

Metodi di Krylov

Metodi per la soluzione numerica di sistemi di equazioni lineari non simmetriche applicabili in un sottospazio di Krylov.

Fabio Archenti Fabio Camagni Andrea Favero
Lorenzo Fiamingo Stefano Gallivanone Nazariy Nashkolnyy

29/04/2020

Bibliografia

- [1] Quarteroni, Saleri, and Gervasio.
Calcolo Scientifico: Esercizi e problemi risolti con MATLAB e Octave.
UNITEXT. Springer Milan, 2012.
- [2] Trefethen, and Bau.
Numerical Linear Algebra.
SIAM, 1997.
- [3] Telichevesky, Kundert, and White.
Efficient Steady-State Analysis based on Matrix-Free Krylov-Subspace Methods.
Scientific report, June 1995.

Sommario

Introduzione

Metodo del Gradiente Coniugato

GMRES

Introduzione

- ▶ I metodi di Krylov risolvono sistemi lineari $Ax = b$, dove:
 - ▶ $A \in \mathbb{C}^{n \times n}$ è una matrice quadrata
 - ▶ $x \in \mathbb{C}^n$ è il vettore delle incognite
 - ▶ $b \in \mathbb{C}^n$ è il vettore dei termini noti
- ▶ Ad ogni passo dell'iterazione viene calcolata una soluzione approssimata x_k appartenente al sottospazio di Krylov:
- ▶ Si definisce **sottospazio di Krylov** il sottospazio generato:

$$\mathcal{K}_n(A, b) = \text{span}\{b, Ab, A^2b, \dots, A^{n-1}b\}, \quad n \geq 1$$

.

Introduzione (II)

- ▶ x_0 è la **guess** iniziale al passo 0.
- ▶ Definiamo la **soluzione** del sistema cercata

$$x^* = A^{-1}b$$

- ▶ Definiamo il **residuo** del sistema al passo k

$$r_k = b - Ax_k$$

- ▶ Definiamo l'**errore** del sistema al passo k

$$e_k = x_k - x^*$$

Possiamo notare come $r_k = -A^{-1}e_k$.

Metodi di discesa

- Data A matrice simmetrica e definita positiva, possiamo definire la funzione (detta **forma quadratica**) $\phi : \mathbb{R}^n \rightarrow \mathbb{R}$

$$\phi(x) = \frac{1}{2}x^T A x - x^T b$$

- Questa funzione (se le ipotesi su A sono verificate) è una funzione convessa e ammette un unico punto x^* di **minimo assoluto**.

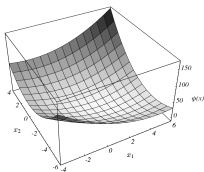


Figura 1: Forma quadratica per una matrice definita positiva

Metodi di discesa (II)

- ▶ Calcolo del **gradiente** di $\phi(x)$: $\nabla\phi(x) = Ax - b$ (con $\nabla(b^T x) = b$ e $\nabla(x^T Ax) = 2Ax$).
- ▶ Nel punto x^* il gradiente si annulla: $\nabla\phi(x^*) = Ax^* - b = 0$.
Risolvere il problema di minimo su $\phi(x)$ (trovando x^*) equivale quindi a risolvere il sistema $Ax = b$.
- ▶ Assegnato un $x_0 \in \mathbb{R}^n$, si procede per ogni k fino a convergenza:
 1. determinando una direzione di discesa $d_k \in \mathbb{R}^n$
 2. determinando un passo $\alpha_k \in \mathbb{R}$
 3. ponendo $x_{k+1} = x_k + \alpha_k d_k$
- ▶ L'approccio sarà lo stesso per il metodo del Gradiente e per il metodo del Gradiente Coniugato, ma sceglieremo direzioni di discesa d_k diverse per ogni metodo.

Metodo del Gradiente

- ▶ Per il metodo del Gradiente la direzione d_k è il residuo $r_k = b - Ax_k$.
- ▶ $\nabla\phi(x) = Ax - b$: $d_k = r_k = -\nabla\phi(x)$.
- ▶ Si può dimostrare che il passo di discesa ottimale è $\alpha_k = \frac{d_k^T r_k}{d_k^T A d_k}$.
- ▶ Si aggiorna la soluzione: $x_{k+1} = x_k + \alpha_k d_k$.
- ▶ E infine si aggiorna il residuo: $r_{k+1} = r_k - \alpha_k A d_k$.

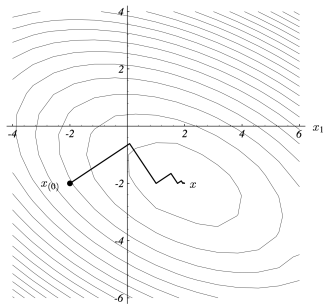


Figura 2: Iterazioni del metodo del gradiente

Metodo del Gradiente Coniugato - Passo di discesa

- Il **passo di discesa** più conveniente (che minimizza $\phi(x_{k+1})$) è

$$\alpha_k = \frac{d_k^T r_k}{d_k^T A d_k}$$

- Dimostrazione:

- Sostituendo la soluzione al passo $k + 1$ $x_{k+1} = x_k + \alpha_k d_k$ nella forma quadratica

$$\phi(x_{k+1}) = \phi(x_k + \alpha_k d_k) = \frac{1}{2} \left(d_k^T A d_k \right) \alpha_k^2 - d_k^T \underbrace{(b - A x_k)}_{=r_k} \alpha_k + \frac{1}{2} x_k^T A x_k - x_k^T b$$

- Derivando e imponendo la condizione di stazionarietà

$$\frac{d\phi}{d\alpha_k}(x_k + \alpha_k d_k) = \left(d_k^T A d_k \right) \alpha_k - d_k^T r_k = 0$$

- Ricaviamo α_k ottenendo la formula cercata.

Calcolo del residuo

- Calcoliamo il **residuo** $r_k = b - Ax_k$:

$$r_{k+1} = r_k - \alpha_k Ad_k$$

- Dimostrazione:

- $r_{k+1} = -Ae_{k+1}$
 - $r_{k+1} = -A(e_k + \alpha_k d_k)$
 - $r_{k+1} = r_k - \alpha_k Ad_k$
- Questa formula, in caso di matrici di grandi dimensioni, potrebbe essere soggetta ad errori di troncamento. Una buona soluzione potrebbe essere l'utilizzare $r_k = b - Ax_k$ dopo alcune iterazioni.

Direzione di discesa

- ▶ Utilizziamo l'**ortogonalizzazione di Gram-Schmidt** per ottenere direzioni $\{d_k\}$ A -ortogonali a partire dai residui $\{r_k\}$
 - ▶ $d_0 = r_0$
 - ▶ $d_{k+1} = r_{k+1} + \sum_{h=0}^k \beta_{kh} d_h$
- ▶ Dato che AK_k è incluso in \mathcal{K}_{k+1} , il fatto che r_{k+1} sia ortogonale a \mathcal{K}_{k+1} implica che r_{k+1} sia A -ortogonale a AK_k , e quindi a tutte le precedenti direzioni di discesa d , esclusa d_k .
- ▶ Di conseguenza **non sarà necessario memorizzare le vecchie direzioni** per assicurare la A -ortogonalità delle nuove.
- ▶ La **direzione di discesa** diviene

$$d_{k+1} = r_{k+1} + \beta_{k+1} d_k$$

dove

$$\beta_{k+1} = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$$

Passo di discesa (II)

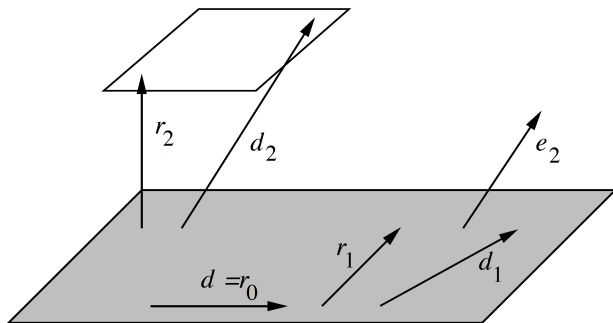


Figura 3: Il piano scuro è il sottospazio \mathcal{K}_2 . Notiamo che r_2 e d_2 puntano su un piano parallelo a \mathcal{K}_2 (come conseguenza dell'ortogonalizzazione)

► Dalla Figura 3 $d_k^T r_k = r_k^T r_k$, riscriviamo il passo di discesa come

$$\alpha_k = \frac{r_k^T r_k}{d_k^T A d_k}$$

Velocità di convergenza

- Il metodo del gradiente coniugato converge al massimo in n iterazioni (in aritmetica esatta), e si ottiene che

$$\|e_{(k)}\|_A \leq \frac{2c^k}{1 + c^{2k}} \|e_{(0)}\|_A$$

$$\|e_{(k)}\|_A \leq 2c^k \|e_{(0)}\|_A$$

$$c = \frac{\sqrt{K(A)} - 1}{\sqrt{K(A)} + 1}.$$

$K(A)$ è il numero di condizionamento in norma 2 di A

- Come metodo iterativo, viene arrestato quando il residuo relativo $E_{(k)} = \frac{\|r_{(k)}\|}{\|r_{(0)}\|}$ è minore di una tolleranza data.

Precondizionamento (I)

- ▶ La velocità di convergenza dipende quindi dal numero di condizionamento $K(A)$.
- ▶ Per accelerare la convergenza dobbiamo diminuire il numero di condizionamento ($K(P^{-1}A) \ll K(A)$)
- ▶ Possiamo risolvere il **sistema precondizionato** (equivalente a quello dato, $Ax = b$)

$$M^{-1}Ax = M^{-1}b$$

- ▶ In generale se M e A sono simmetriche e definite non è detto che lo sia $M^{-1}A$, ma per ogni M simmetrica, definita positiva esiste almeno una matrice P tale che $PP^T = M$.
- ▶ Dato che $M^{-1}A$ e $P^{-1}AP^T$ hanno gli stessi autovalori possiamo scrivere il sistema equivalente

$$P^{-1}AP^{-T}\hat{x} = P^{-1}b, \hat{x} = P^T x$$

- ▶ $P^{-1}AP^{-T}$ è simmetrica e definita positiva.

Precondizionamento (II)

- ▶ Per tornare al caso del primo sistema preconditionato sfruttiamo l'equazione $M^{-1} = P^{-T}P^{-1}$
- ▶ Scegliendo M stiamo operando un *trade-off* tra la diminuzione del numero di condizionamento e il costo computazionale della risoluzione del sistema $s_{(k)} = M^{-1}r_{(k)}$ per trovare il residuo preconditionato $s_{(k)}$.
- ▶ Esempio di algoritmi applicabili per calcolare M sono:
 - ▶ Precondizionamento di Jacobi
 - ▶ Precondizionamento incompleto di Cholesky
- ▶ La convergenza del metodo preconditionato è più veloce, poichè $K(A)$ è stato sostituito con $K(P^{-1}A)$:

$$\|e_{(k)}\|_A \leq \frac{2c^k}{1 + c^{2k}} \|e_{(0)}\|_A, c = \frac{\sqrt{K(P^{-1}A)} - 1}{\sqrt{K(P^{-1}A)} + 1}.$$

Metodo del gradiente coniugato (senza preconditionamento)

- ▶ $\mathbf{d}_0 = \mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$
- ▶ $\alpha_k = \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{d}_k^T \mathbf{A} \mathbf{d}_k}$
- ▶ $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$
- ▶ $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{A} \mathbf{d}_k$
- ▶ $\beta_{k+1} = \frac{\mathbf{r}_{k+1}^T \mathbf{r}_{k+1}}{\mathbf{r}_k^T \mathbf{r}_k}$
- ▶ $\mathbf{d}_{k+1} = \mathbf{r}_{k+1} + \beta_{k+1} \mathbf{d}_k$

Implementazione Matlab (senza preconditionamento)

Dati come input una matrice A , il vettore dei termini noti b , una guess iniziale x , un numero massimo di iterazioni k_{\max} e una tolleranza ϵ

```
k = 0;
r = b-A*x; d = r;
delta = r'*r; delta0 = delta;
while k < kmax && delta > e^2*delta0
    q = A*d;
    alpha = delta/(d'*q);
    x = x+alpha*d;
    r = r-alpha*q;
    deltaold = delta;
    delta = r'*r;
    beta = delta/deltaold;
    d = r+beta*d;
    k=k+1;
end
```

Metodo del gradiente coniugato (con preconditionamento)

- ▶ $r_0 = b - Ax_0$
- ▶ $d_0 = M^{-1}r_0$
- ▶ $\alpha_k = \frac{r_k^T M^{-1}r_k}{d_k^T A d_k}$
- ▶ $x_{k+1} = x_k + \alpha_k d_k$
- ▶ $r_{k+1} = r_k - \alpha_k A d_k$
- ▶ $\beta_{k+1} = \frac{r_{k+1}^T M^{-1}r_{k+1}}{r_k^T M^{-1}r_k}$
- ▶ $d_{k+1} = M^{-1}r_{k+1} + \beta_{k+1}d_k$

Implementazione Matlab (con preconditionamento)

Dati come input una matrice A , il vettore dei termini noti b , una guess iniziale x , la matrice di preconditionamento M , un numero massimo di iterazioni k_{\max} e una tolleranza ϵ

```
k = 0
r = b-A*x;
d = M\r; %%piu' performante di inv(M)*r
delta = r'*r; delta0 = delta;
while k < kmax && delta > epsilon*delta0
    q = A*d;
    alpha = delta/(d'*q);
    x = x+alpha*d;
    r = r-alpha*q;
    s = M\r;
    deltaold = delta;
    delta = r'*s;
    beta = delta/deltaold;
    d = s+beta*d;
    k=k+1;
end
```

Considerazioni finali

Svantaggi

- In quanto metodi iterativi, sono soggetti a errori di troncamento.

Vantaggi

- Possibilità di implementare questi metodi in versione *matrix-free*, ovvero ad ogni iterazione basta solo calcolare un prodotto matrice-vettore, e altre operazioni meno computazionalmente costose.
- Se la matrice è **sparsa**, è possibile memorizzare i suoi elementi a un costo limitato ($O(n)$ invece di $O(n^2)$). Non è detto però i suoi fattori siano matrici sparse, quindi non è consigliabile applicare metodi di fattorizzazione.

Confronto Gradiente - Gradiente Coniugato

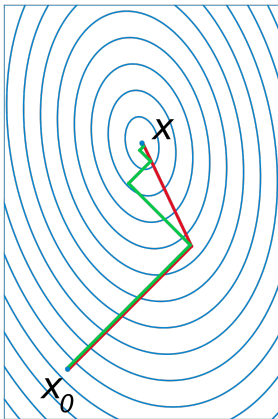


Figura 4: Confronto tra il numero di iterazioni dei due metodi (in verde il gradiente e in rosso il gradiente coniugato)

Come si può osservare dalla figura, il metodo del Gradiente esegue molte più iterazioni rispetto al metodo del Gradiente Coniugato.

Generalized Minimal RESidual

risoluzione di un sistema lineare tramite sottospazio di Krylov

GMRES

- **GMRES** (Generalized Minimal RESidual) è uno dei metodi di Krylov utilizzato per risolvere il sistema lineare $Ax = b$ tramite il sottospazio \mathcal{K}_n in modo tale da rendere minima la norma Euclidea del residuo:

$$\|b - Ax_n\|_2.$$

- Sia K_n la matrice $m \times n$ di Krylov, composta da spazi colonne AK_n . Inanzitutto si moltiplica K_n per A , per cui il problema diventa trovare un vettore $c \in \mathbb{C}^n$, tale per cui il residuo $\|AK_n c - b\|_2$ sia minimo.
- Questa formula è risolvibile con fattorizzazione QR, ma AK_n diventa **numericamente instabile** al crescere di n , quindi si utilizza **l'iterazione di Arnoldi** per costruire una base di vettori ortonormali.

L'iterazione di Arnoldi (I)

- Si definisce Q_n come la matrice, i cui vettori formano una base ortonormale di \mathcal{K}_n . Il problema consiste ora nella ricerca di $y \in \mathbb{C}^n$ tale per cui:

$$\|AQ_n y - b\|_2 = \textit{minimo},$$

riducendo così le dimensioni del problema da $m \times n$ a $(n+1) \times n$.

- L'iterazione di Arnoldi dimostra che $AQ_n = Q_{n+1}\mathcal{H}_n$ (con \mathcal{H}_n la matrice alta-sinistra, $(n+1) \times n$, della matrice di Hessenberg). Il problema quindi diventa:

$$\|Q_{n+1}\mathcal{H}_n y - b\|_2 = \textit{minimo}.$$

L'iterazione di Arnoldi (II)

- Moltiplicando entrambi i membri per Q_{n+1}^* si ottiene:

$$\|\mathcal{H}_n y - Q_{n+1}^* b\|_2 = \textit{minimo}.$$

- Si osserva che $Q_{n+1}^* b = \|b\| e_1$, con $e_1 = (1, 0, 0, \dots)^*$, quindi il problema finale è:

$$\|\mathcal{H}_n y - \|b\| e_1\|_2 = \textit{minimo}.$$

Questa equazione è risolvibile con l'approssimazione ai minimi quadrati.

Generalized Minimal RESidual

applicazione nel calcolo polinomiale

GMRES Polinomi

- ▶ Il metodo GMRES è utilizzabile anche nel calcolo polinomiale.
- ▶ Dato $p_n \in P_n$, con $p_n(0) = 1$ si definisce $x_n = q_n(A)b$ e $r_n = p_n(A)b$. Il problema consiste nel calcolare $p_n \in P_n$ tale che

$$\|r_n\| = \textit{minimo}.$$

Velocità di convergenza

Quanto deve valere n (numero di iterazioni) per soddisfare la tolleranza?

$$\frac{\|r_n\|}{\|b\|} < \text{tolleranza}$$

- Osservazione 1: convergenza monotona $\|r_{n+1}\| \leq \|r_n\|$.
- Osservazione 2: per $n \rightarrow \infty$, $\|r_\infty\| = 0$, a meno di errori di arrotondamento.
- Essendo $\|r_n\| = \|p_n(A)b\| \leq \|p_n(A)\|\|b\|$ risulta che solo $p_n(A)$ influenza la velocità di convergenza:

$$\frac{\|r_n\|}{\|b\|} \leq \inf \|p_n(A)\|,$$

con $p_n \in P_n$.

Analisi stazionaria efficiente basata sul metodo matrix-free nei sottospazi di Krylov

Matrix Free GMRES

Esempio applicativo del metodo: **circuiti analogici**

- ▶ Più circuiti \rightarrow più problemi per risolverli
- ▶ Risposta del circuito dipende dagli ingressi (ampiezze, frequenze) \rightarrow migliaia di punti da verificare
- ▶ N equazioni $\rightarrow N^3$ flops per Gauss, N^2 flops per metodi iterativi
- ▶ Matrix-Free VS Iterativi: se $N=400$ $t_{It} \approx 10t_{MF}$

Matrix Free GMRES

Algoritmo per calcolare la distorsione nei circuiti analogici

- I dati si ricavano simulando il circuito e misurando i valori ottenuti in un periodo T .

$$f(v(t), t) = i(v(t)) + q'(v(t)) + u(t) = 0 \quad (1)$$

$$v(T) - v(0) = 0 \quad (2)$$

- Dalla (1) e (2) si ricava

$$\phi(v(0), 0, T) - v(0) = 0 \quad (3)$$

- Dove

$$\phi = \int_0^T f(v(t), t) dt$$

$${}^0v, i, q, u \in \mathbb{R}^N$$

Matrix Free GMRES

Iterazione di Newton (I)

- Tramite il metodo di Newton alla (3) si ottiene l'iterazione

$$\mathbf{v}_0^j = \mathbf{v}_0^{j-1} - [J_\phi(\mathbf{v}_0^{j-1}, 0, T) - I]^{-1}[\phi(\mathbf{v}_0^{j-1}, 0, T) - \mathbf{v}_0^{j-1}] \quad (4)$$

con $\mathbf{v}_0 = \mathbf{v}(t = 0)$.

- Utilizzando opportuni passaggi(vedi dac95 paragrafo 2) si ricava la formula

$$\left[\frac{C(\mathbf{v}_m^{l-1})}{h_m} + G(\mathbf{v}_m^{l-1}) \right] (\mathbf{v}_m^l - \mathbf{v}_m^{l-1}) = -\frac{1}{h_m} (q(\mathbf{v}_m^{l-1}) - q(\mathbf{v}_{m-1})) - i(\mathbf{v}_m^{l-1}) - u_m \quad (8)$$

Matrix Free GMRES

Iterazione di Newton (II)

Siano

- ▶ $m = \text{istante temporale}$
- ▶ $h_m = t_m - t_{m-1}$
- ▶ $C = \frac{dq}{dv}$
- ▶ $G = \frac{di}{dv}$
- ▶ $J_f = \frac{C(v_m^{l-1})}{h_m} + G(v_m^{l-1})$

La sensibilità si esprime come

$$J_f(v_m) \frac{dv_m}{dv_0} = \frac{C(v_{m-1})}{h_m} \frac{dv_{m-1}}{dv_0} \quad (11)$$

L'iterazione di Newton richiede circa N^2 flops.¹

¹ $N = \dim(v)$

Matrix Free GMRES

Algoritmo (I). Per risolvere $Ax = b$ con GMRES

$x_0 =$ Guess iniziale

$p_0 = b - Ax_0$ (direzione)

$k = 1$

do {

$p^k = Ap^{k-1}$ (nuova direzione)

$p^k = p^k - \sum_{j=0}^{k-1} \beta_{k,j} p^j$ (ortogonalizza)

cerca α_k in $x^k = x^{k-1} + \alpha_k p^k$ tale che

$\|r^k\| = \|b - Ax^k\|$ minimo

$k = k + 1$

} while($\|r^k\| < tolerance_{GMRES}$)

$x^k =$ soluzione

Matrix Free GMRES

- ▶ Il costo elevato dell'iterazione di Newton è causato dal calcolo di $A = J_\phi - I$: $\dim(A) = N^2$
- ▶ Il GMRES matrix free consente di risolvere il sistema senza memorizzare l'intera matrice A
- ▶ Così facendo il calcolo di $J_\phi p^{k-1}$ richiede circa N operazioni²

² $J_\phi p^{k-1}$ serve a calcolare Ap^{k-1} in ogni step k dell'iterazione

Matrix Free GMRES

- $Ap^{k-1} = (J_\phi - I)p^{k-1}$ è approssimabile a meno di un errore ϵ

$$\frac{\phi(v_0 + \epsilon p^{k-1}, 0, T) - \phi(v_0)}{\epsilon} - p^{k-1} \quad (12)$$

- L'algoritmo utilizzato per risolvere la (12) richiede lo stesso numero di operazioni necessario a calcolare una sola colonna di J_ϕ con il metodo di Newton
- Salvare $C(v_m)$ e fattorizzare LU $J_f(v_m)$ ad ogni passo per applicare:

$$J_f(v_m) \frac{dv_m}{dv_0} = \frac{C(v_{m-1})}{h_m} \frac{dv_{m-1}}{dv_0} \quad (11)$$

Matrix Free GMRES

Algoritmo (II). Per risolvere $Ax = b$ matrix free

$v_0 =$ Guess iniziale

For $j = 1$ to Max Newton {

$\int_0^T (1) dt$ con $v_0 = v_0^{j-1}$

For $m = 1$ to M {

Risolvi la (6)

Salva $J_f(v_m)$ e $C(v_m)$

}

Risolvi $(J_\phi(v^{j-1}) - I)\delta v^j = v^{j-1} - \phi(v^{j-1})$ con GMRES

Nel GMRES calcola $p^{k+1} = J_\phi(v^{j-1})p^k$ usando

$p^{k+1} = p^k$

For $m = 1$ to M risolvi $J_f(v_m)p^{k+1} = C(v_{m-1})p^{k+1}$

$v^j = v^{j-1} + \delta v^j$

se $\|\delta v^j\| < \text{tolleranza}_{Newton}$ return

}

Matrix Free GMRES

circuit	eqns	it	GE	GMRES	MF	GE/MF
xtal	29	3	0.50	0.50	0.39	1.28
mixer	24	4	1.85	1.74	1.20	1.54
dbmixer	100	4	4.15	4.07	1.34	3.09
lmixer	126	3	3.72	3.63	1.03	3.61
cheby	237	4	23.39	21.97	3.01	7.96
scf	377	6	2962	2954	281.4	10.52

Figura 5: Comparazione⁴ di metodi diversi per la risoluzione dei circuiti elettronici.

⁴Risultati ottenuti con HP712/80 workstation.

Matrix Free GMRES

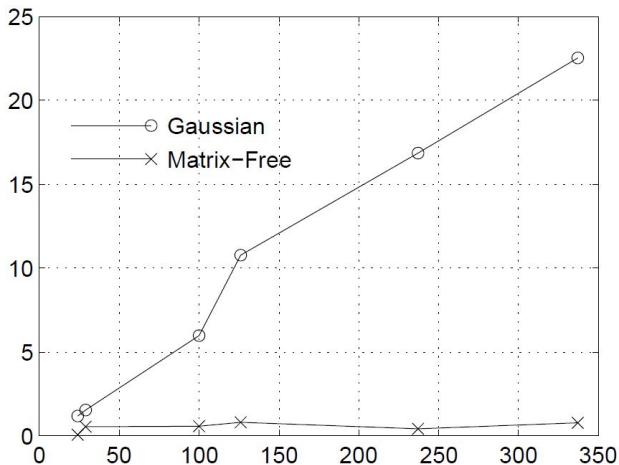


Figura 6: Rapporto tra lo "shooting update time" e il costo di elaborazione del calcolo di una singola operazione.

Matrix Free GMRES

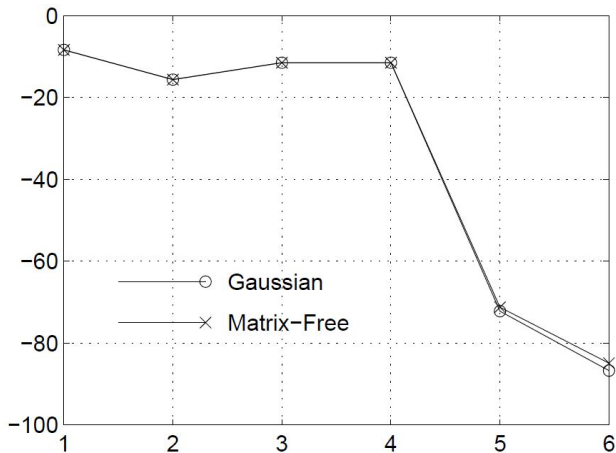


Figura 7: Il grado di convergenza del metodo di Newton comparato al metodo Matrix Free.