# Metodi di Krylov

Loredana Cascarano

Luglio 2024

# Introduzione ai Metodi di Krylov

- I metodi di Krylov sono algoritmi iterativi per la risoluzione di grandi sistemi lineari $Ax = b$ e problemi agli autovalori.
- L'idea chiave è di cercare una soluzione approssimata all'interno di un sottospazio di Krylov, generato da:

$$K_m(A, b) = \text{span}\{b, Ab, A^2b, \ldots, A^{m-1}b\}$$

- Sono particolarmente utili quando la matrice $A$ è di grandi dimensioni o sparsa.

# Sottospazi di Krylov

- Un sottospazio di Krylov di ordine $m$ associato alla matrice $A$ e al vettore $b$ è definito come:

$$K_m(A, b) = \text{span}\{b, Ab, A^2b, \ldots, A^{m-1}b\}$$

- È un sottospazio di dimensione al massimo $m$ che contiene informazioni utili sulla soluzione del sistema lineare.
- I metodi di Krylov sfruttano questi sottospazi per trovare soluzioni approssimate efficienti.

# Metodo di Arnoldi

- L'iterazione di Arnoldi è un metodo di Krylov che costruisce una base ortonormale per il sottospazio di Krylov $K_m(A, b)$.
- Questa base viene utilizzata per ridurre il problema originale ad un problema di dimensioni molto minori, coinvolgendo una matrice di Hessenberg superiore.
- La matrice di Hessenberg superiore ottenuta ha gli stessi autovalori di $A$, facilitando l'approssimazione degli autovalori e autovettori di $A$.

# Matrice Upper-Hessenberg

- Una matrice $H$ è detta upper-Hessenberg se ha elementi nulli sotto la prima sottodiagonale: $h_{ij} = 0$ per $i > j + 1$.
- Esempio:

$$H = \begin{pmatrix} h_{11} & h_{12} & h_{13} & \cdots & h_{1n} \\ h_{21} & h_{22} & h_{23} & \cdots & h_{2n} \\ 0 & h_{32} & h_{33} & \cdots & h_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & h_{nn} \end{pmatrix}$$

- Questa struttura è cruciale nell'iterazione di Arnoldi e semplifica il calcolo degli autovalori.

# Implementazione del Metodo di Arnoldi (Python)

```python
import numpy as np

def arnoldi_iteration(A, b, n):
    m = len(A)
    H = np.zeros((n+1, n))
    Q = np.zeros((m, n+1))
    #Normalizzazione del vettore iniziale
    Q[:, 0] = b / np.linalg.norm(b)
    for k in range(n):
        v = np.dot(A, Q[:, k])
        for j in range(k+1):
            H[j, k] = np.dot(Q[:, j].T, v)
            #Ortogonalizzazione di Gram-Schmidt
            v = v - H[j, k] * Q[:, j]
        H[k+1, k] = np.linalg.norm(v)
        if H[k+1, k] != 0 and k+1 < m:
            #Normalizzazione
            Q[:, k+1] = v / H[k+1, k]
    return Q, H
```

# Esempio di Applicazione

```
A = np.array([[4, 1, 0],
              [1, 4, 1],
              [0, 1, 4]])
b = np.array([1, 0, 0])

Q, H = arnoldi_iteration(A, b, 2)

print("Q:\n", Q)
print("H:\n", H)
```

# Conclusione

- I metodi di Krylov sono strumenti potenti per risolvere problemi lineari di grandi dimensioni.
- L'iterazione di Arnoldi è un metodo fondamentale per la costruzione di sottospazi di Krylov e la riduzione della dimensionalità del problema.
- L'implementazione in Python dimostra la praticità di questi metodi nella risoluzione di problemi reali.