_{j+1})] \cdot \{y\}_{j+1} + [A(\{y\}_j, x_j)] \cdot \{y\}_j) \\ [B_a] \cdot \{y\}_0 + [B_b] \cdot \{y\}_j = \{\alpha\} \end{cases} \quad j = 1, 2, \dots, J-1 \quad (21)$$

Il sistema di equazioni (21) può essere scritto in forma matriciale come

$$[K]\{y\} = \{f\}$$

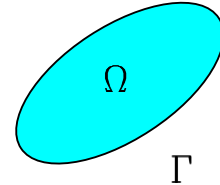
dove il vettore $\{y\}$ contiene i valori delle funzioni incognite nei J nodi, la matrice [K] contiene il set delle funzioni incognite, che dipende dalla soluzione $\{y\}$ in

4.2 Risoluzione numerica tramite il metodo del point collocation

Il metodo di discretizzazione *point collocation* prevede il passaggio da una formulazione forte (SF) di un problema ad una integrale debole (WF).

A partire da un sistema generico:

$$\begin{cases} A(u) = 0 \rightarrow \text{dominio } \Omega \\ B(u) = 0 \rightarrow \text{contorno } \Gamma \end{cases} \quad \text{SF} \quad (25)$$



la formulazione integrale si ottiene ponendo:

$$\int_{\Omega} w(x) \cdot A(u) d\Omega + \int_{\Gamma} w(x) \cdot B(u) d\Gamma = 0 \quad \text{WF} \quad (26)$$

dove $w(x)$ è una generica funzione test.

In particolare nel metodo point collocation si pone $w_j = \delta_j$ (dove δ_j è la delta di Dirach).

Contemporaneamente la funzione incognita u viene discretizzata e approssimata tramite delle funzioni di forma:

$$u(x) = u^h(x) = N_i(x) \cdot a_i \quad (27)$$

dove \bar{a} rappresenta il vettore dei parametri incogniti ed N sono le funzioni interpolanti.

Il sistema di equazioni differenziali caratterizzanti il nostro problema

$$B(y', y, x) = \frac{dy(x)}{dx} - A(y, x) \cdot y(x) - f = 0 \quad \text{con } 0 \leq x \leq L \quad (8)$$

è stato rappresentato in forma integrale come:

$$\int_L w^T(x) \cdot B(y, y') dx = 0 \quad (28)$$

Introdotte le condizioni:

$$\begin{cases} y(x) = y^h(x) = \tilde{N}(x) \cdot \underline{a} + N_* \cdot \Delta L \\ w_j = \delta_j \end{cases} \quad N_i(x) \in C^0 \quad j=1, \dots, n \quad (29)$$

Si ottiene il seguente sistema di equazioni:

$$B(y^h) = B(\tilde{N} \cdot \underline{a}) \Big|_{x=x_j} = 0 \quad y=1, \dots, n \quad (30)$$

La scelta delle funzioni di forma deve rispettare i vincoli delle condizioni al contorno:

$$\begin{cases} u_p(x=0) = 0 \\ u_c(x=0) = 0 \\ \sigma_c(x=L) = 0 \\ u_p(x=L) = \Delta L \end{cases}$$

(18)

- Funzione di forma N_1 per l'incognita σ_p

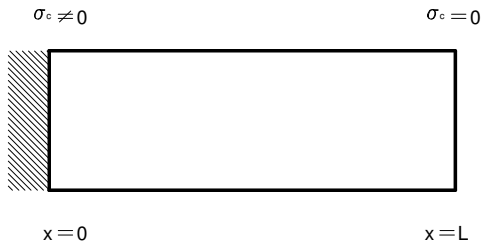


$$N_1^1(x) = \frac{x}{L}$$

$$N_1^2(x) = \frac{L-x}{L} \quad i=3, \dots, n$$

$$N_1^i(x) = \sin\left(\frac{i\pi x}{L}\right)$$

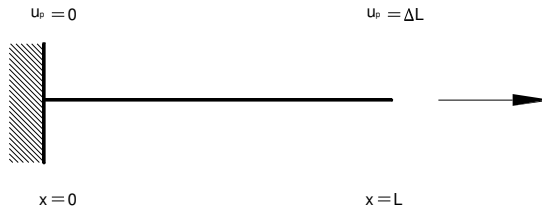
- Funzione di forma N_2 per l'incognita σ_c



$$N_2^1(x) = \frac{L-x}{L} \quad i=2, \dots, n$$

$$N_2^i(x) = \sin\left(\frac{i\pi x}{L}\right)$$

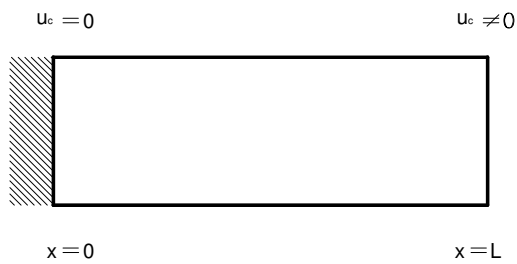
- Funzione di forma N_3 per l'incognita u_p



$$N_3^i(x) = \sin\left(\frac{i\pi x}{L}\right) \quad i=1, \dots, n$$

$$N_*(x) = \frac{x}{L} \cdot \Delta L$$

- Funzione di forma N_4 per l'incognita u_c



$$N_4^1(x) = \frac{x}{L}$$

$$N_4^i(x) = \sin\left(\frac{i\pi x}{L}\right) \quad i=2, \dots, n$$

È possibile quindi esprimere il vettore delle incognite y come:

$$y^h = \begin{bmatrix} [N_1] \\ [N_2] \\ [N_3] \\ [N_4] \end{bmatrix} \cdot \begin{bmatrix} [a_1] \\ [a_2] \\ [a_3] \\ [a_4] \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ x/L \\ 0 \end{bmatrix} \cdot \Delta L \quad (31)$$

Le funzioni di forma sono state scelte appartenenti alla classe C^0 e facilmente derivabili.

Il sistema di equazione (30) assume quindi la forma:

$$N' \cdot a + N'_* \cdot \Delta L - A \cdot N \cdot a - A \cdot N_* \cdot \Delta L = 0 \Big|_{x=x_j} \quad j=1, \dots, n \quad (32)$$

$$[N' - AN] \cdot a = [AN_* - N'_*] \cdot \Delta L \Big|_{x=x_j} \quad j=1, \dots, n \quad (33)$$

che può essere scritto in forma matriciale:

$$\begin{bmatrix} [k_{x1}] \\ \dots \\ \dots \\ \dots \\ [k_{xn}] \end{bmatrix} \cdot \{a\} = \begin{bmatrix} [f_1] \\ \dots \\ \dots \\ \dots \\ [f_n] \end{bmatrix} \quad (34)$$

dove

$$k_{xi} = [N' - AN] \Big|_{x=xi} \quad \text{e} \quad f_i = [AN_* - N'_*] \Big|_{x=xi},$$

oppure in forma compatta:

$$[K]\{a\} = \{f\} \quad (35)$$

5. Risoluzione del sistema mediante il metodo di Newton-Raphson

Per la risoluzione del sistema di equazioni non lineari è stato adottato il metodo di Newton-Raphson (procedimento incrementale iterativo).

Per poter applicare tale procedimento occorre trasformare il vettore delle incognite $\{a\}$ in un vettore incrementale $\{\Delta a\}$. Nello stesso modo viene trasformato il vettore dei termini noti $\{f\}$.

Il sistema (35) assume la seguente espressione:

$$[K]\{\Delta a\} = \{\Delta f\} \quad (36)$$

Si opera cioè una suddivisione dello spostamento ΔL in incrementi con passo minore; in particolare nella nostra analisi si è assunto un $\Delta L=0,5\text{mm}$, imposto con incrementi di passo pari a $0,002\text{mm}$.

Per ogni incremento il metodo di Newton-Raphson innesca un processo iterativo che risolve il sistema.

Riassumendo si eseguono i seguenti passi:

- Suddivisione di ΔL in incrementi di $0,002\text{mm}$;
- Applico il primo incremento;
- Calcolo il vettore noto $\{\Delta f\}_1$;
- Deduzione del valore di $\{\Delta a\}_1$ tramite ciclo iterativo.

Il ciclo iterativo si innesca risolvendo il seguente sistema di equazioni:

$$[K]_0 \{\Delta a\}_1^1 = \{\Delta f\}_1 \quad (37)$$

Nella relazione l'indice inferiore indica che si opera con il primo incremento di carico, mentre l'indice superiore numera l'iterazione.

Risolto tale sistema con il metodo L,U,P e calcolati quindi gli incrementi $\{\Delta a\}_1^1$ si calcolano i totali:

$$\{a\}_1^1 = \{a\}_0 + \{\Delta a\}_1^1 \quad (38)$$

dove $\{a\}_0$ è il vettore che caratterizza la configurazione iniziale della struttura.

La matrice delle rigidezze tangente $[K]_i^1$ e $[\Delta f]_i^1$ sono calcolati per la configurazione della struttura $\{a\}_i^1$ ottenuta al termine della prima iterazione.

In particolare, per ogni punto di discretizzazione del dominio, viene calcolato lo scorrimento relativo $s_p = u_p - u_c$ e quindi la rigidezza dell'interfaccia. Quest'ultimo valore viene inserito nella (9), matrice $[A]$, e successivamente va ad aggiornare i termini del sistema.

Quindi il vettore degli squilibri è dato dalla seguente differenza:

$$\{RE\}_1^1 = \{\Delta f\}_1^1 - \{\Delta f\}_1^1 \quad (39)$$

Il passo successivo consiste nell'applicare ai vari nodi il vettore degli squilibri risolvendo nuovamente il sistema di equazioni:

$$[K]_1^1 \{\Delta a\}_1^2 = \{RE\}_1^1 \quad (40)$$

È da notare che, operando in questa maniera, si sta compiendo una seconda iterazione all'interno del primo incremento di carico.

Risolto il sistema il vettore incognito vale:

$$\{a\}_1^2 = \{a\}_0 + \{\Delta a\}_1^1 + \{\Delta a\}_1^2 \quad (41)$$

A questo punto non resta che ripetere operazioni analoghe a quelle sopra descritte per ottenere un nuovo vettore degli squilibri $\{RE\}_1^2$; il procedimento iterativo prosegue fino a che gli squilibri nodali sono sufficientemente piccoli rispetto al vettore $\{\Delta f\}_1^1$, ed esattamente vengono confrontati i moduli del vettore dei residui e del vettore dei carichi applicati ad inizio del passo:

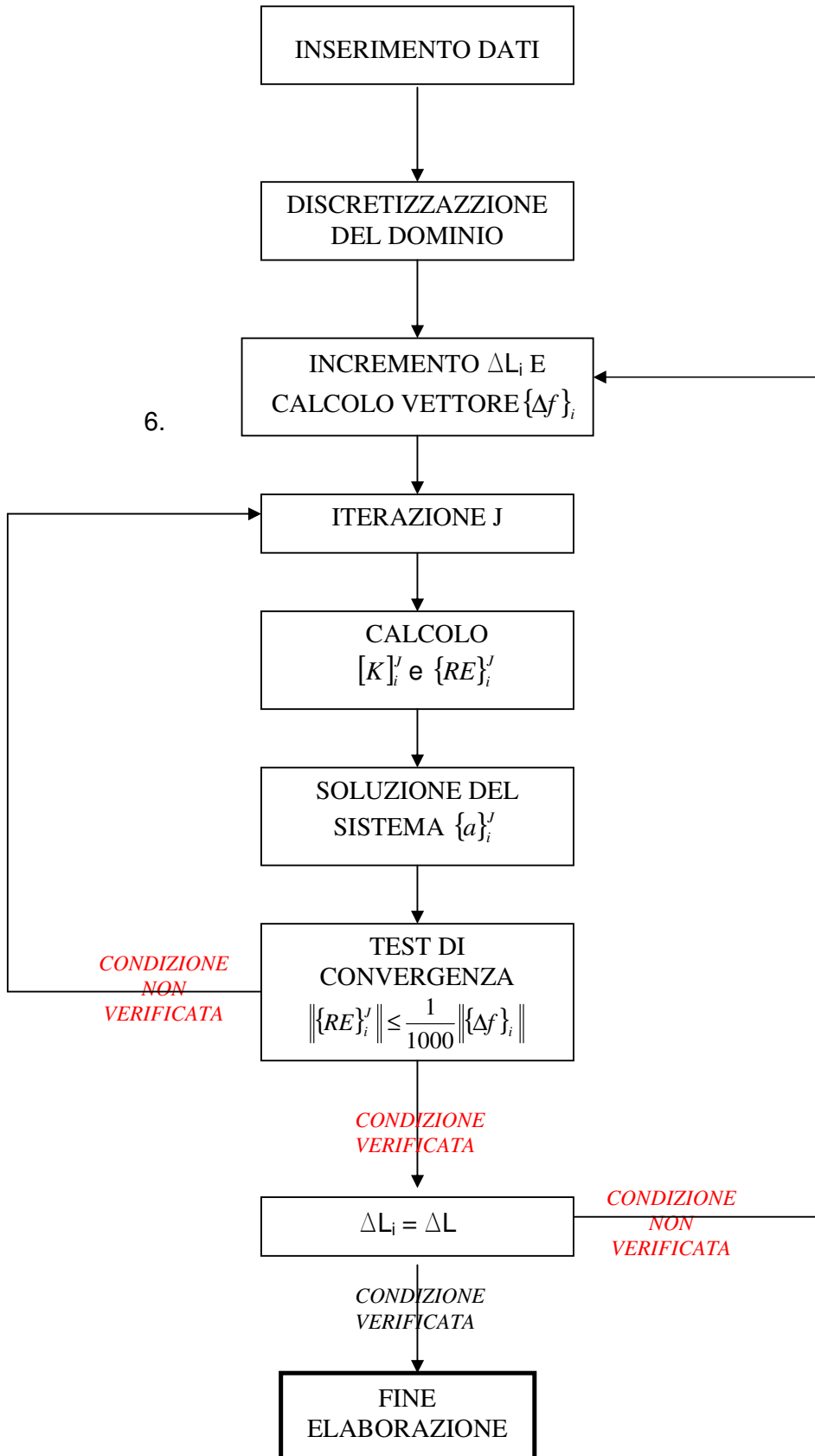
$$\|RE\| \leq \frac{1}{1000} \|\{\Delta f\}_1^1\| \quad (42)$$

Se tale condizione è soddisfatta si opera un secondo incremento di carico $[\Delta f]_2$ ripetendo all'interno di tale incremento una serie di operazioni analoghe a quelle precedentemente descritte.

Il processo incrementale si arresta una volta raggiunto il valore di spostamento dell'estremo libero della placca pari a ΔL .

Il metodo di Newton-Raphson converge con rapidità, ma presenta l'inconveniente che ad ogni iterazione è necessario aggiornare l'espressione della matrice di rigidezza e del termine noto.

5.1 Schematizzazione procedimento di Newton-Raphson



6. Compilazione del programma in MATLAB

```

%Caratteristiche dei materiali utilizzati
%Calcestruzzo
Ec=30700;
Ac=30000;
%Elemento FRP
l=332;
bp=80;
Ap=96;
Ep=165000;
kp=188.65;
Kp=bp*Ap;
%Numero elementi di discretizzazione
n=150;

%Grandezze incognite
SigmaC=0;
SigmaP=0;
Up=0;
Uc=0;
%vettore delle incognite
Inc=[SigmaP;SigmaC;Up;Uc];
%Legame costitutivo dell'interfaccia calcestruzzo-FRP
%A=6.43;
%B=0.044;
%C=4.437;
Sp=Up-Uc;
%Tau=Sp*(A/B)*(C/((C-1)*(Sp/B)^C));
Tau=Sp*((6.43*4.437)/0.044)/(3.437+((Sp/0.044)^4.437));

%Introduzione vettore incognito dei parametri
a=zeros(4*n,1);
a1=zeros(n,1);
a2=zeros(n,1);
a3=zeros(n,1);
a4=zeros(n,1);
a=[a1
    a2
    a3
    a4];

%Espressione della matrice A
A=[0 0 Kp -Kp
    0 0 -Kp Kp
    1/Ep 0 0 0
    0 1/Ec 0 0];

%Rappresentazione della matrice K e vettore F
N=zeros(4,4*n);
Nder=zeros(4,4*n);
Nstar=zeros(4,1);
Nstarder=zeros(4,1);

nodi=zeros(n,1);
dist=l/(n-1);
for i=1:n-1
    q(i,1)=i;
end
nodi=[0;
    q*dist];

for passo=1:n
    x=nodi(passo,1);
    for i=1:n
        N1=sin((i*pi*x)/l);
        N2=sin((i*pi*x)/l);
        N3=sin((i*pi*x)/l);
        N4=sin((i*pi*x)/l);

        N1der=cos(i*pi*x/l);
        N2der=cos(i*pi*x/l);
        N3der=cos(i*pi*x/l);
        N4der=cos(i*pi*x/l);

        N(1,i)=N1;
        N(2,i+n)=N2;
        N(3,i+2*n)=N3;
        N(4,i+3*n)=N4;

        Nder(1,i)=N1der;
        Nder(2,i+n)=N2der;
        Nder(3,i+2*n)=N3der;
        Nder(4,i+3*n)=N4der;

        N(1,n-1)=100*x/l;
        Nder(1,n-1)=100/l;
        N(1,n)=(100*(1-x)/l);
        Nder(1,n)=-(100/l);

        N(2,2*n)=(100*(1-x)/l);
        Nder(2,2*n)=-(100/l);

        N(4,4*n)=100*x/l;
        Nder(4,4*n)=100/l;
    end
    k(:,passo)=Nder-(A*N);
    N5=x/l;
    N5der=1/l;
    Nstar(3,1)=N5;
    Nstarder(3,1)=N5der;
    f(:,passo)=A*Nstar-Nstarder;
end

K=zeros(4*n,4*n);
K=k(:,1);
F=zeros(4*n,1);
F=f(:,1);
for passo=2:n
    K=[K;k(:,passo)];
    F=[F;f(:,passo)];
end

%Risoluzione numerica
deltaL=0.0001;
Sp=deltaL;
Tau=Sp*((6.43*4.437)/0.044)/(3.437+((Sp/0.044)^4.437));
kp=Tau/Sp;
Kp=bp*Ap;

A=[0 0 Kp -Kp
    0 0 -Kp Kp
    1/Ep 0 0 0
    0 1/Ec 0 0];

%%INIZIO CICLO iterativo
RE=F;
iter=0;

```

```

FF0=deltaL*F;
while norm(RE)>(norm(FF0))/1000
iter=iter+1;
if iter==1
N= zeros (4,4*n);
Nder= zeros (4,4*n);
Nstar=zeros (4,1);
Nstarder=zeros (4,1);

for passo=1:n
x=nodi(passo,1);
for i=1:n
N1= sin((i*pi*x)/l);
N2= sin((i*pi*x)/l);
N3= sin((i*pi*x)/l);
N4= sin((i*pi*x)/l);

N1der=cos(i*pi*x/l);
N2der=cos(i*pi*x/l);
N3der=cos(i*pi*x/l);
N4der=cos(i*pi*x/l);

N(1,i)=N1;
N(2,i+n)=N2;
N(3,i+2*n)=N3;
N(4,i+3*n)=N4;

Nder(1,i)=N1der;
Nder(2,i+n)=N2der;
Nder(3,i+2*n)=N3der;
Nder(4,i+3*n)=N4der;

N(1,n-1)=100*x/l;
Nder(1,n-1)=100/l;
N(1,n)=(100*(1-x)/l);
Nder(1,n)=- (100/l);

N(2,2*n)=(100*(1-x)/l);
Nder(2,2*n)=- (100/l);

N(4,4*n)=100*x/l;
Nder(4,4*n)=100/l;
end
k(:, :, passo)=Nder-(A(:, :, 1)*N);
N5= x/l;
N5der=1/l;
Nstar(3,1)=N5;
Nstarder(3,1)=N5der;
f(:, :, passo)=A(:, :, 1)*Nstar-Nstarder;
end
K= zeros (4*n,4*n);
K=k(:, :, 1);
F=zeros (4*n,1);
F=f(:, :, 1);
for passo=2:n
K=[K;k(:, :, passo)];
F=[F;f(:, :, passo)];
end
FF0=deltaL*F;
G0=FF0;
[L,U,P]=lu(K);
y=L\ (P*FF0);
a(:, :, iter)=U\y;
elseif iter>1
FFF=FF0-FF(:, :, iter-1);
[L,U,P]=lu(K);
y=L\ (P*FFF);
a(:, :, iter)=U\y;

```

```

end
aa=zeros(4*n,1);
for j=1:iter
aa=aa+a(:, :, iter);
end
cont=1;
gg(:, cont)=aa;
a3=aa(2*n+1:3*n,1);
a4=aa(3*n+1:4*n,1);

for passo=1:n
x=nodi(passo,1);
M3=zeros(n,1);
M4=zeros(n,1);
for i=1:n
m3= sin((i*pi*x)/l);
m4= sin((i*pi*x)/l);
M3(i,1)=m3;
M4(i,1)=m4;
M4(n,1)=100*x/l;
end
Up(passo,1)=M3*a3+x/l;
Uc(passo,1)=M4*a4;
end
sp=(Up-Uc);
for i=1:n
Sp=sp(i,1);
if Sp<0.0001
kp(i,1)=188.65;
elseif Sp>0.0001
Tau=Sp*((6.43*4.437)/0.044)/(3.437+((Sp/0.044)^4.437));
kp(i,1)=Tau/Sp;
end
Kp(i,1)=kp(i,1)*bp/AP;
A(:, :, i)= [0 0 Kp(i,1) -Kp(i,1)
0 0 -Kp(i,1) Kp(i,1)
1/Ep 0 0 0
0 1/Ec 0 0];
end

for passo=1:n
x=nodi(passo,1);
for i=1:n
N1= sin((i*pi*x)/l);
N2= sin((i*pi*x)/l);
N3= sin((i*pi*x)/l);
N4= sin((i*pi*x)/l);

N1der=cos(i*pi*x/l);
N2der=cos(i*pi*x/l);
N3der=cos(i*pi*x/l);
N4der=cos(i*pi*x/l);

N(1,i)=N1;
N(2,i+n)=N2;
N(3,i+2*n)=N3;
N(4,i+3*n)=N4;

Nder(1,i)=N1der;
Nder(2,i+n)=N2der;
Nder(3,i+2*n)=N3der;
Nder(4,i+3*n)=N4der;

N(1,n-1)=100*x/l;
Nder(1,n-1)=100/l;
N(1,n)=(100*(1-x)/l);
Nder(1,n)=- (100/l);

```

```

N(2,2*n)=(100*(1-x)/l);
Nder(2,2*n)=- (100/l);

N(4,4*n)=100*x/l;
Nder(4,4*n)=100/l;
end
k(:, :, passo)=Nder-(A(:, :, passo)*N);
N5= x/l;
N5der=1/l;
Nstar(3,1)=N5;
Nstarder(3,1)=N5der;
f(:, :, passo)=A(:, :, passo)*Nstar-Nstarder;
end
K= zeros (4*n,4*n);
K=k(:, :, 1);
F=zeros (4*n,1);
F=f(:, :, 1);
for passo=2:n
    K=[K;k(:, :, passo)];
    F=[F;f(:, :, passo)];
end

FF(:, :, iter)=deltaL *F;
RE=FF0-FF(:, :, iter);
end

%INIZIO CICLO INCREMENTALE
for deltaL=0.0101:0.005:1.0001
    FF0=(deltaL/0.0001)*G0;
    RE=FF0;
    iter=0;
    cont=1+cont;
    aa=0;

    while norm(RE)>(norm(FF0))/1000
        iter=iter+1;
        if iter==1
            [L,U,P]=lu(K);
            y=L\u(P*FF0);
            a(:, :, iter)=U\u;
        elseif iter>1
            FFF=FF0-FF(:, :, iter-1);
            [L,U,P]=lu(K);
            y=L\u(P*FFF);
            a(:, :, iter)=U\u;
        end

        for j=1:iter
            aa=aa+a(:, :, iter);
        end
        gg(:, cont)=aa;
        a3=aa(2*n+1:3*n,1);
        a4=aa(3*n+1:4*n,1);

        for passo=1:n
            x=nodi(passo,1);
            M3=zeros(n,1);
            M4=zeros(n,1);
            for i=1:n
                m3= sin((i*pi*x)/l);
                m4= sin((i*pi*x)/l);
                M3(i,1)=m3;
                M4(i,1)=m4;
                M4(n,1)=100*x/l;
            end
            Up(passo,1)=M3*a3+x/l;
            Uc(passo,1)=M4*a4;
        end
end

```

```

sp=(Up-Uc);
for i=1:n
    Sp=sp(i,1);
    if Sp<0.0001
        kp(i,1)=188.65;
    elseif Sp>0.0001
        Tau=Sp*((6.43*4.437)/0.044)/(3.437+((Sp/0.044)^4.437));
        kp(i,1)=Tau/Sp;
    end
    Kp(i,1)=kp(i,1)*bp/AP;
    A(:, :, i)= [0 0 Kp(i,1) -Kp(i,1)
                0 0 -Kp(i,1) Kp(i,1)
                1/Ep 0 0 0
                0 1/Ec 0 0];
end

for passo=1:n
    x=nodi(passo,1);
    for i=1:n
        N1= sin((i*pi*x)/l);
        N2= sin((i*pi*x)/l);
        N3= sin((i*pi*x)/l);
        N4= sin((i*pi*x)/l);

        N1der=cos(i*pi*x/l);
        N2der=cos(i*pi*x/l);
        N3der=cos(i*pi*x/l);
        N4der=cos(i*pi*x/l);

        N(1,i)=N1;
        N(2,i+n)=N2;
        N(3,i+2*n)=N3;
        N(4,i+3*n)=N4;

        Nder(1,i)=N1der;
        Nder(2,i+n)=N2der;
        Nder(3,i+2*n)=N3der;
        Nder(4,i+3*n)=N4der;

        N(1,n-1)=100*x/l;
        Nder(1,n-1)=100/l;
        N(1,n)=(100*(1-x)/l);
        Nder(1,n)=- (100/l);

        N(2,2*n)=(100*(1-x)/l);
        Nder(2,2*n)=- (100/l);

        N(4,4*n)=100*x/l;
        Nder(4,4*n)=100/l;
    end
    k(:, :, passo)=Nder-(A(:, :, passo)*N);
    N5= x/l;
    N5der=1/l;
    Nstar(3,1)=N5;
    Nstarder(3,1)=N5der;
    f(:, :, passo)=A(:, :, passo)*Nstar-Nstarder;
end
K= zeros (4*n,4*n);
K=k(:, :, 1);
F=zeros (4*n,1);
F=f(:, :, 1);
for passo=2:n
    K=[K;k(:, :, passo)];
    F=[F;f(:, :, passo)];
end

FF(:, :, iter)=deltaL *F;
RE=FF0-FF(:, :, iter);
end

```

```
end
end

%creazione vettore sigma
NN1=zeros(1,4*n);
NN1(1,n-1)=100;
sigma=NN1*gg;
```

Diagramma FORZA-SPOSTAMENTO

