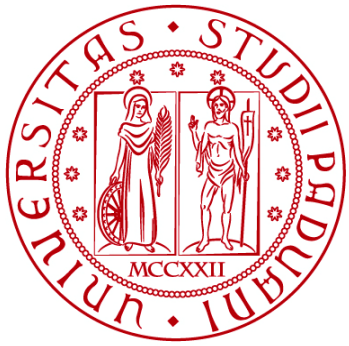
UNIVERSITÀ DEGLI STUDI DI PADOVA
DIPARTIMENTO DI INGEGNERIA CIVILE, EDILE E AMBIENTALE
Department Of Civil, Environmental and Architectural Engineering

Corso di Laurea in Ingegneria Civile



TESI DI LAUREA

IMPLEMENTAZIONE DEL METODO GMRES PRECONDIZIONATO PER LA SOLUZIONE DI SISTEMI LINEARI E APPLICAZIONE ALLA MECCANICA DI MEZZI FRATTURATI

Relatore: **Chiar.mo PROF. MASSIMILIANO FERRONATO**

Laureando: **GIULIA RUBINI**

ANNO ACCADEMICO 2021-2022

Sommario

Introduzione

La risoluzione dei sistemi lineari è un problema molto studiato, in quanto equazioni del tipo

$$Ax = b \tag{1}$$

scaturiscono dalla modellazione di un gran numero di problemi ingegneristici e di fisica matematica. Nella scrittura (1) x è il vettore delle incognite del problema, b è il vettore dei termini noti, solitamente determinato dalle condizioni al contorno o dalle forzanti del problema, e A è la matrice i cui coefficienti contengono le informazioni necessarie a mettere in relazione le incognite del problema con i termini noti. In particolare, ogniqualvolta un problema viene modellizzato utilizzando equazioni differenziali, la sua approssimazione numerica produce la soluzione di una sequenza di sistemi lineari del tipo (1).

I metodi per risolvere i sistemi lineari sono svariati e si possono dividere in due grandi categorie. La prima categoria è quella dei metodi diretti, che costruiscono la soluzione x mediante un numero finito, potenzialmente molto grande, di operazioni elementari. La seconda categoria invece è quella dei metodi iterativi, che si basano sulla creazione di una successione di approssimazioni che, sotto determinate condizioni, converge alla soluzione del sistema.

Tra i più importanti metodi diretti ci sono il metodo dell'eliminazione di Gauss e la fattorizzazione LDU. Questi metodi sono stati sviluppati per primi e sono particolarmente efficaci per la soluzione di sistemi densi e di piccole dimensioni.

Con la crescente disponibilità a basso prezzo di importanti risorse computazionali, che stimola la formulazione di problemi discreti sempre più grandi, si è reso necessario sviluppare strumenti matematici in grado di stare al passo con la dimensione crescente dei sistemi lineari ottenuti. Essendo il costo computazionale dei metodi diretti proibitivo per la risoluzione di sistemi di grandi dimensioni, si privilegia in genere l'utilizzo dei metodi iterativi.

I vantaggi dei metodi iterativi sono particolarmente accentuati per la risoluzione di sistemi sparsi, ovvero con un'alta percentuale di entrate nulle rispetto a quelle non nulle. Infatti, mentre nei metodi diretti il numero di non zeri può aumentare a seguito delle operazioni effettuate, nei metodi iterativi le matrici vengono

utilizzate solamente per le operazioni di prodotto matrice-vettore, per le quali servono solamente i coefficienti non nulli. Le matrici sparse possono essere memorizzate sotto forma di tre vettori che contengono i valori delle entrate non nulle e informazioni sul loro indice di riga e di colonna; in tal modo anche la memoria necessaria per la memorizzazione di una matrice diventa drasticamente minore.

Le applicazioni ingegneristiche di questi metodi sono ovviamente numerosissime. In questo lavoro di tesi si considera il problema della meccanica del contatto, dalla cui modellazione agli elementi finiti scaturiscono matrici a blocchi di grandi dimensioni, indefinite e fortemente sparse.

In questo lavoro verrà introdotto e illustrato l'algoritmo di uno dei metodi iterativi più diffusi per la soluzione di sistemi con matrici generiche, il GMRES, e ne verrà mostrata un'applicazione ad un problema di meccanica del contatto, fornendo i risultati ottenuti e una loro interpretazione.

1. Metodi iterativi e GMRES

1.1. Metodi iterativi

Un metodo iterativo si basa sulla costruzione di una successione il cui limite tenda alla soluzione cercata:

$$\lim_{m\to\infty} x_m = x \tag{1.1}$$

dove x è la soluzione esatta del sistema $Ax = b$. La successione è costruita partendo da una soluzione iniziale x_0, e ad ogni passo viene calcolata la nuova soluzione x_m sulla base dei dati del problema e utilizzando le approssimazioni precedenti. Un metodo iterativo giunge teoricamente alla soluzione esatta in un numero infinito di passi, a meno che si verifichi quello che viene detto un "lucky breakdown"; pertanto la successione viene arrestata quando la soluzione corrente è ritenuta abbastanza buona, ovvero quando l'errore stimato scende sotto una certa tolleranza imposta.

Esistono differenti tipi di metodi iterativi. Una prima categoria è quella dei metodi iterativi stazionari, che creano una successione del tipo

$$x_{k+1} = Ex_k + q \tag{1.2}$$

dove E è una matrice detta di iterazione e q un vettore dipendente dal termine noto del sistema da risolvere. Di questa categoria di metodi fanno parte il metodo di Jacobi e il metodo di Seidel, che si distinguono per la diversa matrice di iterazione scelta.

Un'altra categoria di metodi iterativi è quella dei metodi del gradiente. Questi sono metodi di risoluzione di problemi di ottimizzazione, ovvero di ricerca del minimo di una funzione. Essendo, geometricamente, la direzione del gradiente quella lungo la quale la funzione varia più velocemente, questi metodi iterativi calcolano quindi ad ogni passo il minimo locale lungo la direzione del gradiente. Di questa categoria fanno parte, per esempio, i metodi Steepest Descent e Gradiente Coniugato.

Una terza categoria di metodi iterativi sono i metodi proiettivi, che si basano sulla ricerca di una soluzione approssimata in sottospazi di dimensione crescente

mediante l'utilizzo di operatori di proiezione. Gli spazi in cui viene cercata la soluzione sono chiamati spazi di Krylov e sono generati da una base del tipo $v, Av, \ldots., A^{m-1}v$, e $K_m(A, v)$ viene detto spazio di Krylov indotto da A e v. L'operazione di proiezione di un vettore consiste, dal punto di vista matematico, nella sua ortogonalizzazione rispetto ad un sottospazio della stessa dimensione di quello in cui esiste dato vettore. Un metodo proiettivo cerca una soluzione in uno spazio di ricerca di Krylov indotto da A e r_0, essendo $r_0 = b - Ax_0$ il residuo iniziale, e impone sul vettore residuo corrente r_m le condizioni di ortogonalità ad uno spazio $\mathcal{L}_m$, detto $spazio\ test$:

$$x_m \in x_0 + \mathcal{K}_m(A, r_0), \qquad r_m \perp \mathcal{L}_m \tag{1.3}$$

I vari metodi proiettivi differiscono per la scelta dello spazio $test$ e del precondizionatore. Degli esempi sono il metodo Bi-CGStab, Bi-Conjugate Gradient Stabilized, che utilizza lo spazio $test$ $\mathcal{L}_m = \mathcal{K}_m(A^T, r_0)$, e il MINRES, che utilizza invece $\mathcal{L}_m = A\mathcal{K}_m(A, r_0)$. (Gambolati, Ferronato, 1994)

1.2. GMRES

Il metodo GMRES fu introdotto da Saad e Schultz nel 1986, come generalizzazione a matrici non simmetriche e indefinite del MINRES. Esso è un metodo proiettivo che utilizza come spazio $test$ lo spazio

$$\mathcal{L}_m = A\mathcal{K}_m(A, r_0) = span\{Ar_0, A^2r_0, \ldots, A^m r_0\} \tag{1.4}$$

Si può dimostrare che trovare la soluzione $x_m \in x_0 + \mathcal{K}_m(A, r_0)$ imponendo l'ortogonalità di r_m rispetto a uno spazio $\mathcal{L}_m$ definito come nella (1.4) corrisponde a minimizzare la norma euclidea del residuo r_m. (Gambolati, Ferronato, 1994)

Quindi la soluzione cercata dal metodo GMRES è del tipo

$$x_m = x_0 + y, \;\; y \in K_m(A, r_0) \tag{1.5}$$

con $y = c_1 r_0 + \ldots + c_m A^{m-1} r_0 = \sum_{k=1}^{m} c_k A^{k-1} r_0$, tale per cui $\| r_m \|_2$ sia minima.

Tuttavia, essendo la successione $y_{k+1} = Ay_k$ convergente all'autovettore associato al raggio spettrale di A, i vettori y_k tendono ad essere paralleli tra loro, e quindi linearmente dipendenti, all'aumentare di k. Per questo motivo per applicare

il GMRES bisogna costruire una base V_m ortonormale di K_m. La soluzione sarà quindi del tipo $x_m = x_0 + V_m z_m$, con $z_m \in \mathbb{R}^m$, tale che $\| r_m \|_2$ sia minima.

1.3. Implementazione

Per calcolare una base ortonormale dello spazio di Krylov viene utilizzato il processo di ortogonalizzazione di Gram-Schmidt. Questo processo, data una base formata dagli m vettori linearmente indipendenti $g_1, g_2, \dots, g_m$, calcola gli m vettori di una base ortonormale dello stesso spazio come $\hat{v}_{k+1} = g_{k+1} - \sum_{j=1}^{k} h_{j,k} v_j$, con $v_{k+1} = \frac{\hat{v}_{k+1}}{\|\hat{v}_{k+1}\|_2}$ e $h_{j,k} = g_{k+1}^T v_j$. Applicando questo processo ai vettori base dello spazio di Krylov indotto da A e r_0 si ottiene $\hat{v}_{k+1} = A v_k - \sum_{j=1}^{k} h_{j,k} v_j$, con $h_{j,k} = v_k^T A^T V_j = v_j^T A v_k$ e quindi $\| \hat{v}_{k+1} \|_2 \, v_{k+1} = A v_k - \sum_{j=1}^{k} h_{j,k} v_j$. Infine, pre-moltiplicando per v_{k+1}^T, essendo tutti i v_k tra loro perpendicolari, si ottiene

$$\| \hat{v}_{k+1} \|_2 = v_{k+1}^T A v_k = h_{k+1,k} \tag{1.6}$$

L'algoritmo utilizzato per costruire questa base è l'algoritmo modificato di Gram-Schmidt, che parte da un vettore iniziale corrispondente al vettore residuo iniziale normalizzato, $v_1 = \frac{r_0}{\|r_0\|_2}$, e costruisce successivamente gli altri vettori nel seguente modo:

```
for k = 1 ... m
    w_{k+1} = A v_k
    for j = 1 ... k
        h_{j,k} = w_{k+1}^T v_j
        w_{k+1} = w_{k+1} - h_{j,k} v_j
    h_{k+1,k} = || w_{k+1} ||_2
    v_{k+1} = w_{k+1} / h_{k+1,k}
```

I vettori $v_1, \dots, v_{m+1}$ così ottenuti costituiscono la base ortonormale dello spazio di Krylov, e possono essere raggruppati in una matrice V_m di dimensione nxm. Il

vettore y può quindi essere scritto come una combinazione lineare dei vettori della nuova base:

$$y = z_1 v_1 + z_2 v_2 + \ldots + z_m v_m \tag{1.7}$$

e quindi come il prodotto tra la matrice V_m e un vettore contenente i coefficienti:

$$y = V_m z \tag{1.8}$$

Essendo, dalla (1.6), $Av_k = \sum_{j=1}^{k+1} h_{j,k}\, v_j$, si ottiene la scrittura $AV_m = V_{m+1}\bar{H}_m$, dove $\bar{H}_m$ è una matrice di Hessenberg di dimensione $(m+1)\mathrm{x}m$ le cui entrate sono i coefficienti $h_{j,k}$. Possiamo allora riscrivere il vettore residuo all'm-esima iterazione in questo modo:

$$\begin{aligned} r_m &= b - A(x_0 + V_m z) \\ &= b - Ax_0 - AV_m z \\ &= r_0 - V_{m+1}\bar{H}_m z \end{aligned} \tag{1.9}$$

$$r_0 = \beta V_{m+1} i_1, \text{ dove } i_1 = (1, 0, \ldots, 0)^T \tag{1.10}$$

Si ha quindi

$$r_m = V_{m+1}(\beta i_1 - \bar{H}_m z) \tag{1.11}$$

Ricordando che l'obbiettivo dell'algoritmo è quello di minimizzare la norma del vettore residuo ad ogni iterazione, si può riscrivere la norma come:

$$\begin{aligned} \| r_m \|_2 &= \sqrt{r_m^T r_m} \\ &= \sqrt{(\beta i_1 - \bar{H}_m z)^T V_{m+1}^T V_{m+1} (\beta i_1 - \bar{H}_m z)} \\ &= \| \beta i_1 - \bar{H}_m z \|_2 \end{aligned} \tag{1.12}$$

Un metodo agevole per minimizzare questa norma è effettuare la fattorizzazione QR della matrice $\bar{H}_m$, che consiste nel calcolare R, matrice triangolare alta, e Q, matrice ortogonale, tali che $\overline{H}_m = QR$.

Questa operazione può essere svolta con diversi algoritmi. Quello più utilizzato e più efficiente sfrutta le rotazioni di Givens per eliminare colonna per colonna gli

elementi della matrice $\overline{H}_m$ al di sotto della diagonale. La matrice Q che si ottiene da questa fattorizzazione è l'accumulazione delle m matrici di rotazione Ω_m, tale che $Q\overline{H}_m = R$.

La rotazione nel piano viene effettuata applicando una matrice di rotazione del tipo:

$$\Omega_j = \begin{pmatrix} 1 & 0 & & & & & & 0 \\ 0 & 1 & & & & & & \\ & & \ddots & & & & & \\ & & & c_i & s_i & & & \\ & & & -s_i & c_i & & & \\ & & & & & \ddots & & \\ & & & & & & 1 & 0 \\ 0 & & & & & & 0 & 1 \end{pmatrix} \tag{1.13}$$

che ruota i vettori i_j e i_{j+1} di un angolo θ_j, dove $c_j = \cos(\theta_j)$, $s_j = \sin(\theta_j)$.

$$c_j = \frac{h_{j,j}}{\sqrt{h_{j,j}^2 + h_{j+1,j}^2}} \; ; s_j = \frac{h_{j+1,j}}{\sqrt{h_{j,j}^2 + h_{j+1,j}^2}} \tag{1.14}$$

Ad ogni passo j allora verrà calcolata l'applicazione della matrice Ω_j alla j-esima colonna della matrice $\overline{H}_m$, riutilizzando le rotazioni degli j-1 passi già calcolati, nel seguente modo:

for $i = 1{:}\, j-1$

$temp = c_i\, h_{i,j} + s_i\, h_{i+1,j}$

$h_{i+1,j} = -s_i h_{i,j} + c_i h_{i+1,j}$

$h_{i,j} = temp$

$c_j = \frac{h_{j,j}}{\sqrt{h_{j,j}^2 + h_{j+1,j}^2}}$

$s_j = \frac{h_{j+1,j}}{\sqrt{h_{j,j}^2 + h_{j+1,j}^2}}$

$h_{j,j} = c_j h_{j,j} + s_j h_{j+1,j}$

$h_{j+1,j} = 0$

Essendo la matrice $Q_m = \Omega_m \Omega_{m-1} \dots \Omega_1$ unitaria e tale che $Q_m \overline{H}_m = R_m$, posso riscrivere la norma da minimizzare (1.12) come

$$\| \beta i_1 - \bar{H}_m z \|_2 = \| Q_m [\beta i_1 - \bar{H}_m z] \|_2 = \| Q_m \beta i_1 - R_m z \|_2 \qquad (1.15)$$

Allora, chiamando $\overline{g}_m$ il vettore:

$$\overline{g}_m = Q_m(\beta i_1) = [\gamma_1, \gamma_2, \dots, \gamma_{m+1}] \qquad (1.16)$$

Si ottiene la seguente uguaglianza:

$$\| Q_m \beta i_1 - R_m z \|_2 = \| \overline{g}_m - R_m z \|_2 = \| r_m \|_2 \qquad (1.17)$$

Il problema da risolvere diventa allora la ricerca del minimo

$$\min \| \overline{g}_m - R_m z \|_2 \qquad (1.18)$$

Essendo per costruzione l'ultima riga di R_m nulla, l'argomento della norma nella (1.18) si può scrivere nel seguente modo:

$$\begin{bmatrix} g_m \\ \gamma_{m+1} \end{bmatrix} - \begin{bmatrix} \widetilde{R}_m \\ 0 \end{bmatrix} \begin{bmatrix} z_m \\ 0 \end{bmatrix} \qquad (1.19)$$

ed è quindi facile vedere che il vettore z_m che la rende minima sarà quello che risolve il sistema $\widetilde{R}_m z_m = g_m$, dove $\widetilde{R}_m$ è la matrice triangolare alta ottenuta rimuovendo l'ultima riga nulla di R_m, e g_m è il vettore ottenuto rimuovendo l'$(m+1)$-esima componente di $\overline{g}_m$. È immediato allora vedere che ad ogni passo la norma del vettore residuo r_m è pari a γ_{m+1}. (Saad, Schultz, 1986)

Effettuando la fattorizzazione QR tramite le rotazioni di Givens si può ottenere una stima dell'errore corrente senza dover calcolare esplicitamente la soluzione x_m, operazione che richiederebbe un costo computazionale non trascurabile. L'algoritmo così ideato dunque si fermerà automaticamente una volta che il valore assoluto di γ_{m+1} scende sotto una tolleranza prefissata, e solo a questo punto verrà calcolata la soluzione approssimata

$$z_m = \widetilde{R}_m^{-1} g_m \qquad (1.20)$$

$$x_m = x_0 + V_m z_m \tag{1.21}$$

1.4. Alcune proprietà del GMRES

Il GMRES gode della proprietà di ottimo, ovvero giunge alla soluzione esatta al massimo in n iterazioni, trascurando gli errori di arrotondamento. Infatti, per definizione del metodo, la soluzione viene cercata come la migliore in un sottospazio di dimensione sempre maggiore ad ogni iterazione. È pertanto ovvio che alla n-esima iterazione la soluzione verrà cercata come la migliore nello spazio n-dimensionale, e corrisponde quindi alla soluzione esatta. Tuttavia, solitamente, il metodo converge in un numero di iterazioni molto inferiore rispetto ad n. Se così non fosse, non sarebbe un metodo particolarmente vantaggioso, poiché ad ogni iterazione vengono utilizzati tutti i vettori v_k calcolati ai passi precedenti e quindi sia il costo computazionale che la memoria necessaria a memorizzare i vettori crescono velocemente con l'aumentare dei passi necessari alla convergenza.

Per accelerare la convergenza è possibile precondizionare il sistema originale $Ax = b$ in modo tale da ottenere un sistema equivalente $By = c$ la cui matrice abbia proprietà di convergenza migliori rispetto ad A e si cerca quindi di ottenere una matrice B che abbia gli autovalori il più possibile raggruppati attorno ad un valore non nullo. Anche se per il GMRES si può provare che la distribuzione degli autovalori della matrice del sistema non è l'unica cosa che influisce sulla convergenza e che un importante ruolo lo giocano anche gli autovettori, è comunque provato sperimentalmente che un precondizionamento opportuno possa portare il metodo a convergere in un numero di iterazioni significativamente minore rispetto al sistema non precondizionato.

Il precondizionamento del sistema può essere ottenuto pre-moltiplicando o post-moltiplicando per una matrice M^{-1}, oppure nel caso che M^{-1} fosse fattorizzabile si può sia pre-moltiplicare che post-moltiplicare. Il precondizionamento del tipo $M^{-1}Ax = M^{-1}b$ viene detto precondizionamento sinistro. Quello del tipo $AM^{-1}y = b,\ x = M^{-1}y$ viene detto precondizionamento destro. Solitamente

viene preferito il precondizionamento destro al sinistro, poiché quest'ultimo nella sua applicazione va a modificare la norma del residuo, parametro sul quale viene imposta la tolleranza. Quindi nell'applicazione del metodo GMRES con un precondizionamento sinistro non si ha un controllo diretto su $\| r_0 \|_2$, bensì sulla norma del residuo precondizionato $\| M^{-1} r_0 \|_2$.

L'applicazione del precondizionatore comporta il calcolo aggiuntivo di un vettore $s = M^{-1} v$, per un vettore v, e dal punto di vista computazionale risulta quindi vantaggioso quando il numero di iterazioni necessarie a soddisfare la tolleranza imposta diminuisce rispetto al sistema non precondizionato.

Alcuni esempi di precondizionatori che si possono utilizzare sono i precondizionatori di Jacobi e Seidel, che approssimano l'inversa di A attraverso opportune fattorizzazioni, oppure la decomposta incompleta di Cholesky, che consiste nel calcolare il fattore triangolare di Cholesky incompleto, $\tilde{L}$, che viene calcolato solamente in corrispondenza delle entrate non nulle della matrice A. Con questo precondizionamento la matrice $M^{-1} = \left(\tilde{L}\tilde{L}^T\right)^{-1}$ viene applicata ad un certo vettore v mediante la risoluzione di due sistemi triangolari, uno alto e uno basso. Un' altra categoria di metodi di precondizionamento sono i metodi Multigrid, che consistono nel proiettare il problema in uno spazio di dimensione ridotta nel quale sia possible trovare la soluzione in modo più agevole. Il risultato dell'applicazione di questo tipo di precondizionatore è la soluzione del problema ridotto riscalata alla dimensione del problema originale tramite dei processi di interpolazione. Un'ulteriore classe di precondizionatori è quella delle inverse approssimate; questi precondizionatori utilizzano algoritmi che calcolano esplicitamente un'approssimazione dell'inversa della matrice A, che vengono poi applicate direttamente con un prodotto matrice-vettore. (Gambolati, Ferronato, 1994)

2. Applicazione ingegneristica

2.1. Meccanica del contatto

La meccanica del contatto si occupa delle interazioni tra due corpi o tra due parti dello stesso corpo. Quando si verifica una qualsiasi interazione tra due corpi solidi, questi esercitano una forza reciproca e si crea una azione di attrito tangenziale proporzionale, attraverso un coefficiente μ, alla sollecitazione normale esistente sulla superficie di contatto. Di norma, vale la legge di attrito statico finché la forza tangenziale supera quella di attrito. In questo caso si verificherà uno scorrimento relativo tra i due corpi.

La formulazione del problema della meccanica del contatto richiede che siano rispettate le leggi statico-dinamiche dell'attrito e la legge dell'impenetrabilità dei corpi solidi. Considerando condizioni quasi-statiche e deformazioni infinitesime, l'equilibrio di un corpo solido deformabile è descritto dalle seguenti formule:

$$-\nabla \cdot \sigma(u) = b \tag{2.1}$$

$$t_N = t \cdot n_f \leq 0, \qquad g_N = [[u]] \cdot n_f \geq 0, \qquad t_N g_N = 0 \tag{2.2}$$

$$\| t_N \|_2 \leq \tau_{MAX}(t_N), \qquad g_T . t_T = \tau_{MAX}(t_N) \| g_T \|_2, \tag{2.3}$$

dove la (2.1) è la legge di conservazione della quantità di moto, la (2.2) l'impenetrabilità dei corpi e la (2.3) la legge di attrito statico-dinamica; u è il vettore degli spostamenti, ovvero l'incognita del problema; b il vettore delle forze esterne di volume; ∇ l'operatore divergenza; $t = t_n n_f + t_T$ la tensione sulla superficie di contatto, [[u]] la discontinuità del vettore spostamento in corrispondenza della superficie di contatto.

Le equazioni (2.1)-(2.3) definiscono quindi un problema di ottimizzazione con vincolo di disuguaglianza, essendo la funzione (2.1) soggetta alle (2.2) e (2.3), dette condizioni di Karush-Kuhn-Tucker. Queste equazioni vincolari impongono un'adesione dei corpi a contatto in fase di compressione, infatti se così non fosse il problema si disaccoppierebbe, realizzandosi un'apertura della frattura o un distacco dei corpi in questione. Esse impongono inoltre l'impenetrabilità dei corpi solidi e delineano un limite superiore per la componente tangenziale del vettore

attrito, al raggiungimento del quale si verifica uno scorrimento relativo dei corpi lungo la superficie di contatto.

Il problema continuo della meccanica del contatto descritto dalle equazioni (2.1)-(2.3) può essere discretizzato mediante il metodo di Newton-Raphson, che consiste nella costruzione del seguente schema iterativo:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \tag{2.4}$$

Se applicato ad una funzione in uno spazio di dimensione n lo schema diventa:

$$x_{k+1} = x_k + \Delta x_k \tag{2.5}$$

$$\Delta x_k = -J(x_k)^{-1} f(x_k) \tag{2.6}$$

dove J è lo Jacobiano del problema. Il problema si riduce allora alla risoluzione, ad ogni passo, del sistema lineare (2.6), la cui incognita è la correzione da applicare alla soluzione corrente per trovare la nuova soluzione. In particolare, nel problema considerato l'incognita è lo spostamento u e J è la matrice di rigidezza.

La formulazione matematica di questo problema può essere effettuata con due diversi approcci. Il primo, chiamato approccio "penalty", si basa sull'idea di modellare il contatto con una molla rigida che si oppone alla penetrazione e allo scorrimento dei corpi. Viene quindi considerata possibile una penetrazione dei corpi infinitesima Δu, contrastata da una forza opposta $F = K\Delta u$. Saranno presenti due rigidezze, una tangenziale K_T e una normale K_N, il cui valore deve essere abbastanza grande da ridurre Δu a quasi zero e simulare la condizione di rigidezza infinita. Questo approccio risulta semplice da applicare e scaturisce nella risoluzione di un sistema lineare la cui matrice di rigidezza è simmetrica e definita positiva, quindi risolvibile con metodi iterativi tipo il Gradiente Coniugato. Tuttavia, come illustrato, non garantisce la completa impenetrabilità dei corpi e di conseguenza risulta meno preciso. Inoltre, la matrice può essere soggetta a problemi di malcondizionamento dovuto ai valori molto grandi delle costanti di rigidezza delle molle, con una convergenza non lineare molto più difficoltosa e irregolare.

L'altra possibile formulazione fa uso dei moltiplicatori di Lagrange per ricondursi ad un problema di ottimo libero che rispetti esattamente i vincoli del

problema. Il metodo dei moltiplicatori di Lagrange permette di passare da un problema di ottimo vincolato del tipo:

$$\min f(x), \quad g(x) = 0 \tag{2.7}$$

Ad un problema di ricerca dell'ottimo libero per una funzione ausiliaria detta Lagrangiana:

$$\mathcal{L}(x, \lambda) = f(x) - \lambda g(x) \tag{2.8}$$

essendo λ i moltiplicatori di Lagrange.

Per quanto riguarda nello specifico il problema della meccanica del contatto, i moltiplicatori di Lagrange assumono il significato fisico di tensione di contatto sulla frattura, mentre l'equazione di cui si vuole ricavare il minimo è l'energia potenziale totale del sistema.

Questa seconda formulazione porta quindi a risultati più accurati rispetto al metodo *penalty*, essendo il vincolo di impenetrabilità rispettato analiticamente, e la convergenza non lineare è generalmente più liscia. La maggior accuratezza però richiede un maggior costo computazionale nella risoluzione del sistema discreto così ottenuto, a causa dell'aumento del numero di incognite e dal cambiamento della natura del problema, che da simmetrico definito positivo diventa indefinito con una classica struttura a punto sella.

L'aggiunta dell'incognita dei moltiplicatori λ, essenziale per poter rimuovere la condizione di vincolo, porta ad avere un sistema lineare con matrice a blocchi:

$$J = \begin{bmatrix} A & B_1 \\ B_2^T & 0 \end{bmatrix} \tag{2.9}$$

dove il blocco (1,1) è la matrice di rigidezza, e i blocchi (2,1) e (1,2) sono i contributi aggiuntivi che includono i vincoli.

Una matrice come la (2.9) risulta essere una matrice di punto sella, i cui autovalori sono quindi sia negativi che positivi. Inoltre, la simmetria si ha solo quando i blocchi B_1 e B_2 sono uguali, situazione che tipicamente si verifica quando non si ha dissipazione di energia, ovvero nel caso statico.

Problemi di questo tipo si verificano anche in altre applicazioni, e una loro efficacie risoluzione si basa sull'utilizzo di opportune tecniche di precondizionamento che velocizzino la convergenza dei metodi iterativi. In particolare, essendo questo problema indefinito e, talvolta, non simmetrico, il metodo iterativo più frequentemente utilizzato è GMRES, introdotto nel capitolo 1.

3. Precondizionamento del sistema

3.1. RACP- Reverse Augmented Constrained Preconditioner

Un modo per precondizionare un sistema a blocchi del tipo

$$\begin{bmatrix} A & B \\ B^T & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \tag{3.1}$$

che abbia il blocco A simmetrico e semidefinito positivo e il blocco B di rango massimo, consiste nel premoltiplicare la matrice (2.6) per l'inversa della matrice "stabilizzata"

$$\tilde{\mathcal{A}} = \begin{bmatrix} A & B \\ B^T & -C \end{bmatrix} \tag{3.2}$$

Si può dimostrare che se A è non singolare, scegliendo la matrice C uguale al complemento di Schur

$$C = S = B^T A^{-1} B \tag{3.3}$$

gli autovalori della matrice precondizionata $\tilde{\mathcal{A}}^{-1}\mathcal{A}$ si raggruppano attorno a due numeri interi, in particolare si avrà un autovalore pari a 1 con molteplicità n_u e un autovalore pari a 0.5 con molteplicità n_t, essendo n_u e n_t il numero di colonne del blocco A e del blocco B rispettivamente. Infatti, tali autovalori devono soddisfare il sistema $\mathcal{A}$v=$\lambda\tilde{\mathcal{A}}$ v, ovvero:

$$\begin{cases} Av_u + Bv_t = \lambda Av_u + \lambda Bv_t \\ B^T v_u = \lambda B^T v_u - \lambda C v_t \end{cases} \tag{3.4}$$

Dalla prima equazione del sistema si ricava $\lambda = 1$ o $Av_u = -Bv_t$. Se $\lambda = 1$, dalla seconda equazione si ha $Cv_t = 0$, mentre v_u può assumere qualsiasi valore. L'autovalore 1 ha quindi molteplicità pari ha n_u. Se invece $\lambda \neq 1$, dalla prima equazione si ha $v_u = -A^{-1}Bv_t$, che inserito nella seconda equazione e sostituendo con la matrice $C = B^T A^{-1} B$, diventa:

$$C^{-1}Cv_t = 2\lambda v_t \tag{3.5}$$

da cui si ricava che l'autovalore $\lambda = 0.5$ ha molteplicità n_t.

È quindi chiaro che per un caso di questo tipo il precondizionatore migliore è quello con la matrice aumentata dal blocco C pari al complemento di Schur. Ottenendo infatti degli autovalori così raggruppati è garantita una convergenza veloce, in al più in due iterazioni se gli autovettori sono tra loro ortogonali. Tuttavia, A può essere un blocco singolare, e quindi diventa impossibile definire C come $B^T A^{-1} B$. Per maggiore generalità possiamo utilizzare un differente approccio che si basa su una fattorizzazione della matrice $\tilde{\mathcal{A}}$:

$$\tilde{\mathcal{A}} = \mathcal{UDL} = \begin{bmatrix} I_u & -BC^{-1} \\ 0 & I_t \end{bmatrix} \begin{bmatrix} S_u & 0 \\ 0 & -C \end{bmatrix} \begin{bmatrix} I_u & 0 \\ -C^{-1}B^T & I_t \end{bmatrix} \tag{3.6}$$

dove S_u, complemento primario di Schur, è una matrice simmetrica e definita positiva data da $S_u = A + BC^{-1}B^T$, e I_u e I_t sono le matrici identità di dimensione n_u e n_t rispettivamente. L'idea consiste nell'utilizzare un'approssimazione della fattorizzazione di $\tilde{\mathcal{A}}$ per applicare $\tilde{\mathcal{A}}^{-1}$ e precondizionare il sistema.

Essendo l'inversa di S_u costosa da calcolare è opportuno applicarne un'opportuna approssimazione $\widetilde{S}_u^{-1}$. Allora la matrice del precondizionamento in forma fattorizzata diventa:

$$\mathcal{M}^{-1} = \mathcal{L}^{-1}\widetilde{D}^{-1}\mathcal{U}^{-1} = \begin{bmatrix} I_u & 0 \\ C^{-1}B^T & I_t \end{bmatrix} \begin{bmatrix} \widetilde{S}_u^{-1} & 0 \\ 0 & -C^{-1} \end{bmatrix} \begin{bmatrix} I_u & BC^{-1} \\ 0 & I_t \end{bmatrix} \tag{3.7}$$

Questa matrice di precondizionamento viene chiamata RACP, Reverse Augmented Constraint Preconditioner. (Franceschini et al., 2022)

L'analisi degli autovalori della matrice del sistema così precondizionato mostra come essi siano anche complessi, e la matrice precondizionata è non simmetrica. Si può quindi ricavare una differente matrice $\mathcal{M}_a$ partendo dalla matrice

$$\overline{\mathcal{A}} = \begin{bmatrix} A & B \\ -B^T & C \end{bmatrix} \tag{3.8}$$

che con la fattorizzazione UDL diventa:

$$\overline{\mathcal{A}} = \mathcal{U}_a\mathcal{D}_a\mathcal{L}_a = \begin{bmatrix} I_u & BC^{-1} \\ 0 & I_t \end{bmatrix} \begin{bmatrix} S_u & 0 \\ 0 & C \end{bmatrix} \begin{bmatrix} I_u & 0 \\ -C^{-1}B^T & I_t \end{bmatrix} \tag{3.9}$$

e quindi, prendendo la sua inversa e approssimando S_u^{-1} con $\widetilde{S}_u^{-1}$ otteniamo

$$\mathcal{M}_a^{-1} = \mathcal{L}_a^{-1}\widetilde{D_a}^{-1}\mathcal{U}_a^{-1} = \begin{bmatrix} I_u & 0 \\ C^{-1}B^T & I_t \end{bmatrix}\begin{bmatrix} \widetilde{S}_u^{-1} & 0 \\ 0 & C^{-1} \end{bmatrix}\begin{bmatrix} I_u & -BC^{-1} \\ 0 & I_t \end{bmatrix} \quad (3.10)$$

Applicando questa matrice al sistema di partenza si ottiene una matrice simmetrica indefinita, con autovalori distribuiti in intervalli quasi simmetrici rispetto all'origine. Tuttavia, questa situazione non risulta favorevole per l'applicazione pratica, poiché lo 0 si trova in mezzo allo spettro.

3.2. Scelta della matrice C

La chiave dell'efficacia del precondizionamento sta nella scelta della matrice C, che deve soddisfare i requisiti in contrasto tra di loro. Come mostrato precedentemente, la matrice C deve assomigliare a S, in modo tale che gli autovalori siano raggruppati attorno a due valori non nulli. Essendo C^{-1} usata anche nel calcolo di S_u, è necessario che sia anche molto sparsa e facilmente invertibile. Per questo motivo C viene scelta come matrice diagonale spettralmente prossima a $S = B^T A^{-1} B$. Un modo efficiente per ottenere tale risultato si basa su un approccio di tipo Augmented Lagrangian applicato a livello locale per ciascuna entrata della diagonale di C.

Supponiamo di valutare l'i-esimo elemento diagonale di S. Esso richiede la soluzione del prodotto $A^{-1}b_i$, dove b_i è la i-esima colonna di B. Poiché b_i è un vettore sparso, le uniche entrate di A^{-1} di interesse per il calcolo di $A^{-1}b_i$ sono quelle che si trovano nelle righe e nelle colonne corrispondenti alle posizioni dei non zeri di b_i, come mostrato:

$$\begin{pmatrix} 0 & 5 & 0 & 0 & 0 \\ 0 & 3 & 0 & 7 & 0 \\ 0 & 0 & 2 & 0 & 9 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ 0 \\ 5 \\ 0 \end{pmatrix} = \begin{pmatrix} 0\,x\,5 \\ 7\,x\,5 \\ 0\,x\,5 \end{pmatrix}$$

Ovviamente non conosciamo il valore esatti di tali coefficienti, perché A^{-1} non è nota. Tuttavia, possiamo pensare di sostituirli con l'inversa della sottomatrice di A costituita dagli elementi posti nelle righe e nelle colonne che corrispondono alle

posizioni dei non zeri di b_i. Questa operazione risulta giustificabile anche dal punto di vista fisico poiché una variazione locale di tensione su una frattura produce effetti di maggior rilievo solamente in prossimità di essa.

Le entrate diagonali di S, quindi, possono essere scritte come:

$$s_{i,i} \cong r(b_i^T)A|_{b_i}^{-1} r(b_i) \quad \forall i \in 1 \dots n_t \tag{3.11}$$

essendo $A|_{b_i}^{-1}$ un blocco quadrato ristretto alle righe e alle colonne di A che intercettano le entrate non nulle di b_i, e $r(\cdot)$ un operatore di restrizione che restituisce solamente le entrate non nulle del vettore su cui viene applicato. Tuttavia, essendo possibile che A sia singolare, è possibile che anche la sua restrizione $A|_{b_i}^{-1}$ lo sia. In questo caso non sarebbe possibile calcolarne l'inversa e si stima allora la dimensione di $s_{i,i}$ attraverso le norme-2:

$$s_{i,i} = \frac{\|r(b_i)\|_2}{\|A|_{b_i}\|_2} \tag{3.12}$$

L'algoritmo per la creazione della matrice C diventa quindi:

```
for j = 1:n_t
    patt = find(b_j)
    b = B_{patt,j}
    AB = full(A_{patt,patt})
    c_{j,j} = ||b||_2^2 / ||AB||_2
```

3.3. Implementazione di GMRES precondizionato

Il metodo di precondizionamento che viene applicato in questo lavoro è il precondizionamento destro, risolvendo quindi il sistema equivalente:

$$AM^{-1}y = b, \quad x = M^{-1}y \tag{3.13}$$

e come matrice di precondizionamento è stata usata la matrice della (3.7). Per applicare quindi il metodo GMRES precondizionato è sufficiente aggiungere una

funzione di applicazione della matrice M^{-1} ad un vettore, usando lo stesso algoritmo del GMRES introdotto nel Capitolo 1.

La prima operazione in GMRES che coinvolge il prodotto matrice-vettore è:

$$w_{k+1} = Av_k \tag{3.14}$$

ed essendo adesso il sistema precondizionato, questa operazione deve diventare

$$w_{k+1} = AM^{-1}v_k \tag{3.15}$$

Per effettuare questa operazione si evita ovviamente di fare un prodotto tra due matrici, poiché, nonostante siano entrambe fortemente sparse, il loro prodotto non lo è. La funzione, quindi, calcola un vettore $s = M^{-1}v_k$ e il vettore cercato w_{k+1} viene calcolato ad ogni passo come $w_{k+1} = As$.

Ad ogni iterazione bisogna quindi risolvere il sistema $Ms = v_k$ per ricavare s. Essendo la matrice M una matrice a blocchi, il prodotto $Ms = v_k$ può essere scritto come due sistemi distinti:

$$As_1 + Bs_2 = v_1 \tag{3.16}$$

$$B^T s_1 - Cs_2 = v_2 \tag{3.17}$$

Ricavando s_2 in funzione di s_1 si ottiene $s_2 = C^{-1}(B^T s_1 - v_2)$ e sostituendolo nella prima equazione si ottiene $(A + BC^{-1}B^T)s_1 = v_1 + BC^{-1}v_2$, dove $S_u = A + BC^{-1}B^T$ è detto il complemento primario di Schur.

Il sistema

$$S_u s_1 = v_1 + BC^{-1}v_2 \tag{3.18}$$

può essere risolto con un metodo diretto oppure con un'approssimazione, per esempio usando la fattorizzazione incompleta di Cholesky.

Nel primo caso, essendo la soluzione del (3.18) calcolata a precisione di macchina, l'applicazione del precondizionatore avviene in modo molto accurato con un corrispondente vantaggio in termini di velocità di convergenza di GMRES; approssimando invece la soluzione del sistema il numero di iterazioni di GMRES generalmente aumenta anche in modo non indifferente. Tuttavia, come mostrato in seguito con i risultati numerici, utilizzare la fattorizzazione incompleta di

Cholesky può portare ad un risparmio significativo di tempo di calcolo, con un beneficio complessivo in termini di efficienza computazionale.

4. Risultati numerici

Il metodo GMRES così precondizionato è stato quindi applicato ad un caso reale di mezzo fratturato. La frattura come mostrata nella figura 1 è stata modellata con elementi finiti esaedrici trilineari, con 5 mesh di raffinatezza crescente.

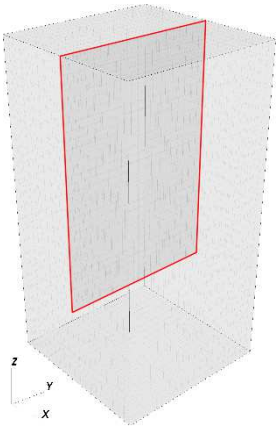


Figura 1: frattura di un corpo solido

Nella tabella seguente sono indicate le dimensioni dei blocchi della matrice del problema che scaturisce dalla modellizzazione con ognuna delle 5 mesh.

Tabella 2: dimensione delle matrici

MESH	n_u	n_t
1	162	27
2	765	90
3	2100	189
4	4455	324
5	8118	495

Al fine di verificarne l'efficienza computazionale, il metodo qui sviluppato è stato applicato alle matrici scaturite dalla modellizzazione agli elementi finiti considerando come vettore dei termini noti b e soluzione iniziale x_0 dei vettori randomizzati sparsi. La convergenza, infatti, non dipende da questi due vettori. Come parametro di tolleranza per il criterio di uscita sul valore del residuo relativo al residuo iniziale è stato imposto un valore di 10^{-8}.

Di seguito sono riportati i profili di convergenza del metodo, con un confronto tra l'applicazione del precondizionatore con la soluzione del sistema con il complemento di Schur mediante un metodo diretto e quello invece calcolato con la fattorizzazione incompleta di Cholesky . In particolare, per il primo caso è stato utilizzato il backslash di MATLAB, e per il secondo caso la funzione "ichol".

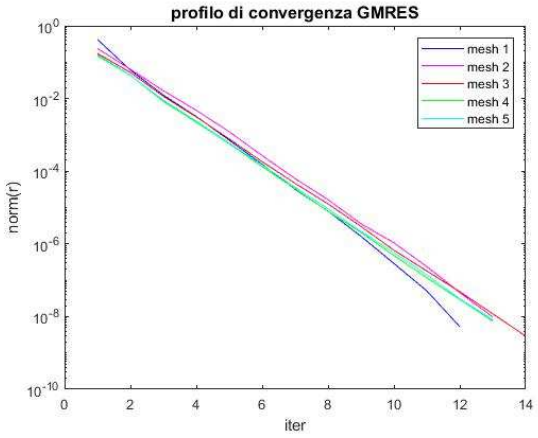


Figura 2: profilo di convergenza- backslash

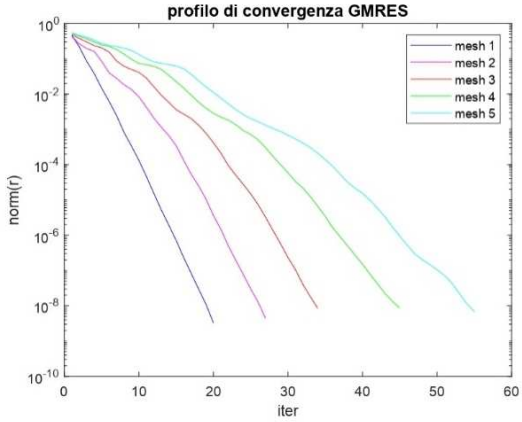


Figura 3: profilo di convergenza-ichol

In questa figura 2 è rappresentato il profilo di convergenza del metodo applicato alle diverse mesh. Il metodo risulta efficacie e l'errore, misurato come norma del vettore residuo, scendo sotto la tolleranza dopo 12-14 iterazioni. Dalla figura 2 si vede che il precondizionatore gode di scalabilità ottimale rispetto alla dimensione del problema, ovvero converge nello stesso numero di iterazioni indipendentemente da essa. Tuttavia, per problemi di grande dimensione diventa difficoltosa la risoluzione del complemento di Schur con il backslash; per questo motivo viene introdotta l'approssimazione con la fattorizzazione incompleta di Cholesky i cui risultati sono riportati nella figura 3. Si evidenzia come, invece, in questo caso il numero di iterazioni cresce con la dimensione del problema, seppur in modo contenuto. Se invece si decidesse di sostituire 'ichol' con un precondizionatore interno scalabile, come per esempio un precondizionatore Multigrid, si potrebbe ripristinare anche la scalabilità globale. Risultati di questa applicazione sono mostrati in (Franceschini et al., 2022).

Nelle seguenti figure 4-8 sono riportati i profili di convergenza per ogni mesh, con un confronto tra il metodo con il calcolo diretto del sistema, in legenda

indicato con “backslash”, e il metodo con il calcolo approssimato con la fattorizzazione di Cholesky, “ichol”.

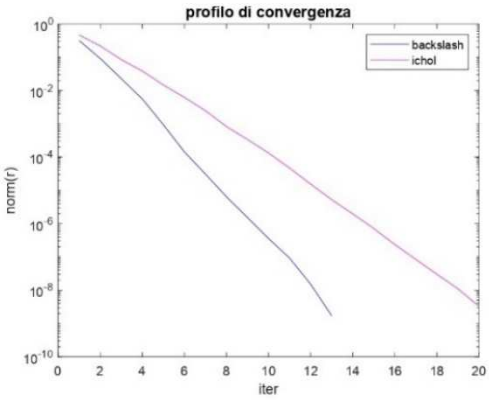


Figura 4: profilo di convergenza-mesh 1

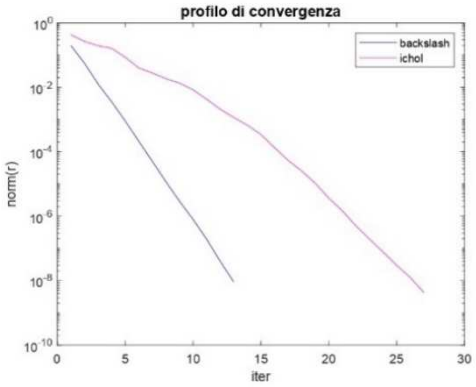


Figura 5: profilo di convergenza-mesh 2

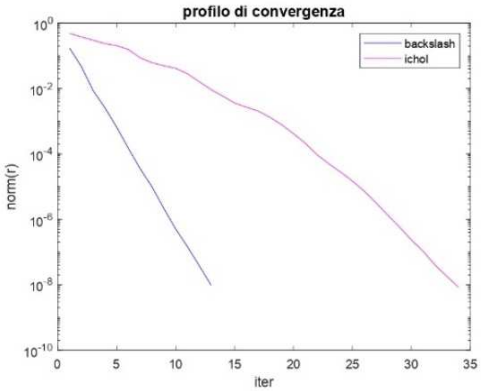


Figura 6: profilo di convergenza-mesh3

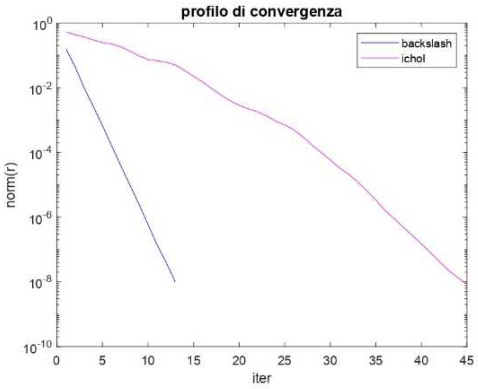


Figura 7: profilo di convergenza-mesh 4

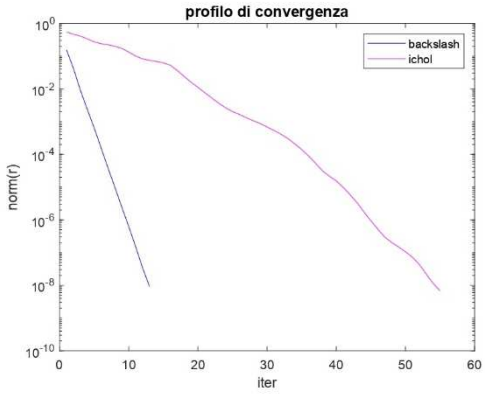


Figura 8: profilo di convergenza-mesh 5

Tuttavia, nonostante il numero maggiore di iterazioni necessario, il costo computazionale è, come già detto precedentemente, minore. Confrontando i tempi di esecuzione dei diversi algoritmi si nota una differenza sostanziale, che aumenta all'aumentare della dimensione del problema. Si può vedere nel seguente grafico come il rapporto tra il tempo di calcolo della soluzione con l'algoritmo "backslash" e quello con l'algoritmo "ichol" sia della dimensione delle unità per il sistema minore, e aumenti fino ad essere 60 volte tanto per il problema di dimensione maggiore. Di seguito sono riportati in una tabella i tempi di esecuzione per la risoluzione di ogni mesh.

Tabella 2: tempi di calcolo

mesh	T(s) backslash	T(s) ichol
1	0.085455	0.066643
2	0.423332	0.045761
3	4.132415	0.223649
4	31.759988	0.845102
5	165.035989	2.034984

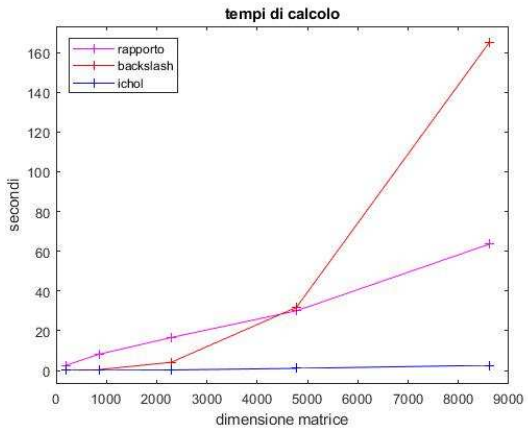


Figura 9: tempi di calcolo

Dalla figura 9 si nota come l'aumento dei tempi di calcoli del metodo applicato con $backslash$ aumenti in modo cubico con il crescere della dimensione del problema. Per problemi di dimensione maggiore raggiungerebbe tempi di calcolo proibitivi. Il metodo $ichol$, invece, richiede tempi di calcolo decisamente minori e risulta quindi efficacie per la risoluzione di sistemi di maggiore dimensione.

5. Conclusioni

In questo lavoro di tesi è stato implementato il metodo GMRES per la risoluzione di sistemi lineari indefiniti e non simmetrici e se ne è vista una sua applicazione ad un problema di meccanica del contatto.

L'implementazione è stata realizzata con MATLAB e gli algoritmi implementati sono riportati di seguito nell'appendice.

Il metodo implementato è stato applicato ad un sistema precondizionato con RACP, effettuando una distinzione tra il calcolo del sistema con S_u con un metodo diretto o con una sua approssimazione fornita dalla fattorizzazione incompleta di Cholesky. I risultati ottenuti mostrano come il metodo così implementato e precondizionato risulti particolarmente efficace, giungendo a risolvere i problemi in poche iterazioni. In particolare, si è visto come il precondizionatore RACP sia dotato di proprietà di ottima scalabilità, che viene meno quando si opera l'approssimazione di S_u. Tuttavia, risulterebbe impossibile risolvere con metodi diretti sistemi con dimensioni di S_u particolarmente grandi. Per questo motivo diventa rilevante l'efficacia dell'approssimazione, che, nonostante porti a perdere l'ottima scalabilità, risultando in un aumento contenuto del numero di iterazioni, porta ad avere dei tempi di calcolo significativamente minori.

I tempi di calcolo della soluzione del problema in cui non si applichi l'approssimazione crescono in modo cubico con l'aumentare delle dimensioni della matrice del sistema lineare e, come si vede nel grafico proposto sopra, diventerebbero proibitivi per problemi di dimensioni maggiori rispetto a quelli proposti.

Il precondizionamento RACP per il metodo GMRES risulta quindi efficace, in particolar modo quando viene utilizzato un precondizionamento per la risoluzione del sistema con il complemento primario di Schur. La scelta di un opportuno metodo di precondizionamento per S_u può inoltre portare a significativi aumenti della velocità di risoluzione dei problemi.

Bibliografia

Franceschini A., Ferronato M., Frigo M., Janna C. ,(2022). A reverse augmented constraint preconditioner for Lagrange multiplier methods in contact mechanics. *Computer Methods in Applied Mechanics and Engineering.*

Gambolati G., Ferronato M. ,(1994) Lezioni di metodi numerici per l'ingegneria. *Libreria Progetto Padova.*

Saad Y. ,(2003). Iterative methods for sparse linear systems. *Society for Industrial and Applied Mathematics.*

Saad Y., Schultz M. H. ,(1986) GMRES: a generalized minimal residual algorithm solving nonsymmetric linear systems. *Society for Industrial and Applied Mathematics.*

Yastrebov V. A. ,(2013) Numerical Methods in Contact Mechanics. *John Wiley & Sons, Inc. USA*

Appendice

```
function [x,e,iter] = gmres_prec(A,B,b,x0,tol,itmax)

%applicazione del gmres con precondizionatore
[~,An]=size(A);
[~,Bn]=size(B);

%calcolo C
[C] = cr_C(A,B,Bn);

prod=prodotto(A,B,0,x0,An,Bn);
r0=b-prod;

beta=norm(r0);

tau=beta;

%inizializzo vettori
sin=zeros(itmax,1);
cos=zeros(itmax,1);
b_norm=norm(b);

%costruisco la matrice V, base ortonormale dello spazio di Krylov
di
%dimensione m. Processo di Gram-Schmidt

V(:,1)=r0/beta;

k=1;
iter=0;
e(iter+1)=beta;

while tau>tol && iter<itmax
    iter=iter+1;

    [s1,s2]=appl_prec(A,B,C,V(:,k),An,Bn);
    [W(:,k+1)] = prodottomatvec(A,B,s1,s2,An,Bn);

    for j=1:k
        H(j,k)= W(:,k+1)'*V(:,j);
        W(:,k+1)=W(:,k+1)-H(j,k)*V(:,j);
    end

    H(k+1,k)=norm(W(:,k+1));
    V(:,k+1)=W(:,k+1)/H(k+1,k);

    %applico le rotazioni di Givens

   [H(1:k+1,k),cos(k),sin(k)]=applRotations(H(1:k+1,k),cos,sin,k);
    beta(k+1)=-sin(k)*beta(k);
    beta(k)=cos(k)*beta(k);
```

```
    e(k)=abs(beta(k+1))/b_norm;

    tau=e(k);
    k=k+1;

end

    beta=beta';
    z=H\beta;
    V(:,k)=[];

    Vz=V*z;

    [vz1,vz2] =applSchur(A,B,C,Vz,An,Bn);
    vz(1:An,1)=vz1;
    vz(An+1:An+Bn,1)=vz2;

    x=x0+vz;

end
          ------------------------------------------------

function [C] = cr_C(A,B,Bn)
%creazione della matrice C

for j=1:Bn
    patt=find(B(:,j));
    b=B(patt,j);
    AB=full(A(patt,patt));
    C(j,j)=(norm(b))^2/norm(AB);
end
end
          ------------------------------------------------

function [v1,v2] =appl_prec(A,B,C,r,An,Bn)
%applicazione del precondizionatore RACP.

r1=r(1:An);
r2=r(An+1:An+Bn);

%calcolo primal Schur complement
Cinv=inv(C);

S=A+B*Cinv*B';

%risolvo il sistema
cr=Cinv*r2;
div=r1+B*cr;
v1=S\div;

%calcolo v2
bv=B'*v1;
rbv=bv-r2;
v2=Cinv*rbv;

end
```

```
-----------------------------------------------


function  [v1,v2] =applSchur_ichol(A,B,C,r,An,Bn)
%applicazione del precondizionatore con fattorizzazione incompleta
di
%Cholesky

r1=r(1:An);
r2=r(An+1:An+Bn);

%calcolo primal Schur complement
Cinv=sparse(inv(C));

S=A+B*Cinv*B';

%risolvo il sistema
cr=Cinv*r2;
div=r1+B*cr;
L=ichol(S);
z=L\div;
v1=L'\z;

%calcolo v2
bv=B'*v1;
rbv=bv-r2;
v2=Cinv*rbv;

end
-----------------------------------------------

function [h,cos_k, sin_k]=applRotations(h,cos,sin,k)
%applicazione delle rotazioni di Givens

for i=1:k-1
    temp=cos(i)*h(i)+sin(i)*h(i+1);
    h(i+1)=-sin(i)*h(i)+cos(i)*h(i+1);
    h(i)=temp;
end

[cos_k,sin_k]=givens_rotation(h(k),h(k+1));

%elimino H(i+1,i)
h(k)=cos_k*h(k)+sin_k*h(k+1);
h(k+1)=0.0;

end
-----------------------------------------------

function [cos,sin]=givens_rotation(v1,v2)
%calcolo sin e cos

t=sqrt(v1^2+v2^2);
cos=v1/t;
sin=v2/t;
end
```