PRESENTAZIONE FONDAMENTI DI CALCOLO NUMERICO

~~[SLIDE -> Calcolo del residuo]~~

~~Come già visto, il residuo si può ricavare dall’applicazione della matrice A all’errore. Dato che l’errore può essere valutato mediante il passo di discesa nel seguente modo, sostituendo e risolvendo l’equazione, otteniamo che il residuo può essere calcolato utilizzando il residuo e la direzione precedente.~~
~~Questa formula non è un’approssimazione, però è soggetta ad errori di troncamento a causa della macchina; soprattutto se la matrice ha una dimensione elevata. Questo errore di troncamento può essere rimosso applicando, ogni tot iterazioni, la formula originale (che utilizza il vettore dei termini noti, e il vettore della soluzione approssimata a quel passo).~~

[RELATORE -> Lorenzo]

[SLIDE -> Direzione di discesa]

Abbiamo già visto precedentemente il sottospazio di Krylov K, creato applicando ripetutamente una matrice a un vettore.
Questo sottospazio ha una proprietà importante: infatti se sappiamo che A K è incluso nel sottospazio K del passo successivo e che il residuo del passo successivo è ortogonale al sottospazio K del passo successivo, allora il residuo del passo successivo è A-ortogonale ad A K.
Di conseguenza il residuo del passo successivo è anche A-ortogonale alle precedenti direzioni di discesa d.
Quindi l’algoritmo di Gram-Schimdt si semplifica e la formula per calcolare la nuova direzione di discesa diviene quella qui mostrata.
È importante notare che non sarà più necessario memorizzare le vecchie direzioni di ricerca.
È proprio questo il punto di forza del complesso coniugato: non dovendo memorizzare le vecchie direzioni, la complessità dello spazio e del tempo si riduce.
Precisamente si riduce da O-grande della dimensione al cubo della matrice, a O-grande degli elementi non nulli della matrice, quindi lineare
È anche interessante notare, che non sono i gradienti a essere coniugati, e che le direzioni non rappresentano la totalità dei gradienti; perciò sarebbe stato più corretto chiamare questo metodo più che del gradiente coniugato delle direzioni coniugate.

[SLIDE -> Passo di discesa (II)]

Alla luce di queste considerazioni possiamo vedere (anche attraverso la figura due) che a ogni passo il residuo e la direzione puntano allo stesso sottospazio.
Questo sottospazio è parallelo a quello di Krylov da cui sono partiti. Nella figura il sottospazio è piano, perché siamo alla seconda iterazione.
Di conseguenza questi due prodotti scalari si eguagliano, e possiamo riscrivere il numeratore della formula per il calcolo del passo di discesa.
Questa sostituzione, ci permette di migliorare le prestazioni dell'algoritmo.

[SLIDE -> Velocità di convergenza]

È complicato effettuare accurate previsioni della convergenza dei metodi iterativi; comunque è possibile ottenere dei limiti utili. Possiamo infatti ottenere la seguente formula dove K di A è il numero di condizionamento della matrice.
Il numero di condizionamento è il rapporto tra il più grande e il più piccolo degli autovalori della matrice. Questo rappresenta il rapporto tra errore commesso sul risultato di un calcolo e incertezza sui dati in ingresso.
~~Un problema è *ben condizionato* quando la soluzione del problema con delle piccole variazioni, non differisce molto dalla soluzione del problema originale.~~
Utilizzando i metodi iterativi raramente ci si trova nella situazione di dover ricercare la soluzione esatta, e molte volte ci si arresta fissando una tolleranza rispetto al residuo.
~~Per la nostra implementazione lavoreremo fissando una tolleranza sul residuo quadrato in modo da evitare il calcolo di una radice quadrata.~~

[SLIDE -> Precondizionamento (I)]

Il precondizionamento è un metodo per migliorare appunto il numero di condizionamento di una matrice: il nostro obbiettivo è quindi quello di raggruppare gli autovalori, ovvero minimizzare la variazione tra l'autovalore maggiore e quello minore.
Intuitivamente lo scopo del precondizionamento è quello di rendere la forma quadratica più sferica.
Per applicare il metodo dobbiamo calcolare il sistema originale A x uguale b mediante la prima equazione che vedete.
Il problema di questa equazione è che M alla meno uno in generale non è simmetrica o definita, ma possiamo comunque trovare una matrice P tale

che P per la sua trasposta sia uguale a una matrice M simmetrica e

instance, by Cholesky factorization.) The matrices $M^{-1}A$ and $E^{-1}AE^{-T}$ have the same eigenvalues. This is true because if v is an eigenvector of $M^{-1}A$ with eigenvalue λ, then $E^T v$ is an eigenvector of $E^{-1}AE^{-T}$ with eigenvalue λ:

$$(E^{-1}AE^{-T})(E^T v) = (E^T E^{-T})E^{-1}Av = E^T M^{-1}Av = \lambda E^T v.$$

definita positiva.
Il sistema equivalente che ne deriva è questo in fondo, e avremo che l'inversa di P per la matrice A per la traslazione dell'inversa P è simmetrica e definita positiva.

[SLIDE -> Precondizionamento (II)]

Nella scelta di M stiamo quindi realizzando un compromesso tra la diminuzione del numero di condizionamento, e il calcolo del residuo precondizionato.
Questo trade-off è conveniente al crescere della dimensione di A, quindi conviene spesso calcolare la matrice M; ad esempio attraverso Jacobi, ovvero il precondizionamento diagonale, in cui M presenta sulla diagonale gli stessi elementi della diagonale di A; e il suo effetto è scalare la matrice lungo gli assi delle coordinate.
Più avanzato è il precondizionamento incompleto di Cholesky. Questo sfrutta la fattorizzazione di Cholesky per fattorizzare A nella forma L per L traslata dove la matrice L è una matrice triangolare bassa.
Della matrice L per il precondizionamento vengono tenuti solo gli elementi che si trovano nelle stesse posizioni degli elementi non nulli della matrice A.
Sfortunatamente il metodo di Cholesky non è sempre stabile.

[SLIDE -> Metodo del gradiente coniugato (senza precondizionamento)]

In questa slide raggruppiamo tutte le formule necessarie per il nostro metodo iterativo.
Con questa visione d'insieme è perciò semplice tradurre il linguaggio matematico in codice Matlab.

[SLIDE -> Implementazione Matlab (senza precondizionamento)]

Notiamo che abbiamo fissato come condizioni per il ciclo un numero massimo di iterazioni e una tolleranza rispetto alla norma quadrata del residuo.

[SLIDE -> Metodo del gradiente coniugato (con precondizionamento)]

Analogamente in questa slide abbiamo raggruppato tutte le formule necessarie per il Metodo del gradiente coniugato con precondizionamento.
Di conseguenza il nostro codice Matlab, è di nuovo facilmente ottenibile.

[SLIDE -> Implementazione Matlab (con precondizionamento)]

Come detto prima notiamo che a ogni iterazione viene calcolato il residuo precondizionato s.

[SLIDE -> Considerazioni finali su gradiente e gradiente coniugato]
A fronte di possibili errori di troncamento, comunque evitabili da ciò che abbiamo visto, abbiamo vantaggi importanti.
In definitiva, quello che possiamo concludere, è che attraverso questi metodi (ovvero del gradiente e del gradiente coniugato) possiamo evitare il calcolo dell'inverso della matrice, molto costoso computazionalmente.
Inoltre, diversamente dai metodi che prevedono fattorizzazione; nei casi di matrice sparse manteniamo il vantaggio di allocazione di memoria che queste ci forniscono.
Questi vantaggi ci permettono di allocare meno memoria: sono convenienti specialmente in problemi di grandi dimensioni, e sono utili in ottica di parallelismo su più CPU.

[SLIDE -> Confronto Gradiente - Gradiente Coniugato]

Confrontando il metodo del gradiente e il metodo del gradiente coniugato possiamo vedere come nel 99% dei casi converga prima il gradiente coniugato.
Dico 99% perché in pochi casi fortunati il gradiente converge alla prima iterazione.
Ad esempio può succedere che se la matrice A ha uguali autovalori otteniamo la soluzione alla prima iterazione. Si può capire intuitivamente perché se la matrice A ha uguali autovalori le linee di livello della forma quadratica sono circolari; e il punto di minimo si trova sulla direzione del gradiente.
Oppure, se il termine di errore in qualche passo ha la direzione di un autovettore, la soluzione converge al passo successivo.