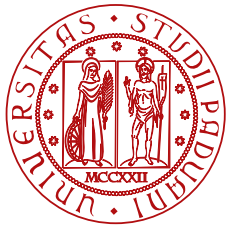


Università degli Studi di Padova

DIPARTIMENTO DI INGEGNERIA INDUSTRIALE

Tesi Magistrale in Ingegneria Aerospaziale

Danno e propagazione delle cricche in un pannello di float glass: simulazione peridinamica con un codice di ricerca

Candidato

Mattia Frighetto

Matricola N. 2097311

Relatore

Prof. Ugo Galvanetto

Università Degli Studi di Padova

Co-relatore

Prof. Mirco Zaccariotto

Università Degli Studi di Padova

Anno Accademico

2023/2024

A mio padre

Sommario

Il float glass è un materiale molto utilizzato in numerosi settori dell'industria. Prevederne il comportamento meccanico e i pattern di frattura è di fondamentale importanza per gli sviluppi futuri del suo utilizzo. Dopo una breve introduzione sulla teoria peridinamica e alcuni brevi cenni alla sua implementazione numerica, si entra nello specifico del codice di ricerca utilizzato e del suo funzionamento. Successivamente si eseguono una serie di simulazioni per verificare l'efficienza del codice e si tenta di paragonare i risultati ottenuti con quelli di un modello analogo pubblicato in letteratura. L'implementazione specifica di alcune condizioni che ha richiesto modifiche al codice, infine, viene riportata nelle appendici a fondo tesi.

Indice

Lista delle Figure

Lista delle Tabelle

Lista dei codici

1

Introduzione

Il float glass, letteralmente "vetro galleggiante", è un materiale che ha una vasta gamma di utilizzi, dall'industria automitive all'ingegneria civile e persino nei pannelli fotovoltaici. Le diverse strutture che possono essere realizzate facendo uso di tale materiale sono solite essere soggette a carichi quasi-statici (pressione del vento, peso della neve) carichi dinamici (grandine) e ambienti termici non stazionari. Per questo motivo le deformazioni, danni e invecchiamento del materiale devono essere costantemente monitorati e studiati con la massima attenzione possibile.
Ci sono diversi approcci disponibili per l'analisi strutturale di componenti in vetro, che siano monolitici o laminari. In generale la resistenza di un vetro può essere stimata in termini di probabilità di rottura per un determianto *maximum equivalent stress*, come per esempio lo stress principale massimo. La meccanica del continuo può essere applicata per predire zone critiche in cui può avere origine una frattura. Tuttavia per l'analisi di resistenza del vetro non è sufficiente stimare l'origine della frattura, infatti si deve considerare anche il processo di diffusione e crescita delle cricche così

come l'interazione tra le stesse. Dopo che nel vetro nascono le cricche e avviene una frammentazione, infatti, si necessita che il materiale supporti ancora dei carichi meccanici fornendo una buona robustezza anche nel regime di post-frattura. La meccanica del continuo con la sua classica formulazione locale non risulta più affidabile per studiare questo regime.
Il problema di fondo della meccanica del continuo classica, infatti, è che la definizione degli sforzi dipende da delle quantità che sono derivate nello spazio, come per esempio il tensore delle deformazioni, che non sono definite nel caso in cui il campo presenti delle discontinuità come quelle dovute alla nascita delle fratture nel materiale. La *peridinamica* è una teoria meccanica del continuo alternativa a quella classica, basata sull'ipotesi che i punti dei corpi deformabli interagiscono tra di loro attraverso forze non locali, che non dipendono dunque da delle derivate nello spazio. Questa teoria permette di modellare con precisione la frattura senza dover conoscere in anticipo il percorso e il punto di partenza della stessa. La crescita della frattura risulta meno sensibile alla direzione della mesh rispetto ai metodi basati sugli elementi finiti e ciò la rende dunque più adatta per lo studio del danneggiamento delle strutture.
La necessità di tenere conto di interazioni non locali nei problemi legati a fenomeni di frattura deriva da delle considerazioni fisiche: la forza di coesione nei materiali infatti è frutto di interazioni a distanza tra le molecole che compongono gli stessi ed è stato dimostrato che nei solidi l'influenza di queste interazoni si propaga a distanze maggiori di quelle della sola molecola più vicina. E' inoltre noto che le caratteristiche di frattura dei materiali sono fortemente influenzate dalle inclusioni di grani e da imperfezioni nel materiale, dalle loro dimensioni e dalla loro densità. La forma-

zione della frattura stessa è dunque influenzata dalla grandezza, dalla distanza e dalla interazione relativa tra le varie inclusioni e imperfezioni.
Le possibilità della teoria del continuo peridinamica sono innumerevoli, è infatti possibile formulare qualsiasi problema di diffusione retto da equazioni alle derivate parziali nello spazio, come la diffusione del calore.
Questa tesi si concentrerà sulla simulazione della propagazione delle cricche in un pannello di float glass, le cui caratteristiche fisiche e meccaniche saranno riportate più avanti. Lo scopo ultimo è quello di riprodurre i risultati di [7], facendo uso di un codice sviluppato dai ricercatori dell'Università degli Studi di Padova, diverso da quello usato in [7].

1.1 Introduzione alla teoria peridinamica

Di seguito si intende riportare una breve base teorica della teoria della peridinamica, facendo riferimento alla letteratura esistente che è stato possibile consultare. In particolare per mantenere coerenza di linguaggio in questo capitolo, si farà riferimento principalmente a [2]. Altre fonti di consultazione sono [1], [9], [3], [6].

1.1.1 Equazioni di bilancio

Si mostrano in questa sezione le equazioni di bilancio per la teoria peridinamica e si confrontano con quelle per la teoria classica.
Dato un corpo deformabile di dominio $\mathcal{B}$ e volume $\mathcal{V}_{\mathcal{B}}$, si definisce un campo vettoriale $\mathbf{x}$ che indica la generica posizione di

riferimento di un punto appartenente al corpo. Si definisce poi un campo vettoriale che indica il campo di spostamenti nel tempo di questi punti $u(\mathbf{x}, t)$. Si definisce infine un campo delle posizioni in configurazione deformata di tali punti come:

$$\mathbf{y}(\mathbf{x}, t) = \mathbf{x} + \mathbf{u}(\mathbf{x}, t)$$

Per ogni campo $\mathbf{x}$ che appartiene al dominio $\mathcal{B}$, sia nel caso di teorie locali che non-locali vale il seguente bilancio (in assenza di forze esterne):

$$\rho_0(\mathbf{x})\ddot{\mathbf{y}}(\mathbf{x}, t) = \mathcal{L}[\mathbf{u}(\mathbf{x}, t), t]$$

dove:

- $\rho_\mathbf{0}$ è la densità di riferimento
- $\mathcal{L}$ è la forza per unità di volume di riferimento dovuta all'interazione con gli altri punti del dominio.

La differenza tra teoria locale e quella non-locale sta nella definizione del termine $\mathcal{L}$. Per la teoria classica infatti si ha che:

$$\mathcal{L}_L[\mathbf{u}(\mathbf{x}, t), t] = \nabla \cdot (\mathbf{P}[\nabla \mathbf{u}(\mathbf{x}, t)]) \tag{1.1}$$

dove $\mathbf{P}$ è il primo tensore di Piola-Kirchhoff della teoria classica. E' nuovamente importante notare che nella teoria locale $\mathcal{L}$ dipende dalle derivate dello spostamento, che nel caso di presenza di discontinuità nel campo degli spostamenti non sono definite.
Per la teoria non locale invece si ha:

$$\mathcal{L}_{NL}[\mathbf{u}(\mathbf{x}, t), t] = \int_{\mathcal{B}} \mathbf{f}(\mathbf{x}, \mathbf{q}, t) d\mathcal{V}_q \tag{1.2}$$

dove $\mathbf{f}(\mathbf{x},\mathbf{q})$ è la cosiddetta *pairwise bond force density function*: immaginiamo $\mathbf{x}$ e $\mathbf{q}$ essere due punti distinti nel dominio $\mathcal{B}$ e dV_x e dV_q dei volumetti che contengono tali punti. La pairwise bond

force è la forza che il materiale in dV_q esercita su dV_x ed è indicata come $\mathbf{f}(\mathbf{x}, \mathbf{q})dV_x dV_q$. Le unità di misura di tale funzione $\mathbf{f}$ sono dunque $Forza/Volume^2$. Il valore di tale funzione è determinato dalla deformazione e dalle proprietà del materiale attraverso un modello costitutivo, che verrà successivamente discusso. La funzione $\mathbf{f}$ in realtà dipende anche dal tempo in problemi dinamici, ma per abbreviare la notazione talvolta questa dipendenza verrà omessa. I due punti sono immaginati come uniti tra di loro da un vettore.
Nella teoria peridinamica questi vettori vengono chiamati *bonds*. Possiamo definire di seguito delle caratteristiche principali che contraddistinguono la teoria peridinamica, che ricordiamo essere una teoria non-locale del continuo.
L'integrale in equazione 1.2 viene calcolato solamente su un dominio sferico $\mathcal{H}_x$ che viene chiamato intorno di $\mathbf{x}$, ed ha la caratteristica di essere centrato in $\mathbf{x}$. Il raggio δ di $\mathcal{H}_x$ è chiamato orizzonte. Il nome orizzonte deriva dal fatto che il punto $\mathbf{x}$ non può "vedere" i punti $\mathbf{q}$ che stanno oltre la distanza δ. I punti che sono separati da una distanza maggiore di quella dell'orizzonte dunque non interagiscono tra loro.

$$\mathcal{H}_x = \{\mathbf{q} \in \mathcal{B} : 0 < |\mathbf{q} - \mathbf{x}| \leq \delta, \delta \in \mathbb{R}\}$$
$$\mathcal{L}_{PD}[\mathbf{u}(\mathbf{x}, t), t] = \int_{\mathcal{H}_x(x)} \mathbf{f}(\mathbf{x}, \mathbf{q}) d\mathcal{V}_q$$

I volumetti che stanno all'interno dell'orizzonte sono chiamati *famiglia* di $\mathbf{x}$.
La pairwise bond force density function nell'ambito della teoria peridinamica gode di una proprietà di simmetria, cioè:

$$\mathbf{f}(\mathbf{x}, \mathbf{q}) = -\mathbf{f}(\mathbf{q}, \mathbf{x}) \quad \forall \mathbf{x}, \mathbf{q}$$

Tale proprietà assicura la conservazione del momento.
Consideriamo infine un volumetto dV_x in equilibrio. Ipotizziamo l'esistenza di un campo $\mathbf{b}(\mathbf{x})$ che chiamiamo *body force density* tale che la forza esterna netta su tale volumetto sia data dal prodotto $\mathbf{b}(\mathbf{x})dV_x$. Se consideriamo che tale volumetto interagisce con tutte le altre particelle di materiale nell'orizzonte la forza totale agente su tale volumetto potrà essere indicata come:

$$\int_{\mathcal{H}_x} [\mathbf{f}(\mathbf{q}, \mathbf{x})dV_x]dV_q + \mathbf{b}(\mathbf{x})dV_x = 0$$

Da qui deriva semplicemente l'equazione di equilibrio peridinamico:

$$\int_{\mathcal{H}_x} \mathbf{f}(\mathbf{q}, \mathbf{x})dV_q + \mathbf{b}(\mathbf{x}) = 0 \tag{1.3}$$

che è valida per ogni punto $\mathbf{x}$ nel dominio.
Nel caso dinamico l'equazione di bilancio della traslazione nella teoria peridinamica diventa invece:

$$\rho(\mathbf{x})\ddot{\mathbf{y}}(\mathbf{x}, t) = \int_{\mathcal{H}_x} \mathbf{f}(\mathbf{q}, \mathbf{x}, t)dV_q + \mathbf{b}(\mathbf{x}, t) \tag{1.4}$$

dove ρ è il campo di densità, mentre $\ddot{\mathbf{y}}$ è il campo di accelerazioni.

1.1.2 Peridinamica bond-based

Come già accennato, il valore della pairwise bond force density function $\mathbf{f}(\mathbf{q},\mathbf{x})$ dipende dalla deformazione e dalle proprietà del materiale attraverso un modello costitutivo. Il più semplice modello peridinamico per il materiale è quello chiamato *bond-based*, nel quale un punto $\mathbf{x}$ e un punto $\mathbf{q}$ scambiano tra di loro forze

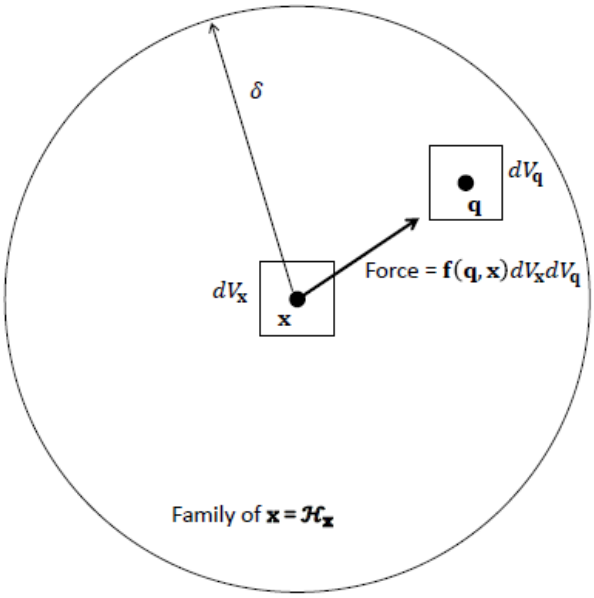


Figura 1.1: Definizione dell'orizzonte peridinamico

interagendo come fossero delle "molle".

$$\mathbf{f}(\mathbf{q},\mathbf{x}) = f(|\mathbf{y}(\mathbf{q}) - \mathbf{y}(\mathbf{x})|, \mathbf{q}, \mathbf{x})\mathbf{M}, \qquad \mathbf{M} = \frac{\mathbf{y}(\mathbf{q}) - \mathbf{y}(\mathbf{q})}{|\mathbf{y}(\mathbf{q}) - \mathbf{y}\mathbf{x}|} \tag{1.5}$$

dove f è una funzione generica, $\mathbf{y}(\mathbf{x})$ è la posizione deformata di $\mathbf{x}$, $\mathbf{M}$ è un vettore unitario nella direzione del legame deformato tra $\mathbf{x}$ e $\mathbf{q}$.
La caratteristica che distingue la teoria bond-based è quella che ogni bond force density function $\mathbf{f}(\mathbf{q},\mathbf{x})$ è indipendente da ogni altro bond connesso con qualsiasi altro punto. Un tipico materiale con formulazione bond-based è dato da:

$$\mathbf{f(q,x)} = \begin{cases} Ce\mathbf{M}, & se \quad |\mathbf{q} - \mathbf{x}| \leq \delta \\ 0 & altrimenti \end{cases} \tag{1.6}$$

Dove C è una costante chiamata *micromodulo*, $\mathbf{M}$ è il vettore unitario nella direzione del legame deformato, definito come in equazione

(1.5), mentre e è il *bond extension*, definito da:

$$e = |\mathbf{y}(\mathbf{q}) - \mathbf{y}(\mathbf{x})| - |\mathbf{q} - \mathbf{x}|$$

In maniera più generale, il micromodulo C può dipendere da entrambi gli estremi del legame e viene scritto come $C(\mathbf{x}, \mathbf{q})$. Il modello (1.6) come già accennato tratta ogni legame come una molla lineare elastica quindi anche le proprietà di bulk del materiale sono elastiche. In questo modo è possibile calibrare C in termini di proprietà che sono misurabili, per rendere la teoria peridinamica coerente con quella classica. Il micromodulo è direttamente legato alla densità di energia ($Energia/Volume^2$) di ogni legame, che è chiamata *micropotenziale* e viene indicato con w. Nell'esempio sopra riportato si può indicare dunque il micropotenziale come:

$$w = \frac{Ce^2}{2} \tag{1.7}$$

Il micropotenziale è infine legato alla bond force density $\mathbf{f}$ da:

$$f = \frac{\partial w}{\partial e} \quad o \quad \mathbf{f} = \frac{\partial w}{\partial \mathbf{Y}}$$

dove $\mathbf{Y} = \mathbf{y}(\mathbf{q}) - \mathbf{y}(\mathbf{x})$.
Il processo di calibrazione viene spiegato in maniera semplificata, per non allontanarsi troppo dal puro scopo di introduzione della teoria. Prendiamo un caso semplificato, dove C è costante. Si suppone di conoscere il modulo di bulk del materiale. Si consideri un punto arbitrario $\mathbf{x}$ interno al corpo: possiamo descrivere la deformazione come

$$\mathbf{y(x)} = (1 + \epsilon)\mathbf{x}, \quad \epsilon \ costante$$

La densità di energia di deformazione W in $\mathbf{x}$ ($Energia/Volume$) è legata al micropotenziale da:

$$W = \frac{1}{2}\int_{\mathcal{H}_x} w dV_q \tag{1.8}$$

Dove il termine $\frac{1}{2}$ deriva dal fatto che ciascun estremo del legame possiede solamente la metà dell'energia posseduta dal legame stesso. Definiamo dunque la distanza tra due punti $\mathbf{q}$ e $\mathbf{x}$ come $\xi = |\mathbf{q} - \mathbf{x}|$. La bond extension, cioè la deformazione del legame, è data da $e = \epsilon\xi$. Combinando 1.7 e 1.8 e otteniamo nel caso 3D:

$$W = \frac{1}{2}\int_{\mathcal{H}_x} \frac{Ce^2}{2} dV_\xi = \frac{\pi C \epsilon^2 \delta^5}{5}$$

Per completare il processo di calibrazione del modello peridinamico in termini di proprietà elastiche (modulo di bulk) dobbiamo porre la densità di energia di deformazione uguale a quella della teoria classica dell'elasticità. In questo modo si avrà che la teoria classica coincide con quella peridinamica. In questo caso dopo aver posto l'uguaglianza si otterrà:

$$C = \frac{45k}{2\pi\delta^5}$$

E' possibile utilizzare modelli di materiale più complessi, fintanto che il problema è ridotto ad un unica incognita, trovando il valore per C che crea la corrispondenza tra le teorie.
La scelta di

$$C(\mathbf{q},\mathbf{x}) = \frac{c}{\xi}$$

dove c è una costante detta *spring constant*, corrisponde per esempio ad una bond force proporzionale al *bond strain*, denotato dalla lettera s. In questo modo si otterrà che la force density e il bond

strain sono dati rispettivamente da:

$$f = cs, \quad s = \frac{e}{\xi} \tag{1.9}$$

E il processo di calibrazione si fa in questo caso rispetto alla costante c:

$$c = \frac{18k}{\pi\delta^4} \tag{1.10}$$

Quest'ultimo materiale, il cui modello è descritto da (1.9) e (1.10) è chiamato *prototype microelastic* (PM) material ed è il modello di materiale più utilizzato in assoluto nell'ambito della peridinamica bond based.

1.1.3 Stati peridinamici

I modelli che descrivono il materiale come quello riportato in (1.6) portano con loro l'assunzione che ogni singolo bond connesso ad un punto **x** risponde in maniera indipendente da tutti gli altri, non considerando dunque le interazioni reciproche che vi possono essere. Questo va a limitare i tipi di risposta del materiale che possono essere riprodotti, indipendentemente dalla funzione **f**. Nello specifico, la risposta elastica alle deformazioni è limitata al caso in cui il modulo di Poisson è $\nu = 0.25$.
Per ottenere una risposta più generica del materiale, dunque, è necessario introdurre delle interdipendenze tra i differenti bond, che vanno oltre le semplici interazioni tra la sola coppia di punti. A tale scopo sono stati introdotti i cosiddetti *stati peridinamici*. Definito $\mathcal{H}_x$ il dominio sferico di raggio δ centrato in **x** ed escluso **x**, uno stato peridinamico è una funzione definita su tutti i bond della famiglia del punto **x** stesso. Il concetto che si vuole esprimere è quello che la force density in un dato legame tra due punti **x** e **q**

dipende non solo dalla deformazione di quel dato legame, ma dalla deformazione di tutti gli altri legami connessi a $\mathbf{x}$ e $\mathbf{q}$. Scriviamo duque:

$$\mathbf{f}(\mathbf{q}, \mathbf{x}) = \mathbf{t}(\mathbf{q}, \mathbf{x}) - \mathbf{t}(\mathbf{x}, \mathbf{q})$$

Dove $\mathbf{t}$ sono i cosiddetti *bond force density vectors* e sono esprimibili come

$$\mathbf{t}(\mathbf{q}, \mathbf{x}) = \underline{\mathbf{T}}[\mathbf{x}]\langle \mathbf{q} - \mathbf{x} \rangle, \quad \mathbf{t}(\mathbf{x}, \mathbf{q}) = \underline{\mathbf{T}}[\mathbf{q}]\langle \mathbf{x} - \mathbf{q} \rangle,$$

Dove con $\underline{\mathbf{T}}$[x] e $\underline{\mathbf{T}}$[q] indichiamo rispettivamente i *force states* in $\mathbf{x}$ e $\mathbf{q}$. Il loro significato è che lo stato $\underline{\mathbf{T}}[\mathbf{x}]$ per esempio, associa ad ogni legame $\mathbf{q}$-$\mathbf{x}$ nella famiglia di $\mathbf{x}$ un qualche bond force density vector $\mathbf{t}$. Per convenzione si indica un legame sul quale uno stato sta "operando" indicandolo tra parentesi angolari "$\langle \cdot \rangle$". Per comprendere meglio la nomenclatura, affermiamo che se un generico $\underline{\mathbf{A}}\langle \xi \rangle$ è un vettore, allora lo stato peridinamico $\underline{\mathbf{A}}$ è detto vettoriale. Esistono dunque stati tensoriali, vettoriali, scalari. Per comprendere meglio possiamo immaginare un force state come un'analogia allo stress tensor in un punto nella teoria classica. L'equazione del bilancio alla traslazione (1.4) facendo uso del formalismo degli stati peridinamici, diventa dunque:

$$\rho(\mathbf{x})\ddot{\mathbf{y}}(\mathbf{x}, t) = \int_{\mathcal{H}_x} (\underline{\mathbf{T}}[\mathbf{x}, t]\langle \xi \rangle - \underline{\mathbf{T}}[\mathbf{q}, t]\langle -\xi \rangle) dV_q + \mathbf{b}(\mathbf{x}, t) \qquad (1.11)$$

Nella peridinamica state-based un modello di materiale associa ad un punto $\mathbf{x}$ la deformazione collettiva di tutta la sua famiglia di punti. Per esprimere questa deformazione si definisce il *deformation state*.

$$\underline{\mathbf{Y}}[\mathbf{x}]\langle \mathbf{q}\text{-}\mathbf{x} \rangle = \mathbf{y}(\mathbf{q}) - \mathbf{y}(\mathbf{x})$$

Quindi il significato dello stato $\underline{\mathbf{Y}}$ è quello di "mappare" ogni bond nella sua conformazione deformata. Introducendo questa notazione facendo uso degli stati, dunque, è possibile esprimere un modello di materiale come:

$$\underline{\mathbf{T}} = \underline{\hat{T}}(\underline{\mathbf{Y}})$$

In questo caso dunque il modello del materiale è la funzione $\underline{\hat{T}}$, che è una funzione di uno stato, valutata in uno stato. Ricapitolando abbiamo dunque ottenuto che, con la teoria state based, la bond force density dipende da uno stato di deformazione che è esso stesso una funzione, piuttosto che da una semplice deformazione di un determinato legame.
Per modellare un materiale elastico si fa uso come per il caso bond-based della densità di energia di deformazione, che in questo caso dipende dalla deformazione della famiglia di $\mathbf{x}$:

$$W(\mathbf{x}) = \hat{W}(\underline{\mathbf{Y}}[\mathbf{x}])$$

Dove in questo caso $\hat{W}$ è una funzione di uno stato valutata in uno scalare. Per legare questa funzione ad un force state si considera una variazione infinitesima $d\hat{W}$ risultante da una variazione infinitesima di uno stato di deformazione $d\underline{\mathbf{Y}}$. La variazione di densità di energia deve essere uguale al lavoro compiuto dalle forze di legame per deformarsi, quindi:

$$d\hat{W} = \underline{T} \bullet d\underline{\mathbf{Y}}$$

Il prodotto interno tra due stati peridinamici è definibile come

$$\underline{\mathbf{A}} \bullet \underline{\mathbf{B}} = \int_{\mathcal{H}_x} \underline{\mathbf{A}}\langle\xi\rangle \cdot \underline{\mathbf{B}}\langle\xi\rangle dV_\xi$$

E' utile definire anche la derivata di Fréchet: se $\Psi(\underline{\mathbf{A}})$ è una funzione di uno stato, allora essa ha una derivata di Fréchet $\Psi_{\underline{\mathbf{A}}}(\underline{\mathbf{A}})$ se per ogni incremento infinitesimo $d\underline{\mathbf{A}}$

$$\Psi(\underline{\mathbf{A}} + d\underline{\mathbf{A}}) = \Psi(\underline{\mathbf{A}}) + \Psi_{\underline{\mathbf{A}}}(\underline{\mathbf{A}}) \bullet d\underline{\mathbf{A}}$$

Tornando alla densità di energia di deformazione, in un materiale elastico, il force state è la derivata di Fréchet della densità di energia di deformazione:

$$\underline{\hat{T}} = \hat{W}_{\underline{\mathbf{Y}}}$$

L'energia totale di deformazione, dunque, dato un corpo $\mathcal{B}$ è data da:

$$E = \int_{\mathcal{B}} W(\mathbf{x}) dV_x = \int_{\mathcal{B}} \hat{W}(\underline{\mathbf{Y}}[\mathbf{x}]) dV_x$$

Quest'ultima equazione ci dice che l'energia totale immagazinata nel corpo è distribuita tra tutti i punti del corpo, proporzionalmente alla deformazione della famiglia di ogni punto. E' importante infine notare che nella trattazione peridinamica state-based le due parti della pairwise bond force $\mathbf{f}(\mathbf{q},\mathbf{x})$ in $\mathbf{q}$ e $\mathbf{x}$ contribuiscono separatamente tra loro alla variazione di densità di energia di deformazione in questi due punti, anche se la densità netta di forza è uguale tra i tali punti. Questo concetto è molto differente rispetto a quello della teoria bond-based, nella quale l'energia nel legame tra $\mathbf{q}$ e $\mathbf{x}$ può essere vista come immagazinata in una molla che li connette. Un modello di materiale bond-based può essere pensato come un caso particolare del modello state-based, definendo:

$$\hat{W}(\underline{\mathbf{Y}}) = \frac{1}{2} \int_{\mathcal{H}_x} w(|\underline{\mathbf{Y}}\langle \mathbf{q}\text{-}\mathbf{x} \rangle|, \mathbf{q}, \mathbf{x}) dV_q$$

Dove con w si intende il micropotenziale del modello bond-based. Un'ultima condizione che viene posta è il fatto che in ogni force state il momento netto rispetto ad $\mathbf{x}$ esercitato dalle forze di legame è nullo. Questo requisito è detto *non-polarità* e si traduce come:

$$\int_{\mathcal{H}} \underline{\mathbf{T}}\langle \xi \rangle \times \underline{\mathbf{Y}}\langle \xi \rangle dV_{\xi} = 0 \tag{1.12}$$

Questa condizione assicura che in assenza di forze esterne il momento angolare è costante in un corpo. E' ora possibile introdurre la formulazione state-based della teoria peridinamica.

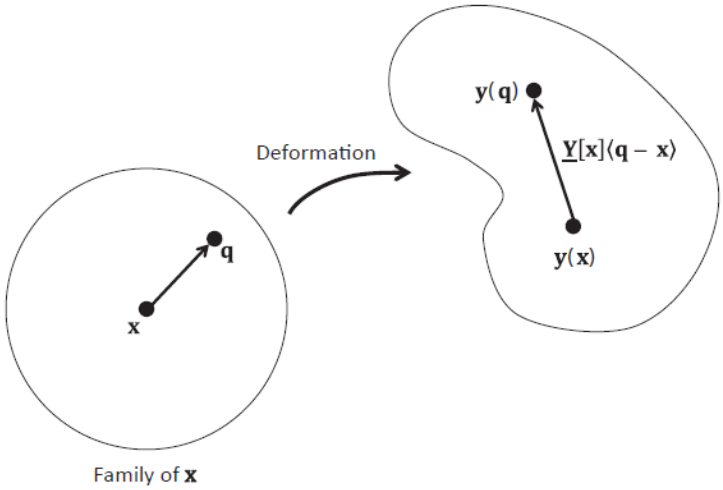


Figura 1.2: Illustrazione grafica del deformation state $\underline{\mathbf{Y}}$

1.1.4 Peridinamica state-based

I materiali state-based si possono dividere in due grandi categorie, *ordinari* e *non-ordinari*. In un modello di materiale ordinario, il bond force density vector $\underline{\mathbf{T}}\langle \xi \rangle$ è sempre parallelo al legame deformato identificato dal vettore $\underline{\mathbf{Y}}\langle \xi \rangle$ in cui con ξ si intende qualsiasi legame che sta nella famiglia. In un materiale non-ordinario questi vettori non sono necessariamente paralleli. I materiali ordinari

soddisfano automaticamente il requisito di non polarità presentato in (1.12) per definizione, per i materiali non-ordinari questa condizione invece va imposta tramite il modello di materiale e non è automaticamente soddisfatta.
Il pannello di float glass che viene utilizzato in [7] è costituito da un modello di materiale di tipo ordinario, pertanto non si ritiene necessario approfondire il modello di tipo non-ordinario.
Se $\underline{\mathbf{T}}$ è il force state in un materiale ordinario, siccome le forze di legame sono sempre parallele ai legami deformati, deve esistere uno stato $\underline{g}$ scalare tale che:

$$\underline{\mathbf{T}}\langle\xi\rangle = \underline{g}\langle\xi\rangle\underline{\mathbf{Y}}\langle\xi\rangle \quad \forall\xi \in \mathcal{H}$$

Se il materiale è anche elastico si ha che $\underline{\mathbf{T}} = \hat{W}_{\underline{\mathbf{Y}}}$. Dalla definizione della derivata di Fréchet dunque, per ogni variazione infinitesima di deformazione si ha:

$$d\hat{W} = \int_{\mathcal{H}} \underline{g}\langle\xi\rangle\underline{\mathbf{Y}}\langle\xi\rangle \cdot d\underline{\mathbf{Y}}\langle\xi\rangle dV_\xi = \frac{1}{2}\int_{\mathcal{H}} \underline{g}\langle\xi\rangle d|\underline{\mathbf{Y}}\langle\xi\rangle|^2 dV_\xi$$

Da cui segue che $\hat{W}$ dipende solo dalla lunghezza deformata dei legami, e

$$\underline{g} = 2\hat{W}_{|\underline{\mathbf{Y}}|^2}$$

Il significato di tale risultato è che in un modello materiale elastico ordinario si considerano solamente le variazioni di lunghezza dei legami e non le rotazioni degli stessi.
Un tipico esempio di modello di materiale elastico ordinario si può ottenere assumendo che la densità di energia di deformazione dipende dalla media pesata degli allungamenti dei legami. Si

definisce dunque l'*extension state*:

$$\underline{e}\langle\xi\rangle = |\underline{\mathbf{Y}}\langle\xi\rangle| - |\xi|$$

E possiamo, per ogni ξ appartenente alla famiglia, definire una dilatazione non-locale:

$$\theta = \frac{3}{m}\int_{\mathcal{H}} \underline{\omega}\langle\xi\rangle|\xi|\underline{e}\langle\xi\rangle dV_\xi, \quad m = \int_{\mathcal{H}} \underline{\omega}\langle\xi\rangle|\xi|^2 dV_\xi \tag{1.13}$$

dove $\underline{\omega}$ è chiamata *influence function* e m è uno scalare di normalizzazione. Il fattore 3 compare perchè per una piccola espansione isotropa $|\underline{e}\langle\xi\rangle| = \epsilon|\xi|$, la dilatazione non-locale coincide con la dilatazione locale, $\theta = 3\epsilon$. Se la densità di energia di deformazione dipende solamente dalla dilatazione non-locale, dopo una serie di calcoli, si ottiene che il force state:

$$\underline{\mathbf{T}}\langle\xi\rangle = \frac{-3p\underline{\omega}\langle\xi\rangle|\xi|}{m}\mathbf{M}, \quad \mathbf{M} = \frac{\underline{\mathbf{Y}}\langle\xi\rangle}{|\underline{\mathbf{Y}}\langle\xi\rangle|} \tag{1.14}$$

Dove con p si indica la pressione non-locale, che è data da:

$$p = \frac{\partial W}{\partial \theta}$$

Linear Peridynamic Solid (LPS)

Il pannello di float glass, sempre in accordo con [7], è modellato come un solido lineare peridinamico. Questa particolare formulazione per il materiale, detta anche LPS, è ottenibile a partire da (1.14), imponendo la pressione non locale uguale a $p = -k\theta$, dove k è il modulo di bulk del materiale. Questa sola imposizione porta alla formulazione del modello di fluido lineare peridinamico

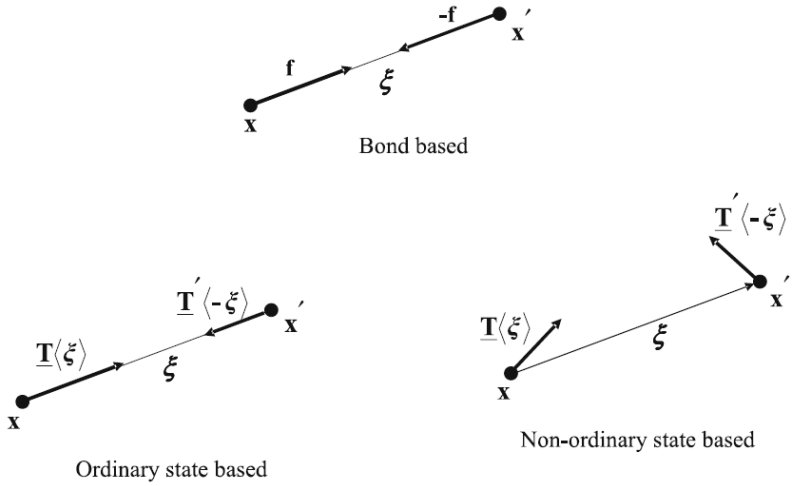


Figura 1.3: Differenze tra modello bond-based, state-based ordinario e state-based non ordinario

(LPF), che può essere facilmente esteso per modellare un solido. Per fare questo è necessario ricavare una sorta di versione peridinamica della shear strain, ottenibile sottraendo all'extension state la parte dilatazionale della deformazione. Si ottiene il cosiddetto *deviatoric extension state*:

$$\underline{e}^d\langle\xi\rangle = \underline{e}\langle\xi\rangle - \frac{\theta|\xi|}{3}$$

In questo modo il force state corrispondente al modello del solido lineare peridinamico è:

$$\underline{\mathbf{T}}\langle\xi\rangle = \left(\frac{3k\theta\underline{\omega}\langle\xi\rangle|\xi|}{m} + \alpha\underline{\omega}\langle\xi\rangle\underline{e}^d\langle\xi\rangle\right)\mathbf{M} \tag{1.15}$$

dove α è una costante che va calibrata in accordo con il modulo di taglio del materiale, in maniera simile a quanto visto per il modello bond-based:

$$\alpha = \frac{15G}{m}$$

La scelta dell'influence function, infine, sebbene arbitraria influenza la risposta non-locale del modello peridinamico state based. Facendo uso di metodi numerici tale influence function può andare a diminuire o aumentare la precisione per un modello.

1.1.5 Danno e frattura

Modelli bond based

Il modo più semplice per modellare il danno facendo uso della peridinamica è attraverso la rottura dei legami. Ciò significa che dopo che un qualche criterio di rottura viene soddisfatto, il bond non è più in grado di supportare una force density diversa da zero. Per applicare un criterio di rottura al materiale PM riportato in (1.9), si introduce una variabile storia-dipendente $\beta(t)$ per ogni legame, che varia in maniera irreversibile tra 0 e 1 quando il legame si rompe:

$$f = cs\beta, \quad \beta(\xi, t) = \begin{cases} 1 & se\ s(t') < s_* \ \forall t' \leq t \\ 0 & altrimenti \end{cases} \tag{1.16}$$

dove s_* è una costante chiamata *critical bond breakage strain* e il materiale risultante è chiamato *prototype microelastic brittle* (PMB) material.
In questo modo il danno ha origine quando la deformazione tensile in un qualche legame supera il valore critico s_*. Dopo la nucleazione del danno la force density immagazzinata in quel legame viene distribuita ai legami vicini, aumentando la probabilità che questi a loro volta si rompano facendo nascere un proceso di danneggiamento progressivo del materiale. La rottura dei legami e il loro percorso viene identificata come una cricca. E' importante notare,

come già detto, che non è conosciuta a priori nè la zona di nucleazione della cricca, nè la sua direzione e velocità di propagazione. E' utile definire il danno nel materiale, che è sostanzialmente la frazione di legami rotti connessi ad un punto materiale:

$$\varphi(\mathbf{x}, t) = \frac{\int_{\mathcal{H}}(1 - \beta(\xi, t))dV_\xi}{\int_{\mathcal{H}} dV_\xi} \tag{1.17}$$

Il valore di s_* infine può essere calibrato a partire dalla *critical energy release rate*, che è misurabile. Si parte calcolando il micropotenziale alla rottura:

$$w_* = \frac{f_* e_*}{2} = \frac{(cs_*)(\xi s_*)}{2} = \frac{c\xi s_*^2}{2}$$

Questo può essere visto come il lavoro su unità di volume^2 richiesto per la rottura del legame. E' possibile successivamente calcolare il lavoro per area unitaria di frattura, G_*:

$$G_* = \int_0^\delta \left\{ \int_0^{2\pi} \int_z^\delta \int_0^{\cos^{-1}(z/\xi)} (w_* \xi^2) \sin\phi d\phi d\xi d\theta \right\} dz = \frac{1}{2} c s_*^2 \left(\frac{\delta^5 \pi}{5} \right) \tag{1.18}$$

Questo integrale è stato derivato per la prima volta da Silling e Askari ([10]) e rappresenta la sommatoria dei lavori richiesti per terminare tutte le interazioni tra due punti rappresentati in figura (1.4) come x_{j-} e x_{k+} che stanno rispettivamente sotto e sopra la zona di frattura. L'integrazione avviene in coordinate sferiche (ξ, θ, ϕ), e rappresenta il volume di tutti i punti $x_(k+)$ al di sopra della superficie della cricca e che stanno all'interno dell'orizzonte di x_{j-}. L'integrale di linea infine include il contributo di tutti i punti x_{j-} tra 0 e l'orizzonte δ. Il valore risultate di G_* permette di ricavare il valore calibrato del critical bond breackage strain. Per esempio nel caso del materiale prototype microelastic (PM), sostituendo il

valore c dell'equazione (1.10) si ottiene un valore del s_*:

$$s_* = \sqrt{\frac{5G_*}{9k\delta}}$$

Dopo la rottura del legame l'energia w_* immagazzinata scompare, non viene convertita, ma rappresenta l'energia che viene consumata da qualche microcricca che nasce e non viene esplicitamente modellata.

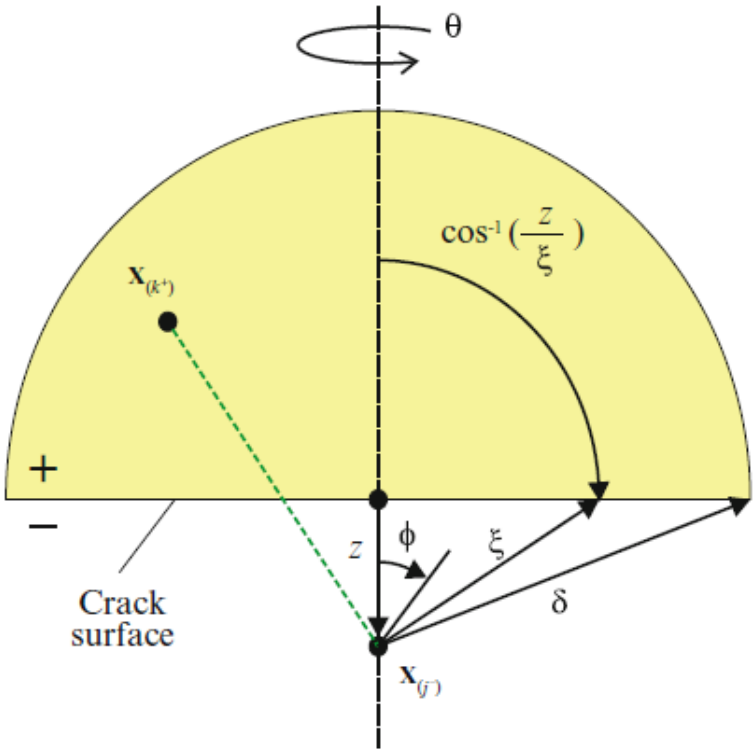


Figura 1.4: Valutazione del lavoro per area unitaria di frattura G_*

Una problematica della modellazione della rottura del legame è che dopo che avvieve la rottura nasce una sorta di effetto di ritorno elastico dovuto alle forze che esercitano i legami rimasti intatti che non sono più equilibrate dal legame che si rompe e nascono dunque delle accelerazioni che risultano in un aumento di energia cinetica del corpo, di solito nella forma di onde a ridotta l'unghez-

za d'onda. Queste onde possono influenzare la modellazione della propagazione delle cricche causando delle fluttuazioni di deformazione nei legami. E' quindi utile, per i modelli numerici, introdurre degli smorzamenti fittizi che riducono queste onde.
La modellazione del danno facendo uso del modello di materiale PMB nella realtà porta con se alcune criticità. Il calcolo di G_*, lavoro per area unitaria di frattura, assume implicitamente che ogni cricca è responsabile per tutti i legami rotti che connettevano due punti che si trovano ai due bordi opposti della cricca stessa. Questa assunzione non è più valida nel caso in cui c'è già un danno preesistente, per esempio derivante da un'altra cricca. In questo caso la nuova cricca dissiperebbe meno energia rispetto al valore G_* calcolato. Un modo per ovviare a questo problema è quello di tenere in considerazione il net damage (1.17). Nei pressi di una singola cricca infatti, il valore di net damage non supera lo 0.5 in quanto un punto ha comunque dei legami che non sono rotti e che stanno dallo stesso lato della cricca di tale punto. Se il net damage è dunque maggiore di 0.5 si aumenta il valore di critical bond breakage strain s_* per tenere conto di una rottura preesistente [2]. Una possibile relazione, per un legame che connette una coppia di punti $\mathbf{x}$ e $\mathbf{q}$ è:

$$s_* = s_{*0} R(\overline{\varphi}), \quad s_{*0} = \sqrt{5G_*/9k\delta}$$

dove

$$R(\overline{\varphi}) = \begin{cases} 1 & se\ 0 \leq \overline{\varphi} \leq 0.5 \\ 0.5/(1 - min(0.95, \overline{\varphi})) & se\ 0.5 \leq \overline{\varphi} \leq 1 \end{cases}$$

e dove $\overline{\varphi}$ è il net damage medio nel legame, definito come:

$$\overline{\varphi} = \frac{\varphi(\mathbf{x}, t) + \varphi(\mathbf{q}, t)}{2}$$

Modelli state based ordinari

I modelli state based ordinari fanno uso di una influence function $\underline{\omega}\langle\xi\rangle$ che controlla quanto ogni legame influisce sul force state, che come già detto a sua volta rappresenta la risposta del materiale alla deformazione collettiva della famiglia. Se teniamo in consideraizione un materiale come il LPS (1.15) l'approccio più semplice per implementare la rottura del legame è quello di imporre $\underline{\omega}\langle\xi\rangle = 0$ quando il legame ξ si rompe e dunque escludendo l'influenza del legame rotto sulla risposta del materiale.
Con riferimento alla modellazione del danno in un modello bond based, un critical bond breakage strain s_* per un modello state based può essere derivato dal critical energy release rate misurato per il dato materiale. Si può calcolare il lavoro per unità d'area di frattura G_* e ottenere per un caso 3D [6]:

$$s_* = \sqrt{\frac{G_*}{(3G + (\frac{3}{4})^4(k - \frac{5G}{3}))\delta}} \tag{1.19}$$

In questo caso il termine G_* differisce da quello della teoria bond-based in base al valore di critical energy release rate w_*.
L'ingrediente fondamentale per derivare il valore di s_* è che ad un legame che raggiunge la sua deformazione critica viene associato un determinato lavoro per unità di volume[2]. Nei modelli state based tuttavia ogni legame risponde alla deformazione collettiva di tutta la famiglia, dunque non c'è una corrispondenza univoca tra la deformazione del legame e il lavoro sul legame singolo. Per questo motivo il concetto di critical bond breakage strain in un modello state based non è del tutto corretto e potrebbe portare a delle approssimazioni.
Foster [3] ha proposto che in un materiale con modello state based

il danno può essere caratterizzato in termini di critical bond work, w_*. Con l'avanzare della deformazione nel tempo il lavoro in ogni legame ξ è calcolato come:

$$w(\xi, t) = \int_0^t \underline{\mathbf{T}}\langle\xi\rangle \cdot \underline{\dot{\mathbf{Y}}}\langle\xi\rangle$$

E la rottura del legame avviene quando:

$$w(\xi, t) = \frac{w_*}{2}$$

Il fattore un mezzo è dovuto al fatto che in un modello state based, se il legame ξ ha origine in $\mathbf{x}$, è allora presente un altro legame $-\xi$ connesso al punto $\mathbf{x} + \xi$ che ha consumato la stessa quantità di lavoro nel momento in cui si rompe.
Per un modello state based, dunque, tenendo conto di (1.16) possiamo riprodurre una relazione simile partendo da (1.15).

$$\underline{\mathbf{T}}'\langle\xi\rangle = \underline{\mathbf{T}}\langle\xi\rangle\beta = \left(\frac{3k\theta\underline{\omega}\langle\xi\rangle|\xi|}{m} + \alpha\underline{\omega}\langle\xi\rangle\underline{e}^d\langle\xi\rangle\right)\mathbf{M} \cdot \beta(\xi, t) \quad (1.20)$$

dove a questo punto:

$$\beta(\xi, t) = \begin{cases} 1 & se\ w(t') < w_*/2 \quad \forall t' \leq t \\ 0 & altrimenti \end{cases}$$

Ad oggi non vi è ancora una "direzione" preferita e si tende ad utilizzare anche per il caso della teoria state based un valore di critical bond breakage strain s_*.

1.1.6 Formulazione 2D della teoria peridinamica

Tutto quel che è stato visto sinora nell'introduzione fa riferimento al caso 3D in quanto la totalità della letteratura consultabile fa riferimento al caso più generico tridimensionale. Può tuttavia essere necessario l'utilizzo di una formulazione bidimensionale o anche monodimensionale della teoria, per adattarla a diversi casi di interesse. Seguendo l'approccio di Bobaru in [2], è possibile riformulare la teoria tenendo conto della dimensionalità del problema con il seguente approccio.
Si tenga in considerazione l'equazione generica del bilancio (1.4): quando si fa riferimento ad un dominio 1D o 2D, piuttosto che riscrivere tutte le equazioni è conveniente cambiare solamente il differenziale dV che appare nell'integrale in base al numero di dimensioni "D". Si definisce dunque:

$$dV = h_D \cdot \begin{cases} volume\ differenziale & se\ D = 3 \\ area\ differenziale & se\ D = 2 \\ lunghezza\ differenziale & se\ D = 1 \end{cases}$$

Dove

$$h_D = \begin{cases} 1 & se\ D = 3 \\ spessore\ della\ piastra & se\ D = 2 \\ sezione\ trasversale\ della\ trave & se\ D = 1 \end{cases}$$

Con questa convenzione le dimensioni di $\mathbf{f}$ sono forza/volume^2 e le dimensioni di ρ sono massa/volume, indipendentemente da D. Geometricamente il dominio $\mathcal{H}_x$ è invece una sfera, un disco o una linea.
Facendo uso di questa accortezza è possibile riscrivere le diver-

se grandezze per il caso bidimensionale, mantenendo invariate le equazioni. Risulta così possibile calcolare gli integrali utili alla calibrazione dei modelli anche nel caso 2D di nostro interesse. Muovendoci direttamente alla formulazione del danno, questa "convenzione" ci permette di calcolare il termine G_*, definito come lavoro per area unitaria di frattura, come:

$$G_* = 2h \int_0^\delta \left\{ \int_z^\delta \int_0^{\cos^{-1}(z/\xi)} (w_* \xi) d\phi d\xi \right\} dz = \frac{1}{2} c s_*^2 (\frac{h\delta^4}{2}) \quad (1.21)$$

Dove h rappresenta lo spessore della piastra, mentre per il caso 2D la spring constant è:

$$c = \frac{12k'}{\pi h \delta^3}, \quad k' = \begin{cases} E/2(1-\nu) & plane\ stress \\ E/2(1-\nu-2\nu^2) & plane\ strain \end{cases}$$

Dove con k' si intende il modulo di bulk bidimensionale. L'integrazione è effettuata su un'area circolare facendo uso di coordinate polari (ξ, ϕ). In questo modo, in maniera analoga a quanto già visto, per il modello bond based si ottiene un valore di critical bond stretch:

$$s_* = \sqrt{\frac{\pi G_*}{3k'\delta}} \quad (1.22)$$

Per il modello state-based, invece:

$$s_* = \sqrt{\frac{G_*}{(\frac{6}{\pi} G + \frac{16}{9\pi^2}(k' - 2G))\delta}} \quad (1.23)$$

1.2 Soluzione numerica

Questa sezione tratta la soluzione numerica della peridinamica, principalmente come fonte si è fatto riferimento al capitolo 5 di [2], scritto da David J. Littlewood e riferito al codice peridinamico *Peridigm* sviluppato dai Sandia National Laboratories. Si cerca dunque di riportare in questa sezione le nozioni principali dando un'impronta qualitativa, senza scendere in quelle di dettaglio per il codice Peridigm. Nel capitolo successivo, invece, si parlerà nello specifico del codice di ricerca utilizzato nell'ambito di questa tesi e delle sue caratteristiche.

1.2.1 Discretizzazione nello spazio

Le equazioni integro-differenziali per la peridinamica possono essere discretizzate nello spazio in diversi modi. L'approccio più utilizzato è il cosiddetto approccio *mesh-free* e si basa sulla suddivisone del dominio di interesse in dei punti di collocazione materiali, detti *nodi*. Ogni nodo è definito dalle sue coordinate iniziali (x, y, z) e dal suo volume, ΔV. Si assume che le varie quantità locali associate al punto materiale siano costanti nello spazio definito dal volume del nodo stesso. Ad ogni nodo nella discretizzazione viene assegnato un suo identificativo univoco. E' anche possibile generare una discretizzazione del tipo mesh-free passando per un modello agli elementi finiti convertendo gli elementi solidi in un vettore contenente $(x, y, z, \Delta V, block\_id)$ dove le coordinate (x, y, z) sono riferite al centroide dell'elemento, ΔV è riferito al volume dell'elemento e $block\_id$ è il riferimento univoco dell'elemento. Questo processo, sebbene semplice, nasconde delle criticità e delle considerazioni che devono essere valutate con

cautela, ma non utilizzando questo genere di approccio si ritiene superfluo riportarle.
La soluzione del problema peridinamico viene dunque ottenuta a partire da (1.11), che viene riportata per praticità:

$$\rho(\mathbf{x})\ddot{\mathbf{y}}(\mathbf{x},t) = \int_{\mathcal{H}_x} (\underline{\mathbf{T}}[\mathbf{x},t]\langle\xi\rangle - \underline{\mathbf{T}}[\mathbf{q},t]\langle-\xi\rangle)dV_q + \mathbf{b}(\mathbf{x},t)$$

Data la discretizzazione finita del corpo, l'integrale viene rimpiazzato dalla sommatoria sul set di nodi che compone il *vicinato* del punto materiale **x** cui è riferita l'equazione.

$$\rho(\mathbf{x})\ddot{\mathbf{y}}(\mathbf{x},t) = \sum_{\mathcal{H}_x} (\underline{\mathbf{T}}[\mathbf{x},t]\langle\xi\rangle - \underline{\mathbf{T}}[\mathbf{q},t]\langle-\xi\rangle)\Delta V_q + \mathbf{b}(\mathbf{x},t) \qquad (1.24)$$

La risposta del sistema è dunque determinata risolvendo questa sommatoria nel tempo per ogni punto materiale nella discretizzazione.

Correzione del volume

La sommatoria (1.24) prevede che vengano inclusi tutti i nodi che stanno all'interno del dominio sferico definito dall'orizzonte e centrato nel punto materiale **x**, andando a creare di fatto una "lista dei vicini". Per identificare i *vicini* del punto materiale è richiesta una "ricerca". Sebbene il concetto sia semplice, nella realtà la sua realizzazione non è così banale. Un approccio approssimativo prevede di creare una sorta di scatola sferica con estensione pari all'orizzonte, dentro la quale vengono identificati i punti appartenenti alla lista dei vicini del punto sul quale è centrata la sfera. Questo approccio tuttavia non tiene conto delle intersezioni parziali tra la sfera dell'orizzonte e i nodi che ne stanno al limite. Seguendo l'approccio di Seleson in [8], il volume di ogni nodo

viene suddiviso in un set di sottovolumi, ognuno dei quali ha un singolo punto materiale localizzato nel suo centroide. Quando un nodo si trova solo parzialmente all'interno della sfera identificata dall'orizzonte peridinamico di un punto, questa suddivisione in sottovolumi permette di tenere in considerazione l'intersezione parziale e dunque la parziale interazione che si ha con il punto materiale di riferimento localizzato al centro della sfera. Dopo che si è trovata questa intersezione parziale si può andare a modificare l'internal force modificando il volume del nodo ΔV_q. L'equazione (1.24) diventa:

$$\rho(\mathbf{x})\ddot{\mathbf{y}}(\mathbf{x},t) = \sum_{\mathcal{H}_x}(\underline{\mathbf{T}}[\mathbf{x},t]\langle\xi\rangle - \underline{\mathbf{T}}[\mathbf{q},t]\langle-\xi\rangle)\nu_q\Delta V_q + \mathbf{b}(\mathbf{x},t) \quad (1.25)$$

Dove ν_q è il fattore di correzione del volume per quel punto materiale.
Questa implementazione che si potrebbe definire come una sorta di *correzione del volume*, dal punto di vista computazionale è molto onerosa, dunque è necessario trovare un giusto compromesso tra la precisione e il tempo di calcolo.
Nel modello studiato nel corso di questa tesi è stato scelto di suddividere i nodi in 10 sottovolumi.

1.2.2 Integrazione nel tempo

Come sopra accennato, la progressione della simulazione è dettata da una routine di integrazione temporale. La durata della simulazione è quindi divisa in un numero finito di time step (o load step) ai quali l'equazione (1.24) viene valutata. E' possibile utilizzare schemi di integrazione di tipo esplicito o di tipo implicito.
Per i primi l'equazione (1.24) viene risolta direttamente per trovare

le accelerazioni nodali, che vengono poi applicate ad ogni nodo per avanzare la simulazione da un time step a quello successivo. Gli schemi di tipo implicito sono invece utilizzati per simulazioni quasi-statiche e trattano gli spostamenti dei nodi come incognite: (1.24) viene risolta per i valori di spostamento che restituiscono una accelerazione nulla per i gradi di libertà nodali che non sono soggetti a condizioni al contorno di tipo cinematico. Il risultato finale per entrambi i casi è una completa descrizione dello stato cinematico e fisico per tutti i punti materiali (nodi) della discretizzazione, per ogni time step della simulazione. Da questi risultati si possono derivare altre quantità di interesse. I metodi numerici espliciti soffrono del problema di essere condizionatamente stabili, il che significa che il time step per assicurarne la stabilità di convergenza deve essere determinato attentamente. Un possibile processo di determinazione del time step critico per il modello PMB è consultable in [10]. Per i metodi impliciti il time step utilizzabile è più ampio. Sebbene la risoluzione di problemi quasi-statici è associata ai metodi di tipo implicito, [4] riporta la possibilità di utilizzare un metodo esplicito di *Adaptive Dynamic Relaxation*, che consente di trovare le soluzioni quasi statiche di un problema andando a smorzarne la risposta a facendo uso di un coefficiente di damping. Il funzionamento di questa tipologia di schemi verrà illustrato successivamente.

1.2.3 Internal Force Density

Una parte fondamentale di un codice peridinamico sono le routine per il calcolo della internal force density. Tale grandezza è determinata tramite leggi costitutive che vanno a calcolare la pairwise bond force density tra i punti materiali che sono legati tra loro basandosi su dati cinematici e, opzionalmente, variabili di stato

del materiale. Gli algoritmi includono l'inizializzazione, calcolo dell'internal force density vero e proprio e il calcolo della matrice di rigidezza. Ognuno di questi algoritmi ha accesso ai dati relativi alla posizione iniziale di ogni nodo, volumi dei nodi, lista dei vicini per ogni nodo, grandezza del time step e tempo corrente. Inoltre vengono utilizzati anche i dati relativi alla velocità e alla posizione dei nodi che vengono aggiornati ad ogni time step. In aggiunta modelli costitutivi possono definire dei campi di dati specifici per il materiale e sono anch'essi aggiornati nel corso della simulazione. L'obbiettivo dell'integrazione nel tempo è quello di aggiornare e avanzare la simulazione da un dato time step (o load step) n, allo step successivo $n + 1$. Tenere traccia dei diversi valori allo step n e $n + 1$ consente ad un modello costitutivo di calcolare i rate della grandezza in questione.
A partire dal calcolo di questa grandezza è possibile trovare la matrice di rigidezza del sistema, derivando le forze di interazione rispetto al vettore di spostamento relativo tra i due punti.

1.2.4 Rottura e danneggiamento del legame

Il legame tra i nodi in un modello peridinamico consente di modellare il danno e la rottura. Seguendo quanto riportato in [10], ad ogni legame nel dominio peridinamico viene assegnato un valore scalare di danno. Un possibile approccio è quello di definire una variabile di danno, d_{ij}, che tiene traccia del danno di un legame che unisce due punti materiali i, j. Questa variabile può assumere un valore tra 0 e 1, dove 0 significa che il legame è intatto, mentre 1 significa la completa rottura del legame. I valori di danneggiamento di ogni legame vengono aggiornati nel corso di tutta l'analisi e incorporati direttamente nel calcolo dell'internal force density. Si potrebbe per esempio pensare all'influence function in un modello

costitutivo state-based che viene modificata dal fattore $(1 - d_{ij})$. In questo modo l'influenza dei legami danneggiati è ridotta, mentre i legami che si sono rotti vengono direttamente esclusi dalla simulazione. L'accumulazione di legami rotti porta alla separazione del materiale e alla formazione delle cricche.
Di seguito si riporta un esempio di funzionamento dell'algoritmo. All'inizio della simulazione i valori di *bond damage* sono posti pari a 0, ad ogni time step (o load step) il bond stretch s, definito come la variazione di lunghezza del legame in rapporto alla lunghezza indeformata dello stesso, viene calcolato per ogni coppia di punti. I valori di bond stretch vengono poi confrontati con un valore critico e se lo superano il legame si rompe. Si pone dunque $d_{ij} = 1$.

2

Il codice di ricerca

In questo capitolo verrà brevemente spiegato il funzionamneto del codice di ricerca utilizzato, riportando i punti salienti e approfondendo brevemente alcune sue caratteristiche.
Il codice é chiamato *ExPS* ed è stato realizzato dal prof. Federico Dalla Barba dell'Università degli Studi di Padova. Esso è organizzato in numerosi moduli, ognuno dei quali svolge delle funzioni specifiche che variano dalla valutazione dello stato del legame, dal calcolo di coefficienti e proprietà del materiale sino alla scrittura dei risultati in file di output che vengono poi rielaborati per essere resi visualizzabili in *paraview*. E' possibile settare il solutore fornendo dei dati, selezionando diverse modalità e modelli costitutivi del materiale e selezionando anche il numero di iterazioni che si desidera effettuare. Questi settaggi vengono messi in pratica semplicemenente andando a inserirli in un file di testo esterno, che il solutore andrà a leggere al suo avvio prendendo così i dati necessari al suo funzionamento. Di seguito si riportano delle parti di questo file in cui è possibile vedere i parametri sui quali si può agire. La definizione dell'orizzonte, delle dimensioni della piastra

e altri parametri riguardanti la geometria infine avviene agendo direttamente sulle righe di codice preposte, le quali si trovano in una cartella separata dal solutore vero e proprio e nella quale avviene la gestione iniziale della geometria, la creazione delle famiglie, dei legami e l'inizializzazione del problema. In appendice (D) è possibile trovare una spiegazione della struttura in moduli e cartelle del codice e una loro breve descrizione.

```
SOLVER SETTINGS
'./DATA/'               I/O path
0                       it_min
150                     it_max
2                       it_out
10                      screen output interval
1.0D0                   CFL
1.0D0                   time step
5000                    maximum number of iteration for ADR solver
1.0D-6                  maximum residual kinetic energy for ADR solver
3                       Flag: 1 => transient RK3 solver, 2 => transient Verlet, 3 => Verlet ADR solver
1                       Flag: 1 => bond-based model, 2 => state-based model
2                       Flag: 1 => 2D plane stress, 2 => 2D plane strain, 3 => 3D
1                       Flag: 1 => linear-elastic behaviour, 2 => bi-linear behaviour, 3 => elasto-plastic behaviour
2                       Flag: 1 => manual time-step, 2 => automatic time-step
2                       Flag: 1 => Gaussian IF, 2 => unitary IF, 3 => conical IF
0                       Flag: 0 => surface correction disabled, 1 => surface correction enabled
0                       Flag: 0 => surface detection disabled, 1 => surface detection enabled
0                       Flag: 0 => crack-detection disabled, 1 => crack-detection enabled
0                       Flag: 0 => surface contact disabled, 1 => surface contact enabled
```

(a) Settaggio del solutore

```
MATERIAL PARAMETERS
72.000000000D+9         Young's modulus                                 SI: [N/m^2]
0.3333333333D+0         Poisson's ratio (state based only)              SI: [-]
2440.0000000D+0         density                                         SI: [kg/m^3]
135.00000000D+0         critical fracture energy release rate           SI: [J/m^2]
1.0000000000D+0         ultimate tensile strength                       SI: [N/m^2]
1.0000000000D+7         internal damping coefficient (relative velocity)  SI: [kg/(m^3 s)]
0.0000000000D+0         external damping coefficient (absolute velocity)  SI: [kg/(m^3 s)]
0.0000000000D+0         Gravity acceleration - x                        SI: [m/s^2]
0.0000000000D+0         Gravity acceleration - y                        SI: [m/s^2]
0.0000000000D+0         Gravity acceleration - z                        SI: [m/s^2]
```

(b) Settaggio dei parametri

Figura 2.1: Possibilità di settaggio del codice per la risoluzione del problema peridinamico

2.1 Discretizzazione nello spazio

Questa prima parte di soluzione del problema avviene come già detto in un ambiente che è esterno al solutore stesso, ma dopo la creazione del corpo peridinamico non basta fare altro che copiare il file di output contenente tutti i dati prodotti dall'inizializzazione

all'interno della cartella contenente il solutore.
Il codice permette di creare delle piastre di diverso genere, utilizzando una costruzione della griglia di nodi caratterizzata da un *grid spacing* uniforme nelle 3 direzioni. E' possibile creare piastre semplici, piastre con un foro circolare all'interno e piastre prefratturate. Per l'interesse di questa tesi ci si limita a illustrare la creazione di una semplice piastra regolare, senza cricche e senza fori. Si comincia definendo il numero di nodi N_x che si vuole avere in direzione x. Automaticamente a partire dalla lughezza del lato L_x la piastra viene suddivisa in nodi uniformemente distanziati tra loro di dx. Nella direzione y, per utilizzare una spaziatura uniforme si suddivide la lunghezza L_y per la distanza dx tenendo conto solo della parte intera del risultato, ricavando il numero di nodi necessario in questa direzione per avere un grid spacing uniforme. Si calcola poi $dy = L_y/N_y$. La stessa cosa avviene nella terza direzione. Questa procedura in realtà non consente sempre di avere una spaziatura uguale in tutte e tre le direzioni, in quanto non è detto che il numero di nodi e la loro distanza consenta di ottenere esattamente la dimensione desiderata della piastra. Comunque il risultato è quello che per ogni direzione la spaziatura è costante tra i nodi in quella direzione. I volumi risultano dunque essere uguali per tutti i nodi e calcolabili come $vlm = dx \cdot dy \cdot dz$. Il numero di nodi complessivo viene calcolato come $N_P = N_x \cdot N_y \cdot N_z$.
Il codice richiede di fornire le coordinate dell'angolo inferiore sinistro (x_c, y_c, z_c) per iniziare la costruzione della griglia di nodi, dopodichè con una serie di algoritmi crea i diversi punti materiali e li distribuisce. Si riporta di seguito parte dell'algoritmo di creazione della piastra:

```
1
2       do i=1,Nx
3         !
```

```
      do j=1,Ny
        !
        do k=1,Nz
          !
          tmp_x=xc+dx*(real(i,rk)-0.5_rk)
          tmp_y=yc+dy*(real(j,rk)-0.5_rk)
          tmp_z=zc+dz*(real(k,rk)-0.5_rk)
          !
          h=h+1
          !
          prd_bdy_idx(h)=0
          !
          prd_rps_x(h)=tmp_x
          prd_rps_y(h)=tmp_y
          prd_rps_z(h)=tmp_z
          !
          prd_pos_x(h)=tmp_x
          prd_pos_y(h)=tmp_y
          prd_pos_z(h)=tmp_z
          prd_vel_x(h)=0.0_rk
          prd_vel_y(h)=0.0_rk
          prd_vel_z(h)=0.0_rk
          !
          prd_prt_vlm(h)=vlm
          !
        enddo
        !
      enddo
      !
    enddo
    !
    N_P=h
```

Codice 2.1: Creazione della griglia di nodi

Praticamente il codice crea una griglia in base al numero di nodi in ogni direzione e associa alla posizone centrale di ogni cella le coordinate (x, y, z) del nodo corrispondente e il suo volume vlm. Questa posizione viene dunque definita come posizione di riferimento. Ogni cella viene infine associata ad un suo identificativo univoco. Si completa l'inizializzazione associando ai nodi una ve-

località iniziale di riferimento nulla.
Unitamente a questo algoritmo (ma non riportato) viene calcolato anche il numero di legami presenti all'interno dell'orizzonte, che essendo unico ed essendo uniforme anche il grid spacing risulta uguale per ogni punto. Si creano così le famiglie dei punti della griglia.
In questo modo si conclude la procedura in inizializzazione, tutte le altre azioni svolte dal solutore e di seguito riportate dunque si intendono avvenire all'interno del solutore vero e proprio e non in un ambiente esterno ad esso.

2.2 Integrazione nel tempo

Il codice ExPS permette di utilizzare diversi algoritmi di integrazione temporale, in particolare per la risoluzione di problemi quasi statici si può utilizzare un algoritmo del tipo *Verlet ADR* dove con ADR si intende *Adaptive Dynamic Relaxation*. Per la risoluzione di problemi di tipo transient invece il codice permette di utilizzare due algoritmi espliciti: *Transient RK3* e *Transient Verlet*. Una caratteristica importante di tale codice è che è in grado di determinare in maniera automatica il time step necessario per la stabilità dell'analisi, andando a rendere più semplice il processo di scelta del time step e rendendolo di fatto non necessario. Questo risulta essere molto importante soprattutto nel caso di schemi espliciti che come è stato precedentemente detto soffrono di instabilità. E' anche possibile impostare manualmente il time step.
L'interfaccia di setting del solutore, ovvero il file di testo precedentemente citato, permette di definire il numero di iterazioni effettuate e il numero di output creati. Per gli schemi Verlet Transient e RK3 transient è possibile scegliere a piacere il numero di iterazioni

massime effettuate e l'intervallo di creazione del file di output. Per lo schema Verlet ADR invece, oltre a queste è possibile assegnare ad ogni iterazione un numero arbitrario di passi di Dynamic Relaxation. Sempre per l'ADR infine è possibile impostare un valore massimo del residuo di velocità, in maniera tale che i passi di Dynamic Relaxation siano inferiori a quelli impostati nel caso in cui il sistema venga smorzato quanto necessario. Il residuo è definito come la massima tra le velocità di tutti i punti materiali del corpo. Sebbene si tratti di schemi di integrazione temporale, nella realtà il tempo è un tempo fittizio, nel senso che non ha alcun valore fisico, quest'ultima affermazione è da ricordare soprattuto successivamente quando si tratterà l'applicazione di carichi varianti nel tempo.
Mentre per gli schemi espliciti generici è presente una vasta manualistica e sono relativamente semplici, si ritiene invece necessario approfondire brevemente il funzionamento dello schema di Adaptive Dynamic Relaxation, che nel codice di ricerca si basa su uno schema di tipo Verlet. Esso è uno schema di tipo esplicito con la caratteristica di essere adatto a trovare soluzioni quasi statiche. Per fare ciò si è fatto riferimento a [6] e [4].

2.2.1 Schemi espliciti

Gli schemi espliciti possono essere diversi tra loro per velocità di convergenza, ordine di accuratezza e altro. Essi si basano comunque tutti sullo stesso principio, che viene qui brevemente esplicato.
Prendiamo in considerazione l'equazione di bilancio peridinamica nella sua forma discretizzata (1.25). Se immaginiamo di essere al

passo n (il tempo sarà $t = n \cdot \Delta t$) la possiamo riscrivere come:

$$\rho(\mathbf{x})\ddot{\mathbf{y}}^n(\mathbf{x}) = \sum_{\mathcal{H}_x}(\underline{\mathbf{T}}^n[\mathbf{x}]\langle\xi\rangle - \underline{\mathbf{T}}^n[\mathbf{q}]\langle-\xi\rangle)\nu_q\Delta V_q + \mathbf{b}^n(\mathbf{x}) \qquad (2.1)$$

Si ricorda che il force state $\underline{\mathbf{T}}$ dipende dal deformation state $\underline{\mathbf{Y}}$. Dopo aver determianto l'accelerazione al passo n, possiamo per esempio utilizzare uno schema alle differenze finite di tipo forward e backward rispettivamente, possiamo calcolare:

$$\dot{\mathbf{y}}^{n+1} = \ddot{\mathbf{y}}^n\Delta t + \dot{\mathbf{y}}^n$$

Che è la velocità al passo $n+1$. Successivamente si puo calcolare:

$$\mathbf{y}^{n+1} = \dot{\mathbf{y}}^{n+1}\Delta t\mathbf{y}^n$$

Che è la posizione al passo $n+1$. In questo modo si può procedere ad aggiornare l'equazione (2.1) per trovare l'accelerazione al passo successivo. Questo procedimento viene fatto per tutti i punti che interagiscono tra loro nell'orizzonte, facendo avanzare l'algoritmo nel tempo.

Implementazione dello schema esplicito Verlet Transient

Si illustra dunque brevemente l'implementazione nel codice dello schema Verlet Transient.

```
1    prd_vel_x(1:N_P)=prd_vel_x(1:N_P)+0.5_rk*prd_dt*rhs_vel_x(1:N_P)
2    prd_vel_y(1:N_P)=prd_vel_y(1:N_P)+0.5_rk*prd_dt*rhs_vel_y(1:N_P)
3    prd_vel_z(1:N_P)=prd_vel_z(1:N_P)+0.5_rk*prd_dt*rhs_vel_z(1:N_P)
4    !
5    call PRD_BCS_ApplyBCs
6    !
7    prd_pos_x(1:N_P)=prd_pos_x(1:N_P)+prd_dt*prd_vel_x(1:N_P)
8    prd_pos_y(1:N_P)=prd_pos_y(1:N_P)+prd_dt*prd_vel_y(1:N_P)
9    prd_pos_z(1:N_P)=prd_pos_z(1:N_P)+prd_dt*prd_vel_z(1:N_P)
```

```
10      !
11      call PRD_MPI_IBD_SndRcvBlock1
12      !
13          !
14          call SOL_SBP_CmptRHS_1
15          !
16          call PRD_MPI_IBD_SndRcvOneRk8(prd_prt_dlt)
17          call PRD_MPI_IBD_SndRcvOneRk8(prd_prt_adt_1)
18          !
19          call SOL_SBP_CmptRHS_2
20          !
21          !
22          !
23      prd_vel_x(1:N_P)=prd_vel_x(1:N_P)+0.5_rk*prd_dt*rhs_vel_x(1:N_P)
24      prd_vel_y(1:N_P)=prd_vel_y(1:N_P)+0.5_rk*prd_dt*rhs_vel_y(1:N_P)
25      prd_vel_z(1:N_P)=prd_vel_z(1:N_P)+0.5_rk*prd_dt*rhs_vel_z(1:N_P)
26      !
27      call PRD_BCS_ApplyBCs
28      !
29      call PRD_MPI_IBD_SndRcvBlock1
30      !
31      call PRD_MSC_UpdtPartSts
32      call PRD_MSC_UpdtBondSts
33      !
34      call PRD_MPI_IBD_SndrcvOneIk1(prd_prt_sts)
35      !
36      if(flag_isp==1) then
37        !
38        call PRD_MSC_UpdtPartPos
39        !
40        call PRD_MPI_IBD_SndRcvBlock0
41        !
42      endif
43      !
44      prd_time=prd_time+prd_dt
45      !
46      return
```

Codice 2.2: Schema di integrazione temporale del tipo Verlet Transient

Questo schema funziona in maniera simile a quanto spiegato so-

pra, con la differenza che la valutazione della velocità avviene in due parti: si parte al passo 0, viene calcolata la velocità e successivamente la posizione al passo 0 dopo aver applicato le condizioni di vincolo. Dopo la chiamata ad una serie di subroutines tra le quali quella dedicata al calcolo del RHS, che vedremo successivamente, viene calcolata la velocità al passo $1/2$. Il ciclo ricomincia, andando a valutare nuovamente la velocità al passo $1/2+1/2$, dunque al passo 1. Con la velocità al passo 1 si calcola la posizione al passo 1 e lo schema dunque procede calcolando ancora la velocità mezzo passo più avanti prima di tornare all'inizio. Il tempo fittizio viene aggiornato alla fine, dopo il calcolo della velocità al passo intermedio.

2.2.2 ADR

Il metodo di dynamic relaxation, come già accennato in precedenza, è basato sul fatto che la soluzione statica è la parte costante della risposta transitoria. Introducendo uno smorzamento (damping) artificiale al sistema la soluzione viene "guidata" al suo stato costante molto velocemente. Non è tuttavia sempre facile determinare il coefficiente di smorzamento più efficace per tutta la soluzione. Usando l'Adaptive Dynamic Relaxation il coefficiente viene determinato così ad ogni step.
Si comincia riscrivendo le equazioni del moto come un sistema di equazioni differenziali ordinarie per tutti i punti del sistema, introducendo termini fittizi di inerzia e di smorzamento.

$$\mathbf{D}\ddot{\mathbf{U}}(\mathbf{X},t) + c\mathbf{D}\dot{\mathbf{U}}(\mathbf{X},t) = \mathbf{F}(\mathbf{U},\mathbf{U'},\mathbf{X},\mathbf{X'}) \tag{2.2}$$

Dove la matrice $\mathbf{D}$ è una matrice di densità fittizia, diagonale, mentre c è il coefficiente di smorzamento. I vettori $\mathbf{X}$ e $\mathbf{U}$ contengono

invece le posizioni iniziali e gli spostamenti dei punti materiali **x** e possono essere riscritte come:

$$\mathbf{X}^T = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}, \quad \mathbf{U}^T = \{\mathbf{y}(\mathbf{x}_1, t), \mathbf{y}(\mathbf{x}_2, t), ..., \mathbf{y}(\mathbf{x}_N, t)\}$$

Dove N è il numero totale di punti nel corpo. Le matrici **X**' e **U**' sono associate invece ai punti che nel corso dell'introduzione a questa tesi abbiamo identificato con **q**.
Si può calcolare velocità e spostamenti al time step successivo utilizzando uno schema centrato come:

$$\dot{\mathbf{U}}^{n+1/2} = \frac{((2 - c^n \Delta t)\dot{\mathbf{U}}^{n-1/2} + 2\Delta t \mathbf{D}^{-1}\mathbf{F}^n)}{2 + c^n \Delta t} \tag{2.3}$$

E

$$\mathbf{U}^{n+1} = \mathbf{U}^n + \Delta t \dot{\mathbf{U}}^{n+1/2} \tag{2.4}$$

Dove con **F** si indica il *force vector*, cioè il vettore delle equazioni (1.25) per ogni punto. Sebbene non si conosca la velocità a $t^{1/2}$, si può assumere per inizializzare l'integrazione che $\mathbf{U}^0 \neq 0$ e $\dot{\mathbf{U}} = 0$, ottenendo come passo iniziale:

$$\dot{\mathbf{U}}^{1/2} = \frac{\Delta t \mathbf{D}^{-1}\mathbf{F}^0}{2}$$

L'unico termine che ha un significato fisico in questo algoritmo è il force vector **F**. Gli altri termini non sono quantità fisicamente rilevanti e i loro valori possono dunque essere scelti per ottenere una convergenza più rapida. Una dimensione di time step $\Delta t = 1$ è solitamente una scelta conveniente. I termini della matrice di densità diagonale **D** possono essere scelti sulla base del teorema di

Greschgorin e possono essere espressi come

$$\lambda_{ii} \geq \frac{1}{4}\Delta t^2 \sum_j |\mathbf{K}_{ij}|$$

Dove $\mathbf{K}_{ij}$ è la matrice di rigidezza del sistema che si sta considerando. Il segno $\geq$ assicura la stabilità dello schema alle differenze finite centrato. La condizione di stabilità è derivata da [11]. Al passo n-esimo l'equazione (2.2) può essere riscritta come:

$$\ddot{\mathbf{U}}^n(\mathbf{X}, t^n) + c^n \dot{\mathbf{U}}^n(\mathbf{X}, t^n) = \mathbf{D}^{-1}\mathbf{F}^n(\mathbf{U}^n, \mathbf{U}'^n, \mathbf{X}, \mathbf{X}') \tag{2.5}$$

Dove il coefficiente di damping al passo n si può scrivere come:

$$c^n = 2\sqrt{((\mathbf{U}^n)^T {}^1\mathbf{K}^n \mathbf{U}^n)/((\mathbf{U}^n)^T \mathbf{U}^n)} \tag{2.6}$$

Dove ${}^1\mathbf{K}^n$ è la matrice di rigidezza locale diagonale, derivabile da:

$${}^1\mathbf{K}^n = -(\mathbf{F}_i^n/\lambda_{ii} - \mathbf{F}_i^{n-1}/\lambda_{ii})/(\Delta t \dot{u}_i^{n-1/2})$$

Implementazione dello schema Verlet ADR

Si vuole ora illustrare brevemente come lo schema Verlet ADR è implementato all'interno del codice di ricerca.

```
1         do i=1,N_P
2     !
3     K_xx=0.0_rk
4     K_yy=0.0_rk
5     K_zz=0.0_rk
6     !
7     if(abs(prd_vel_old_x(i))>zero_rk) K_xx=-(rhs_vel_x(i)-prhs_vel_x
    (i))/(adr_mass*prd_dt*prd_vel_old_x(i))
8     if(abs(prd_vel_old_y(i))>zero_rk) K_yy=-(rhs_vel_y(i)-prhs_vel_y
    (i))/(adr_mass*prd_dt*prd_vel_old_y(i))
9     if(abs(prd_vel_old_z(i))>zero_rk) K_zz=-(rhs_vel_z(i)-prhs_vel_z
    (i))/(adr_mass*prd_dt*prd_vel_old_z(i))
```

```
10      !
11      UU=UU+prd_dsp_x(i)**2+prd_dsp_y(i)**2+prd_dsp_z(i)**2
12      !
13      UKU=UKU+K_xx*prd_dsp_x(i)**2+K_yy*prd_dsp_y(i)**2+K_zz*prd_dsp_z
    (i)**2
14      !
15     enddo
16     !
17     call MPI_ALLREDUCE(MPI_IN_PLACE,UU,1,MPI_DP,MPI_SUM,
    MPI_COMM_WORLD,err)
18     call MPI_ALLREDUCE(MPI_IN_PLACE,UKU,1,MPI_DP,MPI_SUM,
    MPI_COMM_WORLD,err)
19     !
20     if((UU>zero_rk).and.(UKU>zero_rk)) CADR=2.0_rk*sqrt(UKU/UU)
21     !
22     CADR=max(CADR,0.1_rk)
23     !
24     if(prd_time>zero_rk) then
25       !
26       prd_vel_x(1:N_P)=((2.0_rk-CADR*prd_dt)*prd_vel_old_x(1:N_P)+2.0
    _rk*prd_dt*rhs_vel_x(1:N_P)/adr_mass)/(2.0_rk+CADR*prd_dt)
27       prd_vel_y(1:N_P)=((2.0_rk-CADR*prd_dt)*prd_vel_old_y(1:N_P)+2.0
    _rk*prd_dt*rhs_vel_y(1:N_P)/adr_mass)/(2.0_rk+CADR*prd_dt)
28       prd_vel_z(1:N_P)=((2.0_rk-CADR*prd_dt)*prd_vel_old_z(1:N_P)+2.0
    _rk*prd_dt*rhs_vel_z(1:N_P)/adr_mass)/(2.0_rk+CADR*prd_dt)
29       !
30     else
31       !
32       prd_vel_x(1:N_P)=prd_dt*0.5_rk*rhs_vel_x(1:N_P)/adr_mass
33       prd_vel_y(1:N_P)=prd_dt*0.5_rk*rhs_vel_y(1:N_P)/adr_mass
34       prd_vel_z(1:N_P)=prd_dt*0.5_rk*rhs_vel_z(1:N_P)/adr_mass
35       !
36     endif
37     !
38     call PRD_BCS_ApplyBCs
39     !
40     call PRD_MPI_IBD_SndRcvBlock1
41     !
42     prd_pos_x(1:N_P)=prd_rps_x(1:N_P)+prd_dsp_x(1:N_P)+prd_dt*
    prd_vel_x(1:N_P)
43     prd_pos_y(1:N_P)=prd_rps_y(1:N_P)+prd_dsp_y(1:N_P)+prd_dt*
    prd_vel_y(1:N_P)
```

```
44    prd_pos_z(1:N_P)=prd_rps_z(1:N_P)+prd_dsp_z(1:N_P)+prd_dt*
      prd_vel_z(1:N_P)
45    !
46    call PRD_BCS_ApplyBCs
47    !
48    call PRD_MPI_IBD_SndRcvBlock1
49    !
50    prhs_vel_x(1:N_P)=rhs_vel_x(1:N_P)
51    prhs_vel_y(1:N_P)=rhs_vel_y(1:N_P)
52    prhs_vel_z(1:N_P)=rhs_vel_z(1:N_P)
53    !
54    prd_vel_old_x(1:N_P)=prd_vel_x(1:N_P)
55    prd_vel_old_y(1:N_P)=prd_vel_y(1:N_P)
56    prd_vel_old_z(1:N_P)=prd_vel_z(1:N_P)
57    !
58    if(flag_isp==1) then
59      !
60      call PRD_MSC_UpdtPartPos
61      !
62      call PRD_MPI_IBD_SndRcvBlock0
63      !
64    endif
65    !
66    adr_rke=maxval(sqrt(prd_vel_x(1:N_P)**2+prd_vel_y(1:N_P)**2+
      prd_vel_z(1:N_P)**2))
67    !
68    !
69    call MPI_ALLREDUCE(MPI_IN_PLACE,adr_rke,1,MPI_DP,MPI_MAX,
      MPI_COMM_WORLD,err)
70    !
71    prd_time=prd_time+prd_dt
72    !
73    return
```

Codice 2.3: Schema di integrazione temporale del tipo Verlet ADR

Per prima cosa viene calcolatala matrice di rigidezza locale diagonale ${}^1\mathbf{K}^n$ nei suoi 3 termini lungo x, y, z. In questo schema specifico, i termini della matrice di densità diagonale fittizia $\mathbf{D}$, cioe i vari λ_{ii} sono tutti definiti uguali a 0.5 e indicati con il termine "*adr_mass*". Si calcolano poi il numeratore e il denominatore di (2.6), chiamati

rispettivamente come "UKU" e "UU". Il coefficiente di damping è dunque ricavato come "$CADR = 2\sqrt{UKU/UU}$. A questo punto grazie ad un breve ciclo if viene permesso al solutore di inizializzare lo schema al passo 1/2 per la prima iterazione, mentre nel caso non fosse la prima iterazione lo schema procede calcolando la velocità utilizzando la relazione (2.3) e successivamente dopo aver applicato le condizioni al contorno derivate dall'apposito modulo si calcola la posizione aggiornata dei punti utilizzando la relazione (2.4). Prestando attenzione alle relazioni sopra riportate, si nota che la velocità viene ricavata al passo $n + 1/2$ mentre la posizione al passo $n + 1$. Il calcolo del residuo avviene come ultima cosa assieme all'aggiornamento del tempo fittizio, prima di ritornare all'inizio dello schema e ripetere la valutazione al passo successivo. Il termine di accelerazione *rhs_vel_·* è ottenuto tramite la chiamata alle routine di valutazione del RHS dell'equazione, delle quali si parlerà successivamente. Va notato infine come le accelerazioni e le velocità al passo precedente non vengano eliminate, in quanto necessarie per la valutazione del passo successivo, ma vengano conservate in dei vettori appositi chiamati *prhs_vel_·* e *prd_vel_old_·*.

2.3 Rottura del legame

La valutazione del danno e della rottura dei legami è un aspetto molto delicato, sebbene a livello di pura descrizione qualitativa possa sembrare semplice, nella realtà il codice è piuttosto elaborato e si suddivide in diverse parti. Questa valutazione viene effettuata ad ogni step, dopo che le equazioni di bilancio sono state risolte e la posizione dei punti materiali viene aggiornata. La componente principale del ciclo di valutazione viene qui riportata e spiegata.

```
1           csi_x_ij=rps_x_j-rps_x_i
```

```
2           csi_y_ij=rps_y_j-rps_y_i
3           csi_z_ij=rps_z_j-rps_z_i
4           !
5           eta_x_ij=pos_x_j-pos_x_i
6           eta_y_ij=pos_y_j-pos_y_i
7           eta_z_ij=pos_z_j-pos_z_i
8           !
9           csi_ij=sqrt(csi_x_ij**2+csi_y_ij**2+csi_z_ij**2)
10          !
11          eta_ij=sqrt(eta_x_ij**2+eta_y_ij**2+eta_z_ij**2)
12          !
13          str_ij=(eta_ij-csi_ij)/csi_ij
14          !
15          cnd=(prd_bnd_cnt(k,i)==1_ik1).and.(str_ij>lms)
16          !
17          if(cnd) prd_bnd_sts(k,i)=0_ik1
18          !
19          prd_bnd_smx(k,i)=max(prd_bnd_smx(k,i),str_ij)
20          !
21      endif
```

Codice 2.4: Valutazione della rottura dei legami

Nella parte precedente a quella riportata si attivano una serie di cicli *do*, che essenzialmente servono a valutare la posizione del punto i-esimo e quella di ogni punto j-esimo appartenente alla sua famiglia. Dopo questa prima serie di valutazioni della posizione si giunge alla routine vera e propria, nella quale ogni legame all'interno della famiglia del punto i-esimo viene valutato: il codice calcola la lunghezza iniziale di ogni legame, prendendo la posizione di riferimento iniziale dei due punti i, j dove ancora una volta con j si intendono i punti appartenenti alla famiglia di i. Viene così calcolata la lunghezza del legame iniziale in ognuna delle tre direzioni dello spazio, indicata con "csi_·_ij". Successivamente il codice ripete lo stesso calcolo, tenendo però conto delle posizioni al passo di integrazione corrente dei due punti. Quest'ultimo valore è invece indicato come "eta_·_ij". Applicando la radice della som-

ma dei quadrati delle tre componenti si ricavano così le lunghezze iniziali del legame e le lunghezze deformate. Da questi valori si calcola la deformazione del legame, "str_ij". Infine, se nel codice decidiamo di attivare la rottura del legame, si abilita un ciclo *if* che va a valutare se la deformazione del legame è maggiore di quella critica e in tal caso il legame si considera rotto.
Questo modulo viene infine chiamato all'interno del solutore, cosicchè ad ogni passo viene effettuata la valutazione.

2.4 Internal force density

Come detto precedentemente l'internal force density è molto impoortante e la sua valutazione è fondamentale. Nella teoria bond based il processo è piuttosto semplice, in quanto non viene tenuto conto della relazione tra tutti i legami della famiglia, mentre nella teoria state based è piuttosto complesso. Viene brevemenente riportata una sezione di codice che mostra come avviene la valutazione per il modello state-based. Per il modello bond-based il processo è molto simile ma notevolmente più semplice.
Il codice riportato è leggermente modificato per renderlo più comprensibile, nella realtà alcune grandezze vengono valutate in moduli separati con l'ausilio di numerose variabili e poi richiamate all'interno del modulo per il calcolo dell'internal force density. Un'altra nota è che il codice è basato su una serie di cicli *do* che non verranno qui riportati, ma la loro funzionalità è essenzialmente quella di valutare tutte queste grandezze per ogni legame nella famiglia dei diversi punti. La serie di operazioni che viene illustrata in seguito dunque deve ritenersi ripetuta per ogni legame di ogni punto materiale del corpo.

```
1         integer(ik1),dimension(:,:),allocatable :: prd_bnd_sts !
  stat: 0 => broken bond, 1 => unbroken bond
2         !
3         sts_ij=real(prd_bnd_sts(k,i),rk)
4         !
5         ! i-j interaction
6         !
7         ext_sfr_ij=eta_ij-csi_ij
8         !
9         ext_dvt_ij=ext_sfr_ij-dlt_i*csi_ij/3.0_rk
10        !
11        dlt_i=dlt_i+mat_beta*omg_ij*csi_ij*ext_ij/wvl_i*sts_ij*
  vlm_ij
12        !
13        adt_i=adt_i+mat_gamma*mem_omg_ij(k)*ext_dvt_ij*mem_csi_ij(k
  )**2/wvl_i*sts_ij*mem_vlm_ij(k)
14        !
15        fst_ij=(mat_beta*mat_bmd*dlt_i*csi_ij-adt_i+mat_alpha*
  ext_dvt_ij)*omg_ij/wvl_i
16        !
17        ! j-i interaction
18        !
19        wvl_j=prd_wtd_vlm(j)
20        dlt_j=prd_prt_dlt(j)
21        adt_j=prd_prt_adt_1(j)
22        !
23        ext_dvt_ji=ext_sfr_ij-dlt_j*csi_ij/3.0_rk
24        !
25        fst_ji=(mat_beta*mat_bmd*dlt_j*csi_ij-adt_j+mat_alpha*
  ext_dvt_ji)*omg_ij/wvl_j
26        !
27        ! total internal forcing
28        !
29        int_frc_x_i=int_frc_x_i+sts_ij*(fst_ij+fst_ji)*ver_x_ij*
  vlm_ij
30        int_frc_y_i=int_frc_y_i+sts_ij*(fst_ij+fst_ji)*ver_y_ij*
  vlm_ij
31        int_frc_z_i=int_frc_z_i+sts_ij*(fst_ij+fst_ji)*ver_z_ij*
  vlm_ij
32        !
33
```

Codice 2.5: Valutazione dell'internal force density

Per una più semplice comprensione del codice viene fatto riferimento alla equazione (1.15), che permette il calcolo del force state. Viene dunque riportata dopo averla riscritta in una forma più semplice per avere un riferimento e comprendere meglio il codice.

$$\underline{\mathbf{T}}\langle\xi\rangle = \left(3k\theta|\xi| + \alpha\underline{e}^d\langle\xi\rangle m\right)\underline{\omega}\langle\xi\rangle\frac{\mathbf{M}}{m}$$

Per prima cosa, dopo aver valutato la deformazione del legame si definisce una variabile "sts_ij" che può essere 0 o 1 e serve per escludere eventuali legami che sono rotti e che dunque non prendono parte al calcolo dell'internal force density. Questa variabile dipende dal valore "prd_bnd_sts(k,i)" che è il risultato del modulo precedentemente riportato per la verifica della rottura del legame. Il codice procede valutando il deviatoric extension state $\underline{e}^d$, nominato "ext_dvt_ij" e la dilatazione non-locale θ, nominata "dlt_i", dove con "wvl_i" si intende la costante di normalizzazione m. Il termine "adt_i" è un termine addizionale, che serve per "centrare" l'equazione in maniera tale da correggerla tenendo conto della dimensionalità del problema: per esempio nel caso 3D esso si annulla, rendendo l'equazione coincidente a quella qui riportata. Nel caso 2D plane stress o 2D plane strain essa assumerà valori diversi in maniera tale da rendere il risultato concorde con quello della teoria. I vari termini "mat_alpha", "mat_beta", "mat_gamma" sono proprio quei valori che contribuiscono a modificare l'equazione per i vari casi. A questo punto viene calcolato il force state $\underline{\mathbf{T}}\langle\xi\rangle$, denominato come "fst_ij", per ogni punto materiale del corpo. Tutto questo processo viene ripetuto ricavando il valore per il force state $\underline{\mathbf{T}}\langle-\xi\rangle$. Ora che il codice possiede tutti gli ingredienti necessari avviene il calcolo dell'internal force density nelle tre direzioni, facendo uso dei tre versori "ver_·_ij", che sono i corrispettivi di

M in (1.15). E' importante notare come anche qui vi sia il termine moltiplicativo che tiene conto della rottura o meno del legame, andando dunque ad escludere dalla computazione della internal force density quei legami appartenenti alla famiglia che si sono rotti.
In questo modo è stato trovato il primo pezzo dell'equazione del bilancio discretizzata (1.25) per il passo corrente.

2.5 Applicazione di vincoli e carichi e soluzione numerica delle equazioni di bilancio

Il secondo pezzo dell'equazione di bilancio è dato dalle forze esterne che agiscono sul corpo. Vi sono all'interno del codice dei moduli dedicati per l'applicazione delle condizioni di vincolo e di carico. La definizione vera e propria dei carichi e dei vincoli verrà illustrata in seguito, al momento l'interesse si concentrerà sul mostrare come il codice interpreta le condizioni imposte e come le applica, andando poi a completare l'equazione di bilancio (1.25) della quale era già stato trovato il primo termine precedentemente e dunque risolvendo il sistema con uno schema di integrazione temporale tra quelli presentati. Vengono riportate quindi alcune righe di codice semplificate per comprendere meglio il principio di funzionamento.

```
1 do i=1,N_P
2     !
3     ext_frc_x_i=prd_cnc_frc_x(i)+prd_ext_frc_x(i)
4     ext_frc_y_i=prd_cnc_frc_y(i)+prd_ext_frc_y(i)
5     ext_frc_z_i=prd_cnc_frc_z(i)+prd_ext_frc_z(i)
6     !
7     !
```

```
8     rhs_vel_x(i)=(int_frc_x_i+ext_frc_x_i+int_dmp_x_i-mat_edm*vel_x_i
     )*den_inv+grv_x
9     rhs_vel_y(i)=(int_frc_y_i+ext_frc_y_i+int_dmp_y_i-mat_edm*vel_y_i
     )*den_inv+grv_y
10    rhs_vel_z(i)=(int_frc_z_i+ext_frc_z_i+int_dmp_z_i-mat_edm*vel_z_i
     )*den_inv+grv_z
11    !
12 enddo
```

Codice 2.6: Calcolo del RHS

Inanzitutto il codice prende come input le condizioni di carico che sono state inserite nell'apposito modulo, denominate come "prd_ext_frc_·(i)" e le sommma alle forze di contatto "prd_cnc_frc_·(i)". Questo processo viene ripetuto per ogni punto materiale. Il risultato di questa somma non viene inserito in un vettore direttamente, ma in uno scalare denominato "ext_frc_·_i". Questo valore scalare viene successivamente inserito nell'equazione finale di bilancio vera e propria, nella quale sono presenti altri termini come l'internal force density, il cui calcolo è già stato effettuato ed eventualmente una serie di altri valori come damping interni ed esterni e accelerazioni di gravità. In questo modo il solutore calcola il RHS (right hand side), ovvero l'accelerazione che viene poi utilizzata in (1.25). E' importante tenere in considerazione che le forze, per definizione stessa all'interno del codice ma anche per definizione teorica delle equazioni di bilancio della peridinamica, sono forze per unità di volume. Nella descrizione dei carichi va pertanto tenuto conto anche del volume dei nodi ai quali si applicano tali forze. A livello pratico questo valore viene poi utilizzato dal solutore negli schemi temporali (tramite la chiamata alle subroutines di valutazione del RHS), illustrati in precedenza.

3

Simulazione peridinamica

Dopo aver compreso le basi della teoria peridinamica, aver esplorato la sua implementazione numerica e aver visto il funzionamento del codice di ricerca utilizzato, si intende riportare il problema studiato in [7].

3.1 Introduzione al problema

Il float glass, come già detto, è un materiale molto utilizzato nell'industria, e prevederne la frattura e la sua resistenza nel post frattura è di fondamentale importanza per le applicazioni che ha. La resistenza del vetro, in particolare, è determianta dai difetti della sua superficie, ovvero dalle microcricche che possono assumere dimensioni variabili tra i $10\mu m$ e i $300\mu m$ come risultato del processo di produzione. Il float glass è ottenuto colando il vetro fuso in degli stampi di metallo e lasciato raffreddare. Ciò che accade dunque e che nel processo di produzione ci sarà un lato che è a contatto con il metallo, detto *tin side* e l'altro che è a contatto con l'aria ed è detto *air side*. Il metallo può diffondersi all'interno del vetro du-

rante questo processo e come risultato si ha una penetrazione del metallo nel tin side del pannello di float glass che varia tra i $10\mu m$ e i $50\mu m$. A questi piccoli difetti si aggiungono quelli dovuti ai gradienti termici in entrambi i lati del pannello durante il processo di manifattura. Nel complesso dunque accade che solitamente la dimensione media dei difetti nell'air side è inferiore di un fattore 1.5-2.0 rispetto alla dimensione media di quelli nel tin side. Questa differenza di difetti comporta proprietà di resistenza diverse per i due lati del pannello. Per ricavare i parametri del materiale sono stati eseguiti una serie di esperimenti di "double ring bending" nei quali sono stati trovati dei valori statistici per lo stress di frattura di tin side e air side. Una simulazione peridinamica del test di laboratorio infine permette la comparazione dei pattern di frattura dei pannelli. Di seguito si illustra brevemente la procedura di test utilizzata e i risultati ottenuti sperimentalmente in [7], per poi passare al modello peridinamico con il quale si è tentato di riprodurre i pattern di frattura e analizzare il comportamento del codice ExPS rispetto a Peridigm. Lo scopo del riportare i risultati dei test di laboratorio è quello di permettere la comprensione delle varie scelte fatte, nell'imposizione dei vincoli e dei carichi. I risultati di riferimento si possono trovare nell'articolo.

3.1.1 Procedura di test in laboratorio

Il float glass utilizzato in laboratorio è il cosiddetto *soda-lime*, in accordo con la normativa DIN EN 572-1. Il test di double ring bending è invece effettuato secondo la normativa DIN 1288-5 (R45): il load ring ha un raggio di $9mm$ mentre il support ring di $45mm$. Le dimensioni dei pannelli utilizzati per l'esperimento sono di $100x100x2.9mm$. Secondo le specifiche del test il caso valido di frattura è definito quando la cricca si nuclea all'interno del load

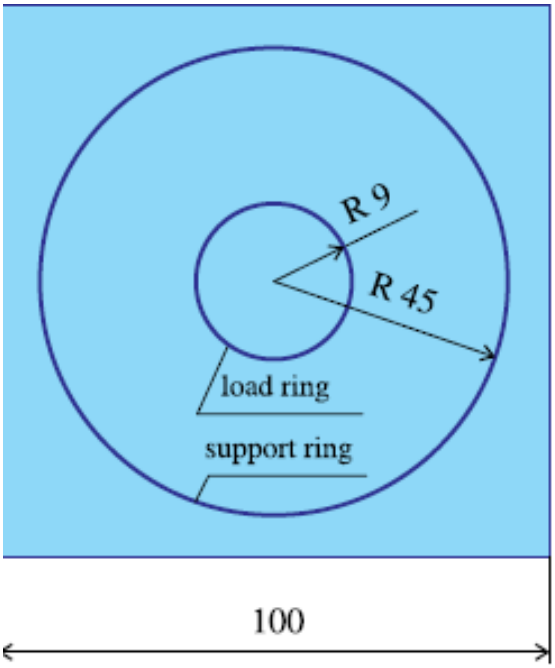


Figura 3.1: Setup per il test di double ring bending

ring. Numerosi campioni di float glass hanno presentato la nucleazione della cricca all'esterno del load ring o sul bordo dello stesso, come visibile in figure (3.2) e (3.4). Vista la distribuzione casuale dei difetti, inoltre, lo stress di frattura varia in un range piuttosto ampio, dunque è richiesto un numero elevato di test.
Per finire, sempre in [7] è dimostrato che i risultati di deflessione del test sono in accordo con quelli della teoria classica delle piastre, consentendo così di validare l'utilizzo del codice peridinamico per questo problema tramite un confronto con un modello FEM.

I risultati trovati, tenendo conto solo dei campioni validi, sono consultabili nella tabella successivamente riportata.

	Fracture strenght, MPa
Tin side	191.7 (185.0...198.7)
Air side	559.1 (438.0...713.6)

Tabella 3.1: Valori di tensione di frattura per tin side e air side, derivati da test di laboratorio

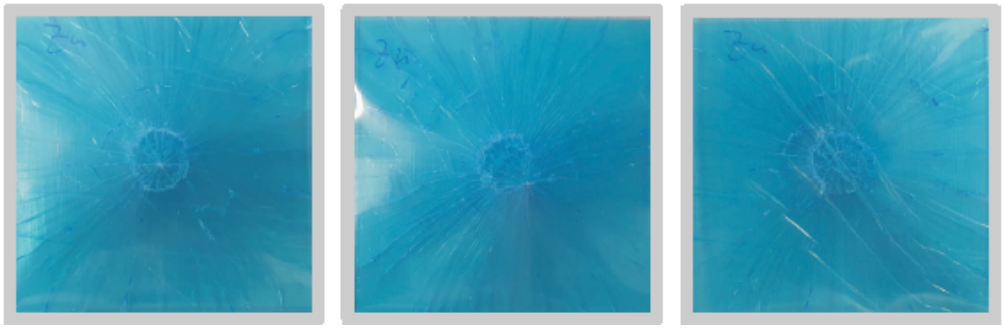


Figura 3.2: Pattern di frattura per il tin side: nucleazione dentro al load ring, sul bordo, fuori (o non identificabile)

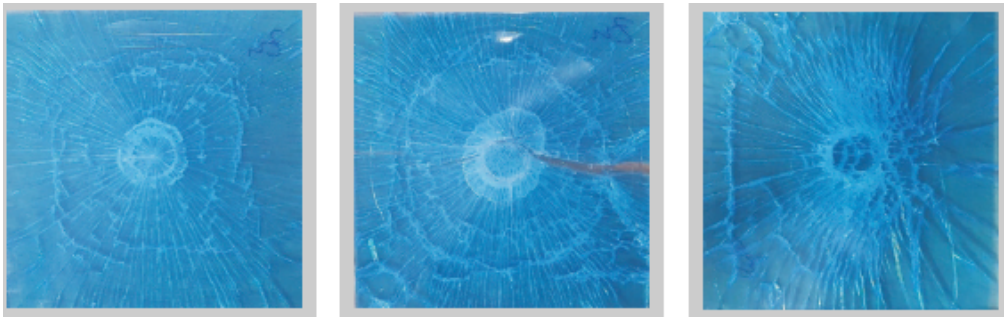


Figura 3.3: Pattern di frattura per l'air side: nucleazione dentro al load ring, sul bordo, fuori (o non identificabile)

3.2 Analisi di deflessione

Prima di procedere con lo studio vero e proprio della propagazione del danno è necessario "calibrare" il modello. Per fare ciò, seguendo quanto di simile fatto in [7] si procede creando un modello agli elementi finiti che fornisca i risultati di deflessione della piastra, in maniera tale da cercare di replicare i risultati tramite l'imposizione di vincoli e carichi al modello peridinamico, così da settare correttamente il codice prima di procedere con l'analisi di interesse vera e propria. Si procede dunque creando un semplice modello tramite l'utilizzo di *MSC Patran*, il cui unico scopo è quello di dare un risultato sul quale fare affidamento per la riproduzione. Non si andrà dunque a discutere sulla realizzazione di tale

modello, ma si presenteranno semplicemente i risultati ottenuti. Viene applicata in accordo con [7] una forza complessiva di $1662N$ sui bordi dell'anello di carico, lungo una circonferenza di raggio $9mm$. Per imitare il vincolo imposto dall'anello di supporto invece si bloccano gli spostamenti in direzione z lungo una circonferenza di raggio $45mm$. Con il risultato che si vuole ottenere ben noto,

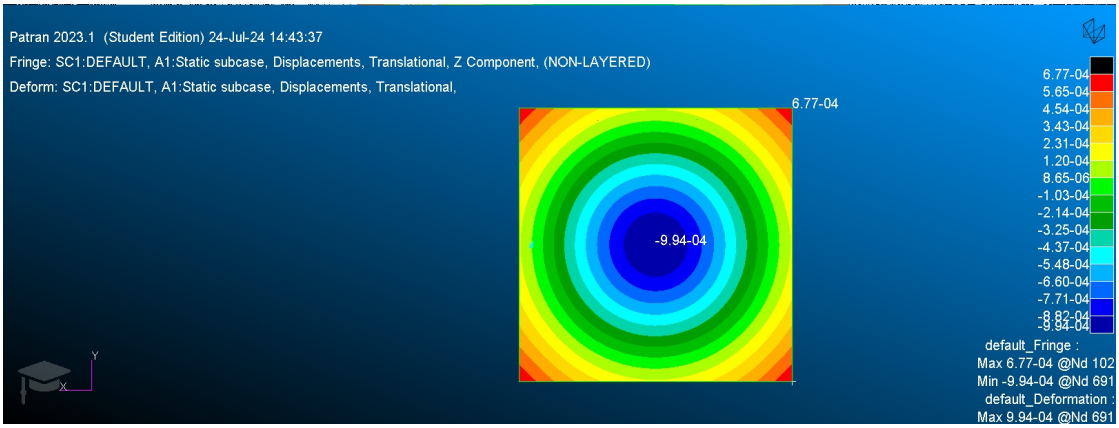


Figura 3.4: Deflessione della piastra valutata con un modello agli elementi finiti

ora si procede con la creazione del modello peridinamico. Per prima cosa si crea una piastra tridimensionale, le caratteristiche di dimensioni, grid spacing e numero di nodi vengono qui riportate. Come valore di orizzonte, in accordo con l'articolo è stato scelto un valore pari a $\delta = 3.015dz = 1,45mm$.
In totale, il numero di nodi è $N_P = 257094$ mentre il volume dei punti materiali è $vlm = 1,128e^{-10}m^3$.
Dopo aver creato la geometria della piastra vanno impostate le proprietà meccaniche della stessa tramite l'apposito file di input del quale si è precedentemente parlato. In accordo con i valori che è stato possibile consultare si è scelto di utilizzare quelli riportati in tabella 3.3.
Per l'impostazione del solutore invece si è scelto un modello di materiale del tipo state-based con influence function Gaussiana e una

	Direzione x	**Direzione y**	**Direzione z**
Dimensione [mm]	100	100	2,9
Grid spacing [mm]	0,483092	0,483092	0,483333
Numero di nodi	207	207	6

Tabella 3.2: Caratteristiche geometriche della piastra nel modello peridinamico

Modulo di Young E	70 GPa
Modulo di Poisson ν	0,22
Densità ρ	2440 kg/m^3

Tabella 3.3: Caratteristiche meccaniche della piastra nel modello peridinamico

legge costitutiva del tipo 3D. Trattandosi di un'analisi quasi-statica lo schema di integrazione temporale è del tipo Verlet ADR. La selezione del time step come già detto non è necessaria in quanto il codice in maniera automatica è in grado di scegliere il time step più adatto ad ogni passo per favorire la convergenza. Per quanto riguarda il numero di iterazioni si è scelto invece un approccio un po' più pratico che teorico: tenendo controllato il risultato in output si è andato a scegliere un numero di iterazioni tale che l'output del sistema fosse stabile e non cambiasse più in maniera apprezzabile tra un passo e l'altro.
L'applicazione delle condizioni di vincolo e carico è una parte critica e fondamentale del problema: i due anelli di carico e di supporto hanno una forma circolare dunque si è cominciato con il creare una variabile corrispondente all'equazione della circoferenza, del tipo $(x - x_c)^2 + (y - y_c)^2 = r^2$. Data la natura discretizzata del problema, infine, non è possibile definire il raggio come un singolo valore, in quanto è molto improbabile che il centro di ogni punto materiale della griglia si trovi esattamente a quella distanza. E' dunque necessario impostare un range di valori entro i quali tutti i centri

dei punti materiali che sono presenti "subiscono" la condizione di vincolo o di carico scelta. Il carico viene applicato solamente ai primi 4 nodi in direzione z della griglia, in maniera tale da coprire almeno un orizzonte.
Le condizioni di vincolo sono state applicate in maniera analoga, variando ovviamenente il raggio della circonferenza e applicandole solamente ai primi 4 nodi in direzione z dal lato opposto della piastra rispetto a quello di applicazione dei carichi. In figura (3.5) è possbile vedere i punti di applicazione di queste condizioni tramite una vista in sezione della piastra. In figura (3.6) è invece possibile vedere le zone circonferenziali di applicazione delle condizioni tramite una vista dall'alto.

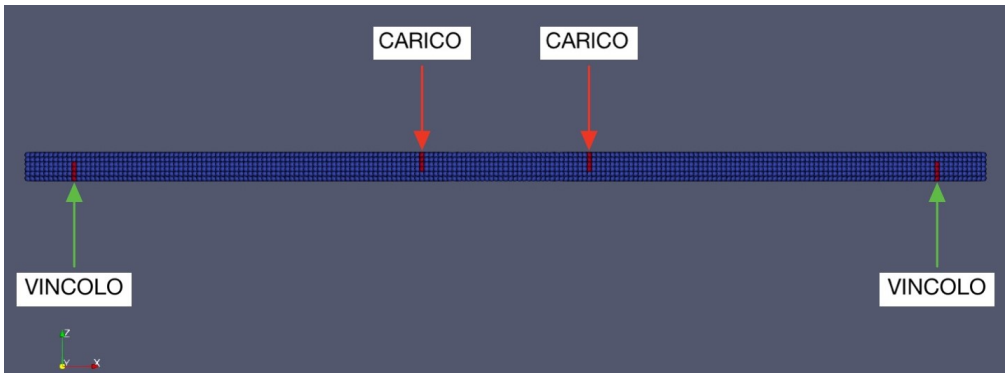


Figura 3.5: Applicazione di vincoli e carichi, vista in sezione

L'implementazone di queste condizioni, così come i range di applicazione delle stesse sono visibili in (Appendice A).
I risultati di deflessione della piastra vengono illustrati nell'immagine (3.7).
Il risultato si presenta diverso da quello atteso: il comportamento della piastra, sia ai bordi che nella zona centrale, si discosta da quello previsto dal modello agli elementi finiti. I motivi per i quali si ha questa differenza tuttavia non sono facilmente identificabili, le complessità introdotte dalla teroia peridinamica, infatti, sono

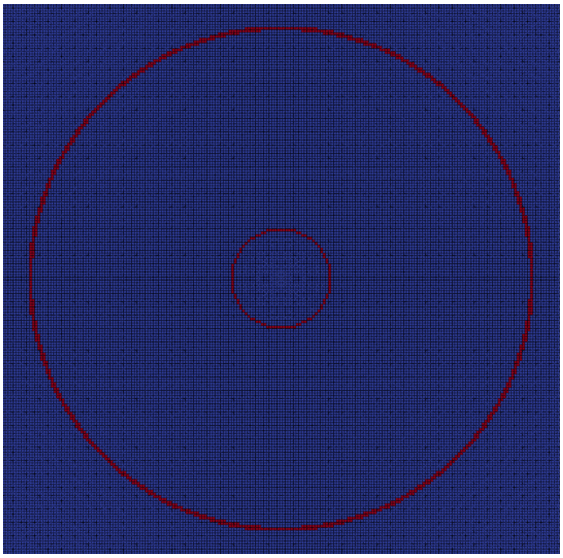


Figura 3.6: Condizioni di vincolo e carico corrispondenti all'anello di supporto e all'anello di carico applicate alla griglia di nodi

molteplici. Sicuramente va tenuto conto del fatto che la teoria peridinamica è una teoria differente da quella classica, sulla quale si basano i metodi agli elementi finiti. La teoria peridinamica inoltre risente dei cosidetti "effetti di bordo", che sicuramente possono andare a diminuire la precisione dei risultati in termini di deflessione. L'entità di questi effetti di bordo risulta tuttavia non quantificabile. Un altro fattore che influenza i risultati è il fatto che le condizioni di vincolo e carico nella realtà andrebbero poste su dei nodi che non fanno parte della griglia della piastra, ma che sono creati appositamente al di sopra e al di sotto di essa nelle zone che corrispondono all'anello di suporto e a quello di carico, così come viene fatto in [7]. La creazione di questi nodi non risulta semplice a livello di implementazione nel codice e richiederebbe delle abilità di programmazione che vanno oltre le conoscenze in possesso da chi sta scrivendo questa tesi. Sicuramente infine vi è un'influenza della discretizzazione spaziale scelta per la griglia di nodi e la conseguente risoluzione numerica del problema peridinamico. Sebbene la discretizzazione scelta viene ritenuta sufficientemente fine ed è

circa la stessa utilizzata in [7], sicuramente in piccola parte anch'essa ha un'influenza sui risultati.

La nuclezione delle cricche tuttavia avviene in modo indipendente

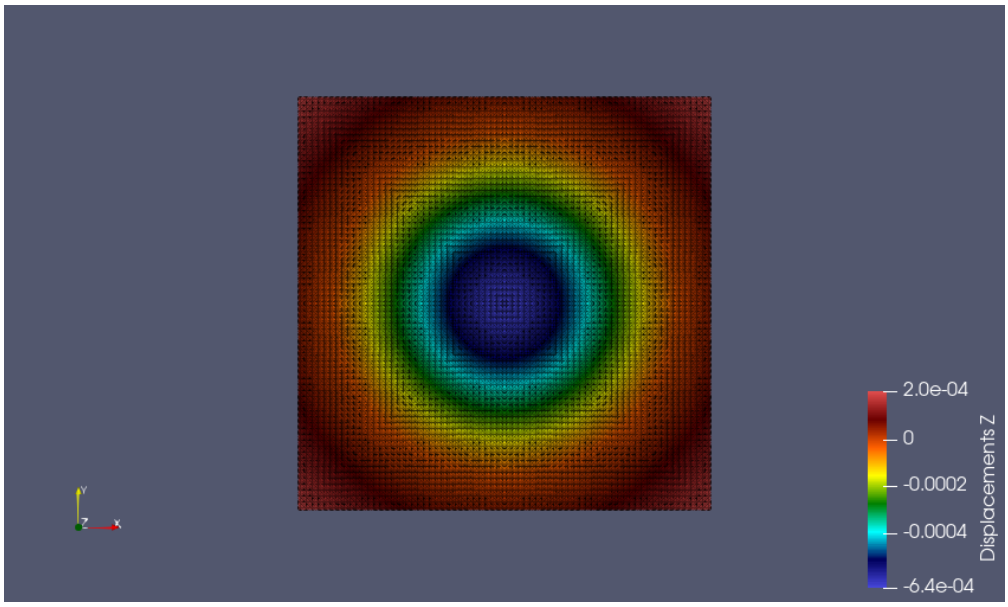


Figura 3.7: Risultati di deflessione per la piastra con il modello peridinamico e schema di integrazione temporale di tipo Verlet ADR

dai risultati di deflessione della piastra. Se ci si interessa dal punto di vista puramente qualitativo nell'indagare i pattern di frattura delle cricche, i risultati di deflessione non vanno a modificare la capacità del codice di caratterizzare o meno i fenomeni di propagazione del danno nel materiale. Per quanto riguarda i valori di forza ai quali avviene il danneggiamento invece non è possibile prevedere a priori se saranno maggiori, minori o uguali a quelli risultanti dai test di laboratorio. In figura (3.8) si può vedere un confronto tra il modello peridinamico e quello FEM, dove si nota chiaramente la discrepanza tra i due risultati, in termini di valori e anche in termini di andamento qualitativo, dal centro piastra al vertice in alto a destra.

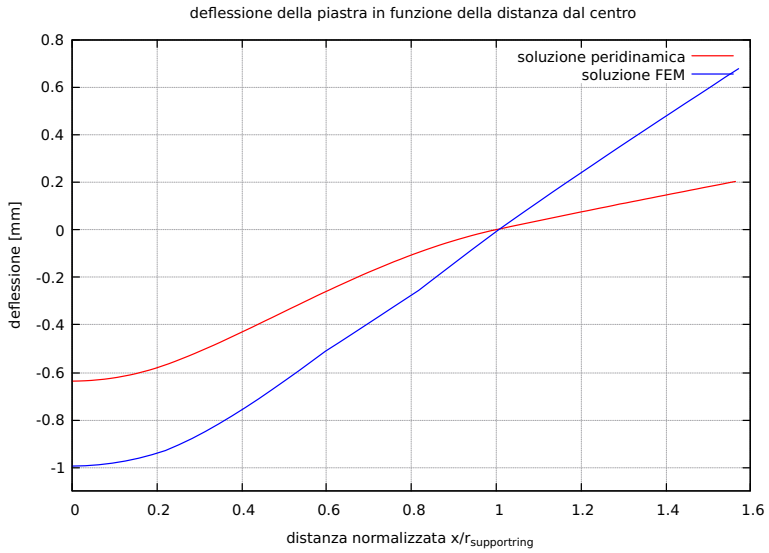


Figura 3.8: Andamento della deflessione in funzione della distanza radiale dal centro della piastra, confronto tra modello peridinamico e FEM

3.3 Analisi del Danno

In questa seconda parte di analisi si va ad effettuare lo studio di interesse vero e proprio. Ora che siamo a conoscenza delle corrette condizioni di vincolo e carico da applicare per replicare le condizioni del test in laboratorio, possiamo procede con l'inserire i dati di interesse per la valutazione della rottura dei legami. Anche per questa seconda analisi lo schema di integrazione temporale scelto è quello di tipo Verlet ADR, in quanto siamo intenzionati a simulare un carico che varia nel tempo in maniera lineare e a valutare dunque la risposta quasi-statica della piastra e non un impatto o un colpo ad alta velocità. L'analisi svolta è in riferimento al tin side della piastra, che è caratterizzato da un carico di rottura inferiore a causa del maggior numero di difetti presenti. L'approccio tuttavia può essere ripetuto per l'air side con le stesse modalità, verificando anche in quel caso la congruenza o meno con i risultati attesi.

A scopo di una corretta comprensione di quanto verrà illustrato in seguito si vuole spiegare il concetto di livello del danno: le sequenze riportate fanno riferimento ad un livello di danno della piastra in funzione del tempo o del carico; tale livello di danno non va inteso come percentuale di legami rotti nella piastra ma bensì come il massimo tra i livelli di danno di tutti i punti materiali presenti nella piastra. Il livello di danno di ogni punto materiale è valutato come numero di legami rotti sui legami totali della sua famiglia.

3.3.1 Setting del codice

Prima di procedere con la presentazione ed il commento dei risultati è importante andare a spiegare come è stato impostato il codice, in quanto vengono fatte delle considerazioni e delle modifiche che è importante comprendere a pieno.
La prima difficoltà è quella relativa all'imposizione dei carichi. Dall'analisi di deflessione siamo stati in grado di comprendere dove e come applicare i carichi e le condizioni di vincolo. Mentre quest'ultime rimangono invariate, le condizioni di carico hanno ora la caratteristica di essere tempo-varianti. Replicando quanto è stato fatto in [7] è stato imposto un carico crescente in maniera lineare di $29N/s$, in corrispondenza dell'anello di carico. E' possibile vedere l'incremento del carico nel tempo in figura (3.9). L'implementazione di questo carico tempo-variante è invece presentata in appendice (B).
La piastra così come è stata creata dal codice è una piastra "perfetta", nel senso che è caratterizzata da un griglia uniforme, in cui tutti i legami tra i punti materiali sono intatti e hanno tutti lo stesso valore di limit bond stretch s_*. Questa perfezione che di per se potrebbe sembrare un vantaggio comporta a delle inconsistenze con la realtà. La frattura infatti nella realtà come è già stato detto

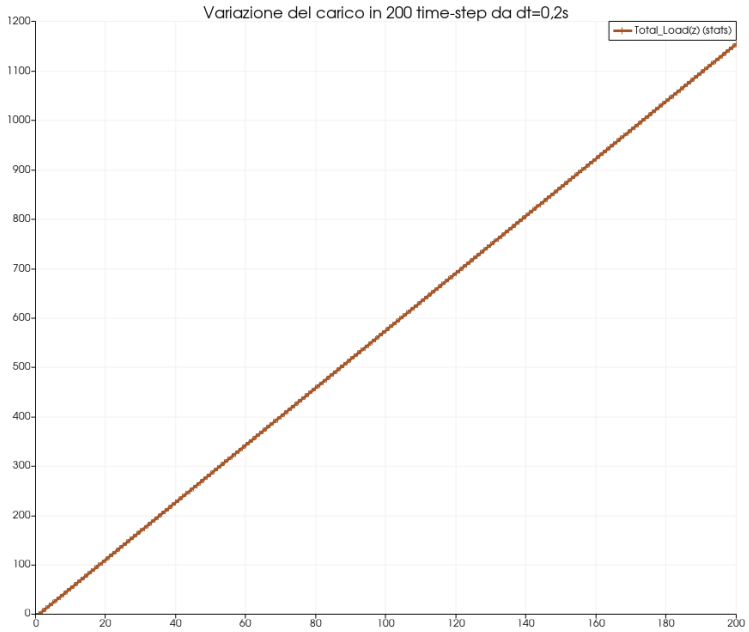


Figura 3.9: Carico totale in direzione z tempo-variante, da 0 a 40s, con un incremento in output di $\Delta t = 0,2$

viene influenzata dai difetti che sono presenti nel materiale e il pattern di propagazione delle cricche dunque va a "seguire" questi difetti, risultato altamente imprevedibile. Se facessimo una simulazione peridinamica senza inserire dei difetti causali nel materiale otterremmo un pattern di propagazione delle cricche simmetrico rispetto alle due direzioni x e y in quanto la frattura non avrebbe una direzione preferenziale in cui diffondersi. Un'immagine esemplificativa di questo è visibile in figura (3.10): in questo modello di piastra non sono stati infatti introdotti dei difetti per romperne la simmetria e si può chiaramente notare che il la propagazione del danno è simmetrica rispetto alle direzioni x e y. Per questo motivo è stato scelto di introdurre dei difetti in maniera random nel materiale, sotto forma di legami rotti. Per l'implementazione di questi difetti si fa riferimento all'appendice (C). La piastra presentata dai ricercatori in [7] non soffre di questo problema, in quanto è derivata a partire da un modello agli elementi finiti con

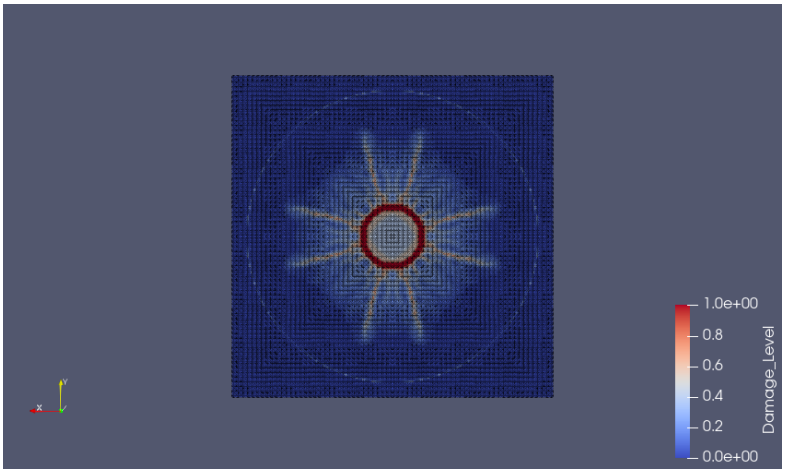


Figura 3.10: Pattern di frattura simmetrico della piastra: nessun difetto introdotto

mesh irregolare, dunque i suoi difetti sono introdotti intrinsecamente nella costruzione della griglia di nodi e non vi è la necessità di agire in questo senso. La nostra piastra è invece caratterizzata da una griglia di nodi regolare e omogenea, va dunque "aiutata" a rompere la simmetria.
Dopo aver impostato queste condizioni preliminari per la corretta riuscita della simulazione, si procede con il setting del codice. Tramite il file di testo presente nel solutore si va ad abilitare il "crack-detection" e anche il "surface-detection". Il parametro fondamentale da inserire è quello dedicato al *critical fracture energy release rate*, che nel corso dell'introduzione abbiamo identificato come il lavoro per area unitaria di frattura G_*. In accordo con [7] questo valore viene impostato come $G_* = 8.25 J/m^2$. Nella realtà quest'ultimo valore nella letteratura non è presente, né per l'air side né per il tin side. Il suo valore è una stima, basato sui risultati di alcuni parametri proposti in [5]. Ci si aspetta anche per questo motivo che i valori derivanti di s_* e di conseguenza di carico critico per un determinato valore di danno saranno differenti da quelli sperimentali.

3.3.2 Valutazione del pattern di frattura

Per prima cosa viene investigato il pattern di frattura del materiale ottenuto grazie alla simulazione peridinamica e confrontato con quelli derivanti dai test in laboratorio. Per questa prima valutazione, il valore di s_* è stato valutato facendo uso della relazione (1.19), che per il caso studiato restituisce un valore:

$$s_* = 3,301 \cdot 10^{-4}$$

Per l'ottenimento di una buona discretizzazione temporale del risultato è stato scelto, lasciando invariato il carico precedentemente presentato, di modificare le richieste in output in maniera tale da avere un risultato ogni 0,20 istanti temporali (fittizi). I risultati di frattura del materiale vengono dunque riportati per i livelli di danno di 0% (istante immediatamente precedente alla nucleazione del danno), 3%, 40%, 67%, 90% e 100%.

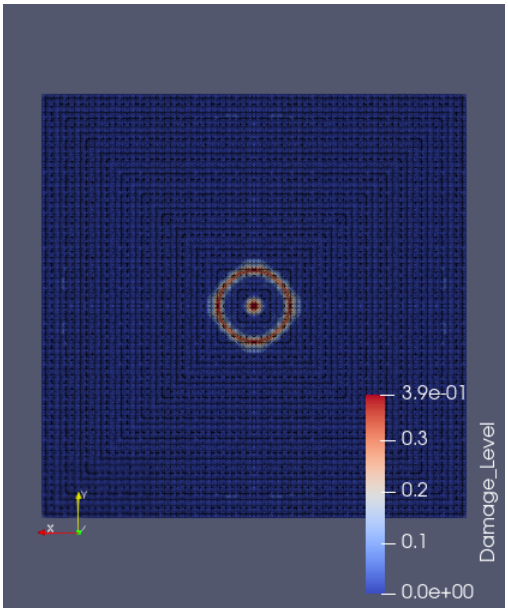
In un confronto con i risultati di laboratorio riportati in [7] questi

Danno	**Tempo [s]**	**Carico Totale [N]**
0%	11	319
3%	11,2	324,8
40%	11,6	336,4
67%	12	348
90%	16,8	487,2
100%	33,2	962,8

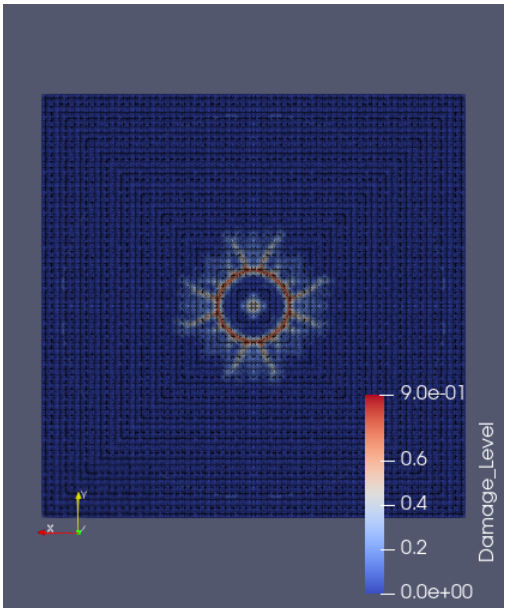
Tabella 3.4: Risultati di frattura per il tin side della piastra, $s_* = 3,301 \cdot 10^{-4}$

valori di carico risultano nettamente inferiori. Non essendo in grado di determinare quanto le condizioni di vincolo e carico differenti da quelle ideali influiscano su questi risultati, ci si può limitare a osservare il pattern di frattura del pannello. L'unico parametro

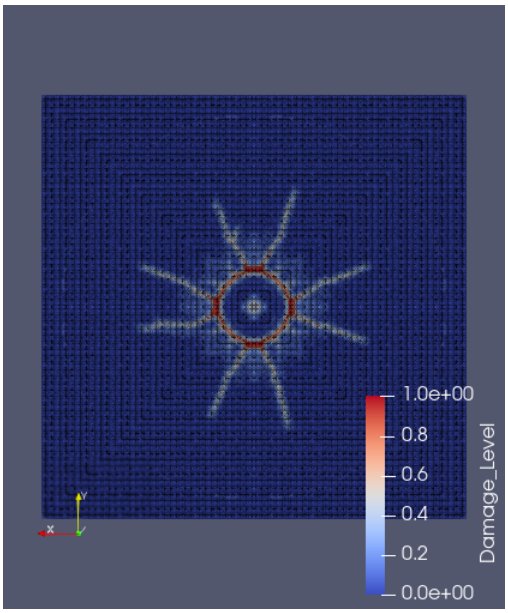
su cui è possibile agire per andare a variare il carico di rottura è quello di limit bond stretch s_*. Dopo aver analizzato il pattern di frattura, riportato per diversi istanti temporali in figura (3.11) per il tin side della piastra, si può notare che esso risulta coerente con quello reale ottenuto dal double ring bending test presentato in figura (3.2). E' interessante notare come l'introduzione di pochi legami danneggiati sia stata sufficiente per andare a eliminare la simmetria della piastra. Il pattern di frattura, però, sebbene non simmetrico risulta comunque chiaramente influenzato dalla regolarità della griglia di nodi, permettendo alle cricche di espandersi maggiormente in direzioni che tendenzialmente sono parallele alle direzioni x e y della piastra. Questa tendenza sarebbe anche potuta essere prevista osservando la discretizzazione dell'anello di carico in figura (3.6), nella quale si vede che le zone dell'anello di carico meglio discretizzate sono quelle ai 4 lati paralleli alle due direzioni principali. Nonostante tutto si è riusciti a simulare in maniera coerente con la realtà delle semplici "caratteristiche" di processi molto complessi e randomici, come la propagazione del danno in un materiale. Le ramificazioni principali infatti vengono ben caratterizzate, mentre quelle più piccole vengono disperse e non si è in grado di notarle chiaramente. L'ultima immagine proposta nella sequenza rappresenta lo stato di frattura del pannello dopo che il carico viene lasciato aumentare e le cricche espandersi per diverso tempo oltre il raggiungimento del danno massimo e si può notare ancora una volta come il pattern sia piuttosto fedele alla realtà, presentando anche i fenomeni delle cricche circonferenziali, tipicamente presenti in questo tipo di test e anch'esse visibili chiaramente in figura (3.2). Questo tipo di cricche non viene invece ottenuto nelle analisi effettuate in [7].
La scelta di richiedere in output il risultato ad ogni t=0,2s (fitti-



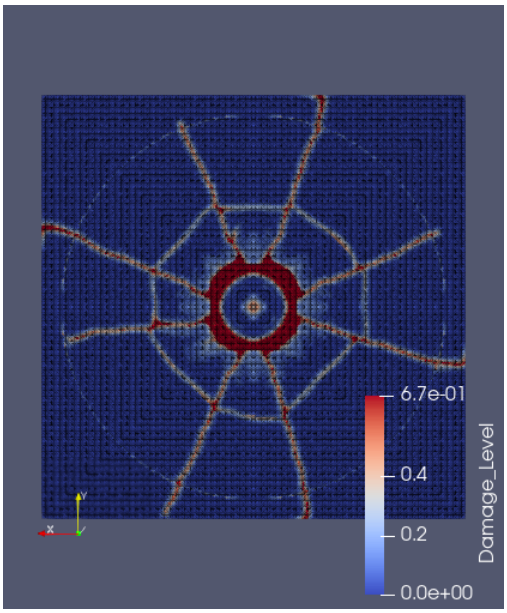
(a) Danno massimo 40%, t=11,6s



(b) Danno massimo 90%, t=16,8s



(c) Danno massimo 100%, t=33,2s

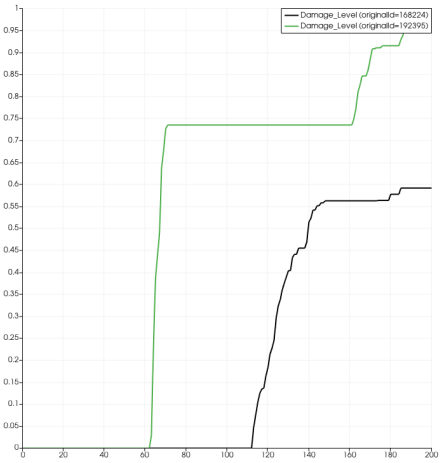


(d) Propagazione del danno dopo t=80s

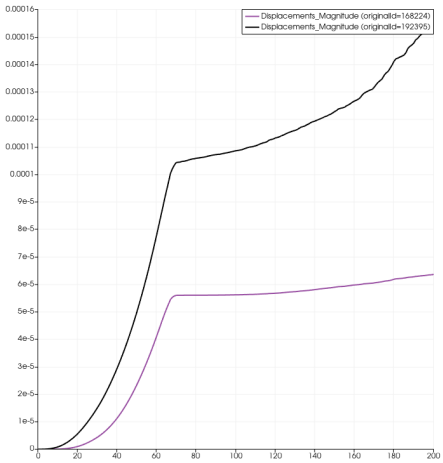
Figura 3.11: Pattern di frattura del tin side della piastra per diversi livelli di danneggiamento, $s_* = 3,301 \cdot 10^{-4}$

zi) sebbene possa sembrare un buon compromesso, in realtà non permette una buona discretizzazione del processo di rottura del materiale soprattutto nelle sue fasi iniziali, quando il livello di danneggiamento massimo cresce molto rapidamente in istanti di

tempo molto ridotti. Infatti è sufficiente osservare dalla tabella (3.4) come nell'intervallo di $\Delta t = 0,4s$ il danneggiamento massimo passi dal 3% al 40%. Per investigare in maniera più approfondita questi fenomeni di propagazione rapida del danno nelle fasi iniziali sarebbe necessario modificare il numero di iterazioni richieste. Visto l'elevato costo computazionale e di memoria di sistema che sarebbe richiesto tuttavia si preferisce non valutare questi aspetti del fenomeno, che potrebbero essere approfonditi in studi futuri.
In figura (3.12) sono allora rappresentati i valori di danno e di modulo del vettore spostamento in funzione del numero di step (ogni step equivale a 0,2 istanti temporali) per due nodi selezionati tra quelli della piastra: uno al centro (ID=192395) e uno nella zona dell'anello di carico (ID=168224). Si può vedere chiaramente che c'è una discontinuità degli spostamenti poco dopo lo step 60, equivalente a circa 12-13 istanti temporali. Ciò che accade in modo qualitativo, è che quando il nodo posto sull'anello di carico raggiunge un livello di danno di circa 70-75% probabilmente vi è l'apertura della cricca, che genera una conseguente discontinuità negli spostamenti del nodo stesso. Di conseguenza anche gli altri punti materiali della piastra subiscono una sorta di effetto di "ritorno elastico" e subiscono a loro volta un brusco spostamento. Anche il punto materiale al centro della piastra infatti presenta nello stesso istante temporale una discontinuità nello spostamento. Questo comportamento è molto in linea con quanto accade nella realtà e giustifica a pieno l'utilizzo della teoria peridinamica per questo genere di problemi.
Una piccola parentesi, infine, va aperta su come sono stati selezionati i valori di danneggiamento riportati in tabella e nelle figure. I nodi della piastra nelle zone in cui è vincolata sono costretti a rimanere fermi: questo comporta che quando la piastra si solleva



(a) Rappresentazione grafica dell'evoluzione del danno nel tempo

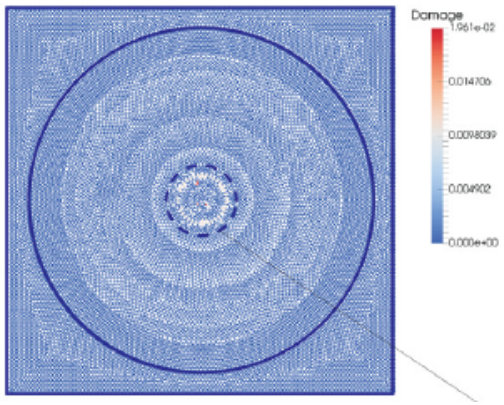


(b) Rappresentazione grafica dell'evoluzione del modulo del vettore spostamento nel tempo

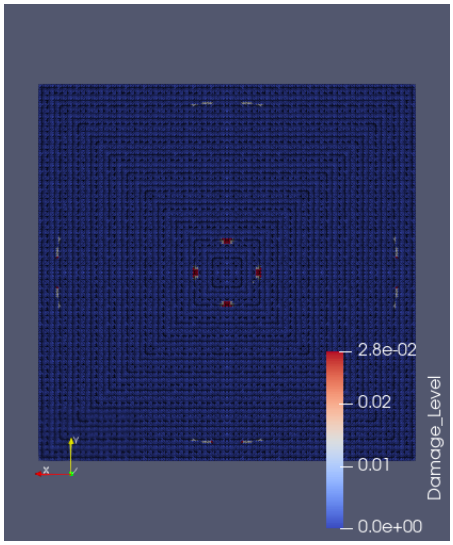
Figura 3.12: Danno e spostamento per due punti selezionati in funzione del tempo

ai bordi a causa del carico posto nella zona centrale i legami tra i nodi corrispondenti alle zone dell'anello di supporto subiscono un allungamento, andando a rompersi. Questo valore di danneggiamento non viene considerato, in quanto è esterno alla zona di interesse e in quanto tali legami non hanno alcuna influenza su ciò che accade al centro della piastra. I valori di danneggiamento riportati dunque fanno riferimento solamente alla zona centrale e non al danneggiamento di quei legami più esterni.
Questo fenomeno fornisce anche la controprova che le cause identificate della non corretta deflessione della piastra ai bordi sembrano essere dovute anche a queste condizioni di vincolo, che non consentono la totale libertà di movimento alla piastra.
Dopo aver visionato i risultati prodotti con il codice di ricerca si vuole fare un confronto con quelli pubblicati in letteratura. La piastra precedentemente presentata è dunque confrontabile con il modello di piastra caratterizzato da 6 nodi lungo lo spessore presentato in [7]. Estrapolando le figure di interesse si può riportare

un confronto tra diversi valori di danno. Il valore di s_* in realtà non è lo stesso tra i due modelli, pertanto ci si limita solamente a commentare la capacità del codice di ricerca di produrre dei risultati paragonabili a quelli prodotti in [7] in termini di pattern di frattura. Ecco che dunque si può trovare che il pattern di frattura del modello di piastra descritto in questa sezione ha delle importanti similitudini con quello del modello di piastra presentato in [7]. Come si può facilmente visionare nelle figure (3.13) e (3.14) infatti, entrambi i modelli presentano una nucleazione del danno nella zona centrale del loadring, risultato che è in accordo anche con i test sperimentali. La propagazione delle cricche che avviene nelle fasi successive poi è ben rappresentata in entrambi i modelli, con la differenza che la tendenza di simmetria del modello sviluppato con il codice di ricerca è molto evidente se confrontata con il modello di [7], nel quale come è già stato detto si fa uso di una griglia di nodi irregolare e dunque altamente asimmetrica. Nel complesso tuttavia è evidente come il codice di ricerca sia perfettamente funzionante e sia in grado di reggere il confronto con altri codici peridinamici come quello presentato in [7] e nel futuro le sue potenzialità potranno essere ampliate agendo opportunamente.
Per ultimare questo primo confronto dei pattern di frattura tra il codice di ricerca e il codice utilizzato in [7] ci si interessa brevemente all'influenza che il valore di critical bond stretch s_* ha sui pattern di propagazione del danno. A livello di carico richiesto il suo impatto è infatti sicuramente molto importante, mentre non si sa quale sia il suo effetto sulla propagazione e sulla nucleazione delle cricche. Prendendo in considerazione figura (3.15) si nota come l'effetto predominante è quello di una propagazione delle cricche inferiore a parità di danno massimo per il caso in cui il valore di s_* è maggiore. Un altro aspetto molto interessante è poi

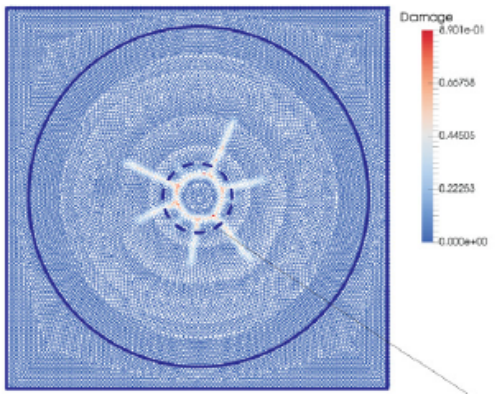


(a) Danno massimo 2% nel modello presentato in [7]

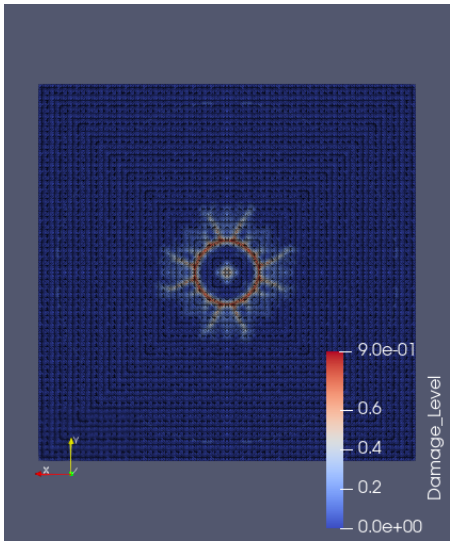


(b) Danno massimo del 3% nel modello creato con il codice di ricerca

Figura 3.13: Confronto tra modello realizzato in [7] e modello realizzato con il codice di ricerca, danno 2%



(a) Danno massimo 90% nel modello presentato in [7]



(b) Danno massimo del 90% nel modello creato con il codice di ricerca

Figura 3.14: Confronto tra modello realizzato in [7] e modello realizzato con il codice di ricerca, danno 90%

il fatto che a parità di discretizzazione spaziale del modello i due pattern siano differenti sebbene nessuno dei dati geometrici e della

discretizzazione del modello vengano cambiati.
Se rapportiamo tali risultati con quelli in figura (3.16) ottenuti tramite il codice di ricerca facendo uso di due modelli analoghi che differiscono tra loro solamente per il valore di s_*, possiamo notare che in questo caso non vi è una netta differenza tra le zone di propagazione delle cricche, mentre vi è invece una netta detta differenza dei pattern di propagazione. Questo risultato è in linea con quello presentato in [7].

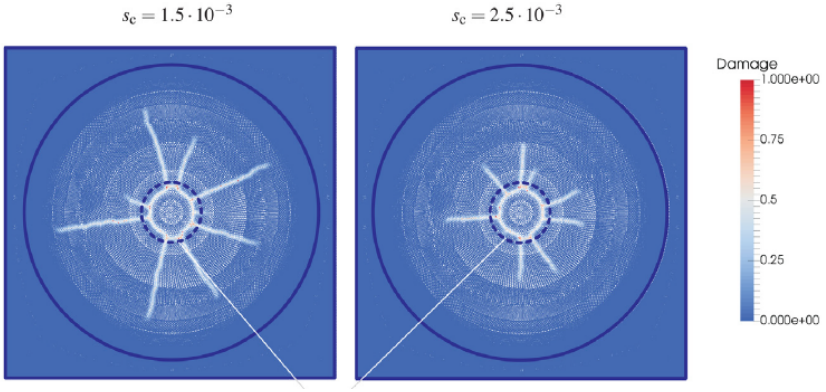
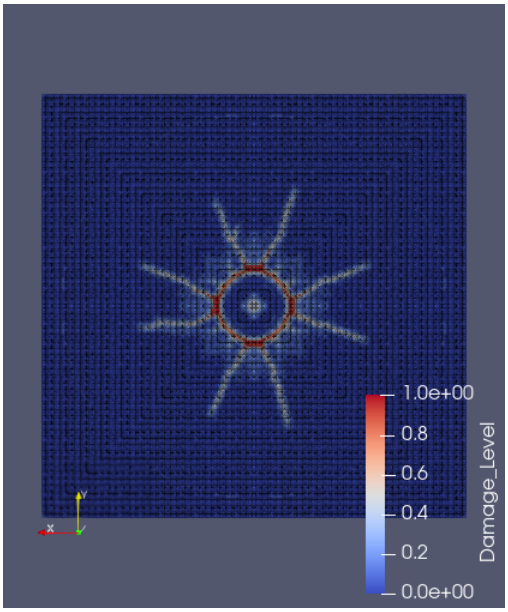


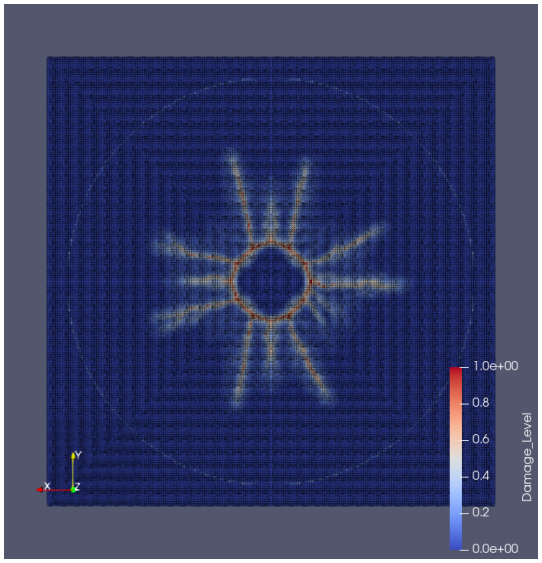
Figura 3.15: Danno massimo 100% per diversi valori di s_*, risultati presentati in [7]

3.3.3 Impatto della discretizzazione spaziale

Ci si potrebbe chiedere quanto aumentare la discretizzazione spaziale della piastra influisca sui risultati ottenuti, sul pattern di frattura e sul carico critico. Vengono dunque creati nuovi modelli della piastra, le cui caratteristiche di dimensioni rimangono analoghe a quelle precedenti, ma la loro discretizzazione spaziale viene aumentata. In maniera analoga a quanto fatto in [7] si lascia invariato il rapporto $\delta/dz = 3.015$. Anche le condizioni di vincolo e carico rimangono invariate. Ripetendo il processo di valutazione presentato nella sezione precedente, a causa del nuovo valore di δ il valore stimato di limit bond stretch varierà secondo la relazione



(a) $s_* = 3,30081 \cdot 10^{-4}$



(b) $s_* = 1 \cdot 10^{-3}$

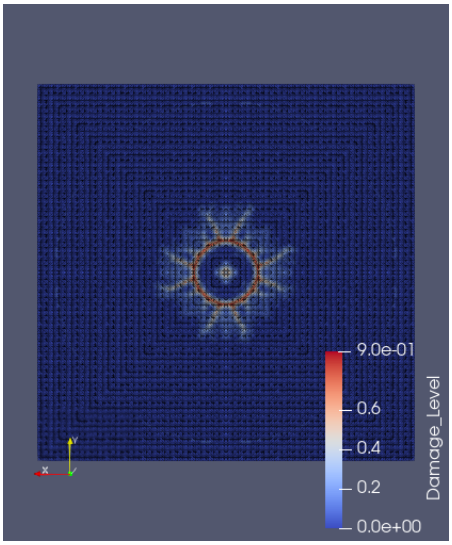
Figura 3.16: Danno massimo 100% per diversi valori di s_*, risultati ottenuti con il codice di ricerca

(1.19), aumentando. Nella tabella di seguito si riportano riassunte tutte le principali proprietà dei tre modelli di piastra che vengono confrontati tra loro.

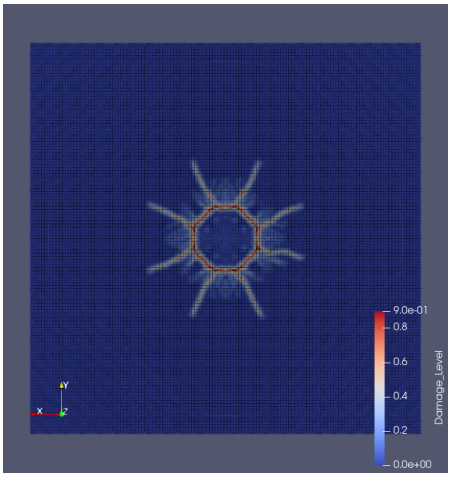
	N. Nodi	**Orizzonte δ [mm]**	**Limit bond stretch s_***
Piastra 1	207x207x6	1,45	$3,30081 \cdot 10^{-4}$
Piastra 2	242x242x7	1,24907	$3,56528 \cdot 10^{-4}$
Piastra 3	276x276x8	1,0929375	$3,811446 \cdot 10^{-4}$

Tabella 3.5: Caratteristiche dei tre modelli di piastra utilizzati per il confronto del danno

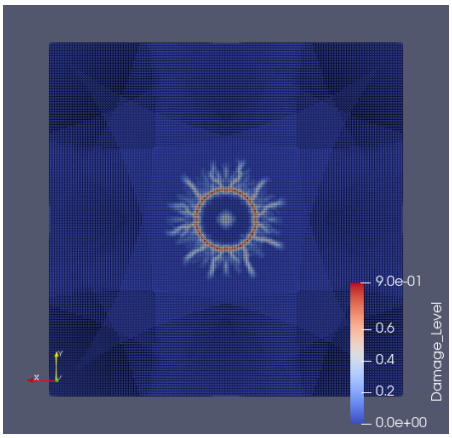
Per prima cosa si farà un breve confronto tra i pattern di frattura, mentre successivamente si cercherà un confronto tra i valori di carico ottenuti, per capire se la discretizzazione spaziale influisce su di essi. Selezionando alcuni istanti dunque, si riporta un confronto del pattern di frattura tra i tre modelli che sono stati creati. Figure (3.17) e (3.18).



(a) Piastra 1 t=16,8s



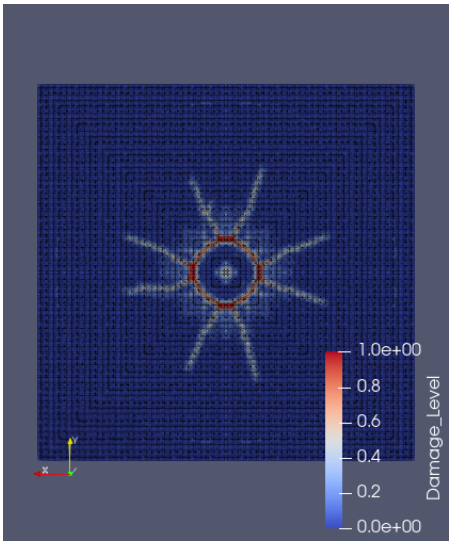
(b) Piastra 2 t=25s



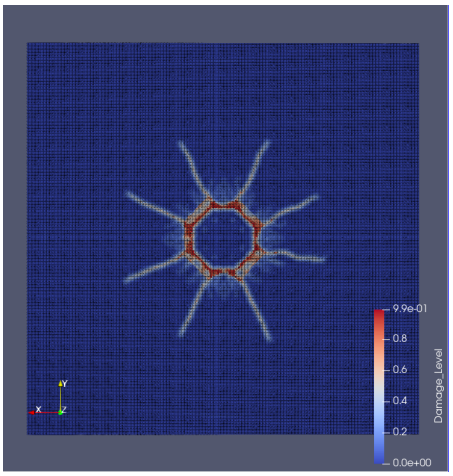
(c) Piastra 3 t=22,25s

Figura 3.17: Pattern di frattura del tin side della piastra per diverse discretizzazioni spaziali, danno 90%.

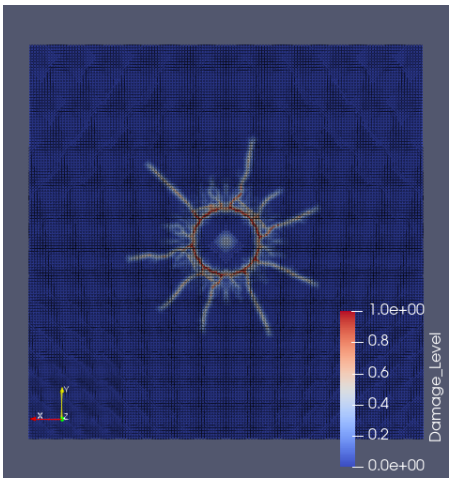
Dalle istantanee si può notare come l'aumento della discretizzazione spaziale, tradotto in un aumento del numero di nodi, vada a rendere il processo di diffusione del danno via via meglio discretizzato. Il processo di criccatura del materiale è ben descritto e a livello di dettagli si può notare come l'aumento della discretizzazione favorisca una miglior visualizzazione del percorso e del pattern delle cricche anche un po' più piccole, che con discretizzazioni più grossolane non sono rappresentate in maniera ben visibile. La



(a) Piastra 1 t=33,2s



(b) Piastra 2 t=40,8s



(c) Piastra 3 t=40s

Figura 3.18: Pattern di frattura del tin side della piastra per diverse discretizzazioni spaziali, danno 100%.

simmetria del problema infine tende a sembrare ridotta, tuttavia questo effetto potrebbe essere in parte derivato dalla localizzazione prescelta per i difetti che essendo casuale non è conoscibile a priori e un altro motivo potrebbe essere che le cricche più piccole essendo meglio discretizzate e più visibili inducano a far sembrare la soluzione meno simmetrica.
Possiamo prendere come confronto i risultati presentati in [7] per

valori di danno del 90% e differenti discretizzazioni spaziali (figura (3.19)). Se andiamo ad analizzare tali risultati e li confrontiamo con quelli in figura (3.17) che sono anch'essi presi al 90% di danno per differenti discretizzazioni, si può notare che l'effetto prevalente che si nota in [7] è che il danno è più localizzato, ovvero a parità di danno massimo le cricche sono molto meno estese facendo uso di una discretizzazione spaziale più fine. Questo effetto non è apprezzabile nei modelli di piastra creati con il codice di ricerca, però è chiaramente visibile come analogamente a figura (3.19) con un numero maggiore di nodi si ha una caratterizzazione migliore delle criccature più piccole della piastra. Anche in questo caso il confronto è puramente riguardante il pattern di frattura in maniera qualitativa, poichè come già fatto presente i risultati in termini di carico differiscono da quelli sperimentali e anche da quelli riportati in [7].

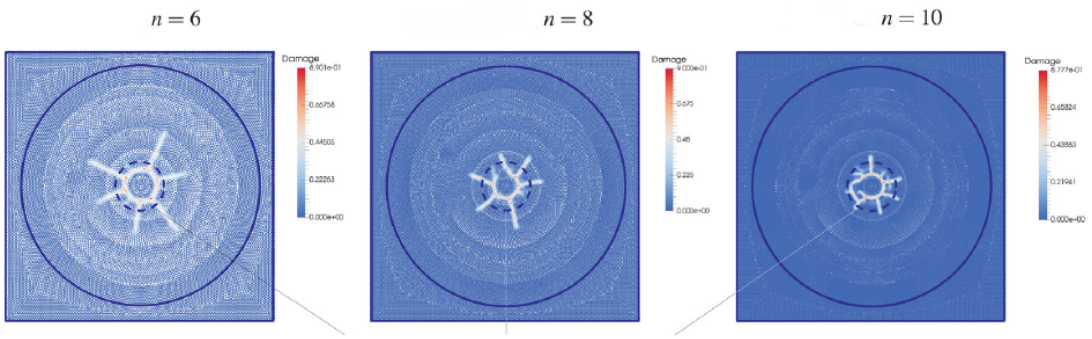


Figura 3.19: Confronto tra discretizzazioni spaziali con 6, 8 e 10 nodi sullo spessore e danno del 90%. Risultati presentati in [7]

Si torna dunque a discutere i risultati ottenuti facendo uso del codice di ricerca. Se andiamo ad analizzare i dati relativi al tempo e al carico corrispondente, notiamo che l'effetto è quello di un aumento del carico necessario per la frattura all'aumento della discretizzazione spaziale, che causa un aumento del valore del limit bond stretch s_*. Questa è solamente una tendenza, in realtà come si può

vedere dal grafico (3.20) per valori di danno fino al 70% circa il tempo di rottura è sempre maggiore per un numero maggiore di nodi, mentre successivamente per le discretizzazioni maggiori c'è una variazione del trend. Questo effetto potrebbe derivare da una gran quantità di variabili, in primis la già citata localizzazione dei difetti introdotti per rompere la simmetria, che va a modificare la distribuzione del carico tra i legami. Indicativamente comunque è interessante vedere che l'effetto prevalente è quello dell'aumento di carico richiesto a causa dell'aumento del limit bond stretch, mentre ci si potrebbe aspettare che in realtà il risultato rimanga invariato.

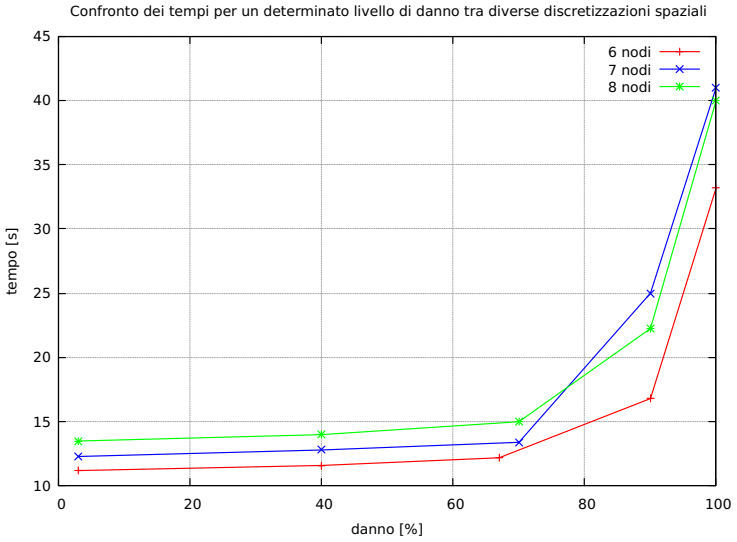


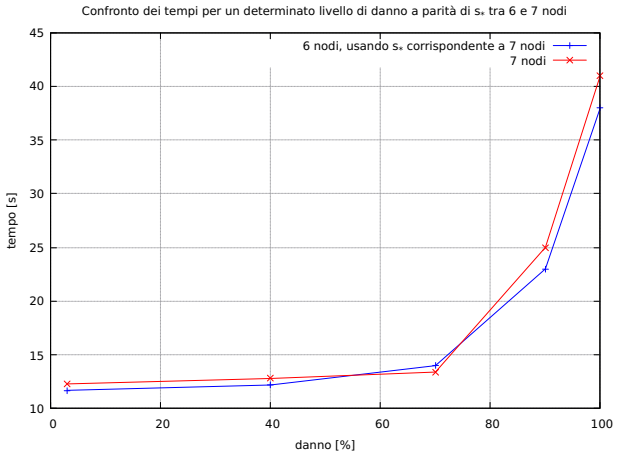
Figura 3.20: Confronto del carico richiesto per differenti discretizzazioni spaziali e conseguenti differenti valori di s_*

Ci si potrebbe poi anche chiedere però se la variazione del valore di s_* è l'unica variabile che entra in gioco e causa l'aumento di carico critico. In [7] è infatti stato notato che fissando un valore di s_* e aumentando la discretizzazione spaziale il valore di carico necessario a parità di livelli di danno cambia. Si vuole dunque verificare se ponendo il valore di limit bond stretch fisso e variando il numero

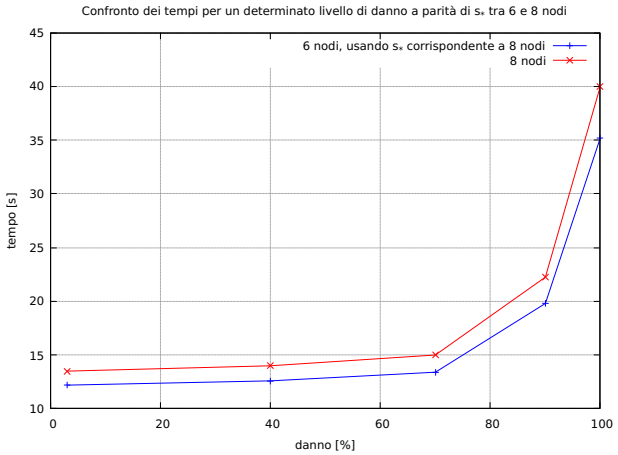
di nodi il risultato cambia.
Scegliamo quindi la discretizzazione più grossolana, con 6 nodi lungo lo spessore della piastra e andiamo ad effettuare delle simulazioni utilizzando i valori di s_* risultati dalle discretizzazioni più fini. In questo modo otteniamo un confronto diretto tra la discretizzazione con 6 nodi nello spessore e quelle rispettivamente con 7 e 8 nodi a parità di s_*. Nei grafici presentati in figura (3.21) si possono vedere i risultati. Notiamo che il carico di rottura in realtà non diminuisce a parità di critical bond stretch s_* con un maggior numero di punti materiali, ma tende ad aumentare. Questo comportamento non è in linea con quello del codice utilizzato in [7], nel quale un aumento della discretizzazione comporta ad una localizzazione del danno e dunque ad un carico richiesto inferiore. Questo risultato dovrebbe essere meglio studiato però, in quanto per semplicità di riproduzione dei risultati non si è andato a variare il range di applicazione delle condizioni di carico e vincolo, che per una discretizzazione maggiore comprendono un numero maggiore di nodi e dunque una distribuzuone più ampia del carico nella zona del loadring, favorendo l'aumento del carico richiesto per la rottura. Tale processo richiederebbe una nuova calibrazione tramite l'analisi di deflessione e potrebbe essere approfondito in studi futuri.
Chiaramente la conseguenza diretta di questo risultato, sebbene differente da quello atteso, è che in base alla discretizzazione spaziale scelta il modello andrebbe calibrato in maniera tale da restituire risultati paragonabili a quelli del test di laboratorio.

3.3.4 Calibrazione del modello

Il processo di calibrazione sopra proposto non viene però eseguito in questa tesi, in quanto non si possono mettere a confronto



(a) Confronto tra la discretizzazione spaziale con 6 e 7 nodi sullo spessore, utilizzando lo stesso valore di $s_* = 3,56528 \cdot 10^{-4}$



(b) Confronto tra la discretizzazione spaziale con 6 e 8 nodi sullo spessore, utilizzando lo stesso valore di $s_* = 3,811446 \cdot 10^{-4}$

Figura 3.21: Confronto dei tempi di danneggiamento per differenti discretizzazioni a parità di s_*

i risultati ottenuti qui con quelli di riferimento. Il motivo è che chiaramente, visti i risultati approssimati nel caso di semplice deflessione della piastra, anche i risultati di frattura sono diversi, ma soprattutto hanno anche un andamento diverso rispetto a quelli reali.

Possiamo verificare quest'ultima affermazione nel modo seguente:

estrapoliamo da [7] i valori di carico ottenuti per $s_* = 2,0 \cdot 10^{-3}$ facendo uso di 6 nodi lungo lo spessore. Dati i pochi valori direttamente reperibili si hanno a disposizione solamente due valori, corrispondenti ad un valore di danno del 2% e del 90%. Quello che si va a fare ora è utilizzare i risultati ottenuti per un modello analogo, caratterizzato da 6 nodi sullo spessore e dallo stesso valore di limit bond stretch utilizzato. Avendo a disposizione un buon numero di dati corrispondenti a diversi livelli di danneggiamento possiamo fare un fit degli stessi.

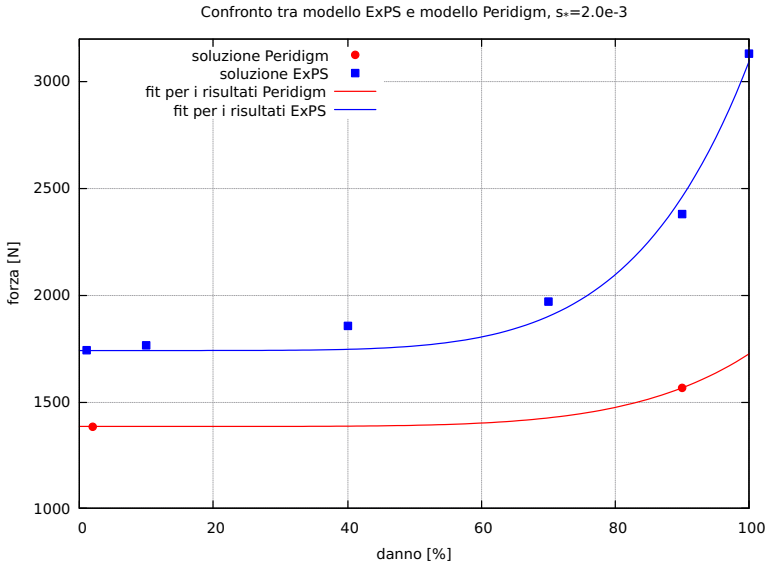


Figura 3.22: Confronto tra l'andamento dei risultati in funzione del livello di danno per il modello ExPS e il modello Peridigm di riferimento

Come si può notare in grafico (3.22) i dati ottenuti con il modello ExPS sono approssimabili in maniera abbastanza accurata con una funzione del tipo $y = C_1 \cdot x^6$. A questo punto, se l'andamento del modello Peridigm fosse lo stesso di quello ottenuto nell'ambito di questa tesi, si avrebbe che una funzione analoga del tipo $y = C_2 \cdot x^6$ ma traslata più in basso seguirebbe lo stesso andamento di quella che approssima i dati del modello ExPS per lo stesso va-

lore della costante, ovvero per $C_1 = C_2 = C$. Ciò però non accade, infatti i valori rilevati sono $C_1 = 1,35682 \cdot 10^{-9}$, mentre la costante $C_2 = 3,41054 \cdot 10^{-10}$. Per questo motivo si conclude che un processo di calibrazione di questo modello con riferimento ai risultati sperimentali non avrebbe alcun senso.
Il massimo che è possibile ottenere viste queste considerazioni è effettuare dei tentativi in modo tale che per un qualche valore di s_* i due modelli ExPS e Peridigm abbiano circa lo stesso livello di danno per circa lo stesso valore di carico. Dopo una serie di tentativi si è ricavato un valore che permette suddetto paragone, ovvero $s_* = 1.5 \cdot 10^{-3}$.

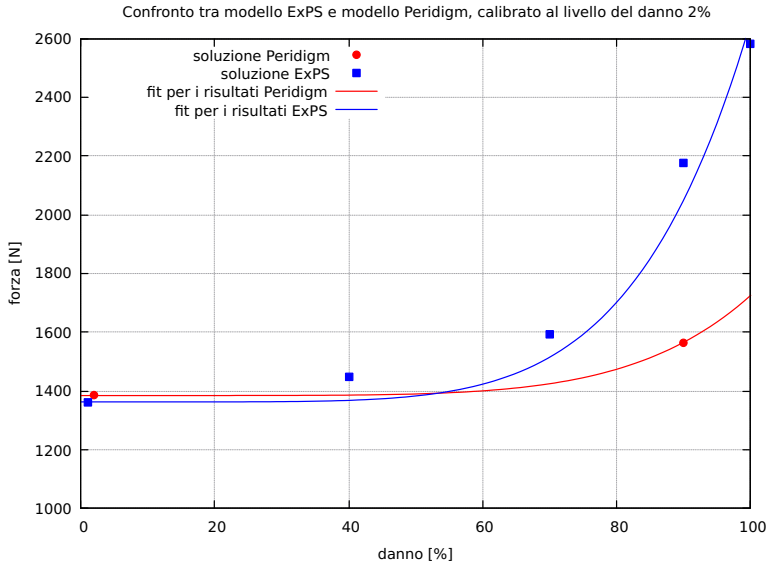


Figura 3.23: Confronto tra l'andamento dei risultati in funzione del livello di danno per il modello ExPS e il modello Peridigm, calibrato al livello di danno 2%

Dall'andamento del modello calibrato al 2% di danno in grafico (3.23) è possibile notare come l'andamento del modello ExPS non venga più così ben rappresentato da una curva del tipo $y = C_1 \cdot x^6$ ma rimane comunque distante dall'andamento del modello presentato in [7]. Questa è un ulteriore conferma del fatto che è pos-

sibile calibrare il modello in riferimento ad un singolo valore di danno o di carico, ma non è possibile una corrispondenza completa dei due. La variazione dell'andamento non va ad escludere che in realtà ci sia un qualche valore di s_* che replichi qualitativamente l'andamento del modello Peridigm, ma nel caso esistesse i valori di carico (o tempo) corrispondente non sarebbero più gli stessi, andando a complicare il problema da un differente punto di vista. La possibilità di variare il valore di critical bond stretch per ottenere dei risultati conformi a quelli di interesse può essere sfruttata per simulare dunque il pattern di frattura dell'air side della piastra per esempio, calibrandolo in maniera da ottenere il livello di danno desiderato in corrispondenza del carico desiderato.

4

Conclusioni e sviluppi futuri

Dopo aver illustrato il funzionamento del codice ExPS e aver tentato di riprodurre quanto fatto in [7], sicuramente si può affermare che sono stati fatti degli importanti passi avanti per quanto riguarda lo sviluppo e lo studio del software di ricerca. Sebbene i risultati ottenuti non siano in linea con quelli teorici o con quelli sperimentali e dunque un buon confronto tra i dati di questa tesi e quelli di riferimento risulta difficoltoso, risulta comunque evidente che il funzionamento del codice è ottimo e sono state individuate alcune migliorie che potrebbero essere apportate per renderne il funzionamento migliore in base alle necessità poste in essere da questa tesi.
In primis osservando quanto prodotto nella sezione dedicata all'analisi di deflessione la prima implementazione che potrebbe favorire al miglioramento del codice è quella legata alla possibilità di creare dei layer di nodi "esterni" alla piastra, con i quali la piastra può interagire grazie alle funzionalità di contact detection già pre-

senti all'interno di esso. In questo modo si potrebbero simulare in maniera più coerente gli anelli di supporto e di carico del double ring bending test e ottenere dei risultati migliori. In secondo luogo un'altra miglioria che sarebbe interessante implementare è legata alla possibilità di importare delle griglie di nodi sulla base di mesh agli elementi finiti, in maniera tale da poter riprodurre delle piastre caratterizzate da un reticolo irregolare, cosicchè l'analisi del danno porti con sè delle imperfezioni intrinseche che vanno a creare dei percorsi non simmetrici per le cricche e riprodurre così questi fenomeni in maniera più naturale.
Tornando ai risultati ottenuti infine, essi sono influenzati chiaramente dalle condizioni di vincolo e carico che sono poste direttamente sui nodi della piastra e dalla regolarità del reticolo, che sebbene sia possibile romperne facilmente la simmetria in realtà l'ordine complessivo dei nodi favorisce a rendere localizzate le imperfezioni introdotte, mantendendo comunque una certa tendenza di simmetria generale. Un confronto dei pattern di danno con quelli reali suggerisce la capacità del solutore di identificare le venature principali delle cricche e di produrre risultati coerenti nonostante le problematiche legate alla simmetria. Anche il comportamento del materiale in termini di discontinuità del campo degli spostamenti è ben rappresentato dal codice. Le analisi condotte per diverse discretizzazioni spaziali infine suggeriscono che la precisione nella rappresentazione dei pattern di frattura aumenta considerevolmente se si aumenta il numero di punti materiali, però allo stesso tempo vengono influenzati anche i risultati in termini di carichi di frattura, suggerendo la necessità di un processo di calibrazione dedicato in base alla discretizzazione spaziale che si sceglie di utilizzare. I valori di carichi richiesti per svariati livelli di danno infine non sono confrontabili con quelli di [7] in quanto

non caratterizzati da un andamento paragonabile e in quanto i dati stessi riportati riguardanti il test di laboratorio non sono sufficienti. L'esperienza accumulata nel corso delle analisi riportate in questa tesi suggerisce inoltre che facendo uso di un normale PC dalle medie prestazioni un numero di nodi totale superiore ai 400.000-500.000 è fortemente sconsigliato, in quanto il tempo di analisi aumenta molto, portando i tempi richiesti ben oltre le 10-12 ore per ogni singola simulazione.

Sviluppi futuri

Interessanti sviluppi futuri legati all'utilizzo di questo software possono essere lo studio di fenomeni più dinamici e istantanei come l'impatto con una massa ad alta velocità oppure addirittura lo studio di fenomeni termici. Nello specifico di questa tesi invece possono essere condotti degli studi riguardanti le fasi più "giovani" della frattura, nelle quali la sua propagazione è molto rapida e il danneggiamento del materiale cresce in modo molto veloce. Lo studio di pattern di frattura più complessi facendo aumentare il carico per numerosi istanti di tempo oltre il danno massimo è una possibilità concreta dispondendo di potenza di calcolo adeguata.
Studi riguardanti il comportamento dell'air side della piastra, che sarà caratterizzato da un valore di s_* nettamente superiore infine sono sicuramente utili e importanti per vedere il comportamento della frattura su un materiale di gran lunga più resistente, ma caratterizzato da distribuzioni di difetti molto localizzate.

Bibliografia

[1] Fabio Biondani. «Sviluppo di tecniche per l'analisi peridinamica della frattura duttile in problemi termomeccanici fortemente accoppiati». Tesi di laurea mag. Politecnico di Milano, 2018.

[2] F. Bobaru et al. *Handbook of Peridynamic Modeling*. Taylor & Francis Group, 2017.

[3] J. Foster, S.A. Silling e W. Chen. «An energy based failure criterion for use with peridynamic states». In: *Internationa Journal for Multiscale Computational Engineering* 9 (2011), pp. 675–688.

[4] B. Kilic e E. Madenci. «An adaptive dynamic relaxation method for quasi-static simulations using the peridynamic theory». In: *Theoretical and Applied Fracture Mechanics* 53 (2010).

[5] M.H. Krohn et al. «Biaxial flexure strenght and dynamic fatigue of soda-lime-silica float glass». In: *Journal of American Ceramic Society* 85 (2002).

[6] E. Madenci e E. Oterkus. *Peridynamic theory and Its Applications*. Springer Science, 2014.

[7] K. Naumenko, M. Pander e M. Würkner. «Damage patterns in float glass plates: Experiments and peridynamics analy-

sis». In: *Journal of Theoretical and Applied Fracture Mechanics* 118 (2022).

[8] Pablo Seleson. «Improved one-point quadrature algorithms for two-dimensional peridynamic models based on analytical calculations». In: *Computer Methods in Applied Mechanics and Engineering* 282 (2014).

[9] S.A Silling et al. «Peridynamic States and Constitutive Modeling». In: *Journal of Elasticity* (2007).

[10] S.A. Silling e E. Askari. «A meshfree method based on the peridynamic model of solid mechanics». In: *Computer and Structures* 83 (2005).

[11] P. Underwood. «Dynamic Relaxation». In: *Computational methods for Transient Analysis* 1 (1983).

A

Imposizione di Vincoli e Carichi per l'analisi di deflessione

In questa prima appendice si andrà a speigare come vengono applicati i carichi e i vincoli utilizzati per l'analisi di deflessione della piastra agendo direttamente sul codice.

A.1 Condizioni di Carico

Le condizioni di carico vengono applicate all'interno del modulo **loads.f90**. Si scorre all'interno del modulo fino all'inizio della subroutine chiamata **PRD_LDS_ApplyLoads**. All'interno di questa subroutine possono essere poste le condizioni di carico. Di seguito si riporta quello che va scritto:

```
cnt = 0.0_rk
!
```

```
do i=1,N_P
   !
   ldrng=(prd_rps_x(i)-0.05_rk)**2.0_rk + (prd_rps_y(i)-0.05_rk)**2.0
   _rk
      !
      if (ldrng<0.000082_rk .and. ldrng>0.000074_rk) then
      !
      if (prd_rps_z(i)>0.0009_rk) then
      !
      cnt = cnt + 1.0_rk
      !
      endif
      !
      endif
   !
enddo
!
do i=1,N_P
   !
   ldrng=(prd_rps_x(i)-0.05_rk)**2.0_rk + (prd_rps_y(i)-0.05_rk)**2.0
   _rk
      !
      if (ldrng<0.000082_rk .and. ldrng>0.000074_rk) then
      !
      if (prd_rps_z(i)>0.0009_rk) then
      !
      prd_ext_frc_z(i) = -1662.0_rk / (cnt * prd_prt_vlm(i))
      !
      endif
      !
      endif
   !
enddo
```

Codice A.1: Applicazione delle condizoni di carico

La forza esterna come gia detto è definita come una forza su unità di volume. Per mantenere la correttezza a livello di unità di misura dunque non è sufficiente definire il carico come è stato fatto per il modello agli elementi finiti. Il valore "prd_ext_frc_z(i)" corrisponde alla forza per unità di volume che viene applicata ad ogni

nodo. E' dunque necessario dividere i $1662N$ per il volume dei punti materiali. Va prestata anche attenzione al fatto che la forza non viene applicata ad un singolo nodo, ma ad un insieme di N nodi e bisogna dunque dividere la forza anche per il numero di questi nodi. La formula permette di rendere il ragionamento più chiaro:

$$F_{tot} = 1662N = F_{vol} \cdot vlm \cdot N \longrightarrow F_{vol} = \frac{1662}{vlm \cdot N}$$

Il volume dei nodi è noto dalla geometria della griglia, mentre per ottenere il numero di nodi si è fatto uso di un contatore, visibile nella prima parte del codice sopra riportato.

A.2 Condizioni di Vincolo

Le condizioni di vincolo sono accessibili attraverso il modulo chiamato **bcs.f90**. All'interno del modulo si scorre fino all'inizio della subroutine chiamata **PRD_BCS_ApplyBCs**. Qui possiamo definire le condizioni di vincolo. Di seguito si riporta il codice:

```
do i=1,N_P
   !
   crf=(prd_rps_x(i)-0.05_rk)**2.0_rk + (prd_rps_y(i)-0.05_rk)**2.0
   _rk
      !
      if (crf<0.002060_rk .and. crf>0.001990_rk) then
      !
      if (prd_rps_z(i)<0.0020_rk) then
      !
      prd_pos_z(i) = prd_rps_z(i)
      !
      endif
      !
      endif
   !
```

```
15 enddo
```

Codice A.2: Applicazione delle condizoni di vincolo

Dove si definisce che la posizione aggiornata dei punti di applicazione del vincolo rimanga uguale a quella di riferimento, che tradotto significa semplicemente che essi non si possono muovere.

B

Modifica del codice per l'implementazione dei carichi tempo varianti

In questa seconda appendice viene spiegato come andare a modificare il codice in maneira tale da poter applicare dei carichi tempo-varianti. L'applicazione di condizioni tempo-varianti non è così diretta come può sembrare. Innanzitutto il codice applica le condizioni di carico solamente nella sua fase di iniziazione, dunque definendo un carico proporzionale al tempo si andrebbe semplicemente a moltiplicare per un valore costante, che all'inizio della simulazione è 0.

B.1 Modifica del codice per implementare i carichi tempo-varianti

Per ovviare a questo problema è necessario addentrarsi tra le righe del codice. Si accede al modulo **main.f90**, dove in base base alle impostazioni che scegliamo per il solutore vengono attivate diverse parti del codice con le chiamate a tutti gli altri moduli presenti. Dopo essere scesi fino alla parte di codice in cui viene attivato lo schema Verlet ADR, alla riga 185, lo modifichiamo aggiungendo la chiamata al modulo nel quale abbiamo definito le condizioni di vincolo, in maniera tale che ad ogni passo di Dynamic Relaxation le condizioni di carico vengano prese dal modulo dedicato, anzichè solamente nella fase di inizializzazione.

```
1   do while(it<it_max)
2   !
3   it=it+1
4   !
5   ! update loads or BCs here
6   !
7   IOchk=mod(it,it_out)==0
8      !
9      adr_rke=inft_rk
10     !
11     adr_it=0
12     !
13     call PRD_MSC_UpdtPartSts
14     call PRD_MSC_UpdtBondSts
15     !
16     call PRD_MPI_IBD_SndrcvOneIk1(prd_prt_sts)
17     !
18     do while((adr_rke>adr_rke_max).and.(adr_it<adr_it_max))
19       !
20       adr_it=adr_it+1
21       !
22       call PRD_LDS_ApplyLoads
23       !
24       MPI_time_1=MPI_WTIME()
```

```
25          !
26      !
27      ![codice continua...]
```

Codice B.1: Modifica per l'applicazione di carichi tempo-varianti

Quando selezioniamo lo schema ADR, il solutore come già accennato ci permette di scegliere il numero massimo di passi di Dynamic Relaxation "adr_it_max" che ad ogni iterazione lo schema ADR compie. Dopo che il solutore compie un passo di Dynamic Relaxation, illustrato nel capitolo (2) procede a quello successivo facendo salire il contatore "adr_it" di 1. In questo momento, all'inizo del passo successivo "adr_it+1" le condizioni di carico vengono aggiornate ogni volta.

B.2 Definizione dei Carichi tempo-varianti

Dopo aver applicato questa modifica il passo successivo è quello di definire i carichi tempo varianti. Tale implementazione potrebbe essere fatta semplicemente andando a moltiplicare il carico di interesse per il tempo "prd_time", aggiornato ad ogni passo dello schema Verlet ADR. Tuttavia questo approccio non risulta conveniente. Il motivo principale è quello che il time step non è un numero comodo per il nostro scopo, in quanto spesso ha numerose cifre dopo la virgola e anche perchè in realtà il tempo è fittizio, dunque non avendo un significato fisico risulta solo più complicato definire i carichi senza andare a trarne alcun vantaggio. La soluzione scelta è più elegante. Per applicare i carichi si torna ancora una volta nel modulo **loads.f90** all'interno della subroutine PRD_LDS_ApplyLoads. Di seguito si riporta il codice, dove il contatore non viene riportato in quanto uguale a quello già utilizzato per il carico statico.

```
do i=1,N_P
   !
   ldrng=(prd_rps_x(i)-0.05_rk)**2.0_rk + (prd_rps_y(i)-0.05_rk)**2.0
   _rk
      !
      if (ldrng<0.000082_rk .and. ldrng>0.000074_rk) then
      !
      if (prd_rps_z(i)>0.0009_rk) then
      !
      prd_ext_frc_z(i) = -29.0_rk / (cnt * prd_prt_vlm(i)) * (
   prd_time/prd_dt) * (0.01_rk)
      !
      endif
      !
      endif
   !
enddo
```

Codice B.2: Applicazione delle condizoni di carico tempo-varianti

In questo modo, il rapporto tra il tempo totale e il time-step definisce il numero di iterazioni, secondo la relazione:

$$prd\_time = it \cdot prd\_dt$$

Il numero totale di iterazioni complessivo è quello dato dal numero di passi di Dynamic Relaxation che si sceglie, per il numero di iterazioni vere e proprie "it_max". In questo modo, moltiplicando per un fattore a scelta è possibile regolare il carico, suddividendolo in più step, come se variasse nel tempo. In questo caso è stato scelto un fattore moltiplicativo di $m = 0.01$, in maniera tale che ogni 100 iterazioni complessive il carico aumentasse di $29N$. A questo punto se si seleziona il numero di passi "adr_it" in maniera appropriata, si avrà che ad ogni iterata di output il carico aumenta di 29N, o di quanto si è interessati ad ottenere, in base all'output richiesto.
Questo approccio va in pratica a simulare il carico tempo-variante suddividendolo in 100 load-step. L'unica accortezza per il suo fun-

zionamento è impostare un valore di resiudo tale che il solutore non si fermi prima, andando così a far perdere traccia del carico effettivo che è stato applicato. Nella realtà questa implementazione presenta un piccolo errore, in quanto al primo passo della prima iterazione il carico risulta nullo dunque lo schema va subito a convergenza facendo perdere una iterazione. L'effetto è quello che ad ogni iterata di output il carico in realtà è maggiore di $29/100 = 0.29N$ rispetto a quello previsto.
La simulazione ha un costo computazionale molto elevato: si potrebbe suddividere il carico in più load step, ma una run di prova ha dimostrato che raddoppiando il numero di load step il risultato è rimasto lo stesso, dunque è stato ritenuto accettabile la suddivisione in 100 load step per non avere un costo computazionale eccessivo.

C

Introduzione di difetti per la rottura della simmetria del problema

In questa terza appendice viene spiegato come introdurre dei difetti nella piastra in maniera tale da rompere la simmetria del problema. L'approccio scelto è stato quello di andare a introdurre dei legami rotti all'interno della famiglia di qualche punto materiale della piastra. Questo approccio è stato scelto per la sua semplicità, in quanto è implementabile in pochi istanti e non richiede modifiche sostanziali al codice. Un approccio più mirato come quello di introdurre dei legami indeboliti random nella piastra, tramite la variazione del valore di limit bond stretch sarebbe stato molto complesso da implementare e avrebbe richiesto molto lavoro.
Si procede dunque ad applicare questa condizione dal modulo delle condizioni di carico **loads.f90**. Per prima cosa si vuole conoscere quali sono i punti materiali che stanno nella zona che vogliamo

indebolire. Lasciando pure tutte le condizioni di caric che abbiamo definito precedentemente ancora applicate, andiamo a scrivere sotto di esse le seguenti righe:

```
do i=1,N_P
!
crf=(prd_rps_x(i)-0.05_rk)**2.0_rk + (prd_rps_y(i)-0.05_rk)**2.0
_rk
!
if (crf<0.000082_rk .and. crf>0.000074_rk) then
!
if (prd_rps_z(i)>0.0009_rk) then
!
print*,prd_glb_idx(i)
!
endif
!
endif
!
enddo
```

Codice C.1: Ricerca dei legami da rompere

Quello che andiamo a fare è sostanzialmente definire una zona tramite uno o più cicli *if*, in questo caso quella dell'anello di carico stesso, e chiediamo che nel terminale venga stampato l'identificativo univoco di tale punto.
A questo punto, si va a far partire il solutore senza curarsi di come è stato effettuato il setting e si va a vedere nella finestra del terminale quel che viene stampato. Se si scorre si osservano una serie di numeri incolonnati: sono gli identificativi dei punti materiali che sono stati richiesti tramite i cicli *if*. Ora che si conoscono si entra nuovamente nel modulo **loads.f90**, si commentano le righe precedentemente inserite e si va a inserire una rottura del legame nella famiglia dei punti materiali che selezioniamo a caso tra la lista stampata. Ecco come fare:

```
do i=1,N_P
!
```

```
3    !crf=(prd_rps_x(i)-0.05_rk)**2.0_rk + (prd_rps_y(i)-0.05_rk)**2.0
     _rk
4    !
5    !if (crf<0.000082_rk .and. crf>0.000074_rk) then
6    !
7    !if (prd_rps_z(i)>0.0009_rk) then
8    !
9    !print*,prd_glb_idx(i)
10   !
11   !endif
12   !
13   !endif
14   !
15   if (prd_glb_idx(i)==150873) prd_bnd_sts(10,i)=0_ik1
16   if (prd_glb_idx(i)==150899) prd_bnd_sts(7,i)=0_ik1
17   if (prd_glb_idx(i)==111127) prd_bnd_sts(13,i)=0_ik1
18   if (prd_glb_idx(i)==148451) prd_bnd_sts(5,i)=0_ik1
19   if (prd_glb_idx(i)==137178) prd_bnd_sts(2,i)=0_ik1
20   if (prd_glb_idx(i)==164544) prd_bnd_sts(1,i)=0_ik1
21   if (prd_glb_idx(i)==129971) prd_bnd_sts(17,i)=0_ik1
22   if (prd_glb_idx(i)==98752) prd_bnd_sts(11,i)=0_ik1
23   if (prd_glb_idx(i)==149620) prd_bnd_sts(17,i)=0_ik1
24   if (prd_glb_idx(i)==150936) prd_bnd_sts(11,i)=0_ik1
25   !
26   enddo
```

Codice C.2: Rottura dei legami

Quello che viene fatto sostanzialmente è: per quel dato punto materiale *i*-esimo identificato dal suo *prd_glb_idx(i)*, si seleziona il legame *n* della sua famiglia, e lo si pone come rotto. La variabile *prd_bnd_sts(n,i)* è sostanzialmente una flag, che se uguale a 1 indica che il legame è intatto e se uguale a 0 il legame è rotto e dunque non viene computato per il calcolo dell'internal force density.
Questa operazione è stata ripetuta selezionando un po' di legami nella zona dell'anello di carico e in quella immediatamente esterna ad esso, in maniera tale da rompere la simmetria del problema.
In realtà basterebbe anche un solo legame per rompere la suddetta simmetria, ma dopo un paio di run di prova si è notato che sebbene

la simmetria fosse rotta comunque il risultato era molto tendente al simmetrico. Si è dunque cercato un compromesso che permettesse al problema di essere abbastanza "distante" dalla condizione di simmetria ma che allo stesso tempo non andasse a falsare in maniera eccessiva i risultati in termini di carico di rottura.

D

Descrizione della struttura del codice e modifiche minori

In questa ultima appendice ci si propone di spiegare brevememnte come è strutturato il codice, i diversi moduli presenti e alcune modifiche minori che sono state utilizzate per svolegere alcune delle analisi riportate in questa tesi. In questo modo è facilmente possibile andare a trovare le parti di codice che si desidera modificare e/o verificare.
Per prima cosa il codice è suddiviso in tre cartelle principali, la prima contiene i moduli che abbiamo definito come esterni al solutore vero e proprio, che permettono la creazione della geometria della piastra, della griglia di nodi, delle famiglie e di tutti quei parametri che sono legati alla geometria del corpo. Questa cartella è nominata **InitiExPS**. Successivamente vi è un'altra cartella dentro la quale vi è il solutore vero e proprio. Qui troviamo tutti i moduli

dedicati alla soluzione del problema peridinamico e anche il file di testo nel quale inserire i dati relativi al setting del solutore, come le caratteristiche meccaniche del materiale, gli algoritmi di risoluzione, i modelli costitutivi e tanto altro. Questa seconda cartella è denominata **WrkDir**. L'ultima cartella invece è dedicata solamente al post-processing dei risultati ottenuti. Il solutore infatti restituisce in output i risultati sotto forma di file in codice binario. Tramite i moduli presenti in questa cartella è possibile prendere tali file e trasformarli in file **.vtk**, accessibili tramite paraview, così da renderli visualizzabili. Questa cartella è denominata **PostExPS**. Per ognuna di queste cartelle infine dopo aver effettuato il setting e le modifiche di cui si ha bisogno, è necessario digitare sul terminale il comando *make clean*, per "pulire" le cartelle da tutti i file secondari creati durante la run precedente e che non sono necessari al funzionamento. Dopodichè si digita il comando *make*, in maniera tale da aggiornare tutti i moduli e creare i file eseguibili. Eseguendo questi file infine il codice esegue quello per cui è stato programmato.

D.1 InitiExPS

Innanzitutto la prima cosa che va fatta è creare all'interno di questa cartella una cartella vuota chiamata **DATA**. Qui verranno conservati tutti i file contenenti le informazioni riguardanti la geometria della piastra che verrà creata. Troviamo poi all'interno il file **opt.dat**. Tramite questo file è possibile selezionare il valore dell'orizzonte peridinamico nella forma $m = \delta/dx$. Vi è poi una cartella denominata **SRC**. Al suo interno vi sono i moduli di creazione della geometria:

- **init.f90**: Qui si può creare la geometria della piastra, sce-

gliendo dimensioni, numero di nodi e tanto altro. Oltre alle piastre semplici e regolari si possono creare piastre forate, pre-fratturate e altro.

- **main.f90**: Qui si sceglie quale delle diverse piastre che abbiamo impostato precedentemente va creata. E' sufficiente commentare le chiamate alle diverse subroutines di creazione lasciando scommentata solo quella di interesse

- **io.f90**: Qui vi sono tutte le routine che vanno a prendere i dati in output e a scriverli sui file binari che verranno immagazzinati in una cartella chiamata DATA. Non va modificato nulla in questo file.

- **common.f90**: Qui sono definite tutte le variabili che i vari moduli utilizzano nel corso della run del codice. Non va modificato per non alterare il funzionamento dello stesso.

La creazione della geometria può infine venire controllata e verificata tramite uno script Python, così da verificare che la geometria sia corrispondente a quella che si vorrebbe creare, senza dover fare una run di prova a andare a convertire i risultati per visualizzarli in paraview.

D.2 WrkDir

In questa cartella come già detto è contenuto il solutore vero e proprio. Per il funzionamento del solutore è necessario fare una copia della cartella **DATA** presente in **InitiExPS** all'interno di questa cartella. Qui possiamo poi trovare il file **opt.dat** nel quale come già accennato è possibile inserire i parametri relativi a caratteristiche del materiale, setting del solutore e alcune impostazioni avanzate come il rilevamento del danno, il funzionamento in parallelo del codice ed altro. Successivamente c'è la cartella denominata **SRC**: al suo interno troviamo tutti i moduli che compongono il solutore vero e proprio.

- **loads.f90** e **bcs.f90**: Questi sono i moduli dedicati all'applicazione delle condizioni di vincolo e carico e che sono già stati discussi nel dettaglio nelle precedenti appendici.
- **mat.f90**: Qui sono calcolate tutte le proprietà del materiale che non sono state inserite tramite le varie formule esistenti, oltre a quantità specifiche della teoria peridinamica, come il limit bond stretch s_* ed altro. E' proprio da questo modulo che si è intervenuti per modificare il valore di s_* per eseguire i diversi confronti.
- **sbd.f90**: Qui sono contenute una serie di routine specifiche per la formulazione state-based della peridinamica, generalmente non è consigliato intervenire modificando questo modulo
- **scaling.f90**: Qui sono contenuti una serie di fattori di scala per le diverse quantità calcolate e/o utilizzate dal solutore. Può capitare che in paraview i carichi applicati non corrispondano a quelli attesi, questo a causa di fattori moltiplicativi che vengono utilizzati per garantire stabilità agli algoritmi temporali. Modificarli non sempre ha portato a buoni risultati, la raccomandazione è tenere traccia del valore di questi fattori correttivi per poi riscalare i valori di cui si è interessati direttamente da paraview usando le sue funzioni.
- **misc.90**: Qui sono contenute una serie di routine varie, principalmente utilizzate per aggiornare tra un passo e l'altro i valori delle diverse variabili come la posizione, la velocità o lo stato di danneggiamento di un legame. Per qualche interesse specifico si può intervenire sul codice, prestando molta attenzione.
- **init.f90**: Qui sono contenute le routine che permettono l'iniziazione del codice e che caratterizzano l'interfaccia del terminale, stampando le varie informazioni come il numero di iterazioni, il tempoo corrente, ma anche le varie proprietà inserite e i settaggi scelti per il solutore.
- **solver.f90**: Qui sono contenuti i diversi schemi di integrazione temporale e gli algoritmi per il calcolo del RHS, dell'internal forcing, del damping e altro.
- **main.f90**. Qui è gestito il solutore nel suo complesso: vengono effettuate le chiamate alle varie routine e ai vari moduli

in funzione del setting prescelto e viene fatta avanzare la simulazione da un passo a quello successivo. Intervenire su questo modulo è molto delicato e ne è stato discusso nelle appendici dedicate.

- **mpi.f90**: Qui sono riportate tutte le routine dedicate al funzionamento in parallelo del codice. E' fortemente sconsigliato modificarne il contenuto.
- **io.f90**: Qui, analogamente a quanto accadeva nella cartella InitiExPS, vengono gestiti i file di input della geometria e creati i file di output con i risultati della simulazione.
- **types.f90**: Qui vengono assegnati le tipologie delle varie variabili (integer, real, ...) e non si deve intervenire per non creare malfunzionamenti imprevisti nel codice.
- **common.f90**: Qui, come in tutti gli altri moduli presenti in questa cartella che cominciano per "common" sono definite tutte le variabili e le flag che i diversi moduli utilizzano durante la run. Questi moduli non vanno modificati, a meno che non si intenda aggiungere delle variabili.

D.3 PostExPS

In quest'ultima cartella sono contenuti un file **opt.dat**, che è necessario impostare in maniera tale che il numero di iterazioni si corrispondente a quelle scelte in output dal solutore, in maniera tale che tutti i file binari prodotti vengano convertiti, e una cartella **SRC** che contiene i vari moduli dedicati alla conversione dei suddetti file binari. Non essendo di interesse per la simulazione non è necessario riportarne il contenuto ed il funzionamento. Dopo aver fatto la conversione, nella cartella **DATA** presente in **WrkDir** si troveranno, oltre a tutti i file binari prodotti dalla simulazione, anche i file **.vtk** pronti per essere caricati in paraview e visualizzati.

Ringraziamenti

Vorrei dedicare questo spazio a chi, con dedizione e pazienza, ha contribuito alla realizzazione di questo elaborato.

Un ringraziamento particolare va al mio relatore, il Prof. Ugo Galvanetto che mi ha seguito, con la sua infinita disponibilità, attraverso ogni step della realizzazione dell'elaborato, non mancando mai di darmi preziosi consigli.

Grazie anche al mio correlatore, il Prof. Mirco Zaccariotto, per avermi saputo consigliare il materiale bibliografico necessario.

Grazie infine al Dr. Federico Dalla Barba, che con generosità e pazienza ha saputo aiutarmi e consigliarmi. Senza di lui nulla di tutto questo sarebbe stato possibile.