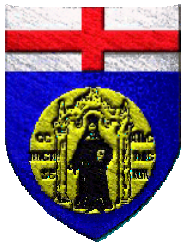
UNIVERSITA' DEGLI STUDI DI GENOVA

FACOLTA' DI INGEGNERIA



25 Gennaio 2001

TESI DI LAUREA

Ottimizzazione delle variabili caratteristiche di un magnete superconduttore mediante analisi FEM pilotate da Algoritmi Genetici

RELATORE : Chiar.mo Prof. Ing. Alessandro Rebora

CORRELATORE : Dott. Stefania Farinon

ALLIEVO : Luca Reina

Optimum design of a superconducting magnet through FEM analyses driven by a Genetic Algorithm

Keywords: optimum design, FEM analysis, DOE, genetic algorithms

Abstract

This work dealt with the optimum design of the coil layout of a superconducting magnet. This magnet will be used to deflect heavy particles for cancer therapy. The goal of the optimization was the improvement of the magnetic field uniformity, needed to guide the particle beam onto the target with a very high precision. At first, a Design of Experiment (DOE) was carried out in order to evaluate the importance of each design variable. The optimization was performed using FEM analyses driven by a Genetic Algorithm. This approach was chosen because the topology of the response surface would have deceived a traditional gradient method. The 2D analysis showed that the design requirements were fulfilled: the magnetic field uniformity was brought to 2‰. The 3D analysis results perfectly agree with 2D calculations in the middle of the magnet. Magnetic field results obtained by the 3D analysis show that a study of the geometry of the magnet ends will be needed in order to optimize the trajectory of the particle beam.

Ringraziamenti

Ringrazio tutto il gruppo di lavoro dell'INFN con il quale ho trascorso questi mesi, per avermi dato la possibilitá di compiere un'esperienza formativa ed estremamente gratificante.

Un ringraziamento particolare desidero porgerlo a Stefania Farinon per la competenza e la grande simpatia dimostrate, e soprattutto per l'incommensurabile aiuto che mi ha fornito, dedicandomi il suo tempo in ogni situazione.

Un ringraziamento sentito va al Prof. Alessandro Rebora, per la grande disponibilitá e per la fiducia riposta in me, grazie alla quale ho potuto affrontare questo lavoro con tecniche innovative.

Un doveroso ringraziamento a Pasquale Fabbricatore e a Riccardo Musenich, per l'aiuto e la competenza dimostrate, e a tutti coloro che hanno gentilmente messo a disposizione il loro tempo.

Infine, ma non per ultimo, un ringraziamento speciale ai miei genitori per avermi dato la possibilitá di giungere sino alla laurea, accompagnandomi lungo il percorso e condividendo con me sia le gratificazioni, sia i sacrifici.

“La conoscenza é lo strumento piú potente e straordinario
che l’Uomo puó avere nelle proprie mani.
La forza di volontá é il suo piú efficace complemento.”

Luca Reina

Ai miei genitori

INDICE

Prefazione

Oggetto di questa tesi è stata l'ottimizzazione dei parametri caratteristici di un magnete superconduttore di una testata isocentrica (*gantry*), una struttura meccanica rotante che, muovendo la parte terminale di una linea di trasporto di particelle pesanti (adroni), permette di irradiare da differenti direzioni un paziente affetto da tumore, tecnica nota con il nome di *Adroterapia*.

Tale studio è stato motivato dalla necessità di disporre, per la deflessione del fascio di particelle, di un magnete di tipo superconduttore. Tali magneti sono caratterizzati da peso e da ingombri nettamente inferiori rispetto a quelli dei magneti convenzionali, gli unici attualmente impiegati in questo tipo di applicazioni. Questa soluzione permette di ridurre sensibilmente le dimensioni della struttura meccanica di sostegno, con conseguente abbattimento dei costi dell'apparecchiatura.

Obiettivo dell'ottimizzazione è stata la determinazione della geometria dei conduttori, della loro disposizione e delle correnti in esse circolanti, al fine di ottenere un campo magnetico il più omogeneo possibile, per poter "guidare" il fascio di particelle sul bersaglio tumorale con elevata precisione.

E' stata quindi condotta un'analisi strutturale per valutare le deformazioni indotte dalle forze di Lorentz sulla struttura meccanica di contenimento delle bobine.

I passi fondamentali seguiti possono essere così sintetizzati:

- costruzione di un modello agli Elementi Finiti 2D ed uno 3D completamente parametrici;

- esplorazione preliminare delle configurazioni progettuali possibili, per l'individuazione delle variabili più significative. Questa fase è stata condotta con algoritmi per la progettazione di piani di esperimenti, tecniche note con il nome di *Design of Experiment* (DOE);
- ottimizzazione mediante simulazioni numeriche agli Elementi Finiti pilotate da Algoritmi Genetici;
- affinamento locale della ricerca attraverso algoritmi gradientali tradizionali.

Il primo capitolo introduce la tecnica dell'Adroterapia e fornisce una panoramica sulle apparecchiature utilizzate e sulle difficoltà tecnologiche ad esse connesse.

Nel capitolo 2 viene descritto il magnete superconduttore oggetto di questa tesi, dalle specifiche tecniche sino alla costruzione del modello parametrico agli Elementi Finiti utilizzato per simularne il comportamento.

Il capitolo 3 descrive la tecnica del Design of Experiment.

Il capitolo 4 è interamente dedicato agli Algoritmi Genetici, dalle origini storiche alle implementazioni più recenti.

Il capitolo 5 tratta dell'ottimizzazione tramite l'algoritmo genetico: si descrive in dettaglio l'implementazione, i file di input e di output e il test su funzioni note. Questo capitolo si propone di fornire al lettore tutte le conoscenze necessarie per utilizzare questo potente strumento matematico qualora ne avesse la necessità.

Il capitolo 6 descrive in maniera esaustiva tutte le tecniche di ottimizzazione fornite dal codice ANSYS. Viene inoltre descritto l'affinamento della configurazione del magnete effettuata con gli algoritmi gradientali di ANSYS a partire dal risultato ottenuto con l'algoritmo genetico.

Gli ultimi due capitoli sono dedicati, rispettivamente, al modello FEM tridimensionale del magnete e all'analisi delle sollecitazioni meccaniche indotte dalle forze elettromagnetiche sulla struttura di contenimento delle bobine.

Questo lavoro, commissionato dalla fondazione TERA, è stato svolto presso l'Istituto Nazionale di Fisica Nucleare in collaborazione con il Dipartimento di Costruzione di Macchine dell'Università di Genova.

La fondazione TERA, fondata nel 1992, è uno dei pochi enti non-profit riconosciuti dal Ministero della Sanità, ed ha come scopo lo sviluppo, in Italia e all'estero, delle tecniche di radioterapia basate sull'uso di particelle adroniche e, più in generale, delle applicazioni della fisica alla medicina e alla biologia.

1

Introduzione

1.1 Storia e sviluppo dell'adroterapia

Il termine "adroterapia" fu utilizzato e accettato per la prima volta al First International Symposium on Hadrontherapy, tenuto a Como nell'ottobre 1993, in precedenza erano stati utilizzati i termini di *radioterapia con particelle pesanti, terapia con particelle, terapia con radiazioni adroniche.*

Per adroterapia si intende la moderna tecnica di radioterapia oncologica che utilizza le radiazioni prodotte da tutte le particelle non elementari fatte di quark: i protoni e i neutroni sono gli adroni più noti. Anche gli ioni, nuclei di atomi elettricamente carichi, sono adroni, ma sono attualmente meno impiegati in radioterapia dei protoni e dei neutroni.

Gli adroni possiedono caratteristiche fisiche che consentono un trattamento di tumori e di malformazioni arterovenose intracerebrali "conforme" al bersaglio che si vuole raggiungere: fasci ben collimati di protoni e ioni distruggono le cellule tumorali con precisione millimetrica, risparmiando i tessuti sani circostanti. Ciò risulta invece impossibile con le radiazioni convenzionali (elettroni e raggi X) che possiedono una precisione solo centimetrica.

I protoni sono particolarmente indicati per trattare tutti quei tumori che sono

vicini ad organi critici che non devono essere irradiati, gli ioni carbonio sono indicati per distruggere - sempre con precisione millimetrica - anche i tumori che sono "radioresistenti" sia ai raggi X che ai protoni.

La disponibilità degli adroni per uso terapeutico è sorta con l'introduzione di acceleratori i cui parametri principali (variazione di energia e intensità di corrente) sono stati ottimizzati per l'uso clinico. L'acceleratore più idoneo a tal fine è un acceleratore lineare (linac) che viene comunemente usato nel mondo della fisica delle particelle come iniettore di acceleratori circolari (ciclotroni e sincrotroni). Un linac possiede al suo interno una sorgente di particelle a radiofrequenza. Le particelle sono focalizzate tramite lenti elettrostatiche in un magnete dove vengono impacchettate e accelerate e successivamente iniettate nell'acceleratore circolare. Questo è in grado di produrre all'estrazione fasci adronici di elevata energia (300 MeV per i protoni, 4500 MeV per gli ioni) mediante una catena di elementi magnetici che focalizzano e curvano la traiettoria delle particelle. La caratteristica peculiare di un sincrotrone è quella di poter variare l'energia di incidenza del fascio in piccoli intervalli da un impulso a quello successivo, in accordo con le esigenze cliniche. Questa possibilità lo fa preferire ad un ciclotrone che è in grado sì di accelerare il fascio ai valori richiesti dall'uso terapico, ma con una energia costante. Considerando inoltre che l'utilizzo di un unico linac come acceleratore (che arrivi ad almeno 375 MeV) comporterebbe strutture lunghissime, la soluzione migliore è l'uso combinato di un linac e di un sincrotrone che permette, conservando quelle caratteristiche del fascio richieste dall'uso terapico, l'ottimizzazione della superficie occupata, la minimizzazione delle dimensioni e del numero di magneti necessari.

Contributo importante allo sviluppo dell'adroterapia è stato anche l'affinamento del sistema di distribuzione della dose posto al termine della linea di trasporto del fascio. Questa struttura, chiamata *nozzle*, comprende i dispositivi di somministrazione del fascio, la strumentazione dosimetrica, i collimatori, il tubo a raggi X per il controllo della centratura, i laser per l'allineamento del paziente e, se necessario altri dispositivi quali i filtri compensatori.

Tale sistema è indispensabile sia per la terapia con fasci fissi orizzontali sia per quella che prevede l'uso di testate isocentriche rotanti (gantry).

Si vuole puntualizzare che la disponibilità dei soli fasci fissi (che implica la necessità di allineare il paziente al fascio di radiazioni, e non il contrario) e per la quasi totalità dei casi, di energie relativamente basse, inferiori a 100 MeV e quindi non utili per il trattamento di lesioni profonde, ha limitato per molto tempo le indicazioni cliniche della terapia con gli adroni.

1.2 Confronto con la radioterapia convenzionale

La ragione prima che giustifica l'uso in radioterapia di fasci di adroni elettricamente carichi (e cioè protoni e ioni) è la favorevole distribuzione della dose di energia assorbita in profondità (fig.1.1).

Il picco di energia visibile in fig.1.1 è noto con il nome di *Picco di Bragg*. Protoni e ioni, essendo particelle pesanti (la massa di un nucleide è circa 1800 volte quella di un elettrone) ed elettricamente cariche, quando penetrano nella materia rallentano ma non deviano molto dalla direzione iniziale e presentano un picco di dose stretto e alto alla fine del loro percorso, diversamente da quello che accade con i raggi X. Il picco di Bragg è troppo stretto per coprire un tumore di qualche centimetro di diametro; si può però ottenere un picco allargato (*Spread Out Bragg Peak = SOBP*) sovrapponendo molti picchi stretti dovuti ad adroni carichi (protoni oppure ioni); occorre quindi variare l'energia del fascio di adroni carichi, riducendola in piccoli passi, o inserendo nel fascio spessori variabili di materiale, oppure variando l'energia dell'acceleratore (rispettivamente sistema passivo e attivo) in modo da ottenere l'effetto voluto (fig.1.2 che si riferisce a un fascio di ioni).

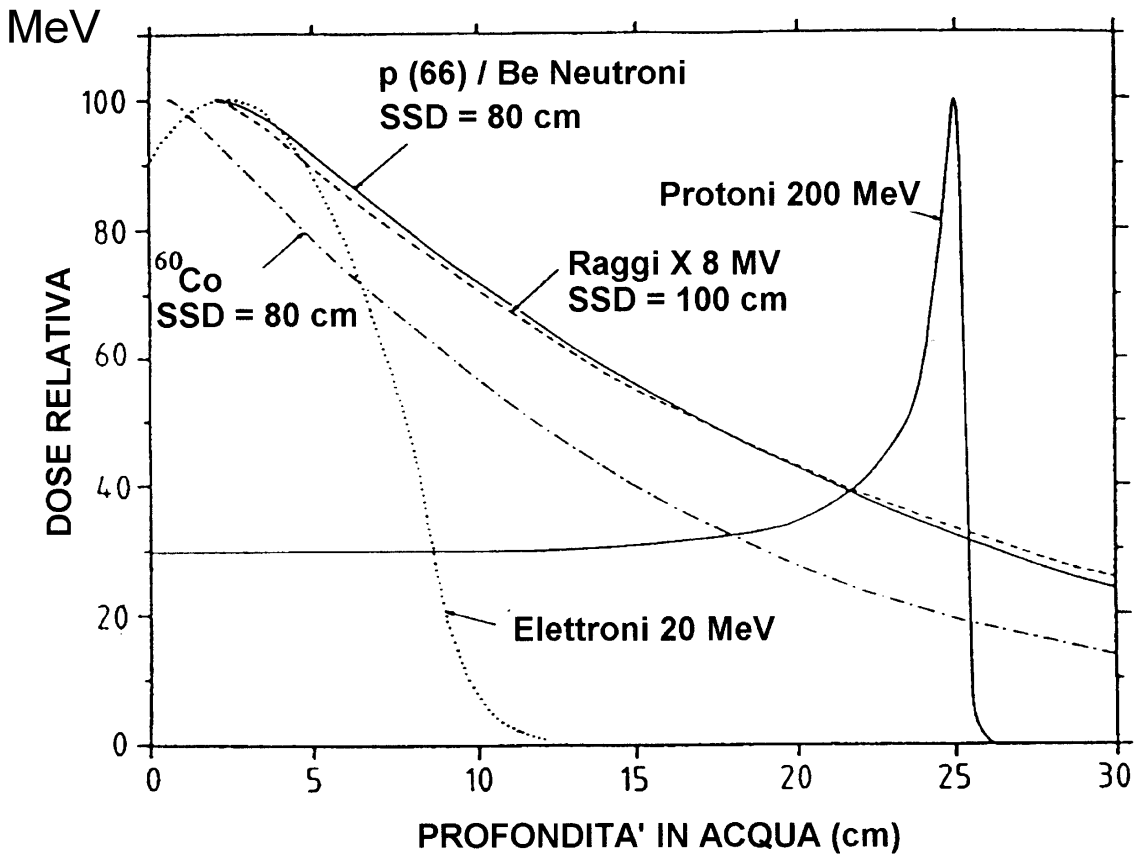


Fig.1.1 *Curve dose-profondità per neutroni veloci, protoni da 200 MeV, per fotoni (e cioè per raggi X prodotti da un linac per elettroni da 8 MeV) e per raggi gamma emessi da una sorgente di Cobalto. Il picco di Bragg dei protoni è alto e stretto perché i protoni sono pesanti, elettricamente carichi e monomagnetici.*

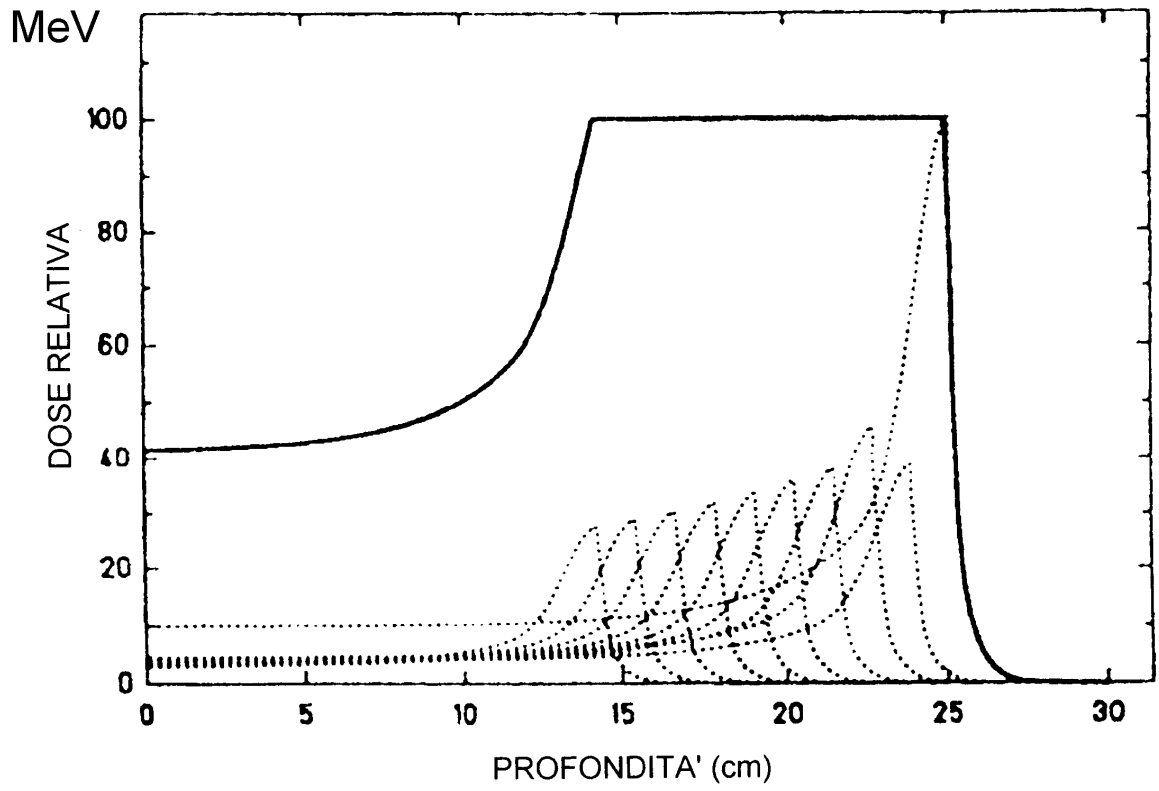


Fig.1.2 *Un picco di Bragg allargato (Spread Out Bragg Peak = SOBP) è ottenuto come sovrapposizione di molti picchi stretti dovuti ad adroni carichi (protoni oppure ioni); la necessaria variazione di energia si ottiene o inserendo nel fascio spessori variabili di materiale, oppure variando l'energia dell'acceleratore (rispettivamente sistema passivo e attivo).*

Grazie al caratteristico picco di Bragg, il fascio di adroni rilascia la dose al tumore con grande selettività**,** cioè cede la maggior parte della propria energia distruttiva (indicata in rosso nella fig.1.3*)* al bersaglio, recando in tal modo meno danni ai tessuti sani circostanti e permettendo al tempo stesso di fornire una dose molto elevata, cosa non ottenibile con la radioterapia convenzionale a raggi X (che infatti cedono la maggior parte dell'energia in gioco ai tessuti sani più esterni, rilasciandone al tumore solo una piccola parte). Questa proprietà dell'adroterapia è particolarmente importante in casi nei quali il tumore è localizzato presso organi vitali che non devono essere irradiati. Inoltre nella terapia convenzionale la

distanza tra il magnete d'uscita del fascio e il corpo del paziente deve essere ridotta al minimo. Ciò non avviene con gli adroni che invece consentono un certo "spazio libero". Infatti, a parità di carica, le forze repulsive degli adroni, essendo inversamente proporzionali al quadrato del raggio medio della particella, sono nettamente minori rispetto a quelle degli elettroni, che cominciano a disperdersi non appena cessa l'azione del campo magnetico.

Oltre al fatto che con gli adroni carichi, dotati di una specifica selettività fisica, si possono effettuare trattamenti "conformi" (con precisione millimetrica) al bersaglio individuato dal radioterapista, vi è un'altra ragione per la quale si utilizzano tali particelle.

Infatti, già a metà degli anni trenta, fu dimostrato che per un gran numero di sistemi biologici i fasci di adroni hanno efficacia biologica relativa (EBR) diversa e quasi sempre maggiore dei raggi X. Per "efficacia biologica" si intende il danno biologico arrecato dalle particelle alle molecole colpite, fortemente dipendente dal tipo di radiazione utilizzato. Gli effetti biologici del passaggio di radiazioni ionizzanti in un mezzo biologico sono principalmente dovuti alla ionizzazione degli atomi e delle molecole. Questi processi fisici, che avvengono in tempi molto brevi possono agire direttamente sulle biomolecole (in particolar modo sul DNA), rompendone i legami chimici, oppure possono produrre radicali liberi per idrolisi dell’acqua contenuta in alta percentuale nel tessuto umano. Fino a un certo limite di energia irradiante, le molecole tumorali spezzate sono rimpiazzate mediante effetti di natura biologica (rigenerazione delle cellule) e pertanto non si osserva alcuna alterazione evidente nei tessuti tumorali.

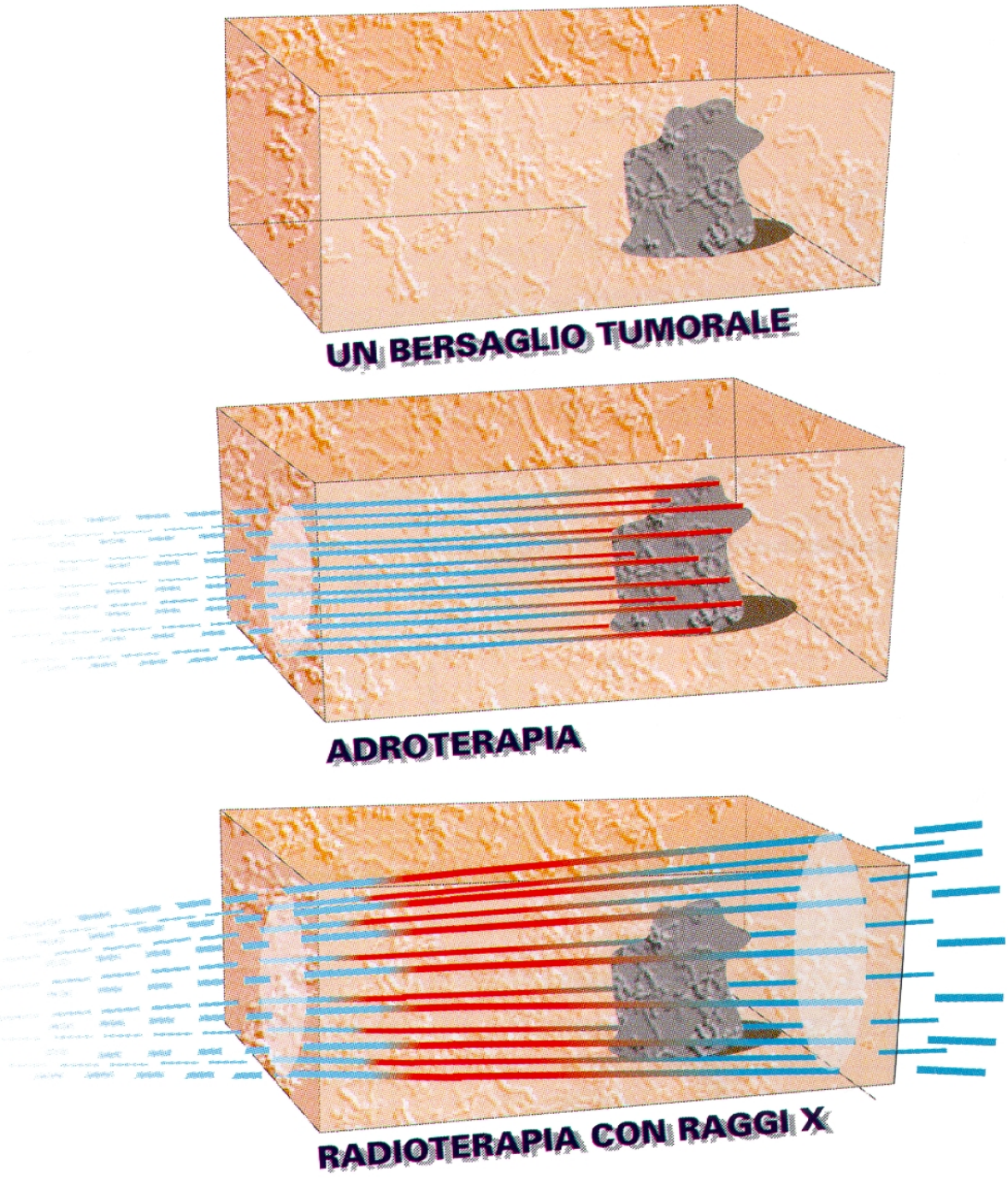


Fig.1.3 *confronto tra terapia con adroni e con raggi X.*

Se la "riparazione" invece non ha successo, o risulta incompleta, gli effetti che si possono manifestare sono i seguenti:

1. morte della cellula;
2. danneggiamento delle funzionalità principali della cellula, come ad esempio le capacità riproduttive;
3. alterazione permanente della cellula che si trasmette alla successiva generazione, cioè effetti genetici.

L'entità e la natura dell'effetto risultano strettamente correlate al tipo di radiazione ionizzante.

Ad esempio una dose di protoni riesce a spezzare in maniera più profonda, rispetto ad una dose equivalente di elettroni o di raggi X, le molecole del DNA di una cellula tumorale.

La differenza è racchiusa nel *LET (Linear Energy Transfert)* delle varie particelle, cioè nella differente quantità di dose ceduta per unità di lunghezza di percorso. Le radiazioni intensamente ionizzanti, cioè ad alto LET come gli adroni, producono la rottura contemporanea di numerosi legami nelle biomolecole, impedendone la riparazione, mentre per bassi valori di LET (è il caso della radioterapia convenzionale) si ha un effetto distruttivo più basso, con maggior probabilità che le cellule irradiate si ristabiliscano.

1.3 Sistemi attivi e passivi di variazione dell'energia

In alcuni acceleratori di particelle (sincrotroni o linac) l'energia degli adroni può essere variata cambiando i parametri della macchina (sistema attivo). Nel caso dei ciclotroni, che invece accelerano adroni a energia fissa, è necessario porre sul percorso del fascio assorbitori di spessore variabile per ottenere un SOBP (sistema passivo).

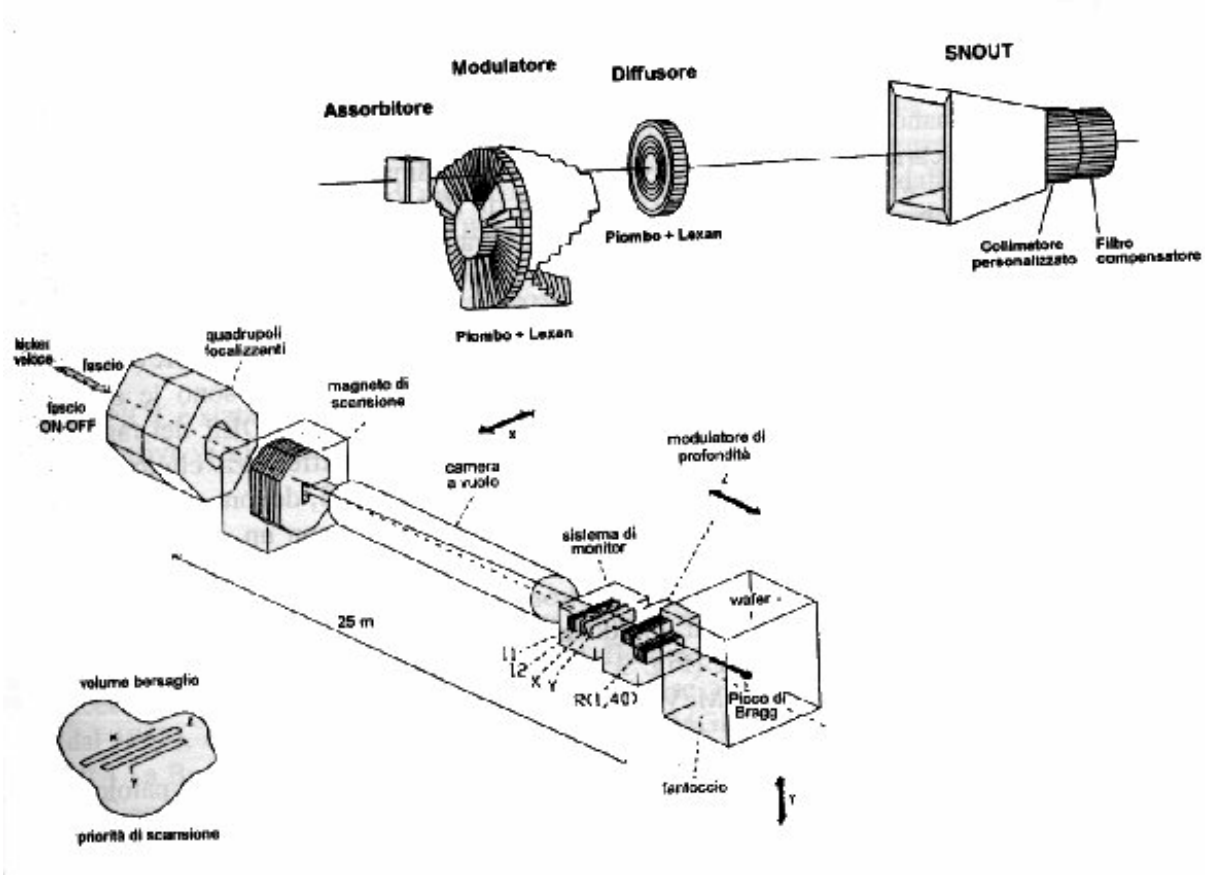


Fig.1.4 *La figura mostra in modo schematico i metodi passivi e attivi di distribuzione della dose di energia.*

Sino al 1995 tutti i trattamenti sono stati fatti con sistemi passivi di distribuzione della dose. Come mostrato nella parte alta della fig.1.4, gli adroni sono diffusi da un primo *"scatterer"* per appiattire la distribuzione di dose; poi il fascio è collimato e l'energia adattata alla forma del tumore nella direzione del fascio con opportuni assorbitori. Dal 1996 due sistemi attivi molto sofisticati sono entrati in funzione presso due centri di ricerca fisica nucleare: il PSI di Zurigo (protoni) e il GSI presso Darmstad (ioni).

In particolare al PSI è entrata in funzione la prima testata dotata di un sistema a scansione per il fascio di protoni: il bersaglio è suddiviso in molte migliaia di punti e ciascuno di essi viene irradiato in successione (se necessario in più pennellate successive) inviando un fascio, di energia e direzione determinata, che ha una sezione di circa 5 mm. La tecnica è illustrata schematicamente nella parte

inferiore della fig.1.4. Si pensa che le tecniche del futuro per i tumori profondi saranno tutte basate su sistemi attivi di distribuzione della dose, che sfruttano il fatto che gli adroni sono elettricamente carichi e possono quindi essere diretti a piacere per mezzo di campi magnetici (cosa che non è possibile con i raggi X, che sono elettricamente neutri).

1.4 Tecniche di Adroterapia

Variando l'energia di un fascio di adroni è possibile sovrapporre molti picchi di Bragg e ottenere una distribuzione della dose in profondità come quella della fig.1.2, che si riferisce agli ioni carbonio. La dose di energia ceduta scende dall'80% al 20% in qualche millimetro. E' questa selettività fisica che permette di somministrare una dose "conforme" al bersaglio scelto anche con un solo campo di irradiazione; in altre parole, modellando la profondità del suddetto picco, è possibile distribuire approssimativamente il 100% dell'energia al centro del tumore, l'80% sulle zone più esterne, il 40% sulle cellule sane a stretto contatto con quelle tumorali, e una percentuale trascurabile ai restanti tessuti sani circostanti. Per aumentare la precisione e per far sì che tutta la massa tumorale venga raggiunta dagli adroni, occorre una radiazione incidente da più angoli in modo da avere una ricostruzione tridimensionale perfetta del volume del neoplasma; ciò si ottiene utilizzando dei sistemi meccanici di distribuzione del fascio di particelle basati sull'impiego di strutture rotanti.

1.5 Terapia con fasci di ioni

Le caratteristiche che rendono vantaggioso, rispetto ai protoni, l'uso dei fasci di ioni sono un'ancora più ridotta diffusione laterale ed un LET (tasso di perdita di energia delle particelle nella materia) molto più elevato. Infatti, a parità di velocità, uno ione di carica $+Ze$ completamente privo di elettroni ha un LET ben Z^2 volte maggiore di un protone. Da ciò, in generale, consegue una maggiore efficacia biologica relativa (EBR): un gran numero di studi di radiobiologia ha dimostrato

infatti che vi è per la maggior parte delle cellule e degli organi una correlazione positiva tra EBR e LET. Quindi una radiazione ionica presenta, a confronto con le radiazioni convenzionali, il vantaggio della selettività balistica dovuto al picco di Bragg, ed inoltre risulta meno sensibile alle variazioni della pressione parziale di ossigeno a livello cellulare. Infatti le cellule ipossiche, quali sono quelle di certi tipi di tumori, presentano una radioresistenza tale da non consentire alle radiazioni tradizionali, così come a quelle protoniche, la loro distruzione. Infatti anche le radiazioni protoniche presentano nella maggior parte del volume irradiato, tranne che nella parte conclusiva del loro percorso, un LET inferiore ai 100 MeV/cm, valore energetico che si ottiene facilmente con l'irradiazione neutronica.

Rispetto ai neutroni, che esauriscono il loro effetto in meno di un millimetro di tessuto, gli ioni sono vantaggiosi grazie alla della selettività balistica, cosicché l'irradiazione con fasci di ioni mette a disposizione del radioterapista uno strumento nuovo che combina la selettività balistica dei protoni con l'elevato LET dei neutroni.

1.5.1 La scelta dello ione carbonio

Sebbene le proprietà degli ioni fossero note già a partire dagli anni '50, il primo studio clinico fu iniziato solamente nel 1975 presso il San Francisco Medical Center e il Lawrence Berkeley Laboratory (LBL) in California. Fu infatti necessario predisporre non soltanto fasci di ioni accelerati, ma anche attendere lo sviluppo dei sistemi di calcolo basati sulle immagini di tomografia computerizzata. Furono per primi utilizzati gli ioni elio, che presentavano però una EBR relativamente bassa (1.2 -1.4), e gli ioni neon con una elevata efficacia biologica (circa 3.5). Ora si sa che gli ioni neon, nell'attraversamento della materia, frammentano in ioni più piccoli che vanno al di là del picco di Bragg degli ioni neon, producendo una "coda" che rende il picco molto meno pronunciato di quello dei protoni e deteriora le proprietà conformazionali dell'irradiamento.

Inoltre gli ioni neon presentano nei primi strati di tessuto un LET superiore a 250 MeV/cm, quindi troppo elevato. Infatti a questi valori le interazioni con le

cellule sane del "canale di ingresso" sono dannose a tal punto che vengono spesso prodotti gruppi di doppie rotture del DNA, per i quali non sono attivi gli usuali meccanismi cellulari di riparazione.

Quindi la carica Z degli ioni deve essere scelta con molta cura: agli inizi degli anni '90 radiobiologi e radioterapisti sono giunti alla conclusione che la carica ottimale si trova intorno a Z=6, tipica dello ione carbonio. I nuclei di carbonio accelerati a 4500 MeV (e cioè 375 MeV/nucleone) attraversano 25 cm di tessuto prima di fermarsi. Se la perdita di energia fosse costante, il LET sarebbe 180 MeV/cm (4500/25); in realtà esso aumenta molto con la profondità, tanto che per metà del percorso esso è compreso tra 100 e 150 MeV/cm mentre a 3 cm dalla fine (e cioè a 22 cm di profondità) vale circa 1000 MeV/cm.

Si nota allora che un LET di 100 MeV/cm è abbastanza piccolo da permettere la riparazione delle cellule nel canale di ingresso; d'altra parte l'intervallo di LET che va da 200 a 1000 MeV/cm non è ottenibile né con raggi X né con protoni, che raggiungono al massimo un paio di centinaia di MeV/cm.

Quindi le proprietà fisiche di interazione con la materia biologica fanno degli ioni uno strumento nuovo, che va utilizzato con cura, perché irradiando un bersaglio con picco di Bragg allargato (SOBP) parti diverse del tumore sono irradiate necessariamente con ioni che hanno LET molto diversi.

Non è quindi possibile utilizzare un valore unico di efficacia biologica relativa; è necessario invece applicare un modello dettagliato, che tenga conto sia della distribuzione del LET in ogni punto, sia delle interazioni con le cellule degli ioni con diverso LET. Si parlerà così di effetto complessivo del trattamento e si utilizzeranno "superfici di isoeffetto" nell'elaborazione dei piani di somministrazione della dose per mezzo di programmi di simulazione.

1.5.2 Risultati clinici

I soli risultati clinici ottenuti dal 1975 al 1995 sono quelli di Berkeley), dove sono stati trattati circa 2000 pazienti con ioni di elio e circa 500 con ioni di neon. Tali risultati hanno dimostrato l'utilità dell'impiego degli ioni elio nel determinare

incrementi di dose a livello del volume tumorale, sino a valori del 35%. I casi nei quali la radiazione ionica risulta più indicata sono quelli già individuati per i protoni: neoplasie oculari e tumori che arrivano a infiltrare la base del cranio.

Secondo quanto riportato al PTCOG di Detroit (aprile 1996) nei primi 104 pazienti trattati nel centro HIMAC, in Giappone non sono state osservate complicanze gravi e i risultati preliminari, in termini di risposta al trattamento, appaiono incoraggianti Nel 1996 presso il laboratorio di fisica nucleare GSI di Darmstadt è entrato in funzione un sistema di distribuzione attiva di un fascio di ioni carbonio; i trattamenti sono iniziati nel 1997.

Esistono quindi le strutture di base per trattare e seguire qualche centinaio di pazienti all'anno; tuttavia tenuto conto dei diversi anni necessari alla raccolta delle informazioni sul controllo locale e sulla sopravvivenza, conclusioni definitive sulle patologie elettive e i trattamenti più mirati, saranno disponibili verso l'anno 2000.

Sulla base di considerazioni radiobiologiche è stato stimato che circa il 10% dei tumori che richiedono una radioterapia potrebbero trarre vantaggio dall'impiego di radiazioni ad alto LET. In Italia tale percentuale corrisponde a circa 12000 pazienti/anno. Tale dato non differisce sostanzialmente dalle stime effettuate da Gademann sulla popolazione europea che, se proiettate sulla popolazione italiana, indicherebbero in circa 14000 i possibili candidati a radioterapia con radiazioni ad alto LET.

1.6 Precisione nell'Adroterapia

Le caratteristiche di selettività fisica dei protoni e degli ioni (cessione di energia alla fine del percorso) e la possibilità di modulazione ("spread-out") dell'ampiezza del picco di Bragg – accompagnata dalla possibilità delle moderne tecniche diagnostiche che permettono di delineare il bersaglio con elevata precisione – ben si attagliano alla elaborazione dei piani di trattamento che si avvicinano alla condizione ideale, nella quale la cessione di energia avviene all'interno del volume bersaglio tumorale, con minimo coinvolgimento degli organi e tessuti sani circostanti. L'impiego di protoni o di ioni è quindi idoneo alla

realizzazione di una radioterapia di elevata precisione , che consenta, attraverso la conformazione della distribuzione della dose alle caratteristiche del volume da trattare, di somministrare dosi totali più elevate.

Tale precisione si lega in maniera inscindibile con la realizzazione della testata isocentrica (gantry).

Nella progettazione di un gantry, infatti, bisogna fare in modo che le deformazioni, indotte dalla gravità su tale struttura, rimangano a livelli molto bassi (< di 1 mm nelle tre dimensioni) in modo che il modesto spostamento del magnete dalla posizione nominale non distorca sensibilmente la traiettoria del fascio adronico, che deve avere precisione millimetrica.

Si capisce che variando anche di poco tale traiettoria, gli adroni potrebbero colpire con la maggior parte della loro energia i tessuti sani circostanti il tumore distruggendoli in maniera irreparabile e danneggiando in alcuni casi parti vitali.

1.7 Futuro dell'Adroterapia

In conclusione, la terapia ionica trova il suo fondamento essenzialmente in dati fisici e radiobiologici; la protonterapia ha al suo attivo una consistente attività clinica, ma mancano ancora, soprattutto per le patologie a maggiore incidenza, i risultati definitivi di studi clinici controllati. In ogni caso, le favorevoli prospettive della terapia con fasci di ioni e la provata efficacia della protonterapia hanno contribuito alla diffusione di nuovi centri di adroterapia in tutto il mondo.

2

Il magnete superconduttore

2.1 Considerazioni introduttive

Oggetto di questa tesi è stata l'ottimizzazione di alcuni dei parametri caratteristici di un magnete superconduttore, componente principale di una testata isocentrica (*gantry*), cioé di quella struttura meccanica rotante che, muovendo la parte terminale di una linea di trasporto di particelle pesanti (adroni), permette di irradiare da differenti direzioni un paziente affetto da tumore. La figura 2.1 mostra, a livello schematico, la parte terminale della linea di trasporto. ***La deflessione finale di 90 gradi si ottiene grazie ad un dipolo come quello studiato in questa tesi.***

Allo stato attuale, per le applicazioni nel campo dell'Adroterapia, vengono utilizzati gantries con magneti convenzionali resistivi, vale a dire magneti non superconduttori. L'intensità di corrente circolante nelle bobine di un magnete resistivo ha un limite massimo dovuto al riscaldamento che si genera nel conduttore a causa della resistenza ohmica che gli elettroni incontrano nel loro moto. A livello macroscopico questo fenomeno si traduce in produzione di calore per effetto Joule; se il calore generato è eccessivo il conduttore si può danneggiare irrimediabilmente. I magneti superconduttori si differenziano da quelli

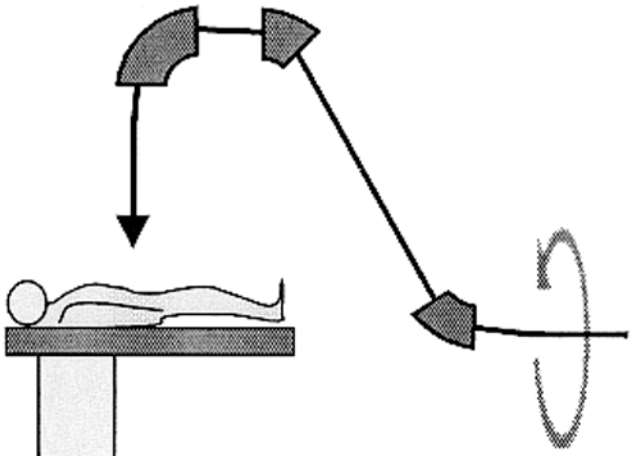


Fig. 2.1 *Schema di gantry isocentrico.*

convenzionali perché utilizzano bobine costituite da filamenti di materiali superconduttori. Un cavo superconduttore viene prodotto a partire da sottilissimi filamenti di materiale superconduttore, tipicamente NbTi, del diametro di circa 40 μm. Un gran numero di questi filamenti (>500) vengono poi attorcigliati a formare uno strand, il cui diametro é dell'ordine di 1 mm. Infine, attraverso un processo di estrusione, un certo numero di questi strand (~30) viene inserito in una matrice di rame, a formare il conduttore finale.

Il vantaggio dei magneti superconduttori rispetto ai tradizionali magneti resistivi é che i materiali superconduttori presentano una resistenza al passaggio della corrente che a tutti i fini pratici puó essere considerata nulla, purché si trovino al di sotto di una temperatura limite, detta temperatura critica. Poiché il campo magnetico generato è direttamente proporzionale all'intensità di corrente, le elevate correnti raggiungibili con l'impiego di tali materiali generano intensi campi magnetici. Ne segue che anche la curvatura che può essere impressa al fascio di particelle risulta maggiore. In generale, infatti, per deflettere lungo una traiettoria circolare con raggio di curvatura R una particella con carica q, massa m e velocità v, occorre un campo magnetico, perpendicolare alla direzione del moto della particella, di intensità:

$$B = \frac{mv}{qR} \tag{2.1}$$

Rispetto ai tradizionali magneti resistivi, l'utilizzo di magneti superconduttori funzionanti con campi magnetici dell'ordine dei 4 T, permette di ridurre notevolmente il raggio di curvatura del fascio, con conseguente riduzione del peso e degli ingombri, non solo del magnete, ma di tutta la struttura meccanica rotante di sostegno, rendendo quindi molto più conveniente il ricorso a questo tipo di macchine.

Il rovescio della medaglia dei magneti superconduttori è dato dalla grande sensibilità ad ogni instabilità, sia meccanica che termica. Per questo motivo gli strand superconduttivi sono inseriti all'interno di una matrice di rame che garantisce ai filamenti un supporto meccanico, un bypass elettrico ad alta conduttività ed un pozzo di calore. Infatti, un superconduttore può, in seguito ad una perturbazione, subire localmente una transizione allo stato normale, diventando cosí altamente resistivo. In questo caso la corrente, avendone la possibilità, fluirà nel materiale a resistenza minore, ossia nella matrice di rame. Se la perturbazione non è troppo grande, una volta eliminata la causa perturbatrice il filamento può ritornare nel suo stato superconduttivo. Qualora ciò non avvenga (si parla in questo caso di *quench* del magnete) è necessario interrompere l'erogazione della corrente onde evitare che il cavo, divenuto un normale conduttore, si danneggi irreparabilmente a causa del surriscaldamento.

2.2 Le specifiche tecniche

La tabella 2.2 raccoglie i parametri principali del gantry, le caratteristiche di un cavo superconduttore, possibile candidato per l'avvolgimento, e una stima degli ingombri e del costo complessivo.

Raggio di curvatura del dipolo	1.6 m
Arco di curvatura del dipolo	90°
Campo magnetico	4 T
Temperatura operativa	4.5 K
Raffreddamento	Mediante cryo-cooler
Densità media di corrente	$2 \cdot 10^8$ A / m^2
Densità di corrente critica	$3 \cdot 10^8$ A / m^2
Corrente operativa	600 A
Corrente critica	1400 A (4.5 K, 5 T)
Conduttori	Piattine NbTi 3x1 mm^2
Rapporto Cu / NbTi	6.5
Diametro medio	550 mm
Campo massimo sui conduttori	6 T
Induttanza	25 H
Ingombro massa fredda	750 mm
Ingombro totale (con criostato)	1000 mm
Peso Massa Fredda	2400 Kg
Costo	2.5-3 mld di lire

Tab 2.1 *Specifiche tecniche del gantry.*

2.3 Il sistema di raffreddamento

E' stato scelto il sistema di raffreddamento e termostatazione detto "cryogenic free", ossia senza liquidi criogenici, come ad esempio l'elio liquido, a diretto contatto con il magnete. Si ipotizza, invece, di raffreddare i conduttori tramite un circuito di tubi nei quali circola elio allo stato gassoso; i tubi sono quindi disposti in prossimità delle cave che ospitano i conduttori stessi. Questa scelta, che

indubbiamente complica la criogenia del magnete dal punto di vista della progettazione, è dettata da due motivi: il primo è che il gantry che ospita il magnete deve ruotare, per cui risulta difficile pensare ad un contenitore per liquidi criogenici che partecipi al moto di rotazione del gantry; il secondo motivo è dato dal fatto che il gantry deve funzionare in un ambiente ospedaliero dove generalmente i liquidi criogenici non sono graditi, in quanto occorrerebbe dover provvedere al rifornimento periodico ed a tutte le necessarie precauzioni per evitare la saturazione dell'ambiente con il gas evaporato (in condizioni normali o, peggio, in caso di quench).

La potenza refrigerante è fornita tramite un criogeneratore a due stadi che funziona attraverso un ciclo di compressione-espansione di elio (per esempio il ciclo Gifford-McMahon), mantenendo a 4.5 K la temperatura del gas di elio circolante nel circuito di raffreddamento. Quest'ultimo é costituito da un circuito chiuso di tubi di raffreddamento che vengono posizionati il piú possibile vicino al magnete, ossia sulla superficie esterna della struttura di contenimento. Il raffreddamento avviene per conduzione attraverso la struttura meccanica del magnete per la quale si è deciso di usare una lega di alluminio. L'alluminio presenta infatti una conducibilità termica relativamente elevata (0.1 W/cmK contro 0.002 W/cmK dell'acciaio a 4.5 K) ottimizzando quindi lo scambio termico tra il magnete ed i tubi di raffreddamento.

2.4 La geometria iniziale

Le bobine sono il componente più critico del magnete. La loro forma è studiata in maniera tale da generare un campo magnetico il più omogeneo possibile in almeno in un "tubo" di sezione rettangolare 60×200 mm^2 probabilemente centrata sull'asse del magnete, in modo da guidare il fascio di particelle sul bersaglio tumorale con elevata precisione.

Il magnete oggetto di questa tesi é stato concepito inizialmente con 12 bobine superconduttrici, disposte simmetricamente rispetto al piano equatoriale del magnete stesso (fig. 2.1). Ogni bobina è conformata a "race track" piegato (fig 2.2),

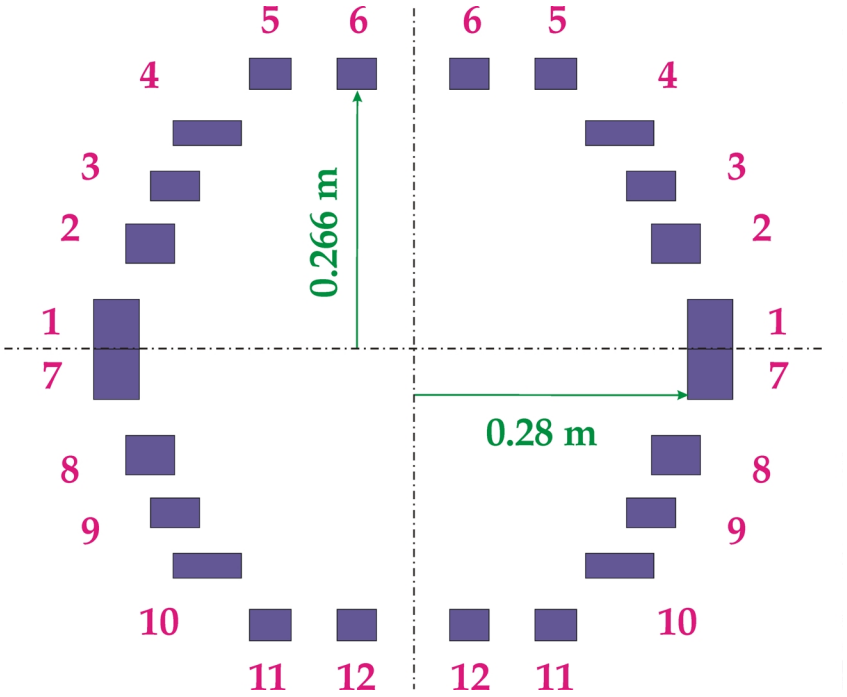


Fig 2.1 *Sezione trasversale delle bobine.*

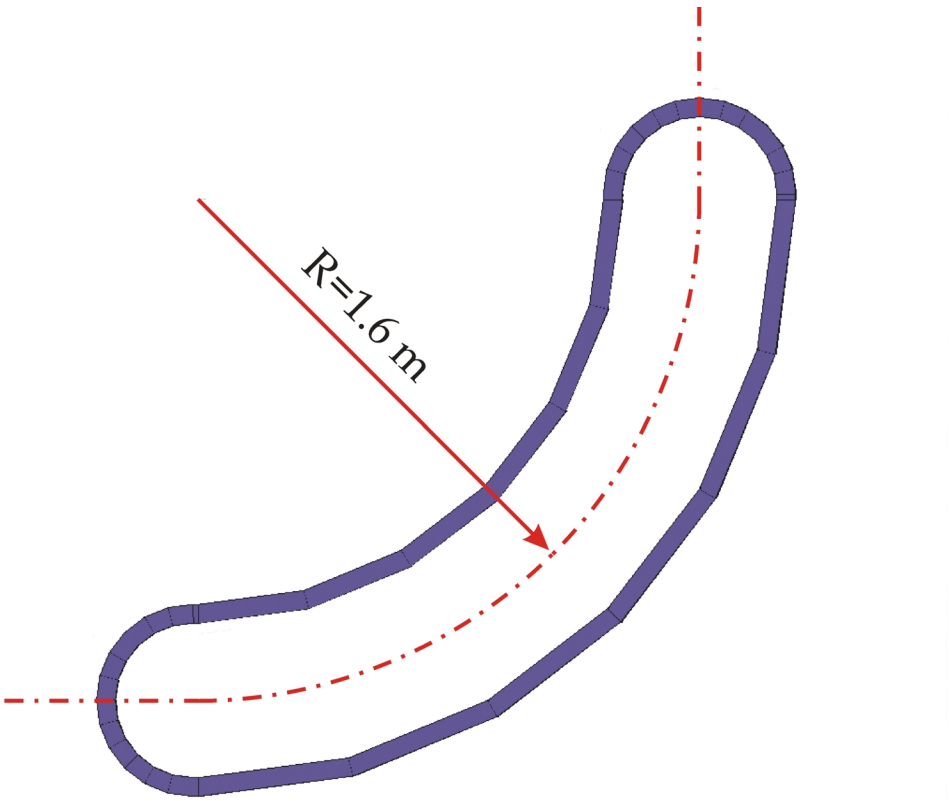


Fig 2.2 *Bobina tipo "racetrack" piegato.*

forma adottata frequentemente per le bobine dei dipoli di deflessione. Per ogni bobina, il raggio di curvatura, valutato in corrispondenza dell'asse della bobina stessa, é pari a 1.6 m, e questo valore é proprio il raggio di curvatura della traiettoria ad arco di circonferenza che le particelle cariche devono percorrrere.

2.5 Il modello agli Elementi Finiti

L'approccio piú conveniente, quando si é nella fase iniziale di un progetto, é quello di cercare di semplificare il piú possibile il modello di studio, raggiungendo poi gradatamente un sempre maggior livello di dettaglio. Nel caso del gantry descritto nei paragrafi precedenti, l'unico elemento di semplificazione non approssimato che si ha a disposizione é il piano di simmetria equatoriale, che permette di costruire il modello tridimensionale di solo metà magnete. Per ridurre drasticamente il livello di complicazione, passando quindi ad un modello bidimensionale, é necessario introdurre un'approssimazione. Supponendo infatti che gli effetti di bordo, ossia della richiusura dei conduttori, siano scarsamente influenti sulla distribuzione di campo magnetico al centro del magnete, é possibile costruire un modello 2D assial-simmetrico, mostrato in fig. 2.3, che in realtà simula 12 anelli, concentrici a due a due, disposti a varie distanza dal piano equatoriale. Come verrà dimostrato nel Capitolo 7, questo modello é assolutamente efficace nel simulare l'andamento del campo magnetico al centro del dipolo, e, in generale, é tanto più esatto quanto più ci si allontana dalle estremità del magnete.
La fig. 2.3 inoltre fornisce una descrizione schematica dell'asse di simmetria equatoriale e normale, dell'asse di simmetria cilindrica del modello e degli assi x e y, definizioni che verrano successivamente utilizzate.

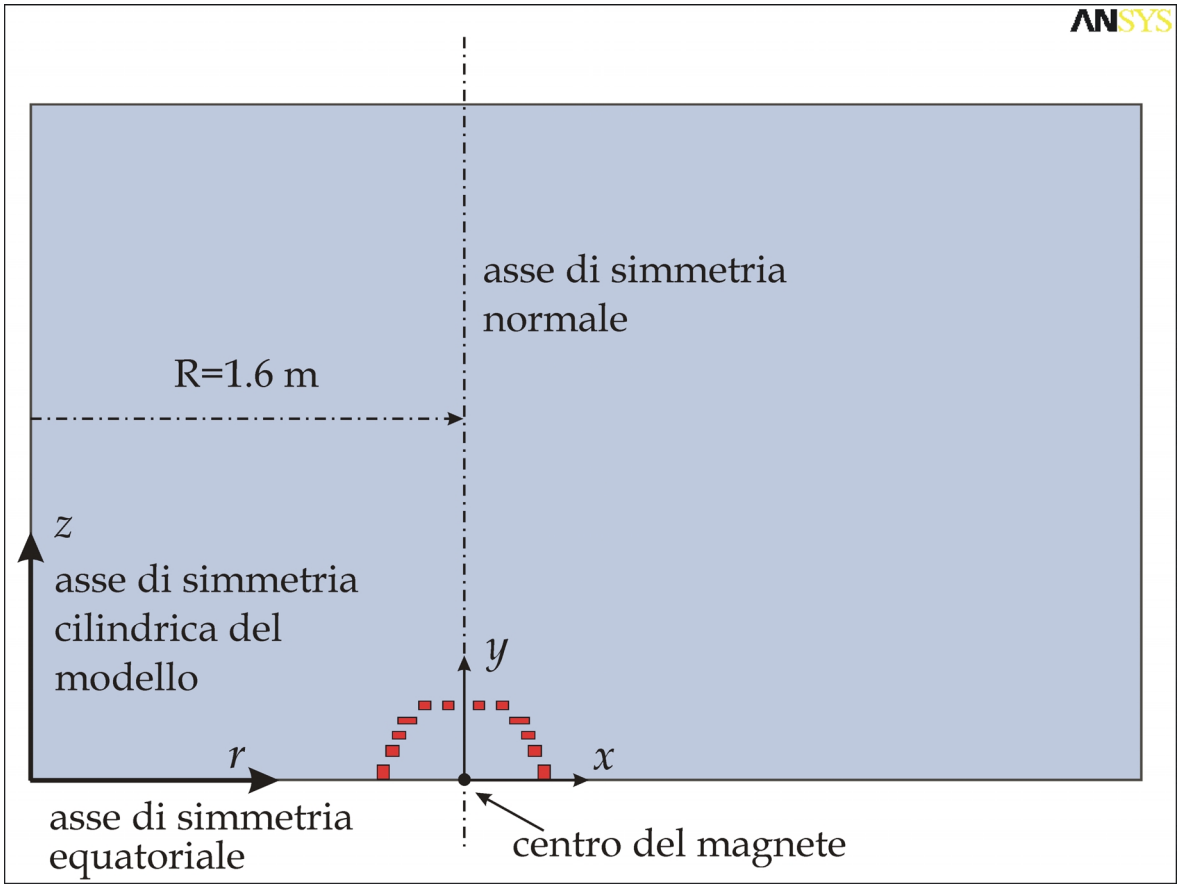


Fig 2.3 *Schema del modello agli elementi finiti 2D.*

Per il calcolo del campo magnetico prodotto dalla corrente circolante nelle bobine è stato utilizzato il codice agli elementi finiti ANSYS, tramite gli elementi PLANE13, adatti all'analisi magneto-statica bidimensionale. Il codice di calcolo ANSYS determina il campo magnetico, prodotto da una corrente costante, attraverso le equazioni di Maxwell:

$$\nabla \times H = J \tag{2.1}$$

$$\nabla \cdot B = 0\,, \tag{2.2}$$

dove H (espresso in A/m) rappresenta il campo magnetico, J (espresso in A/m^2) la densità di corrente e B (espresso in T) l'induzione magnetica. E' da notare che, poiché nel vuoto vale la semplice relazione $B=\mu_0 H$, dove

$\mu_0=4\pi\cdot10^{-7}$ H / m é la permeabilità magnetica del vuoto, il termine *campo magnetico* viene convenzionalmente adottato anche per intendere l'induzione magnetica *B*.

In realtà, il codice di calcolo ANSYS introduce, per la risoluzione di qualsiasi problema bidimensionale di natura elettromagnetica, il grado di libertà potenziale vettore *A* (espresso in T·m), definito come:

$$B = \nabla \times A. \tag{2.3}$$

Poiché in due dimensioni il campo magnetico *B* avrà due sole componenti, B_x e B_y, ne segue immediatamente dalle proprietà del rotore, che il potenziale vettore avrà un'unica componente, perpendicolare al piano *xy*, ossia A_z. Di qui discende il vantaggio di utilizzare il potenziale vettore per la risoluzione di problemi magnetici bidimensionali: con un unico grado di libertà infatti si riesce a descrivere completamente qualsiasi analisi magnetica ($B_x = \frac{\partial A_z}{\partial y}$ e $B_y = -\frac{\partial A_z}{\partial x}$).

In due dimensioni, inoltre, il potenziale vettore assume un particolare significato; infatti le isolinee a potenziale vettore costante sono equivalenti alle linee di forza del campo magnetico, nel senso che in ogni punto del piano il campo magnetico risulta essere tangente ad esse, ed il loro addensamento é proporzionale all'intensità del campo stesso.

E' importante ricordare che in un'analisi magnetostatica occorre che venga modellata anche l'aria, in quanto, a differenza di qualsiasi carico di natura meccanica, le onde elettromagnetiche si trasmettono anche in aria. E' necessario allora costruire una "scatola" di aria che racchiuda il modello: tale scatola deve estendersi fino a dove il campo magnetico assume valori trascurabili. Un'alternativa a questo tipo di modellazione è rappresentata dall'utilizzo dei cosiddetti *Infinite Boundary Elements*, che, se utilizzati correttamente, permettono di ridurre notevolmente le dimensioni della "scatola", il numero di nodi / elementi del modello e conseguentemente i tempi di calcolo.

Infine, in questa analisi l'unica proprietà che viene considerata, sia per l'aria

che per i conduttori, è la permeabilità magnetica relativa al vuoto del mezzo considerato, $\mu_r=\mu/\mu_o$. In assenza di mezzi permeabili, come ad esempio il ferro, per i quali $\mu_r\neq1$, l'unica proprietà da introdurre é $\mu_r=1$.

2.6 Condizioni al contorno

Le condizioni al contorno per le analisi magnetostatiche sono le seguenti: le linee di forza del campo magnetico devono essere parallele al bordo della "scatola" d'aria (si impone che la componente A_z del potenziale vettore A sia nulla) e perpendicolari al piano di simmetria, ossia al piano equatoriale del magnete (condizione automaticamente verificata).

2.7 Prestazioni del modello preliminare

L'uniformità del campo magnetico è l'obiettivo fondamentale che deve guidare la progettazione del magnete e dei suoi sottoinsiemi. L'uniformità deve essere la migliore possibile per poter guidare il fascio di particelle sul bersaglio tumorale con la massima precisione. E' richiesta un'uniformità del 2‰ in una regione rettangolare di dimensioni 60x200 mm^2, possibilmente centrata sull'asse del magnete. La necessità di avere un campo magnetico omogeneo in corrispondenza di una regione cosí estesa e' dettata dal fatto che il fascio di particelle impiegato per il bombardamento delle cellule tumorali, non rimane fisso durante il trattamento del paziente, ma viene spostato sul piano equatoriale del magnete al fine di eseguire una scansione di tutta la massa tumorale.

L'analisi agli elementi finiti, condotta sulla configurazione "preliminare" dalla quale ha preso avvio questa tesi, mostra che il campo magnetico generato è caratterizzato da una uniformità che è peggiore di un ordine di grandezza rispetto a quella richiesta. Il campo magnetico all'interno della regione di interesse presenta uno scostamento, rispetto al valore al centro della regione, pari a –0.8% +1.8%.

La figura 2.4 mostra come varia l'uniformità di campo al variare delle

dimensioni prefissate Δx e Δy per la zona di valutazione dell'uniformità stessa.

La regione di migliore uniformità non è centrata sull'asse del magnete, ma è situata all'esterno rispetto all'asse stesso, come si vede nella figura 2.5, che mostra le curve di livello del campo magnetico all'interno di tale regione.

L'andamento del campo magnetico calcolato lungo direzioni parallele all'asse *x* (asse di simmetria del magnete) presenta un andamento con un massimo verso il centro. Allontanandosi dall'asse del magnete verso la periferia in entrambi i versi, il campo, dopo aver raggiunto un minimo relativo, si impenna raggiungendo valori prossimi ai 5 T. Questo andamento è mostrato dalla figura 2.6: i valori in rosso rappresentano il campo calcolato sull'asse *x*, i valori in blu rappresentano il campo sull'ordinata *y*=30 mm, che corrisponde al bordo superiore della regione di uniformità. L'ascissa rappresenta la distanza dell'asse di simmetria cilindrica del magnete (1.3 – 1.9 m).

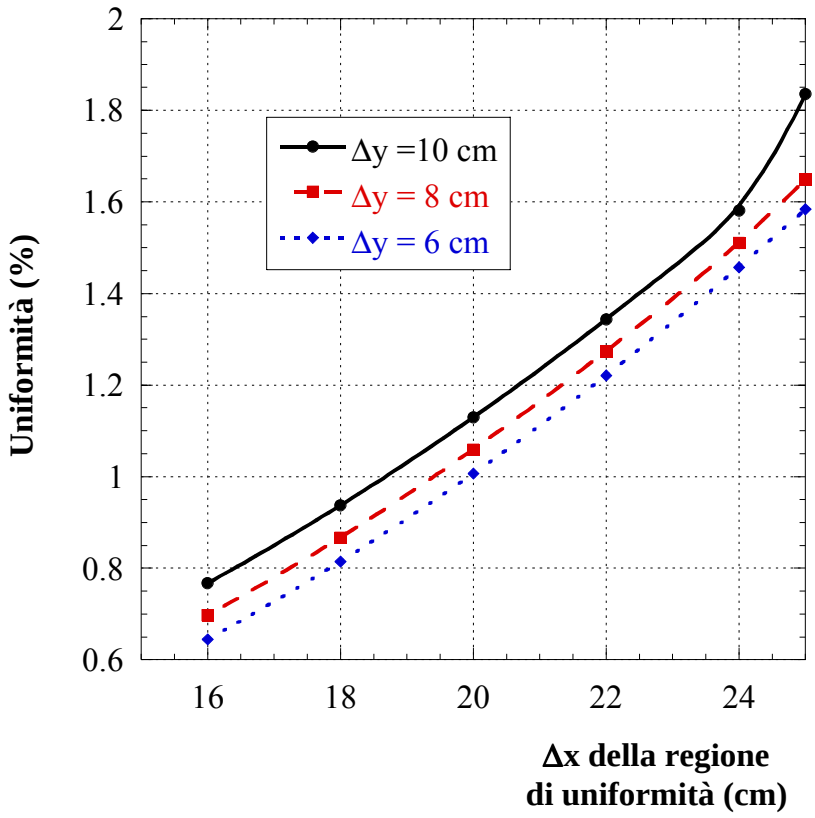


Fig 2.4 *Uniformità del campo magnetico in funzione delle dimensioni della regione di riferimento.*

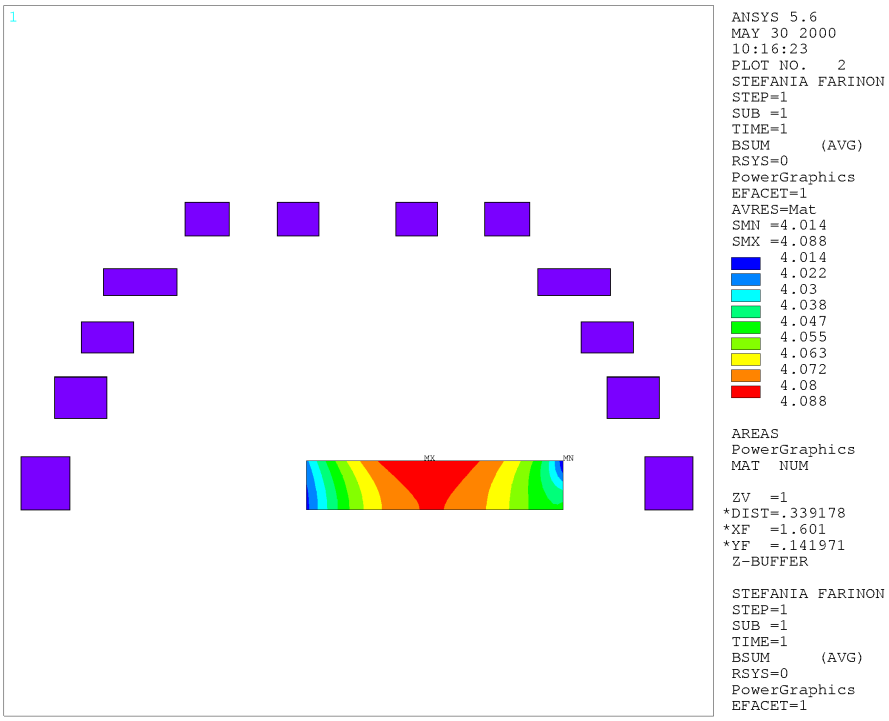


Fig 2.5 *Curve di livello del campo magnetico all'interno della regione di uniformità.*

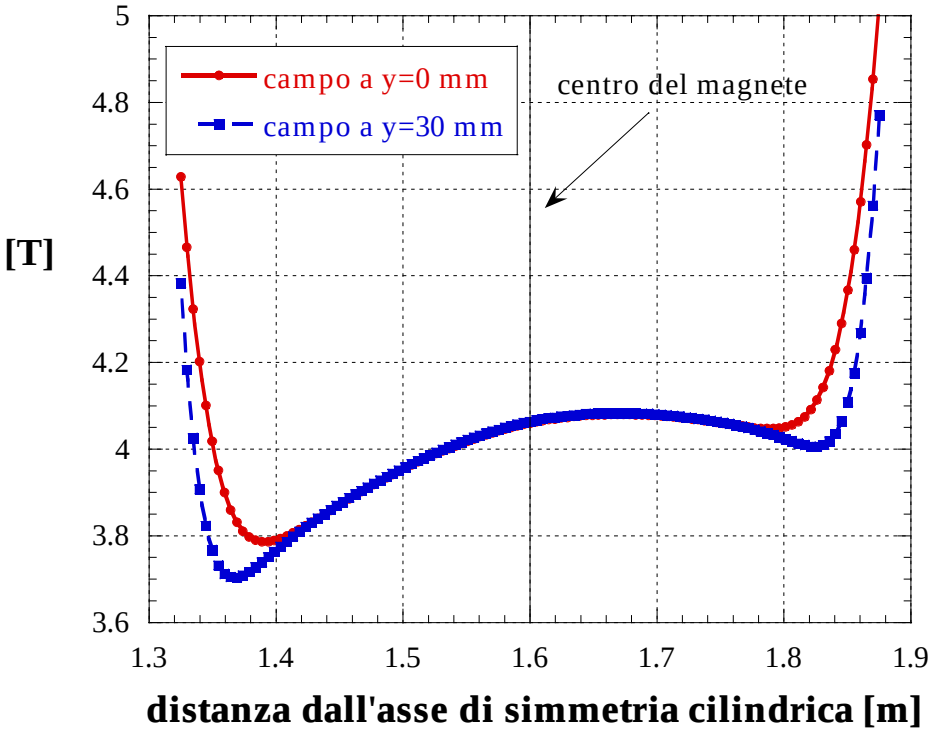


Fig 2.6 *Andamento del campo magnetico.*

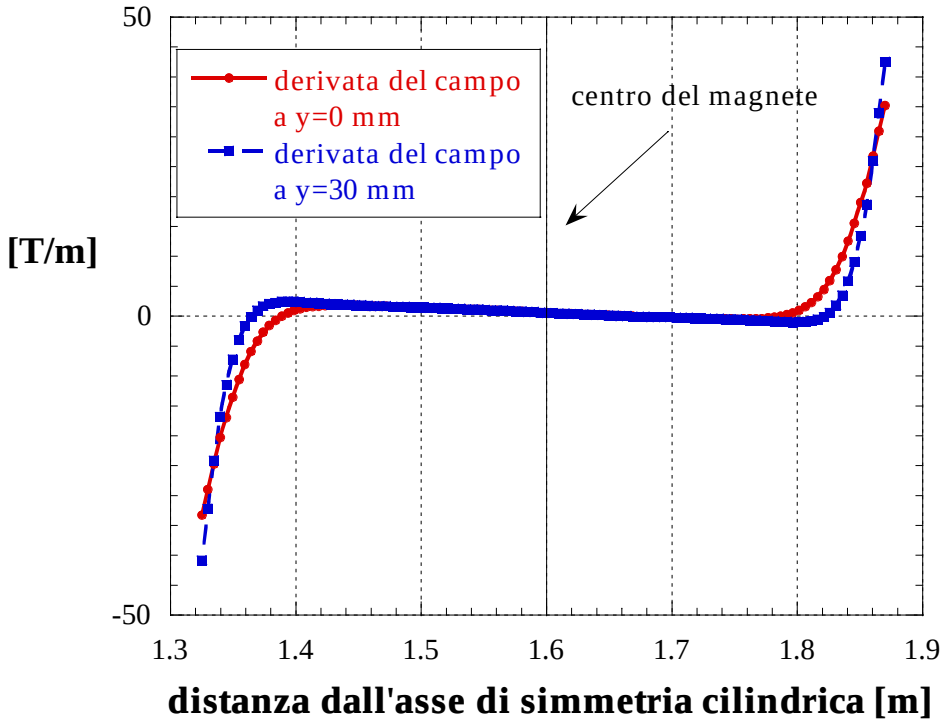


Fig 2.7 *Andamento della derivata del campo magnetico.*

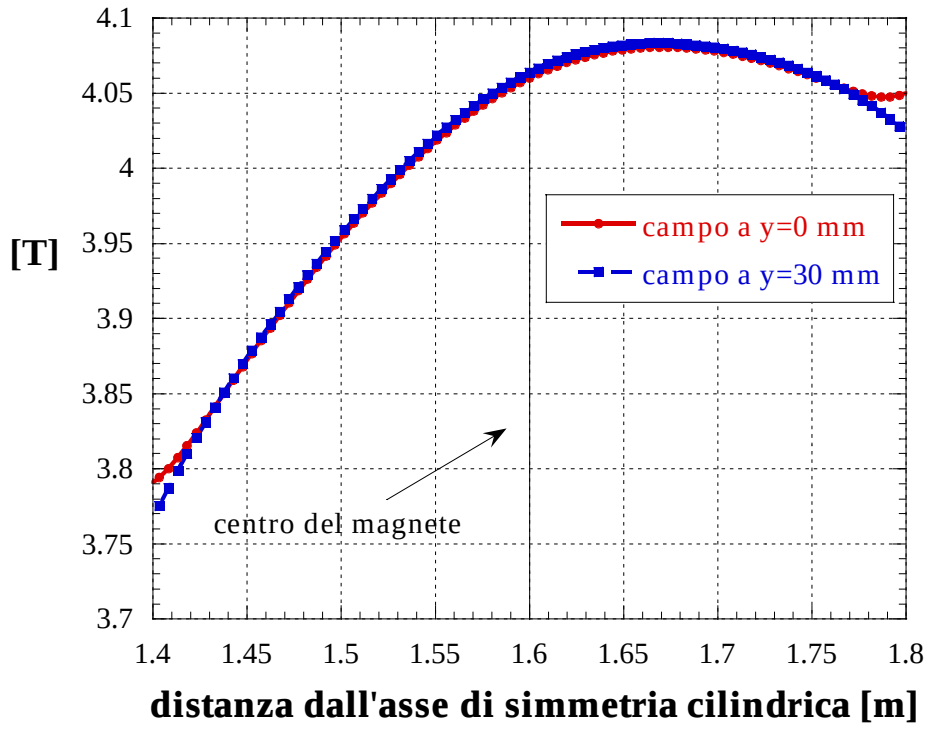


Fig 2.7 *Andamento del campo magnetico.*

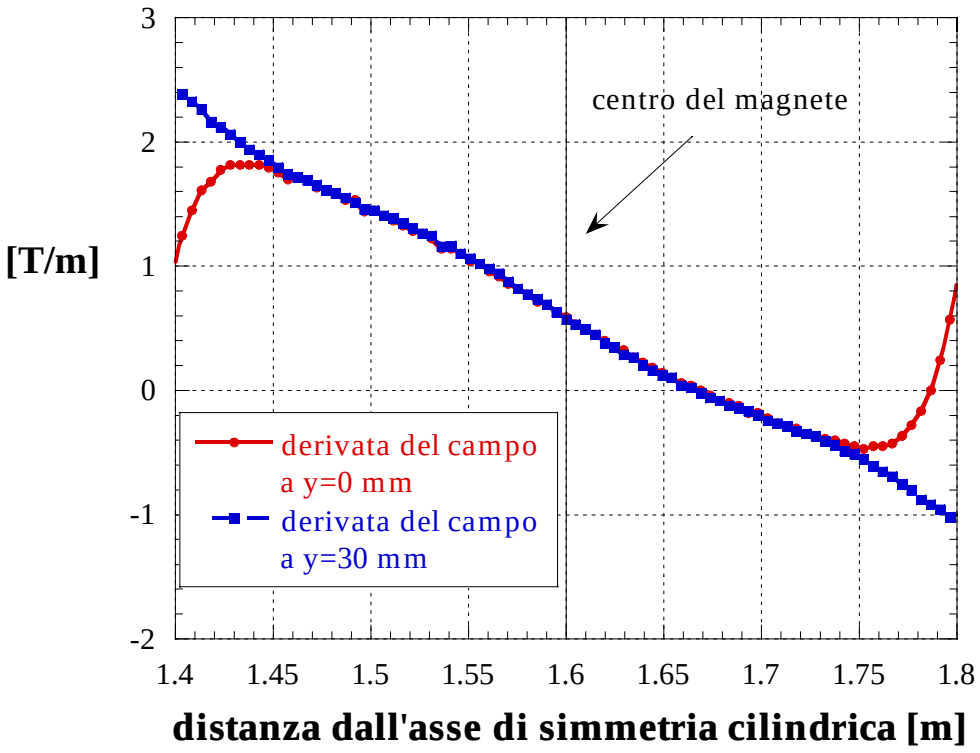


Fig 2.8 *Andamento della derivata del campo magnetico.*

Le figure 2.8 e 2.9 rappresentano le stesse grandezze, campo e derivata del campo, limitando la distanza dall'asse del magnete a 20 cm, anziché 30, in entrambi i versi.

La figura 2.9 visualizza, sotto forma di curve di livello, il campo magnetico generato dal magnete, all'interno del magnete stesso e all'esterno. E' importante che il campo magnetico esterno sia il più ridotto possibile in quanto il magnete dovrà essere collocato in un ambiente ospedaliero per il quale sono previste delle norme di legge che impongono precisi limiti all'intensità dei campi ai quali vengono esposti pazienti e personale.

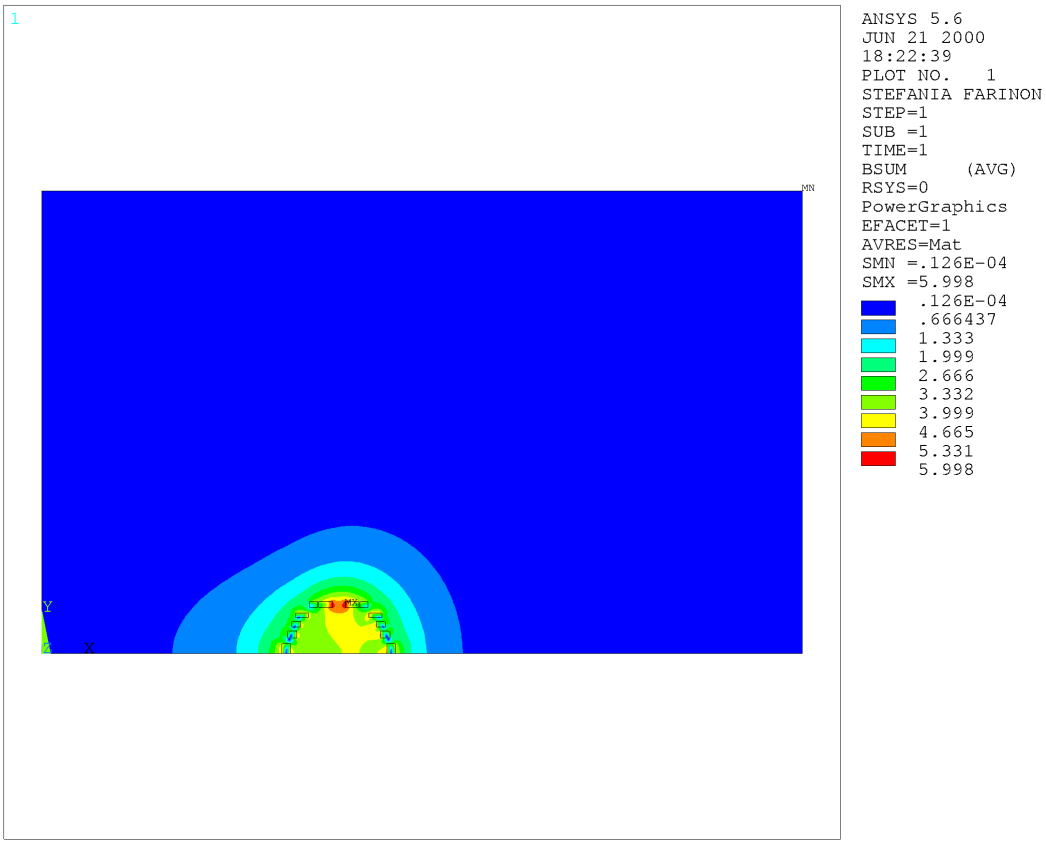


Fig 2.9 *Distribuzione del campo magnetico.*

Le analisi condotte evidenziano che l'uniformità del campo magnetico risulta insufficiente per guidare il fascio di particelle con la voluta precisione, inoltre la regione di uniformità non è centrata sull'asse del magnete.

Il lavoro di ottimizzazione necessario per raggiungere la richiesta uniformità di campo è stato l'oggetto di questa tesi.

3

Vagliatura dei fattori

3.1 Introduzione

Prima di affrontare il lavoro di ottimizzazione vero e proprio ho ritenuto essenziale studiare in dettaglio il contributo al campo magnetico dovuto alle singole bobine. Tale conoscenza è indispensabile per individuare da subito il ruolo svolto dalle variabili di progetto e poter condurre un lavoro di ottimizzazione ragionato.

I risultati ottenuti da questo studio preliminare sono stati particolarmente interessanti, infatti hanno mostrato che i contributi al campo magnetico dovuti alle singole bobine possono essere ricondotti a tre sole tipologie di andamenti, descritti rispettivamente da:

- curve concave;
- curve convesse;
- curve con due massimi;

allontanando progressivamente una bobina dall'asse del magnete si ottiene un andamento del campo che varia con continuità dal tipo concavo a quello convesso

passando per il terzo tipo.

Dal momento che le equazioni che descrivono il campo magnetico prodotto da una generica distribuzione di corrente sono lineari, questo consente di affermare che il campo prodotto dall'insieme delle bobine è dato dalla sovrapposizione dei singoli contributi. In altre parole il campo totale è la combinazione lineare dei campi generati dalle singole bobine:

$$B = \sum_i B_i \cdot p_i \tag{3.1}$$

ove $\boldsymbol{B_i}$ è il campo prodotto dalla bobina i-esima percorsa da una corrente unitaria mentre $\boldsymbol{p_i}$ è il peso dato dalla corrente che circola nella bobina i-esima.

In pratica la corrente è un fattore che, per ogni bobina, ci permette di traslare lungo l'asse verticale la curva del campo, a nostro piacimento.

I calcoli magnetici effettuati per lo studio del contributo delle singole bobine sono stati condotti con il codice ANSYS attribuendo una corrente unitaria alla bobina in esame e corrente nulla a tutte le altre. Queste analisi magnetiche hanno mostrato che il campo prodotto dalle tre bobine più esterne presenta un andamento descrivibile sostanzialmente dallo stesso tipo di curva. Tale risultato mi ha portato ad indurre che, se avessi lasciato solo una di queste tre bobine, in posizione baricentrica rispetto alle tre di partenza, e l'avessi alimentata con una corrente pari alla somma delle tre, avrei dovuto ottenere, nella regione di interesse, sostanzialmente il medesimo campo, sia in termini di distribuzione sia di intensità. Questo procedimento è giustificato dal fatto che la regione di interesse, ove misuriamo l'intensità del campo magnetico, è sufficientemente distante dalla distribuzione di corrente che lo induce (una sorta di principio di De Saint Venant esteso al magnetismo). I calcoli effettuati hanno dimostrato la validità della mia ipotesi.

Alla luce di questo risultato, ho disegnato una configurazione di prova costituita da sole 8 bobine, anziché 12, ottenendo addirittura un miglioramento,

seppur di entità ridotta, dell'uniformità di campo.

Utilizzando la terminologia dell'algebra vettoriale si potrebbe dire che le 8 bobine rappresentano una *base* per lo spazio vettoriale dei campi magnetici (nella regione di interesse), nel senso che i campi magnetici con le caratteristiche volute possono essere costruiti con una combinazione lineare dei campi di queste sole 8 bobine. L'unico limite a questo processo di "riduzione" è dato da eventuali vincoli di campo locale massimo e dalla corrente massima che può circolare nei conduttori senza che essi si danneggino irreparabilmente a causa del surriscaldamento. Facendo un paragone, è come se, in un'orchestra costituita da sei strumenti (le sei bobine), i tre più lontani dal punto di ascolto (la regione ove misuriamo il campo), fossero tre strumenti dal timbro perfettamente identico (curva del campo); ora, se al posto di questi, suonati all'unisono con un'intensità moderata, ne lasciassimo suonare solo uno, con un'intensità pari alla somma, se il punto di ascolto è sufficientemente lontano, otterremmo lo stesso effetto sonoro. In perfetta analogia, il limite sarebbe costituito dall'intensità massima con cui il singolo strumento può essere suonato (densità di corrente).

Questo approccio di tipo ragionato si è rivelato indispensabile nel corso dell'ottimizzazione, in quanto, a causa della natura del problema e del numero di variabili, la funzione obiettivo è risultata descritta da una superficie di risposta particolarmente irregolare, caratterizzata da miriadi di minimi locali. Conseguenza di questo è che un algoritmo di tipo esclusivamente gradientale risulta inadeguato per il nostro problema.

Quando si studia un sistema particolarmente complesso a causa del numero elevato di variabili che lo influenzano, è buona norma eseguire una vagliatura dei fattori al fine di determinare la significatività di ognuno di essi. E' infatti possibile che non tutte le variabili di progetto abbiano una influenza rilevante sulla risposta del sistema (uniformità di campo, campo sull'asse e campo massimo), inoltre, se un fattore ha modesta influenza, esso produrrà interazioni con altri fattori del tutto trascurabili.

3.2 Teoria dei progetti fattoriali

Un esperimento consiste nell'assoggettare un sistema a determinati livelli delle variabili d'ingresso (fattori) e nell'osservare il conseguente risultato espresso numericamente (risposta). Un fattore viene definito "quantitativo" se i relativi livelli possono essere associati ad una scala numerica; essi vengono anche suddivisi in continui e discreti. Per un fattore quantitativo discreto le risposte corrispondenti a livelli intermedi sono prive di significato fisico, mentre un fattore quantitativo di natura continua è definito dall'intero intervallo di indagine.

Un qualunque sistema fisico involvente fattori quantitativi è rappresentabile da una o più relazioni matematiche fra le risposte (variabili dipendenti) e i fattori (variabili indipendenti). Una volta effettuata la scelta delle variabili indipendenti e delle variabili dipendenti, una fase di primaria importanza consiste nella scelta del tipo di progetto da impiegare nell'esperimento.

L'esperimento deve assicurare una veridicità nelle risposte; è buona norma mantenere un giusto equilibrio tra una soddisfacente accuratezza di indagine (cercando di avere almeno due replicazioni per ogni prova), ed il costo dell'esperimento. Nel caso la risposta del sistema non sia di tipo stocastico ma sia di tipo deterministico, cioè ad una determinata configurazione delle variabili indipendenti corrisponde sempre il medesimo valore della risposta del sistema (come avviene in un’analisi agli elementi finiti), l’errore sperimentale è nullo e quindi non sono necessarie replicazioni delle prove.

Segue poi la fase di esecuzione dell'esperimento e quindi l'analisi dei dati, per studiare i rapporti intercorrenti tra i fattori e le variabili dipendenti.

Vagliatura	***Analisi dei fattori e scelta dei livelli sperimentali***
	Progetto sperimentale fortemente frazionato
	Analisi della varianza e scelta dei k fattori di interesse
Progetto sperimentale	***Progetto fattoriale completo o frazionato***
	Aggiunta di n punti centrali
	Analisi della varianza e determinazione dell'errore sperimentale
Analisi di regressione	***Scelta del modello di regressione e determinazione dei coefficienti di regressione***
	Test sulla bontà di adattamento della regressione svolta
	Analisi degli errori e tracciamento delle bande di confidenza

Tab 3.1 *Schema procedurale di un progetto fattoriale.*

3.3 Il progetto fattoriale completo 2^k

Il progetto fattoriale consiste, nel caso di problemi multifattore, in una fase preliminare di analisi dell'importanza relativa dei fattori. L'obiettivo primario di questa fase consiste nell'eliminazione, dall'indagine sperimentale, dei fattori di scarsa o nulla influenza sulla risposta esaminata: si terranno perciò in considerazione tutti e soli i fattori di effettivo rilievo nell'ambito del sistema.

Lo schema procedurale è riassunto nella tabella 3.1.

Il progetto più efficiente per l'adattamento con un modello regressivo del secondo ordine è considerato il cosiddetto *Progetto Composto Centrale*, consistente in un fattoriale 2^k (od un fattoriale frazionario della serie 2^k) aumentando i punti di progetto di altri $2k$ punti assiali ed n_0 punti centrali. Il valore dei punti assiali dipende dal numero dei punti costituenti il nucleo centrale del progetto. La replicazione degli n_0 punti centrali ha due scopi:

- Consente la stima dell'errore sperimentale, valutando la risposta in vicinanza del centro del progetto.
- Permette di applicare il test sulla bontà di adattamento del modello del secondo ordine, permettendo la distinzione tra gli errori puramente sperimentali, da quelli dovuti alla mancanza di adattamento del modello imposto.

La metodologia del progetto composto centrale ha lo scopo di fornire il miglior campionamento dei valori delle variabili influenzanti il sistema per ottenere superfici di risposta adattabili con modelli regressivi del secondo ordine. Il progetto composto centrale richiede un numero di prove pari a:

$$2^k + 2 \cdot k + n_0 \tag{3.2}$$

essendo k il numero di variabili indipendenti del sistema (fattori), p il grado di frazionamento del progetto fattoriale, $2k$ il numero di prove assiali, ed n_0 il

numero di replicazioni effettuate in corrispondenza del centro del progetto.

Definiamo ora alcuni termini utilizzati nell'ambito dei progetti fattoriali.

L'*effetto* di un fattore è definito dalla rilevazione di un cambiamento nella risposta prodotto dal cambiamento del livello del fattore stesso; questa influenza viene chiamata *effetto principale* poiché è riferita ai singoli fattori di interesse primario nell'esperimento. In molti esperimenti si può ritenere che la differenza in risposta tra i diversi livelli dei singoli fattori non sia la stessa per i livelli di tutti gli altri fattori; quando questo accade significa che esiste una *interazione tra i fattori*.

Il progetto fattoriale 2^k, involvente k fattori, richiede il minor numero di combinazioni di livelli necessarie per studiare i k fattori, in quanto considera 2 livelli per ogni fattore. I livelli dei fattori vengono convenzionalmente denominati "basso" e "alto". Il progetto 2^k include:

k	effetti principali
k! / [2!(k-2)!]	interazioni a due fattori
k! / [3!(k-3)!]	interazioni a tre fattori
1	interazione tra i k fattori

I livelli alto e basso di ciascun fattore si possono anche indicare rispettivamente con +1 e –1; oppure introducendo una convenzione secondo la quale si indica il livello alto di un fattore con la corrispondente lettera minuscola e quello basso con l'assenza della stessa.

Per poter determinare gli effetti dei fattori, esistono differenti metodi a seconda della difficoltà e complessità dei calcoli del progetto considerato.

Il metodo per la valutazione dell'effetto di un qualsiasi fattore o dell'interazione tra i fattori, solitamente usato nel progetto 2^k, è il *metodo dei contrasti*.

Una formula di carattere generale, utile alla valutazione degli effetti principali e delle interazioni, è la seguente:

CONTRASTO $_{AB...K}$ ***= (a ± 1) (b ± 1)..... (k ± 1)***

Si sviluppano i prodotti delle quantità in parentesi scegliendo opportunamente il segno: si utilizza il segno "-" se il fattore è incluso nell'effetto che si desidera valutare, il segno "+" se il fattore non è incluso.

Valutati i contrasti, si possono determinare gli effetti e le somme dei quadrati, in accordo alle seguenti espressioni generali (è convenzione normalmente adottata indicare gli effetti con le maiuscole dei corrispondenti fattori):

$$AB...K = 2\,(\text{contrasto } AB...K) / n\,2^k$$

$$SS_{AB...K} = (\text{contrasto } AB...K)^2 / (n\,2^k)$$

Dove:

k = numero dei fattori

n = numero delle replicazioni

Si possono infine valutare la somma dei quadrati per l'errore (SS_E) e le medie quadratiche (MS), ed utilizzare la tecnica dell'analisi della varianza per stabilire quantitativamente i pesi degli effetti dei fattori e delle interazioni nell'esperimento in esame:

$$SS_E = \sum_i \sum_j (y_{ij} - y_{i_{media}})^2 \qquad (3.3)$$

dove i = 1…a con a = n° di combinazioni di trattamenti

j = 1…n con n = n° di replicazioni

y_{ij} = osservazione eseguita all'i-esimo trattamento per la j-esima replicazione

$y_{i\ media}$ = media dell'i-esimo trattamento

$MS_{AB...K} = SS_{AB...K}/(\nu_1)$

dove $\nu_1 = 1$ = gradi di libertà della somma dei quadrati dei contrasti

$MS_E = SS_E/\nu_2$

dove $\nu_2 = N-a$ = gradi di libertà dell'errore, con N che indica il numero totale delle osservazioni

Le grandezze statistiche risultanti sono distribuite come una F (teorema di Cochran):

$F\nu 1,\nu 2$ = *(Media quadratica del trattamento/Media quadratica dell'errore)*

Nel caso in cui il numero dei fattori sia molto elevato, risulta scomodo valutare i contrasti con la relativa formula sopra citata; allo scopo di ridurre al minimo le possibilità di errori di calcolo conviene utilizzare la *tavola dei segni*: esistono facili regole mnemoniche per poter ricavare la suddetta tavola. I segni per gli effetti principali sono determinati mediante associazione di segni "+" agli alti livelli e di segni "-" ai bassi livelli.

Una volta valutate tutte le grandezze sopra indicate, è possibile costruire la tabella dell'analisi della varianza, riportata nella tabella seguente, che riassume i calcoli svolti e indica l'influenza o meno di ogni fattore ai fini di una variazione della funzione obiettivo.

ORIGINE DELLA	SOMME DEI QUADRATI	GRADI DI LIBERTÀ	MEDIE QUADRATICHE	Fo

VARIAZIONE				
A	SS_A	1	MS_A	$Fo = MS_A/MS_E$
B	SS_B	1	MS_B	$Fo = MS_B/MS_E$
AB	SS_{AB}	1	MS_{AB}	$Fo = MS_{AB}/MS_E$
...	...	...		
ERRORE	SS_E	N-a	MS_E	
TOTALE	SS_T	$n2^k-1$		

Tab 3.2 *Analisi della varianza.*

Le ipotesi statistiche da controllare sono le seguenti:

1. ipotesi nulla Ho: le medie dei trattamenti non differiscono tra loro, cioè i livelli considerati per il fattore non provocano alcuna variazione sulla popolazione delle osservazioni;

2. ipotesi alternativa H1: le medie dei trattamenti differiscono tra loro per almeno un livello, cioè esiste per lo meno un livello del fattore considerato che risulta avere effetto sulla popolazione delle osservazioni.

Per valutare la significatività dei trattamenti, ci si riferisce al rapporto MS_{TRATT}/MS_E, che è distribuito come una $F_{1,N-a}$. Ciò consente di sottoporre a test l'ipotesi nulla, ad un determinato livello di significatività, utilizzando un test ad una coda della distribuzione F: ovvero l'ipotesi Ho sarà accettata al livello di significatività α se $Fo < F\alpha,\nu 1,\nu 2$, valore tabellato al variare del livello di significatività α.

3.4 La vagliatura dei fattori con ANSYS

Il codice ANSYS permette di effettuare, in automatico, tutte le prove di un progetto fattoriale 2^k (ANSYS non può svolgere le prove aggiuntive previste dal Progetto Composto Centrale). Occorre specificare i fattori ed i relativi intervalli di variazione. Può essere eseguito un progetto fattoriale completo o frazionato (da ½ sino a 1/64), il codice esegue le 2^k prove (2^{k-p} nel caso di progetto frazionato). Attraverso il comando OPLFA viene rappresentato sotto forma di istogramma l'effetto principale dei fattori (MAIN), l'effetto delle interazioni a due (2FAC) ed anche l'effetto delle interazioni a tre (3FAC). Nel capitolo 6 viene descritto in dettaglio l'utilizzo di questo strumento del codice ANSYS.

Il progetto fattoriale è stato utilizzato solo per la vagliatura delle variabili di progetto in quanto la superficie di risposta del nostro sistema, a causa della sua topologia, non poteva essere adattata con un modello regressivo del secondo ordine, utilizzato dai progetti 2^k, ma si sarebbe dovuto ricorrere a modelli di ordine superiore (5^k o addirittura superiori), possibilità non concessa da ANSYS e che comunque avrebbe richiesto tempi di calcolo eccessivi.

Come si fa solitamente in fase di vagliatura, non è stato condotto un progetto fattoriale completo ma uno frazionato ad 1/8, che ci ha permesso di valutare 10 variabili anziché sole 7, limitazioni, queste, del codice ANSYS.

Le variabili del nostro sistema sono (per ogni bobina):

dx_i	Larghezza (sezione trasversale)
dy_i	Altezza (sezione trasversale)
$coox_i$	Distanza dall'asse y
$cooy_i$	Distanza dall'asse x
$curr_i$	Densità di corrente

Il numero di bobine del magnete è 12; dal momento che, grazie alla simmetria rispetto al piano equatoriale, viene studiata solo la metà superiore della struttura, il numero di bobine risulta pari a 6. Di queste sei bobine ne abbiamo eliminata

una, in quanto sovrabbondante, riducendone il numero a 5. Il numero totale di variabili è quindi 5 per ogni bobina e quindi pari a 25. A causa del limite di 10 variabili che potevano essere valutate da ANSYS, sono stato obbligato ad escludere dall'analisi di significatività alcune di esse. Ho scelto di non considerare quelle che definiscono le dimensioni trasversali delle bobine (dx e dy), poiché tali variabili sostanzialmente modificano la distribuzione di densità di corrente, risultato che può essere ottenuto agendo sulle altre variabili (densità di corrente e posizione). Le dimensioni trasversali delle bobine entreranno in gioco nella fase di limatura della geometria, non dimenticando che le variabili dx e dy sono comunque condizionate da vincoli dimensionali costruttivi legati alle dimensioni dei conduttori utilizzati per avvolgere le bobine.

Le 10 variabili scelte per la determinazione della significatività sono le seguenti:

curr1: corrente bobina n. 1

curr2: corrente bobina n. 2

curr3: corrente bobina n. 3

curr4: corrente bobina n. 4

curr5: corrente bobina n. 5

x1: distanza dall'asse y della bobina n. 1

x2: distanza dall'asse y della bobina n. 2

x3: distanza dall'asse y della bobina n. 3

y3: distanza dall'asse x della bobina n. 3

y4: distanza dall'asse x della bobina n. 4

L'analisi di significatività è stata effettuata dal codice ANSYS mediante la determinazione dell'*effetto* dei fattori. Nella tabella 3.1 vengono riportati, sotto forma di istogrammi, gli effetti associati alle 10 variabili (fattori), presi in valore assoluto. Il codice ha determinato anche gli effetti associati alle interazioni a due ed a tre fattori, risultati del tutto trascurabili.

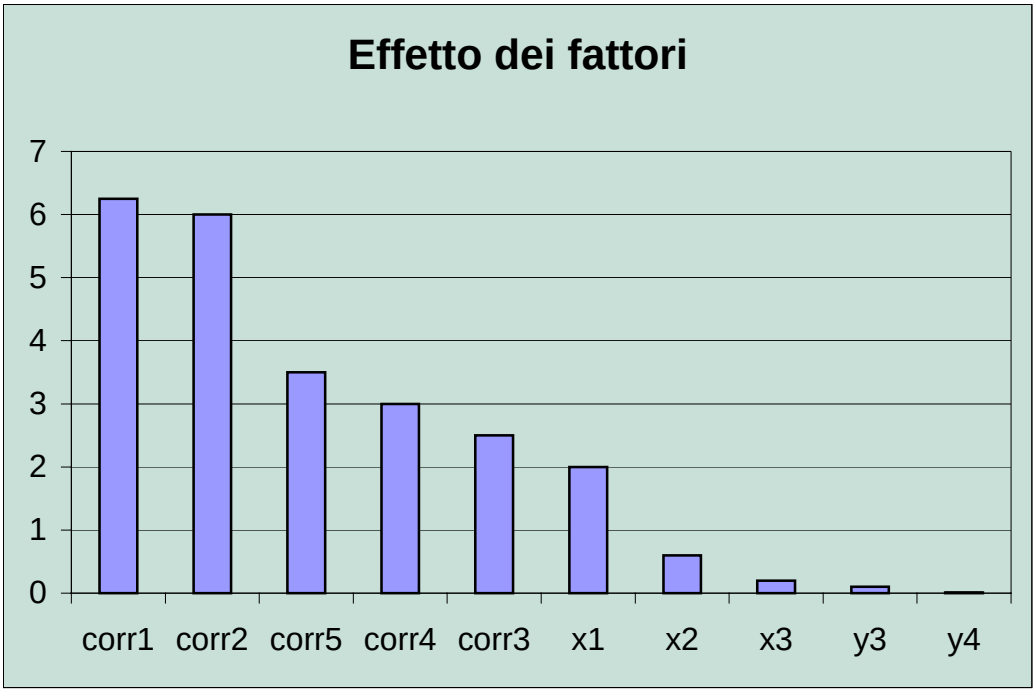


Fig 3.1 *Effetto dei fattori.*

I risultati di questa analisi mostrano che le correnti sono le variabili che più influiscono sull'uniformità del campo magnetico, seguite, per importanza, dalla distanza delle bobine rispetto all'asse y del magnete. Un ruolo di secondaria importanza è invece giocato dalla distanza delle bobine rispetto all'asse x.

Abbiamo ritenuto che questi risultati, associati non tanto ai singoli fattori, bensì alla loro tipologia, fossero risultati aventi validità generale, e che potessero essere quindi assunti validi anche per le variabili non prese in considerazione dalla nostra analisi. In pratica è come se avessimo considerato un magnete costituito da un numero inferiore di bobine.

Il risultato più significativo è la primaria importanza delle correnti circolanti nelle bobine: è su queste che dovrà concentrarsi l'attenzione maggiore del nostro studio.

3.5 Lo sdoppiamento della bobina centrale

Per migliorare le caratteristiche del campo magnetico generato si è deciso di sdoppiare la bobina centrale del magnete. Le motivazioni alla base di questa modifica sostanziale della geometria del dipolo sono le seguenti.

In un dipolo con una curvatura fisica degli avvolgimenti, il contributo al campo magnetico nella zona di interesse è diverso per le parti di avvolgimento a curvatura differente. In particolare le parti a curvatura minore, quelle esterne, danno un contributo minore rispetto a quelle interne, a curvatura maggiore. Questo significa che il centro magnetico tra le due parti con differente curvatura non corrisponde al centro geometrico, ma è spostato verso la parte a curvatura maggiore, quindi verso l'interno. Ciò pone dei problemi in quanto ci costringe a spostare la zona del fascio verso la curvatura maggiore, in una regione vicina agli avvolgimenti, dove è più difficile ottenere buone uniformità di campo. Per evitare questo effetto occorre aumentare la corrente della parte a curvatura minore in modo da compensarne il minor contributo al campo. Il solo modo di avere da una parte del dipolo una corrente maggiore che dal lato opposto è di dividere il dipolo in due bobine distinte. I due dipoli dovranno avere alternativamente un'andata od un ritorno esattamente nelle posizioni occupate dalla bobina che essi hanno sostituito, mentre gli altri due rami saranno posti il più lontano possibile, in modo da non cancellare completamente il contributo dei primi due rami. Otterremo allora un campo magnetico minore di quello dato da una singola bobina, ma bilanciato ed ottimizzabile in relazione al campo generato dalle altre bobine.

Il secondo motivo per cui questa soluzione è vantaggiosa è dovuto al fatto che la bobina centrale pone un problema per il passaggio del fascio di particelle. Se infatti fosse realizzata una bobina piana, le "teste" occuperebbero la zona del fascio. In questi casi occorre realizzare delle teste differenti con configurazione a sella di cavallo. Purtroppo nel caso in cui si scelga un dipolo curvato, si incontrano seri problemi di tipo costruttivo. La sostituzione di un'unica bobina centrale con due dipoli distaccati elimina questo problema, infatti la richiusura delle due

singole bobine non interferisce geometricamente con il fascio di particelle.

Nella figura 3.2 viene mostrato il modello agli elementi finiti della sezione del magnete; si può notare che sul piano equatoriale ora ci sono due dipoli. Sarà con questo modello (assialsimmetrico) che verrà condotto tutto il lavoro di ottimizzazione.

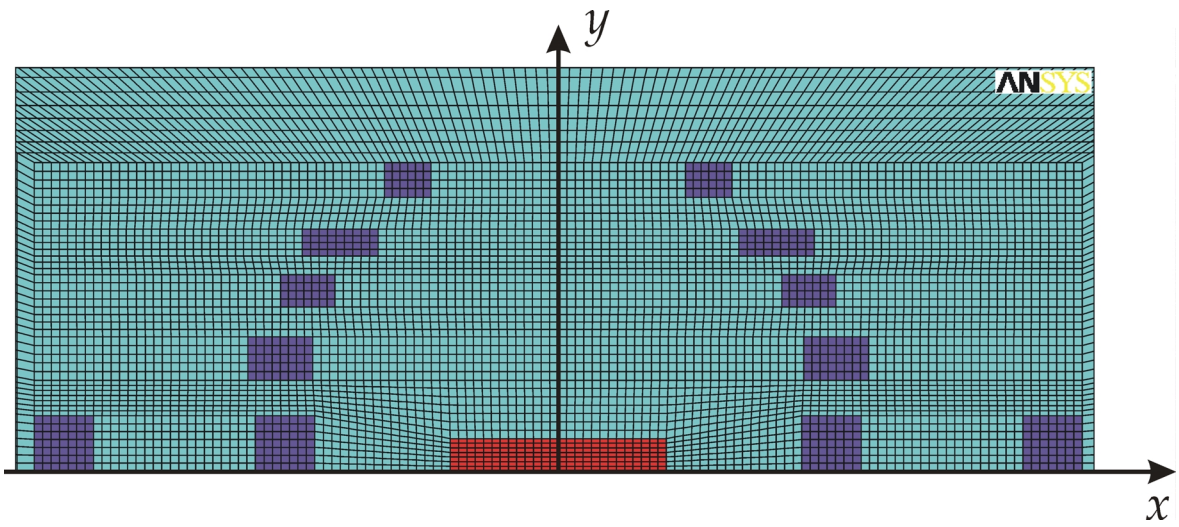


Fig. 3.2 *Modello agli elementi finiti della sezione del magnete.*

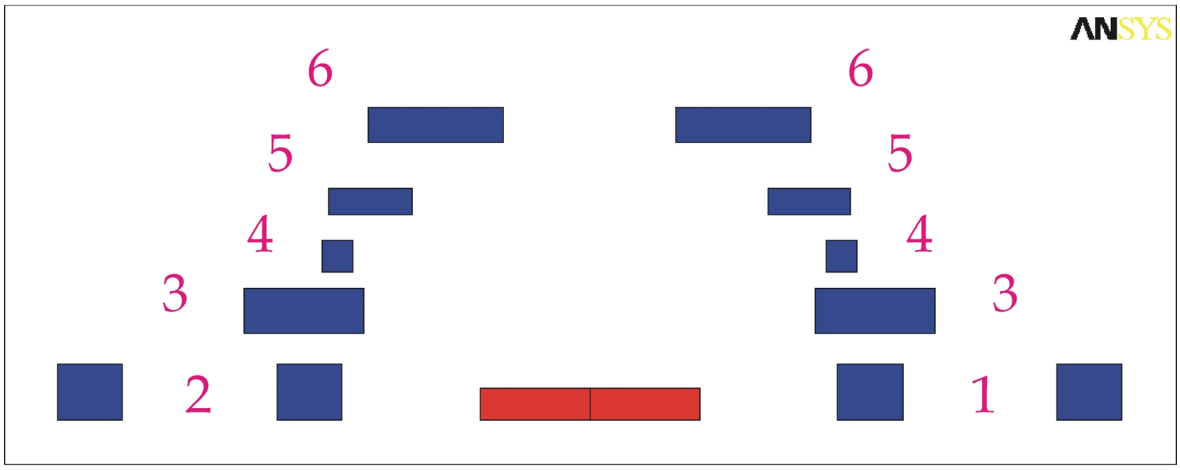


Fig. 3.3 *La nuova numerazione delle bobine*

La nuova numerazione delle bobine é mostrata in figura 3.3. L'area in rosso rappresenta la regione di interesse rispetto alla quale viene calcolata l'uniformitá

del campo magnetico.

4

Gli Algoritmi Genetici

4.1 Introduzione storica

Coloro che per primi si occuparono di cibernetica erano motivati in gran parte dal sogno di instillare, nei programmi per i calcolatori, l'intelligenza e la capacità adattativa di imparare e controllare l'ambiente circostante. Questi pionieri dell'informatica erano interessati alla biologia e alla psicologia tanto quanto all'elettronica e consideravano i sistemi naturali come metafore in grado di guidarli verso la realizzazione del loro sogno.

Queste attività informatiche che traggono spunto dalla biologia hanno avuto alti e bassi nel corso degli anni, ma fin dai primi anni Ottanta sono state tutte oggetto di rinnovato interesse nella comunità dei ricercatori informatici. Hanno così avuto origine il settore delle *Reti Neurali*, dell'*Apprendimento Automatico* e quella che oggi viene chiamata *Computazione Evolutiva*, di cui gli Algoritmi Genetici sono l'esempio più significativo.

4.2 Breve storia della computazione evolutiva

Negli anni ’50 e ’60 parecchi informatici studiarono, indipendentemente l’uno dall’altro, i sistemi evolutivi con l’idea che i meccanismi dell’evoluzione potessero essere usati come strumenti di ottimizzazione per i problemi di ingegneria. L’idea di base di tutti questi sistemi consisteva nel far evolvere una popolazione di soluzioni candidate per un dato problema, usando operatori che si ispiravano alla variabilità genetica ed alla selezione naturale.

Negli anni ’60, Rechenberg introdusse le “strategie evolutive”, metodo che utilizzò per ottimizzare parametri per strutture aerodinamiche.

Nel 1966, Fogel, Owens e Walsh svilupparono la “programmazione evolutiva”, una tecnica che consiste nel rappresentare le soluzione candidate per un dato obiettivo come macchine a stati finiti, che erano fatte evolvere mutandone casualmente i diagrammi di transizione di stato e selezionando i più adatti.

Le strategie evolutive, la programmazione evolutiva e gli algoritmi genetici formano, nel loro insieme, l’ossatura della computazione evolutiva.

Gli algoritmi genetici furono inventati da John Holland negli anni Sessanta e furono sviluppati dallo stesso Holland e dai suoi studenti e colleghi alla University of Michigan negli anni Sessanta e Settanta. A differenza delle strategie evolutive e della programmazione evolutiva, lo scopo originale di Holland non era quello di progettare algoritmi per risolvere problemi specifici, ma piuttosto quello di studiare formalmente il fenomeno dell’adattamento così come avviene in natura e di inventare tecniche per trasferire all’interno dei sistemi informatici i meccanismi dell’adattamento naturale. Il libro di Holland del 1975, *Adaptation in Natural and Artificial Systems*, presentava l’algoritmo genetico come un’astrazione dell’evoluzione biologica e forniva una struttura teorica per la nozione di adattamento nel contesto degli AG. L’algoritmo genetico di Holland è un metodo per passare da una popolazione di “cromosomi” (per esempio stringhe di zero e uno, o “bit”) ad una nuova popolazione usando una specie di ”selezione naturale”, assieme ad operatori di incrocio, mutazione e inversione che prendono

spunto dalla genetica. Ogni cromosoma è fatto di "geni" (per esempio bit), ognuno dei quali rappresenta un'istanza di un particolare "allele" (per esempio 0 o 1). L'operatore di selezione sceglie nella popolazione quei cromosomi che avranno l'opportunità di riprodursi, e in media i cromosomi più adatti produrranno più discendenti di quelli meno adatti. L'incrocio (crossover) scambia parti dei cromosomi, imitando la ricombinazione biologica fra due organismi dotati di un solo cromosoma (aploidi); la mutazione modifica in modo casuale i valori degli alleli in alcune posizioni del cromosoma; infine l'inversione cambia l'ordine di una sezione del cromosoma, cambiando l'ordine nel quale i geni sono sistemati.

L'introduzione da parte di Holland di un algoritmo basato su una popolazione, con incroci, inversioni e mutazioni, fu un'innovazione di grande portata. Le strategie evolutive di Rechenberg partivano con una "popolazione" di due individui, un genitore e un discendente, che era una versione mutata del genitore; le popolazioni con molti individui e gli incroci furono inclusi solamente nel seguito. La programmazione evolutiva usava soltanto la mutazione per introdurre variazioni. Inoltre Holland fu il primo a cercare di fondare la computazione evolutiva su una solida base teorica.

Negli untimi anni c'è stata un'intensa collaborazione tra ricercatori che studiano vari metodi di computazione evolutiva, e i confini tra algoritmi genetici, strategie evolutive, programmazione evolutiva e gli altri approcci evolutivi sono in parte scomparsi.

Perché ispirarsi all'evoluzione naturale per risolvere problemi computazionali? Per risolvere molti problemi ingegneristici è necessario ricercare la soluzione tra un numero enorme di possibili alternative: l'evoluzione biologica è una fonte di ispirazione, in quanto l'evoluzione in sé è, in effetti, un metodo di ricerca all'interno di un grandissimo numero di possibili "soluzioni". Nella biologia l'enorme insieme di possibilità è costituito dall'insieme delle possibili sequenze

genetiche, e le "soluzioni" desiderate sono organismi altamente idonei†, organismi con una forte capacità di sopravvivere e riprodursi nel loro ambiente.

4.3 Un po' di terminologia biologica

E' utile introdurre formalmente i principali termini biologici che saranno utilizzati nel corso di questo capitolo.

Tutti gli organismi viventi sono composti di cellule, e ogni cellula contiene la stessa dotazione di uno o più *cromosomi* – filamenti di DNA- che fungono da "progetto per l'organismo. Un cromosoma può essere idealmente diviso in *geni* – blocchi funzionali di DNA, ciascuno dei quali codifica una particolare proteina. In prima approssimazione, si può pensare che ogni gene codifichi una *caratteristica*, come ad esempio il colore degli occhi. Le diverse possibili "configurazioni" di una caratteristica (per esempio azzurro, marrone, ecc.) sono dette *alleli*. Ogni gene è collocato in un particolare *locus* (posizione) all'interno del cromosoma.

La totalità del materiale genetico (tutti i cromosomi presi insieme) è detto *genoma* dell'organismo. Il termine *genotipo* si riferisce al particolare insieme di geni contenuti in un genoma. Se due individui hanno genomi identici, si dice che hanno lo stesso genotipo. Il genotipo dà luogo, durante lo sviluppo fetale e successivo, al *fenotipo* dell'organismo – le sue caratteristiche fisiche e mentali come il colore degli occhi, l'altezza, le dimensioni del cervello e l'intelligenza.

Gli organismi i cui cromosomi si presentano a coppie sono detti *diploidi*; gli organismi i cui cromosomi sono appaiati sono detti *apolidi*. In natura la maggior parte delle specie che si riproducono sessualmente sono diploidi; fra queste vi sono gli esseri umani, ciascuno dei quali ha 23 paia di cromosomi in ogni cellula (non germinale) del corpo. Nel corso della riproduzione sessuale, si verifica la ricombinazione o incrocio (crossover): in ogni genitore avviene uno scambio di geni tra ogni coppia di cromosomi per formare un *gamete* (un cromosoma singolo),

† "fitness", che traduciamo con "idoneità", è il termine usato da Darwin per esprimere il grado di adattamento di un individuo all'ambiente, la sua idoneità a sopravvivere in esso.

poi i gameti provenienti dai due genitori si uniscono per creare un patrimonio completo di cromosomi diploidi.

Nella riproduzione sessuale aploide, i geni vengono scambiati tra i cromosomi monofilamentari dei due genitori.

Nella maggior parte delle applicazione degli algoritmi genetici si utilizzano individui aploidi, e in particolare individui con un solo cromosoma.

I discendenti sono soggetti alla *mutazione* durante la quale singoli nucleotidi (elementi fondamentali del DNA) possono essere modificati nel passaggio dal genitore al discendente: i cambiamenti spesso derivano da errori di copiatura. L'*idoneità (fitness)* di un organismo è definita tipicamente come la probabilità che l'organismi viva abbastanza per riprodursi *(vitalità)* o come una funzione del numero di discendenti che esso genera *(fertilità).*

Negli algoritmi genetici il termine *cromosoma* si riferisce alla soluzione del problema, codificata spesso in una stringa di bit. I geni sono sia singoli bit sia blocchi di bit adiacenti che codificano un particolare elemento della soluzione candidata. Un allele in una stringa di bit può assumere valore 0 o 1; per gli alfabeti di cardinalità maggiore gli alleli possono assumere più di due valori. L'incrocio consiste nello scambio di materiale genetico tra due genitori aploidi dotati di un solo cromosoma. La mutazione consiste nell'invertire i bit in un locus scelto a caso (o, nel caso di alfabeti più estesi, nel rimpiazzare il simbolo in un locus scelto casualmente con un nuovo simbolo anch'esso scelto a caso).

4.4 Elementi di base degli algoritmi genetici

Non esiste alcuna definizione rigorosa di "algoritmo genetico", accettata da tutti nella comunità della computazione evolutiva, che permetta di distinguere gli AG dagli altri metodi di computazione evolutiva. In ogni caso si puó dire che la maggior parte dei metodi chiamati "AG" hanno in comune almeno i seguenti elementi: una popolazione di cromosomi, la selezione che agisce in base all'idoneità, l'incrocio per produrre nuovi dipendenti e la mutazione casuale.

L'inversione - il quarto elemento degli AG di Holland – è impiegata raramente

nelle implementazioni odierne, e i suoi vantaggi, se ce ne sono, non sono ben definiti.

I cromosomi nella popolazione di un AG sono generalmente rappresentati da stringhe di bit. Ogni cromosoma puó essere pensato come un punto dello spazio di ricerca delle soluzioni candidate. L'AG ha bisogno di una funzione di idoneità che assegni un punteggio (idoneità) ad ogni cromosoma della popolazione corrente.

4.5 Gli operatori degli AG

La forma piú semplice di algoritmo genetico necessita di tre tipi di operatori: selezione, incrocio e mutazione.

Selezione : questo operatore seleziona i cromosomi destinati alla riproduzione. Quanto piú alta é l'idoneità del cromosoma, tanto piú frequentemente esso sarà scelto per la riproduzione.

Incrocio (crossover) : questo operatore sceglie casualmente un locus e scambia fra due cromosomi le sottosequenze che precedono e seguono questo locus per creare due discendenti. L'operatore di incrocio imita grossolanamente la ricombinazione biologica fra due organismi con un solo cromosoma (aploidi).

Mutazione : questo operatore scambia il valore di alcuni bit nel cromosoma. La mutazione puó avvenire in ogni posizione della stringa con una certa probabilità, solitamente bassa rispetto a quella dell'incrocio.

4.6 Un semplice algoritmo genetico

Dato un problema ben definito da risolvere e una rappresentazione delle soluzioni candidate come stringhe di simboli, un semplice AG funziona cosí:

1. Inizia con una popolazione generata casualmente di *n* cromosomi formati da *l* bit (soluzioni candidate per il problema).
2. Calcola l'idoneità *f(x)* di ogni cromosoma *x* della popolazione.

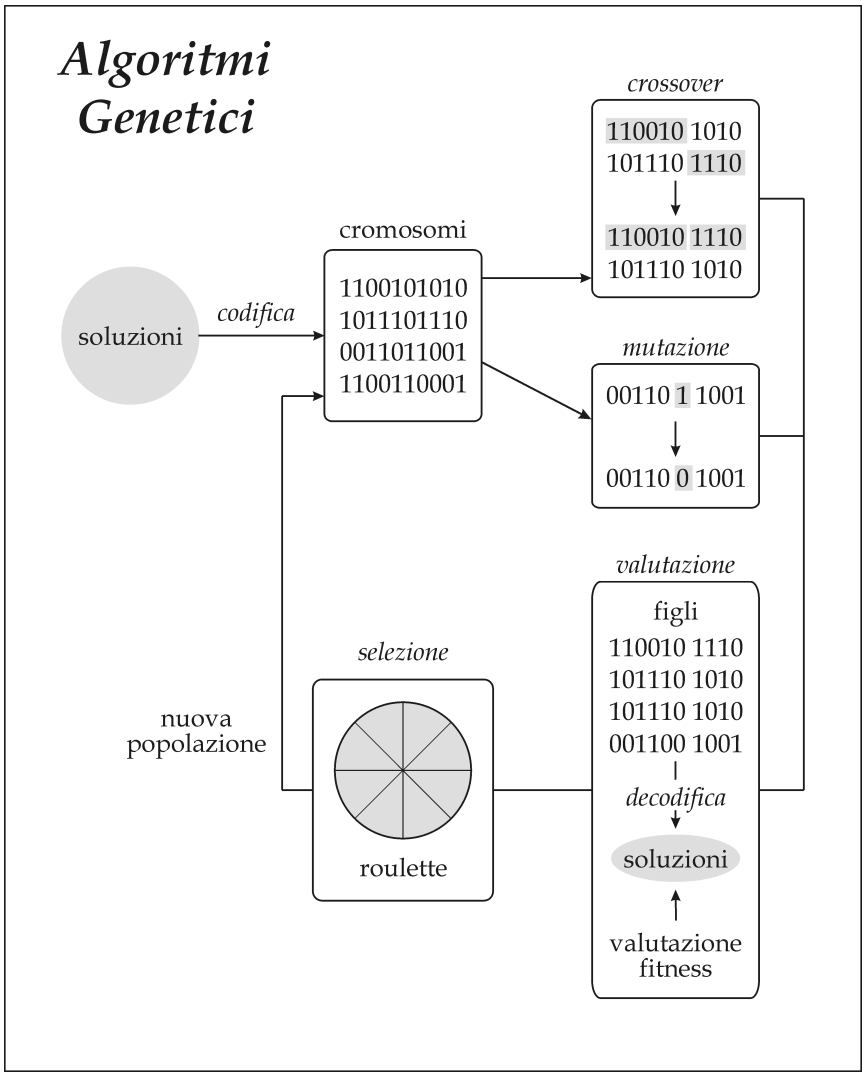


Fig 4.1 *Struttura generale di un algoritmo genetico.*

3. Ripete i tre passaggi seguenti finchè non sono stati creati *n* discendenti, uno per ogni cromosoma:
    a. Seleziona una coppia di cromosomi genitori dalla popolazione medesima, essendo la probabilità di selezione una funzione crescente dell'idoneità. La selezione viene eseguita "con sostituzione", il che significa che lo stesso cromosoma puó essere selezionato piú di una volta per diventare genitore.
    b. Con probabilità p_c (probabilità di incrocio), incrocia la coppia in un punto scelto a caso (con probabilità uniforme) per formare due discendenti. Se non avviene l'incrocio, produce discendenti che sono copie esatte dei rispetivi genitori. (Ci sono versioni di AG in cui si ha incrocio in piú punti).
    c. Muta i due discendenti con una probabilità p_m (probabilità di mutazione) in un locus scelto casualmente.
4. Sostituisce la popolazione corrente con la nuova popolazione.
5. Ritorna al passo 2.

Ogni iterazione di questo processo é chiamata *generazione*. Un AG viene generalmente iterato fra 50 e 500 generazioni. L'intera serie di esecuzioni é chiamata *esecuzione*. Alla fine dell'esecuzione ci sono spesso nella popolazione uno o piú cromosomi di alta idoneità. Poichè la casualità ha un ruolo importante in ogni esecuzione, due esecuzioni con diversi semi di numeri casuali produrranno, in generale, risultati diversi. I ricercatori del settore degli AG riportano spesso statistiche (per esempio la migliore idoneità trovata in un'esecuzione e la generazione nella quale l'individuo con la migliore idoneità é stato trovato) calcolate su molte esecuzioni diverse dell'AG applicato allo stesso problema.

La semplice procedura appena descritta sta alla base della maggior parte degli AG. Ci sono molti dettagli da considerare, come la dimensione della popolazione e la probabilità di incrocio e mutazione, e il successo dell'algoritmo dipende spesso da questi dettagli.

Un metodo comune di selezione negli AG é la *selezione dipendente dall'idoneità*, nella quale il numero di volte che si prevede che l'individuo si riprodurrà é uguale alla sua idoneità divisa per l'idoneità media della popolazione. Un semplice metodo per implementare la selezione dipendente dall'idoneità é il "metodo della roulette" [1], che concettualmente equivale ad associare ad ogni individuo un settore della ruota di una roulette di area proporzionale all'idoneità dell'individuo stesso. La ruota viene fatta girare, la pallina si ferma in un settore, e l'individuo corrispondente viene selezionato.

4.7 Come funzionano gli algoritmi genetici?

La teoria tradizionale degli AG (formulata per la prima volta da Holland nel 1975) ipotizza che, ad un livello descrittivo molto generale, gli AG funzionino scoprendo, favorendo e ricombinando buoni "blocchi costitutivi" dei cromosomi nei quali sono codificate le configurazioni del sistema in esame. L'ipotesi di fondo é che le buone soluzioni tendono ad essere formate da buoni blocchi costitutivi – combinazioni di valori di bit che conferiscono maggiore idoneità alle stringhe in cui si presentano.

Holland (1975) introdusse la nozione di *schema* per formalizzare la nozione informale di blocco costitutivo. Uno schema é un insieme di stringhe di bit che possono essere descritte da un modello costituito da uno, zero e asterischi che rappresentano caratteri jolly. Per esempio, lo schema H=1****1 rappresenta l'insieme di tutte le stringhe di sei bit che cominciano e finiscono con 1. Le stringhe che corrispondono a questo modello vengono dette *istanze* di H. Si dice che lo schema H contiene due bit *definiti* (non asterischi) o, in modo equivalente, che H é di *ordine* 2. La sua *lunghezza di definizione* (la distanza tra i suoi bit definiti piú esterni) é 5. Presupposto fondamentale della teoria tradizionale degli AG é che gli schemi sono, implicitamente, i blocchi costitutivi sui quali l'AG agisce effettivamente per mezzo degli operatori di selelezione, mutazione e incrocio a punto singolo.

L'analisi degli schemi condotta da Holland porta a concludere che un AG, pur

calcolando esplicitamente l'idoneità di soli N membri di una popolazione, stima implicitamente l'idoneità media di un insieme assai piú esteso di schemi, poiché calcola in modo implicito l'idoneità media osservata di tutti gli schemi che si presentano nella popolazione. Questo avviene senza bisogno di tempo di computazione aggiuntivo, oltre a quello necessario al trattamento degli N membri della popolazione. Holland denominó questo meccanismo "parallelismo implicito".

Il *Teorema degli Schemi* formulato da Holland sostiene che schemi brevi e di basso ordine, la cui idoneità media sia superiore alla media generale, saranno rappresentati da un numero di campioni (cioé di istanze valutate) crescente in modo esponenziale nel tempo.

L'incrocio (crossover) é considerato una delle fonti principali della potenza di un AG, per la sua capacità di ricombinare istanze di buoni schemi per formare istanze di schemi di ordine superiore altrettanto buoni o migliori. La supposizione che questo sia il processo mediante il quale gli AG operano é nota con il nome di *Building Block Hypothesis* (Goldberg, 1989). L'effetto della selezione é quello di indirizzare gradualmente la procedura di campionamento verso istanze di schemi la cui idoneità é stimata essere superiore alla media. Qual é il ruolo della mutazione? Secondo Holland la mutazione é ció che impedisce la perdita di diversità in una popolazione. Per esempio, senza la mutazione potrebbe capitare che ciascuna stringa nella popolazione abbia un 1 nella prima posizione, e allora non ci sarebbe modo di ottenere una stringa che inizi con uno zero.

Il Teorema degli Schemi e alcune delle sue presunte implicazioni che esso ha sul comportamento degli AG sono un argomento assai controverso, e di recente sono stati oggetto di molte discussioni critiche nella comunità degli AG.

4.8 Quando occorre utilizzare un algoritmo genetico?

Data una particolare applicazione, come possiamo sapere se un AG é un buon metodo da utilizzare? Non c'é una risposta rigorosa, anche se molti ricercatori concordano nel ritenere che se lo spazio da esplorare é vasto, se si sa che non é

perfettamente liscio e unimodale (vale a dire consistente di un'unica "collina" che sale dolcemente), o non lo si conosce bene, o se la funzione di idoneità (funzione obiettivo) é alterata dal rumore, un AG avrà buone possibilità di essere competitivo. Se lo spazio é liscio o unimodale, un algoritmo di salita lungo il gradiente, come la salita lungo la massima pendenza (steepest ascent), sarà molto piú efficiente di un AG dal momento che quest'ultimo non sfrutta le informazioni della derivata. Se la funzione di idoneità é rumorosa (per esempio se richiede che si effettuino misurazioni soggette a errore sperimentale relative ad un processo del mondo reale) un metodo di ricerca che consideri una sola soluzione candidata per volta, come la salita semplice, potrebbe essere irrecuperabilmente indotto in errore dal rumore, mentre gli AG, dato che funzionano accumulando statistiche di idoneità nel corso di molte generazioni, sono pensati per funzionare bene in presenza di piccole quantità di rumore.

Comunque ***la prestazione di un AG dipenderà molto dai dettagli, come il metodo di codifica delle soluzioni candidate, le probabilità associate agli operatori, la dimensione della popolazione ed il criterio per tradurre numericamente l'idoneità degli individui***.

4.9 Codifica dei problemi per un algoritmo genetico

Come per tutti i metodi di ricerca e apprendimento, il modo con cui le soluzioni candidate sono codificate é, se non *il* principale, almeno uno dei fattori principali del successo di un algoritmo genetico. La maggior parte delle applicazioni degli AG usano, per codificare le soluzioni candidate, stringhe binarie. Di recente sono stati peró effettuati numerosi esperimenti con altri tipi di codifica.

4.9.1 Codifiche binarie

Le codifiche binarie sono le codifiche piú comuni per parecchie ragioni. Una di esse è storica: nei loro primi studi, Holland e i suoi studenti si concentrarono su tali codifiche, e la pratica degli AG ha sostanzialmente seguito questo indirizzo.

Gran parte di questa teoria puó essere estesa a codifiche non binarie, ma queste estensioni non sono state sviluppate con l'accuratezza della teoria originale. Inoltre le euristiche per assegnare valori appropriati ai parametri (ad esempio ai coefficienti di mutazione e di incrocio) sono state generalmente sviluppate nel contesto della codifica binaria.

Nonostante questi vantaggi, le codifiche binarie risultano innaturali per molti problemi (ad esempio per l'evoluzione dei pesi nelle reti neurali).

4.9.2 Codifiche a piú caratteri e a valori reali

Per molte applicazioni é piú naturale formare cromosomi con un alfabeto formato da molti caratteri o con numeri reali. Da molti confronti empirici compiuti tra le codifiche binarie e le codifiche a piú caratteri o a valori reali sono emersi risultati migliori per queste ultime. Ma la prestazione di un AG dipende in gran parte dal problema e dalle particolari scelte di implementazione, e non sono ancora disponibili criteri precisi per prevedere quale codifica funzionerà meglio.

4.9.3 Codifiche ad albero

Gli schemi ad albero (utilizzati ad esempio per la rappresentazione di programmi) hanno parecchi vantaggi, tra cui il fatto che permettono di muoversi in spazi di ricerca illimitati. In linea di principio, infatti, per mezzo dell'incrocio e della mutazione si potrebbero formare alberi di qualsiasi dimensione. Questa assenza di limiti puó portare anche degli svantaggi. Gli alberi possono crescere senza controllo, impedendo la formazione di soluzioni candidate strutturate in modo piú gerarchico, inoltre gli alberi risultanti, essendo grandi, possono essere assai difficili da capire e da semplificare. Gli esperimenti volti a valutare sistematicamente l'utilità delle codifiche ad albero e a confrontarne i risultati con quelli di altre codifiche, sono appena cominciati nell'ambito della comunità della programmazione genetica. Analogamente, ci sono pochissimi tentativi, ancora in embrione, di estendere la teoria degli AG alle codifiche ad albero.

4.9.4 Scelta del tipo di codifica

Queste sono soltanto le codifiche piú comuni, ma una ricerca nella letteratura sugli AG rivelerebbe certamente esperimenti su molte altre.

Come si decide quale sia la codifica giusta per un certo problema? Alcuni ricercatori sostengono che la codifica migliore é la codifica piú naturale per il problema in questione; in un secondo momento si procederà alla progettazione dell'algoritmo genetico che faccia uso di tale codifica. La maggior parte della ricerca attuale parte dal tentativo di indovinare la codifica piú appropriata seguita dalla scelta di una versione dell'AG adatta a tale codifica.

Un'idea affascinante é quella che la codifica stessa si adatti in modo che l'AG possa sfruttarla nel miglior modo possibile.

4.10 Metodi di selezione

Una volta scelta la codifica, la seconda decisione da prendere prima di usare un algoritmo genetico riguarda il metodo si selezione – cioè il modo in cui si scelgono in una popolazione gli individui che generano discendenti per formare la generazione successiva, e il numero di discendenti che ciascuno può generare. Lo scopo della selezione è favorire gli individui più idonei della popolazione nella speranza che i loro discendenti abbiano idoneità ancora maggiore. La selezione deve essere bilanciata dalle variazioni indotte da incrocio e mutazione (equilibrio sfruttamento/esplorazione): una selezione troppo forte produrrà una popolazione dominata da individui subottimali di alta idoneità, con una conseguente riduzione della diversità necessaria per ulteriori mutamenti e progressi; una selezione troppo debole conduce ad un'evoluzione troppo lenta. Come per le codifiche, nella letteratura sugli AG sono stati proposti numerosi schemi di selezione.

4.10.1 Selezione proporzionale all'idoneità con il "metodo della roulette"

L'AG originale di Holland impiegava una selezione proporzionale all'idoneità, in cui il "valore atteso" di un individuo, cioè il numero atteso di volte in cui l'individuo è scelto per la riproduzione, è l'idoneità dell'individuo divisa per l'idoneità media della popolazione. Il metodo di implementazione più comune è il campionamento con il "metodo della roulette" già descritto: ad ogni individuo si assegna una fetta di una ruota di una roulette, e la grandezza di questa fetta è proporzionale all'idoneità dell'individuo. La ruota viene fatta girare N volte, ove N è il numero di individui della popolazione. Ad ogni giro, all'individuo nella cui fetta si ferma la pallina è concesso di essere tra i genitori della generazione successiva.

Questo metodo stocastico porta statisticamente ad avere il numero atteso di discendenti per ogni individuo. Tuttavia, con le popolazioni relativamente piccole impiegate negli AG, il numero di discendenti effettivamente assegnati ad ogni individuo spesso si discosta di molto dal valore atteso (una serie di giri della roulette estremamente improbabile potrebbe addirittura assegnare tutti i discendenti all'individuo peggiore di tutta la popolazione). Un altro inconveniente è dato dal fatto che, nelle fasi iniziali della ricerca, la varianza dell'idoneità nella popolazione è elevata, e c'è un numero ristretto di individui molto più idonei degli altri. Con la selezione proporzionale all'idoneità, questi individui e i loro discendenti si moltiplicano velocemente nella popolazione, e impediscono di fatto all'AG di effettuare ulteriori esplorazioni. Questo fenomeno è conosciuto come "convergenza prematura". In altre parole, la selezione proporzionale all'idoneità spesso enfatizza troppo lo sfruttamento delle stringhe di alta idoneità a scapito dell'esplorazione di altre regioni dello spazio di ricerca. Nelle fasi successive della ricerca, quando tutti gli individui della popolazione sono molto simili (e la varianza dell'idoneità è bassa), non ci sono differenze di idoneità tanto significative da poter essere sfruttate dalla selezione, e l'evoluzione rallenta sin

quasi a fermarsi. Dobbiamo concludere che la velocità dell'evoluzione dipende dalla varianza dell'idoneità della popolazione.

4.10.2 Cambiamento di scala sigma

Per risolvere questi problemi, i ricercatori degli AG hanno tentato diversi metodi di "cambiamento di scala" – metodi per trasformare i valori "grezzi" dell'idoneità nei valori attesi tali da rendere l'AG meno soggetto a convergenza prematura.Un esempio è il *cambiamento di scala sigma*, che mantiene la pressione evolutiva (cioè la misura in cui si concedono discendenti numerosi agli individui altamente idonei) relativamente costante nel corso dell'esecuzione, a prescindere dalla varianza dell'idoneità della popolazione. Con il cambiamento di scala sigma il valore atteso di un individuo è funzione della sua idoneità, della media della popolazione e della deviazione standard.

All'inizio di un'esecuzione, quando la deviazione standard dell'idoneità è generalmente molto alta, gli individui più idonei non saranno i più favoriti nell'assegnazione dei discendenti. Inoltre in momenti successivi dell'esecuzione, quando la popolazione è più omogenea e la deviazione standard più bassa, gli individui più adatti spiccheranno di più, permettendo all'evoluzione di procedere.

4.10.3 L'Elitarismo

L'Elitarismo, introdotto per la prima volta da Kenneth De Long (1975), è un'aggiunta a molti metodi di selezione che impone all'AG di conservare ad ogni generazione un certo numero di individui, i migliori. Questi ultimi potrebbero andare perduti se non fossero scelti per la riproduzione o se fossero distrutti da incrocio e mutazione. Molti sperimentatori hanno notato che l'elitarismo migliora significativamente la prestazione degli AG.

4.10.4 Selezione di Boltzmann

Il cambiamento di scala sigma mantiene la pressione evolutiva più o meno costante nel corso di ogni esecuzione. Spesso però i diversi momenti di

un'esecuzione richiedono pressioni evolutive differenti – per esempio, inizialmente sarebbe opportuno essere generosi, consentendo agli individui meno idonei di riprodursi quasi quanto quelli più idonei, e far procedere lentamente la selezione per mantenere la diversità nella popolazione. In seguito sarebbe meglio rafforzare la selezione per favorire decisamente gli individui ad alta idoneità, presumendo che la fase iniziale, con grande diversità e poca selezione, abbia consentito alla popolazione di individuare la zona giusta nello spazio di ricerca. Un approccio possibile è la *selezione di Boltzmann* (un approccio simile alla ricottura simulata), in cui la "temperatura" che varia continuamente influenza il tasso di selezione secondo uno schema prefissato. La temperatura all'inizio è elevata, il che significa che la pressione evolutiva è bassa (cioè che ogni individuo ha una ragionevole probabilità di riprodursi). Poi la temperatura si abbassa gradualmente, e quindi la pressione evolutiva aumenta gradualmente, il che permette all'algoritmo genetico di circoscrivere con precisione crescente la parte migliore dello spazio di ricerca, pur mantenendo sempre il grado di diversità "appropriato".

Negli ultimi anni sono diventati sempre più comuni approcci diversi quali la *selezione in base al rango* e la *selezione a torneo*.

4.10.5 Selezione in base al rango

La selezione in base al rango è un metodo alternativo il cui scopo è di impedire una convergenza troppo rapida. Nella versione proposta da Baker (1985) gli individui della popolazione sono classificati in base all'idoneità, il valore atteso di un individuo dipende dalla sua posizione nella classifica (il suo "rango") invece che dalla sua idoneità assoluta. In questo caso non c'è bisogno di effettuare un cambiamento di scala sulle idoneità, dato che si trascurano le differenze assolute di idoneità. Scartare le informazioni relative all'idoneità assoluta presenta sia vantaggi (l'uso dell'idoneità assoluta presenta problemi di convergenza), sia svantaggi (in alcuni casi può essere importante sapere che un individuo è molto più idoneo del suo sfidante più vicino). La selezione in base al rango evita che

piccoli gruppi di individui di alta idoneità si assicurino la maggioranza dei discendenti, e quindi riduce la pressione selettiva quando la varianza dell'idoneità è alta. D'altra parte essa mantiene elevata la pressione selettiva quando la varianza è ridotta. La selezione in base al rango ha un potenziale svantaggio: il rallentamento della pressione selettiva significa che in alcuni casi l'AG sarà più lento nel trovare individui di alta idoneità. Tuttavia, in molti casi, la maggiore salvaguardia della diversità derivante dalla selezione in base al rango conduce ad una ricerca più fruttuosa rispetto alla convergenza veloce che può essere prodotta dalla selezione proporzionale alla idoneità.

4.10.6 Selezione a torneo

I metodi proporzionali all'idoneità descritti sopra richiedono due operazioni sull'intera popolazione per ogni generazione: prima il calcolo dell'idoneità media (e, per il cambiamento di scala sigma, della deviazione standard), poi il calcolo del valore atteso di ogni individuo. La classificazione in base al rango richiede il riordinamento dell'intera popolazione, una procedura potenzialmente costosa in termini di tempo. La selezione a torneo è simile alla selezione in base al rango per quanto riguarda la pressione selettiva, ma è più efficiente dal punto di vista computazionale e più adatta per l'implementazione parallela. Si scelgono a caso due individui nella popolazione, poi si sceglie un numero casuale r compreso tra 0 e 1. Se $r < k$ (dove k è un parametro, per esempio 0.75), si sceglie come genitore il più idoneo dei due; altrimenti si sceglie l'individuo meno idoneo. Poi i due sono rimessi nella popolazione originale e possono essere scelti di nuovo.

La selezione a torneo è quella adottata dall'AG utilizzato in questa tesi.

4.10.7 Selezione a stato stazionario

La maggior parte degli AG descritti nella letteratura sono "generazionali" – ad ogni generazione la nuova popolazione è interamente costituita da discendenti di genitori che appartengono alla generazione precedente (anche se alcuni di questi discendenti potrebbero essere identici ai genitori). In alcuni casi, per esempio

nell'elitarismo descritto sopra, le generazioni successive hanno un certo grado di sovrapposizione – una parte della generazione precedente si conserva nella nuova popolazione. La percentuale di individui nuovi in ogni generazione è stata definita *divario generazionale*. Nella selezione a stato stazionario, tra una generazione e l'altra si rimpiazzano solo alcuni degli individui: solitamente un numero ridotto degli individui meno idonei è sostituito da discendenti risultanti dall'incrocio e dalla mutazione degli individui più idonei.

4.11 Operatori genetici

La terza decisione da prendere nell'implementazione di un algoritmo genetico è la scelta di quali operatori genetici utilizzare. Questa decisione dipende ampiamente dalla strategia di codifica. In questa sede prenderemo in esame l'incrocio e la mutazione nel contesto delle codifiche binarie.

4.11.1 L'incrocio

Si potrebbe affermare che la caratteristica specifica degli AG sia l'uso dell'incrocio. L'incrocio in un punto è la forma più semplice: si sceglie a caso una singola posizione di incrocio e si scambiano le parti dei due genitori che seguono e precedono la posizione di incrocio per formare due discendenti.

L'idea è ovviamente la ricombinazione di blocchi appartenenti a stringhe diverse. L'incrocio in un punto ha però alcuni difetti. Per citarne uno, non può

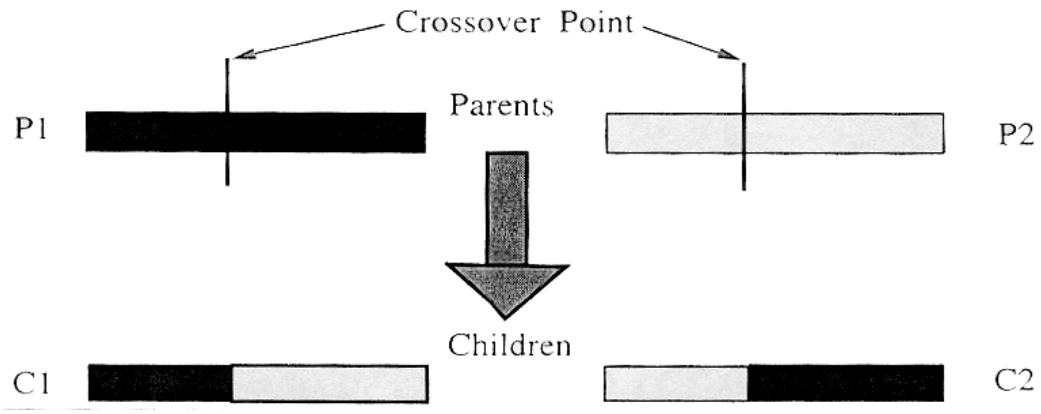


Fig 4.2 *Schema procedurale dell'incrocio ad un punto.*

combinare tutti gli schemi possibili. Per esempio non può in generale combinare istanze di 11*****1 e di ****11** per formare un'istanza di 11**11*1. Analogamente è probabile che gli schemi con elevata lunghezza di definizione siano distrutti dall'incrocio in un punto. L'incrocio in un punto presuppone che gli schemi brevi e di basso ordine siano i blocchi costitutivi funzionali, anche se in generale non si sa in anticipo quali ordinamenti di bit riuniscano insieme i bit legati dal punto di vista funzionale. Per evitare questo fatto molti di coloro che applicano gli AG usano l'incrocio in due punti, in cui si scelgono a caso due posizioni e si scambiano i segmenti compresi.

L'incrocio in due punti ha meno probabilità di distruggere schemi con elevata lunghezza di definizione e può assemblare più schemi dell'incrocio in un punto. Inoltre i segmenti scambiati non contengono necessariamente gli estremi delle stringhe. Alcuni ricercatori sono convinti della superiorità dell'incrocio parametrico uniforme in cui gli scambi si verificano in ogni posizione con probabilità p (in genere $0.5 \leq p \leq 0.8$). L'incrocio parametrico uniforme non ha dipendenza posizionale: qualunque schema contenuto in posizioni diverse dei genitori può essere ricombinato nel discendente.

Come si può scegliere tra queste varianti? Non c'è una risposta semplice, il successo o il fallimento di un particolare operatore di incrocio dipende in modo complicato dalla funzione di idoneità, dalla codifica e da altri particolari dell'AG. La comprensione completa di queste interazioni è un problema molto importante ed è ancora aperto. Nelle più recenti applicazioni degli AG si usano comunemente l'incrocio in due punti o l'incrocio uniforme con $p = 0.7$-0.8.

Considerando i dubbi che sono sorti circa l'importanza degli schemi, come strumento di analisi per la comprensione degli AG, sorge il dubbio se non si debba prendere in considerazione la possibilità che l'incrocio sia utile per un motivo completamente diverso, per esempio perché esso è essenzialmente un operatore di "macro-mutazione" che non fa altro che consentire ampi spostamenti nello spazio di ricerca.

4.11.2 La mutazione

Un'opinione comune nella comunità degli AG, che risale al libro di Holland *Adaption in Natural and Artificial Systems,* è che l'incrocio sia il principale strumento di variazione e innovazione negli AG, mentre la mutazione provvede ad assicurare che la popolazione non assuma valori fissati in certe posizioni, e quindi ricopre un ruolo più defilato. Sono stati svolti alcuni studi comparativi sulla forza della mutazione rispetto all'incrocio: il punto fondamentale non è la scelta tra l'incrocio e la mutazione quanto l'equilibrio tra incrocio, mutazione e selezione. Il giusto equilibrio dipende anche dai particolari della funzione di idoneità e della codifica. Inoltre l'utilità relativa dell'incrocio e della mutazione varia nel corso di ogni esecuzione.

4.11.3 Altri operatori e strategie di accoppiamento

Anche se molte delle applicazioni degli AG usano soltanto l'incrocio e la mutazione, nella letteratura sugli AG sono stati impiegati molti altri operatori, con le relative strategie per applicarli.

De Long introdusse un operatore "sfollamento" tale che ogni discendente appena formato rimpiazzava l'individuo esistente più simile. Questo impediva che nella popolazione ci fossero troppi individui simili ("folle") nello stesso momento. Goldberg e Richardson (1987) hanno ottenuto un risultato simile usando una funzione di "condivisione di idoneità": l'idoneità di ogni individuo veniva ridotta dalla presenza di altri membri della popolazione, e la misura della diminuzione causata da ogni altro membro della popolazione era una funzione crescente della somiglianza tra i due individui. Quindi, gli individui che erano simili a molti altri venivano puniti, mentre gli individui che erano diversi venivano premiati. Goldberg e Richardson hanno dimostrato che in alcuni casi questa tecnica potrebbe indurre una "speciazione" appropriata, poiché ha l'effetto di far convergere i membri della popolazione verso un certo numero di cime diverse nel paesaggio di idoneità invece di farli convergere tutti verso la stessa cima.

Un altro modo per favorire la diversità consiste nell'imporre restrizioni sull'accoppiamento. Per esempio, se si permette di accoppiarsi solo a individui abbastanza simili, tenderanno a formarsi delle "specie" (gruppi di accoppiamento) distinte. Altri ricercatori hanno seguito la strada opposta, vietando gli accoppiamenti tra individui abbastanza simili ("incesto"). Il loro intento era evitare la formazione di specie e mantenere invece la massima diversificazione possibile all'interno della popolazione.

Infine, vi sono stati alcuni esperimenti con restrizioni spaziali all'accoppiamento: la popolazione evolve su un reticolo spaziale, e gli individui hanno la possibilità di accoppiarsi solo con individui che occupano posizioni vicine.

4.11.4 Parametri per gli AG

La quarta decisione da prendere quando si implementa un algoritmo genetico è come scegliere i valori per i vari parametri, come la dimensione della popolazione, il coefficiente di incrocio e il coefficiente di mutazione. Questi parametri in genere interagiscono tra loro non linearmente, per cui non è possibile ottimizzarli uno alla volta. Nella letteratura sulla computazione evolutiva c'è un dibattito talmente ampio sulla scelta dei parametri e sugli approcci per adattarli, che è impossibile anche solo elencare le diverse posizioni. Non ci sono risultati conclusivi su cosa sia meglio; la maggior parte dei ricercatori usa ciò che ha dato i risultati migliori nei casi riportati in letteratura. In questo paragrafo verranno esaminati alcuni degli approcci sperimentali seguiti per trovare i "migliori" valori dei parametri. Dagli esperimenti di De Long è risultato che la miglior dimensione della popolazione era di 50-100 individui, il miglior coefficiente di incrocio era circa 0.6 per coppia di genitori e il miglior coefficiente di mutazione era 0.001 per bit. Alcuni anni dopo Grefenstette (1986) ha notato che, poiché l'AG era utilizzato come procedura di ottimizzazione, poteva essere utilizzato anche per ottimizzare i parametri di un altro AG! I risultati furono i seguenti: dimensione della popolazione 30, coefficiente di incrocio 0.95 e coefficiente di mutazione 0.01.

L'esperimento è stato interessante ma, ancora una volta, dato che si è utilizzata una sequenza di prova specializzata, non è chiaro quanto queste indicazioni siano generalizzabili.

E' verosimile che la dimensione della popolazione e i coefficienti di incrocio e di mutazione ottimali cambino nel corso di un'esecuzione.

4.11.5 Esplorazione e sfruttamento

Un qualsiasi algoritmo di ottimizzazione efficiente deve usare due tecniche per trovare l'estremo assoluto: *esplorazione*, per esaminare nuove e sconosciute aree dello spazio di ricerca, e *sfruttamento*, per fare uso dei punti precedentemente visitati come utili informazioni per trovare punti migliori. Queste due esigenze sono complementari e un buon algoritmo di ricerca deve trovare un buon compromesso tra le due. Una ricerca puramente casuale è buona per l'esplorazione, ma non attua alcuno sfruttamento, mentre un metodo puramente gradientale (hillclimbing) è buono per lo sfruttamento ma fa poca esplorazione. La combinazione di queste due tecniche può essere abbastanza efficace, ma è difficile sapere dove si trova l'equilibrio migliore (cioè quanto sfruttamento bisogna fare per poi arrendersi ed esplorare di più?).

Holland ha dimostrato che un AG combina insieme esplorazione e sfruttamento allo stesso tempo e in modo ottimale. Comunque, sebbene questo sia teoricamente vero, nella pratica ci sono problemi inevitabili. Holland infatti ha fatto certe semplificazioni:

- la popolazione è infinita;
- la funzione di idoneità riflette perfettamente l'idoneità della soluzione;
- i geni nel cromosoma non interagiscono significativamente;

La prima ipotesi non può mai essere verificata in pratica e a causa di ciò il funzionamento dell'AG sarà soggetto ad errori stocastici. Un problema del genere, che si trova anche in natura, è quello della *deriva genetica* (genetic drift): anche in assenza di qualsiasi pressione di selezione (cioè la funzione di idoneità è una costante), i membri della popolazione convergeranno verso qualche punto dello

spazio di ricerca. Questo succede semplicemente a causa dell'accumulazione di errori stocastici. Se un gene diventa predominante e la popolazione è finita, tale gene si può propagare a tutti i membri della popolazione. Una volta che un gene converge in questa maniera, l'incrocio non può più introdurre un nuovo valore per quel gene. Di solito, comunque, l'informazione contenuta nella funzione di idoneità fornisce una pressione selettiva sufficiente a contrastare la deriva genetica. Il genetic drift può essere drasticamente ridotto introducendo la mutazione. Se però il coefficiente di mutazione è troppo elevato le informazioni fornite dalla funzione di idoneità vengono sfruttate meno efficacemente. La seconda e la terza ipotesi possono essere verificate su funzioni test che si comportano bene in laboratorio, ma sono difficili da soddisfare nei problemi reali.

4.11.6 Operatori dinamici

Dal momento che le prestazioni di un AG dipendono fortemente dal bilanciamento *esplorazione-sfruttamento*, durante l'esecuzione dell'algoritmo le impostazioni ottime dei coefficienti di incrocio e mutazione possono variare. Davis ha provato una variazione lineare della probabilità della mutazione e dell'incrocio: mentre l'incrocio decresce durante l'esecuzione, la mutazione cresce. Booker invece utilizza un valore di incrocio variabile dinamicamente, che dipende dalla propagazione dell'idoneità. Quando la popolazione converge, il valore dell'incrocio si riduce per dare più opportunità alla mutazione di trovare nuove variazioni, questo ha un effetto simile alla tecnica lineare di Davis, ma con il vantaggio di essere adattativo.

Davis descrive anche una tecnica adattativa che si basa sul successo degli operatori nel generare discendenti con alta idoneità. Si dà un credito a ciascun operatore quando produce un buon individuo nella popolazione in base agli ultimi 50 accoppiamenti e in seguito, per ogni evento riproduttivo, un operatore viene selezionato in base al suo credito. Il valore del credito varia in maniera adattativa durante il corso dell'evoluzione: se un operatore perde molto credito probabilmente è meno efficace degli altri. In questo modo si può costruire un

algoritmo efficiente senza preoccuparsi troppo di trovare parametri ottimi degli operatori, dato che questi si adattano automaticamente durante l'esecuzione.

4.11.7 Nicchia e speciazione

Negli ecosistemi naturali le differenti specie evolvono per riempire ciascuno la propria nicchia ecologica. La speciazione è il processo nel quale una singola specie si differenzia in due o più specie che occupano differenti nicchie. Negli AG le nicchie sono i massimi della funzione fitness. Alcune volte abbiamo funzioni multimodali di cui si vogliono identificare, non solo il picco più alto, ma un certo numero di picchi (massimi locali). In un AG tradizionale la popolazione tende a convergere verso un unico massimo. Alcune modifiche ai tradizionali AG sono state introdotte per ovviare a questo problema, una di queste è la competizione figli-genitori: se un figlio ha fitness maggiore del genitore, lo rimpiazza. Questo metodo aiuta a mantenere la diversità in quanto le stringhe tendono a rimpiazzare altre che sono simili a loro, e così si previene la convergenza verso un singolo massimo. Un altro approccio è quello che abbassa l'idoneità di individui tutti appartenenti alla stessa nicchia, lasciando inalterata quella dell'individuo con fitness maggiore, in questo modo si ripartisce la fitness degli individui molto simili *(niching* o *sharing)*. E' la tecnica implementata anche nel nostro AG.

4.12 Confronto con altre tecniche

Per i problemi di ottimizzazione esistono molte tecniche di approccio generale. Descriviamone alcune.

4.12.1 Ricerca casuale

L'approccio con la forza bruta per funzioni complicate è la ricerca casuale o enumerata. I punti dello spazio di ricerca sono scelti a caso o in qualche maniera sistematica, ed il loro valore viene calcolato. E' un metodo molto rozzo e di solito viene evitato.

4.12.2 Metodo del gradiente

Sono diversi i metodi che si basano sull'uso delle informazioni sul gradiente della funzione per guidare la direzione della ricerca. Se però la derivata della funzione non può essere calcolata perché la funzione è discontinua, tali metodi non possono evidentemente essere impiegati. Questi metodi sono generalmente detti *hillclimbing*, funzionano bene con funzioni che hanno un solo picco (unimodali), ma per funzioni che presentano numerosi estremi locali (multimodali) hanno il grave inconveniente di fermarsi in un massimo locale.

4.12.3 Ricerca iterata (Stocastic Hillclimber)

Combinando il metodo della ricerca casuale con quello del gradiente si ottiene il metodo cosiddetto della "ricerca iterata". Una volta che un picco è stato trovato, la scalata inizia nuovamente da un altro punto scelto a caso. Questa tecnica ha il vantaggio della semplicità e dà buoni risultati con funzioni che non abbiano molti estremi locali. Comunque, poiché ogni prova è fatta isolatamente, non si ottiene una figura complessiva della forma del dominio. Mentre la ricerca casuale progredisce, si continuano ad allocare lo stesso numero di prove sia in regioni dove sono stati trovati alti valori di idoneità, sia in regioni con bassa idoneità. Un AG, invece, incomincia con una popolazione iniziale casuale, e assegna via via maggiori tentativi alle regioni con più alta idoneità.

4.12.4 Simulated Annealing (ricottura simulata)

Questa tecnica è stata inventata da Kirkpatrick nel 1982 ed è sostanzialmente una versione modificata dell'hillclimbing. Iniziando da un punto scelto a caso nel dominio, viene fatto un movimento casuale: se il movimento porta ad un punto più alto allora è accettato, se porta ad un valore più basso è accettato con probabilità $p(t)$, ove t è il tempo. All'inizio $p(t)$ è vicino al valore 1, poi gradualmente tende a zero. Inizialmente ogni movimento viene accettato, ma la temperatura si riduce e la probabilità di accettare un movimento negativo diminuisce.

Come la tecnica della ricerca casuale, il simulated annealing lavora solo con una soluzione candidata per volta e perciò non costruisce una figura complessiva dello spazio di ricerca e non vengono salvate le informazioni dai precedenti movimenti per guidarci verso la soluzione.

4.12.5I vantaggi degli AG

Il principale vantaggio degli algoritmi genetici per risolvere un problema di ottimo è il fatto che non è necessario conoscere un metodo risolutivo per il problema in questione: **è sufficiente saper riconoscere se una soluzione è migliore o peggiore di un'altra.**

Gli AG possono tranquillamente gestire problemi con 20-30 variabili, cosa difficilmente realizzabile con un algoritmo tradizionale.

Un ulteriore vantaggio è il fatto che gli AG non forniscono un'unica soluzione finale, ma una popolazione di soluzioni. Si pensi al caso in cui si debbano confrontare diverse configurazioni sulla base di caratteristiche difficilmente traducibili in termini matematici, come ad esempio la facilità di lavorazione, l'ergonomia, l'estetica, e così via. Inoltre disporre di un certo numero di soluzioni sub-ottime è molto utile qualora sia opportuno effettuare un'analisi di sensitività costi-benefici.

Ci sono poi campi di applicazione dove i tradizionali metodi deterministici non possono essere applicati, in particolare nelle ottimizzazioni di problemi accoppiati ove vi sia coesistenza di variabili discrete e continue. Si consideri poi che i metodi tradizionali gradientali non sono applicabili a funzioni irregolari come funzioni a scalino, funzioni non derivabili su tutto il dominio, funzioni rumorose, ecc.

In generale si può dire che un AG è in grado di trovare soluzioni generalmente molto buone a quei problemi le cui variabili non potrebbero essere esplorate integralmente da un metodo tradizionale.

La tabella riassume il confronto tra gli AG e l'approccio ad hoc. Il maggior inconveniente di un AG è rappresentato dai tempi di calcolo, anche se, a parziale compensazione di ciò, gli algoritmi genetici si prestano molto bene al calcolo

parallelo. In caso di simulazioni numeriche particolarmente onerose la tendenza attuale è quella di accoppiarli con le Reti Neurali.

	APPROCCIO AD HOC	***AG***
Velocità	*dipende dal problema, generalmente buona*	*medio bassa*
Prestazioni	*dipende dal problema*	*molto buone*
Comprensione del problema	*necessaria*	*non necessaria*
Tempo uomo necessario	*dipende dal problema*	*pochi giorni*
Applicabilità	*bassa*	*totale*
Step intermedi	*non sono soluzioni (occorre aspettare sino alla fine del calcolo)*	*sono soluzioni (il calcolo può essere fermato in ogni momento)*

Tab 4.1 *Confronto tra gli algoritmi genetici e tecniche ad hoc.*

4.12.6 Applicazioni degli AG

Per illustrare la flessibilità degli AG si elencano alcune loro applicazioni.

- Ottimizzazioni nel campo dell'aerodinamica;
- Sintesi di meccanismi;
- Ottimizzazione combinatoria (Problema del commesso viaggiatore);
- Bin packing: disposizione di oggetti o forme su uno spazio limitato (taglio lamiere, cuoio, ecc.);
- Job scheduling: allocazione di risorse umane;
- Image processing;
- Intelligenza artificiale (sistemi di apprendimento);

Infine, tra le numerose applicazioni, una delle più curiose è lo sviluppo di trading systems (sistemi per l'investimento in borsa); in questo caso l'AG viene utilizzato per selezionare le migliori tecniche di investimento facendole sperimentare da migliaia di operatori di borsa virtuali, nel corso di successive generazioni.

5

L’ottimizzazione con gli AG

5.1 Aspetti generali riguardanti l’ottimizzazione

Un’ottimizzazione si rende necessaria ogniqualvolta sia da determinare un *design ottimo.*

Il design ottimo è quello che risponde al valore ottimale di uno dei requisiti stabiliti nelle specifiche di progetto, quali, per esempio, peso, sezione, volume, costo, o qualsiasi altro aspetto predominante ai fini del progetto.

E’ importante sottolineare che non esiste un progetto ottimo in senso assoluto, poichè il raggiungimento contemporaneo di tutti gli obiettivi a cui il progetto dovrebbe tendere, non è, in generale possibile: si avrà un progetto ottimo dal punto di vista del peso, un altro dal punto di vista degli ingombri, e così via.

5.2 Terminologia

Prima di descrivere l’algoritmo di ottimizzazione si forniscono alcune definizioni riguardanti la terminologia di uso generale nei problemi di ottimizzazione.

- **Progetto (design)**: configurazione di una struttura, o di una sua

parte, definita da un insieme di variabili di progetto.

- **Progetto ottimo (optimum design)**:particolare configurazione che comporta il massimo miglioramento di un determinato aspetto, ad esempio il costo, senza compromettere la funzionalità del prodotto.
- **Variabili di progetto (design variables)**: variabili indipendenti x_i che definiscono la geometria (configurazione) di un dato prodotto o sistema, generalmente definite da un estremo superiore $\overline{x}_i$ ed uno inferiore $\underline{x}_i$. Sono spesso grandezze geometriche (quote, spessori, raggi di raccordo, ecc.) che definiscono il progetto.
- **Variabili di stato (state variables)**: sono variabili dipendenti, funzioni delle variabili di progetto, che rappresentano una caratteristica del sistema (ad esempio il peso) e che restringono il campo delle possibili soluzioni ottime. Le variabili di stato costituiscono quindi i vincoli imposti al progetto, sono anch'esse limitate in generale tra un estremo inferiore $\underline{g}_j$ ed uno superiore $\overline{g}_j$.
- **Vincoli (costraints)**: limitazioni imposte alle variabili di stato.
- **Funzione obiettivo (objective function)**: variabile dipendente che rappresenta il particolare aspetto che si intende migliorare (ottimizzare).
- **Progetto accettabile (feasible design)**: configurazione definita da un set di variabili di progetto che soddisfa tutti i vincoli, sia quelli imposti sulle variabili di progetto, sia quelli relativi alle variabili di stato.
- **Progetto inaccettabile (infeasible design)**: configurazione che non rispetta uno o più vincoli.

5.3 Il nostro Algoritmo Genetico

L'algoritmo genetico utilizzato in questa tesi è stato scaricato da un archivio di programmi relativi alla computazione evolutiva presente su internet [ ]. Il codice è stato scritto in Fortran 77 da David Carroll, ricercatore della University of Illinois, in collaborazione con D. Goldberg, una delle massime autorità in materia di algoritmi genetici. Questo fatto mi è sembrato una buona garanzia circa la validità del codice.

Il listato del programma, riportato in appendice, viene fornito gratuitamente a chiunque ne faccia richiesta per utilizzo senza fini di lucro. Oltre al listato del codice, contenuto nel file ga170.f, vengono forniti altri 5 file:

ga.inp	E' il file che contiene tutte le impostazioni necessarie per il funzionamento l'algoritmo genetico
ga.out	E' il file sul quale vengono scritti gli output dell'AG, generazione per generazione.
ga2.inp	Contiene le impostazioni originarie dell'algoritmo (non modificarlo mai)
ga.restart	Su questo file vengono scritte le sequenze binarie che rappresentano gli individui dell'ultima generazione salvata. Serve per far ripartire il calcolo da quel punto.
param.f	Contiene i valori che definiscono la dimensione massima della popolazione, la lunghezza massima del cromosoma associato ad ogni individuo ed il numero massimo di variabili che l'AG accetta.

5.4 Il file di input

Il file ga.inp è il file contenente tutte le impostazioni che l'algoritmo genetico necessita per funzionare, è il vero cuore di tutto il sistema.

Vediamo in dettaglio tutti i parametri tramite i quali l'utente può "personalizzare" l'algoritmo.

irestrt	=0 inizia un nuovo run di calcolo
	=1 riprende da dove era arrivato (legge dal file ga.restart)
microga	=0 opzione 'Micro Genetic Algorithm' disattivata
	=1 attivata
npopsiz	dimensione della popolazione
nparam	numero di variabili di progetto
pmutate	probabilitá assegnata all'operatore Mutazione
maxgen	numero di generazioni
	N.B. il tempo di calcolo e' proporzionale al numero di valutazioni ove
	numero di valutazioni = npopsiz·maxgen
idum	seme di innesco per il generatore di numeri casuali (può assumere valori da -1000 a -20000)
pcross	Probabilitá assegnata all'operatore Crossover
itourny	selezione "a Torneo" (unico tipo di selezione che questo algoritmo genetico rende disponibile)
ielite	=1 opzione "elitarismo"attivata =0 opzione "elitarismo" disattivata (Elitarismo = il migliore individuo viene replicato nella generazione successiva)
icreep	=1 opzione "creep mutation" attivata
	=0 opzione "creep mutation" disattivata
pcreep	probabilita' assegnata all'operatore "creep mutation"
iunifrm	=1 opzione "uniform crossover" attivata
	=0 opzione "uniform crossover" disattivata
iniche	=1 opzione "niching" attivata

	=0 opzione "niching" disattivata
nchild	numero di figli per coppia di genitori (le possibilita' sono: 1 oppure 2)
iskip, iend	parametri solo a scopo di debug, devono essere ignorati
nowrite	=1 evita output dettagliato riguardante i parametri come mutazione, ecc. =0 scrive output dettagliato
kountmx	numero che determina ogni quante generazioni viene riscritto il file di restart
parmin	Array che definisce il limite inferiore delle variabili di progetto
parmax	Array che definisce il limite superiore delle variabili di progetto
nposibl	Array di valori che determinano la minima variazione associata ad ogni variabile di progetto, deve essere una potenza di due. Tale potenza rappresenta anche il numero di bit assegnati a ciascuna variabile di progetto nella codifica binaria. Es.: nposibl=1024,1024,128, significa che le prime due variabili di progetto possono subire (durante le iterazioni) una variazione minima pari a 1/1024 del loro range e ad esse vengono associate due stringhe di 10 bit (2^{10}= 1024), mentre la terza variabile puo' avere una variazione minima di 1/128 del range ed e' rappresentata da una stringa di 7 bit (2^7= 128). Il cromosoma che definisce la configurazione del progetto sara' quindi una stringa di 27 bit. (10+10+7).
nichflg	Array che definisce la possibilitá di "niching" relativamente ad ogni variabile di progetto. Es.: 1,1,0, significa Niching per le prime due variabili, ma non per la terza.

Questa implementazione di AG può utilizzare due differenti operatori di mutazione: la *jump mutation* e la *creep mutation*. La differenza tra questi due

operatori non era molto chiara per cui è stato chiesto un chiarimento direttamente all'autore del codice:

jump mutation: permette una variazione qualunque del valore delle variabili di progetto;

creep mutation: permette una variazione di un solo step (in più o in meno) delle variabili di progetto;

Il *Niching* è una procedura con la quale si riduce la fitness degli individui (configurazioni) troppo vicini all'individuo migliore. Questa procedura vuole evitare che la popolazione converga troppo precocemente ad un massimo locale, si parla in questo caso di *deriva genetica (genetic drift)*. Il nostro codice riduce l'idoneità in base al calcolo della distanza nello spazio multidimensionale delle variabili di progetto tramite una funzione definita da Goldberg che normalizza le distanze tra i parametri.

Quando l'opzione *Micro Genetic Algorithm* é attivata il codice imposta automaticamente alcuni dei parametri del file ga.inp indipendentemente dal'impostazione che compare nel file stesso, piú precisamente vengono poste uguali a zero le probabilitá di mutazione (sia la jump sia la creep mutation), viene disattivata l'opzione niching e viene attivato l'elitarismo.

E' importante osservare che l'intervallo di variazione delle variabili di progetto è discretizzato (agendo sul parametro nposibl), questo significa che l'algoritmo genetico non 'vede' la superficie di risposta vera, bensì un suo campionamento. Proprio per questo fatto è sempre opportuno far seguire all'AG un'analisi gradientale a partire dall'ottimo trovato dall'algoritmo genetico. L'AG infatti non sfrutta le informazioni fornite dalle derivate, e quindi non può sapere se c'è margine per un ulteriore miglioramento. L'utilizzo di un metodo gradientale di affinamento dell'AG fa parte delle tecniche cosiddette di *ibridizzazione* degli AG. A questo proposito è importante comprendere che un valore di nposibl troppo basso determina una perdita di informazioni che può compromettere le prestazioni dell'algoritmo genetico. Infatti, in caso di superficie di risposta particolarmente

irregolare, un campionamento troppo poco fitto può portare a perdere dei picchi che neppure una successiva analisi gradientale può recuperare. Di contro un valore eccessivo, associato ad un elevato numero di variabili, può causare un rallentamento della convergenza dovuto alla aumentata lunghezza del cromosoma.

Il codice, così come viene fornito, accetta una popolazione massima di 200 individui, cromosomi di lunghezza massima pari a 30 bit ed un massimo di 8 variabili. Questi valori possono essere modificati nel file param.f, contestualmente occorrerà modificare alcune istruzioni write ed alcuni format, in particolare, se si modifica nchrome e/o nparam sarà necessario modificare i format con le etichette 1050, 1075, 1275, e 1500. Infine, ogni volta che si modifica il parametro nposibl bisogna modificare il format numero 1075.

L'autore fornisce il codice con una funzione test per il calcolo della fitness degli individui, si tratta della funzione multimodale utilizzata da Goldberg e Richardson durante le loro ricerche sugli AG. Questa funzione ha un numero pari ad nvalley di massimi decrescenti ma un solo massimo globale. Nell'implementazione fornita ad nvalley è assegnato il valore 6.

Le impostazioni suggerite per il calcolo del massimo di questa funzione test sono mostrate in tabella 5.1.

L'algoritmo genetico raggiunge il massimo assoluto in 150 generazioni con una popolazione di 5 individui, per un totale di 150 x 5 = 750 valutazioni di funzione.
Le figure 5.1 e 5.2 seguenti rappresentano l'andamento della fitness del miglior individuo e della fitness media della popolazione.

irestrt	0
microga	1
npopsiz	5
nparam	2
pmutate	0.05
maxgen	150
idum	-1000
pcross	0.5
itourny	1
ielite	1
icreep	1
pcreep	0.02
iunifrm	1
iniche	1
nchild	1
iskip, iend	0,0
nowrite	1
kountmx	5
parmin	-1.0, -1.0
parmax	1.0,1.0
nposibl	32768, 32768
nichflg	8*1

Tab 5.1 *Impostazioni per il calcolo del massimo della funzione test.*

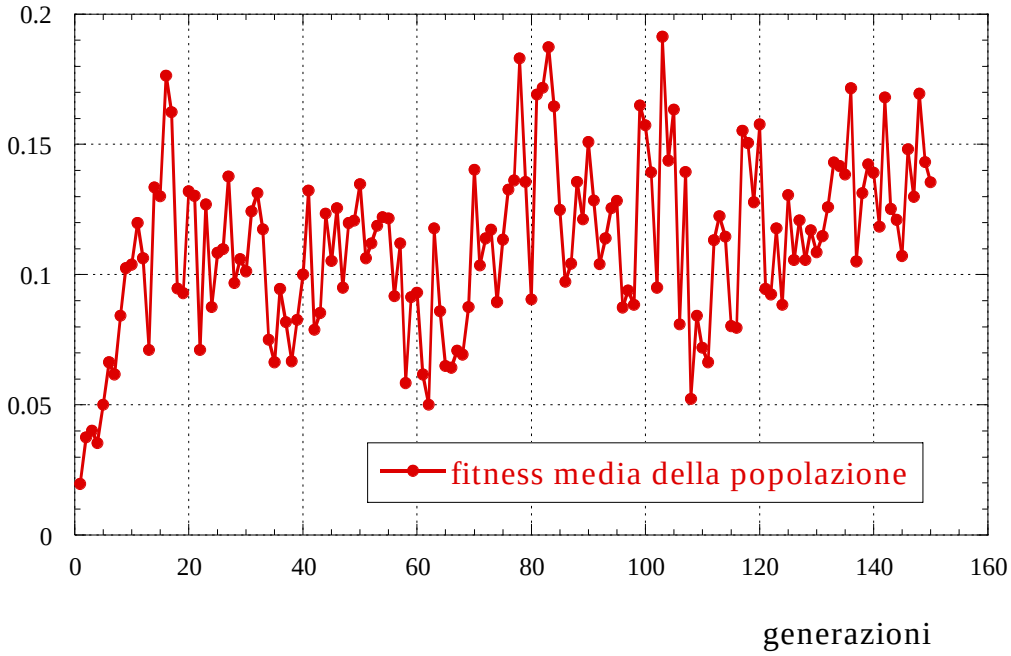


Fig 5.1 *Fitness media della popolazione. In ordinate é mostrato il valore analitico della funzione test.*

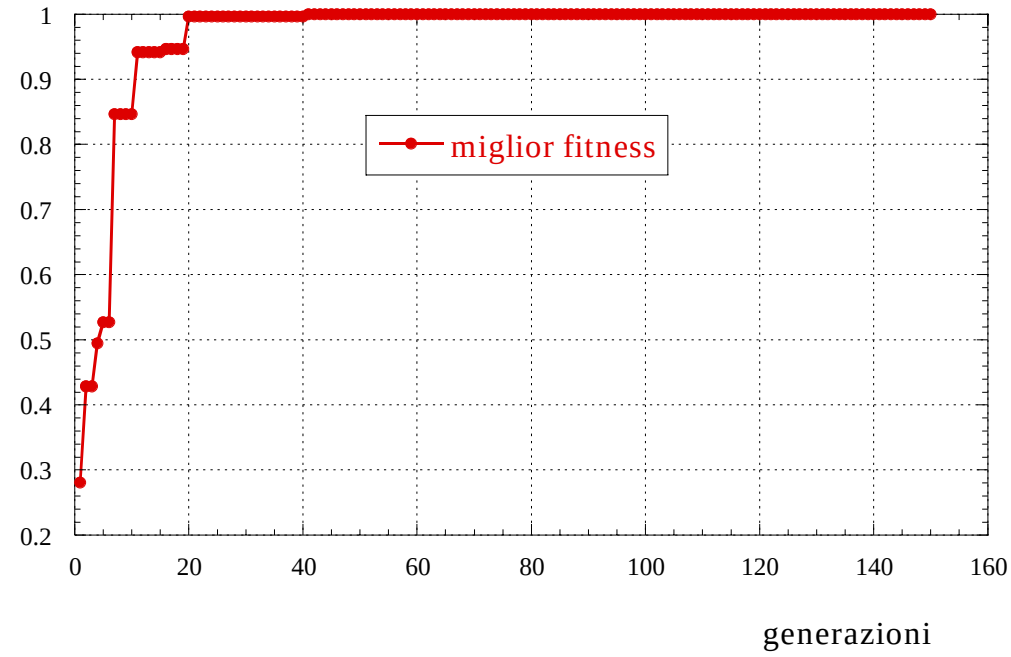


Fig 5.2 *Miglior fitness. In ordinate é mostrato il valore analitico della funzione test.*

Ho provato numerose combinazioni dei parametri di input, in particolare la dimensione della popolazione ed i coefficienti di incrocio (pcross) e mutazione (pmutate). I risultati ottenuti su questa funzione di prova non ci permettono di dire quali siano le impostazioni migliori dei parametri. Inoltre non è possibile fare delle prove facendo variare un solo parametro alla volta, infatti, come è stato detto nel capitolo precedente, tali parametri sono correlati e quindi non possono essere ottimizzati singolarmente. A questo si aggiunga che le impostazioni che si sono rivelate efficaci nella risoluzione di un particolare problema non necessariamente si rivelano tali quando vengono applicate ad un problema differente. Non esiste, al momento, una linea guida per la scelta dei parametri di un AG.

Non accontentandoci dell'efficacia dimostrata dall'AG nell'ottimizzazione della funzione fornita a corredo, lo abbiamo applicato alla ricerca dell'estremo di una particolare funzione, la funzione di Ackley. Si tratta di una funzione multimodale appositamente costruita per verificare l'efficacia degli algoritmi di ricerca degli estremi, ottenuta modulando un esponenziale con seni e coseni. Analiticamente è rappresentata dalla seguente espressione:

$$f = -c_1 \cdot \exp\left(-c_2 \sqrt{\frac{1}{2}\sum_{j=1}^{2} x_j^2} \right) - \exp\left(\frac{1}{2}\sum_{j=1}^{2} \cos(c_3 \cdot x_j) \right) + c_1 + e \qquad (5.1)$$

Dove

c_1=20

c_2=0.2

c_3=2π

e=2.7128

La sua particolare topologia (fig 5.3) ci permette di affermare che rappresenta un buon banco di prova per il nostro algoritmo genetico. Un qualsiasi algoritmo

gradientale cadrebbe inevitabilmente in trappola fermandosi al primo estremo locale. Dal momento che il nostro codice è scritto per la ricerca del massimo, abbiamo ribaltato la funzione di Ackley (fig 5.4). Le prestazioni del nostro AG sono state ottime, riuscendo a trovare il massimo assoluto in 173 generazioni, per

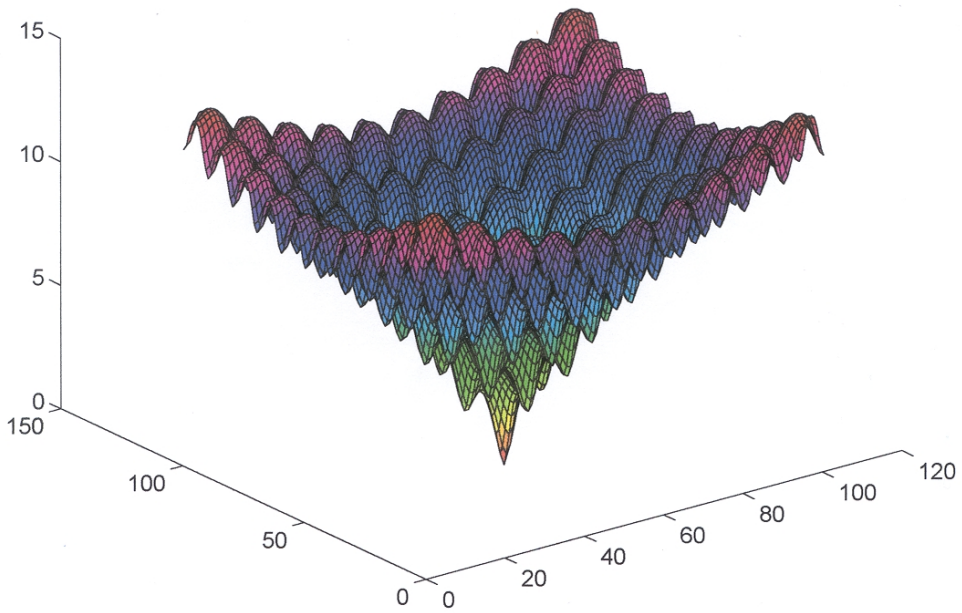


Fig 5.3 *La funzione di Ackley.*

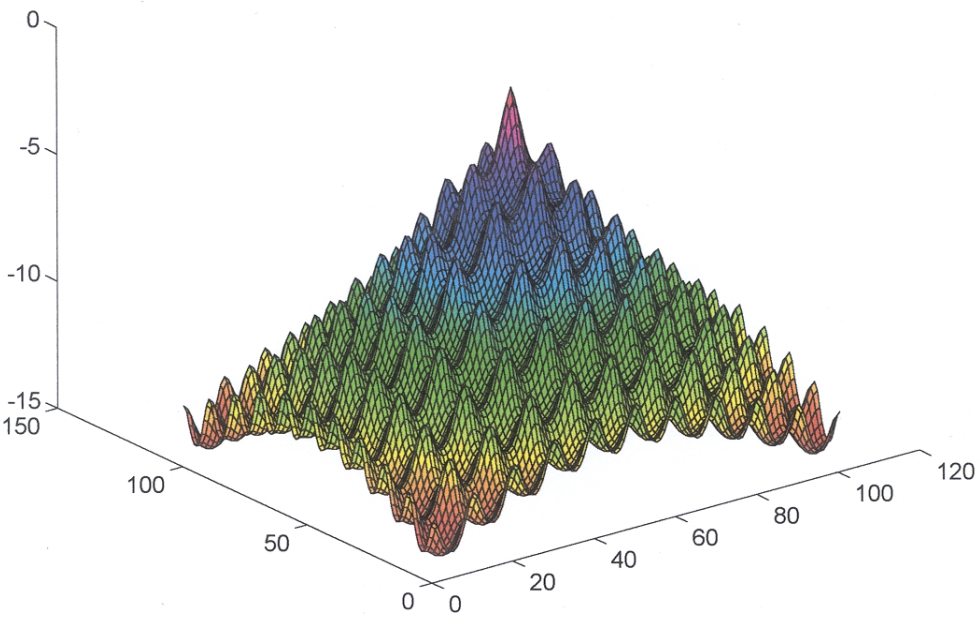


Fig 5.4 *La funzione di Ackley ribaltata.*

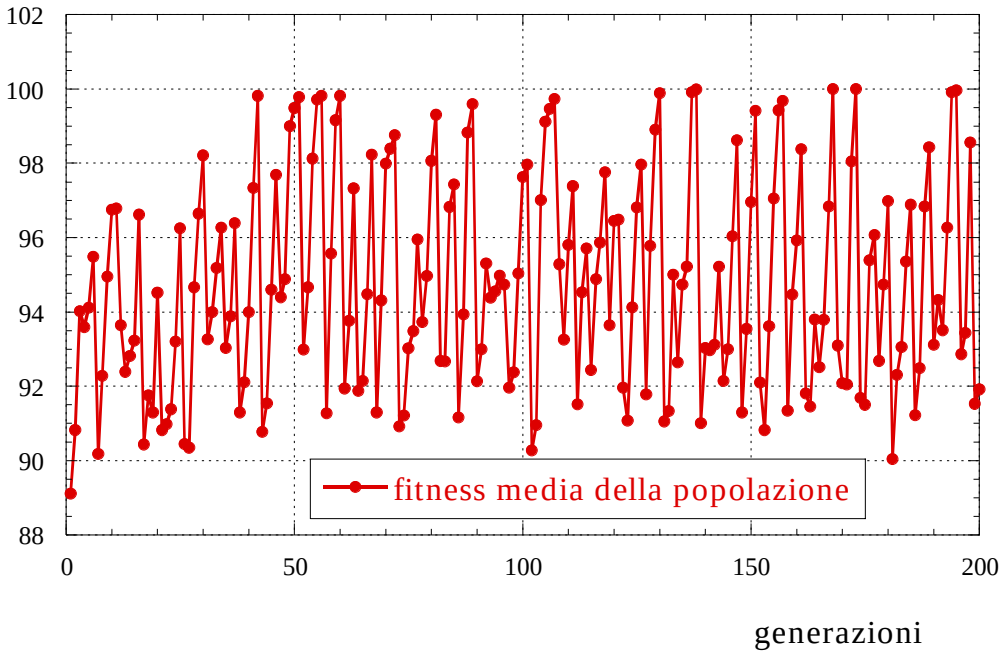


Fig 5.5 *Fitness media della popolazione. In ordinate é mostrato il valore analitico della funzione di Ackley.*

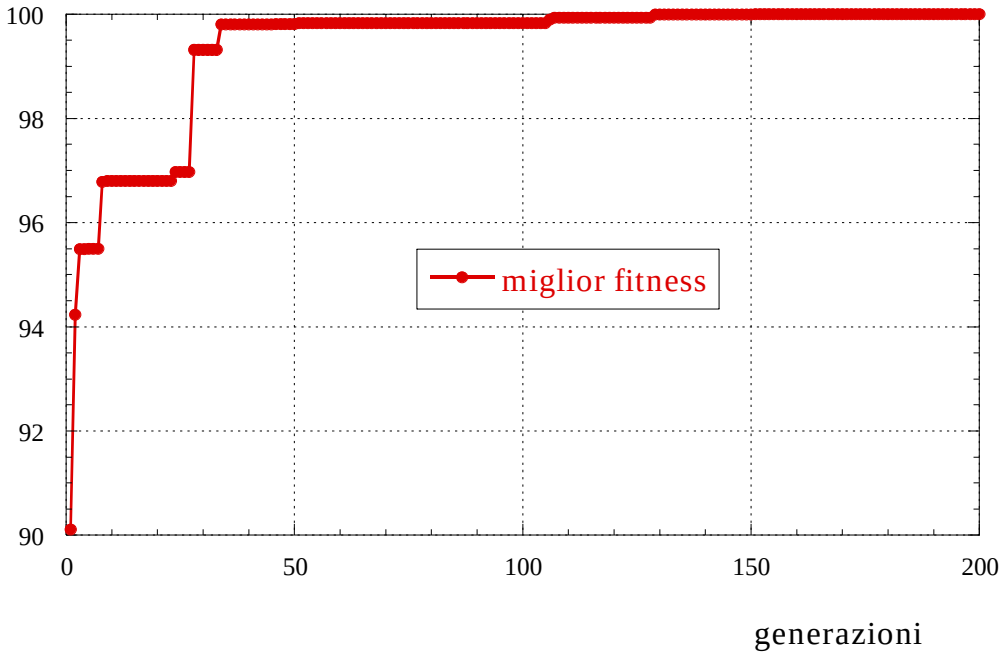


Fig 5.6 *Miglior fitness. In ordinate é mostrato il valore analitico della funzione di Ackley.*

un totale di 865 valutazioni di funzione. Le figure 5.5 e 5.6 mostrano l’andamento della Fitness rispettivamente della popolazione e del miglior individuo.

E’ buona norma far girare l’algoritmo variando ogni volta il seme di innesco dei numeri casuali, al fine di esplorare lo spazio delle variabili di progetto in maniera ogni volta differente. Inoltre, dal momento che durante l’esecuzione intervengono delle generazioni di numeri casuali che influiscono sulle generazioni successive, può succedere che, anche partendo con lo stesso seme di innesco e le medesime impostazioni, si raggiungano risultati differenti in differenti lanci. Anche utilizzando la funzione di Ackley sono stati fatti diversi lanci con differenti set di impostazioni. L’unico risultato certo è il seguente: quando applichiamo l’AG a funzioni matematiche un’impostazione particolarmente efficace è:

microga = 1
npopsiz = 5

5.5 I vincoli in un AG

Come si è detto nel precedente capitolo, la codifica del problema e la funzione con cui rappresentare l’idoneità di un individuo, sono i fattori fondamentali per il successo di un AG. Alcuni ricercatori sostengono che i meccanismi di base degli algoritmi genetici sono così robusti che, entro margini abbastanza grandi, la scelta dei parametri ottimali non è così critica. Quello che è critico nel comportamento di un AG sono proprio la funzione fitness e la codifica. Idealmente vorremmo che la funzione fitness fosse piatta e regolare, purtroppo per la maggior parte dei problemi non è possibile costruirla con questo andamento (e se fosse possibile sarebbe più conveniente usare un metodo gradientale). La regola generale per costruire la funzione fitness è che essa rifletta il valore reale del cromosoma. Nel caso di funzioni matematiche, la fitness è banalmente il valore stesso della funzione. Nel caso di ottimizzazioni di funzioni matematiche non si pone uno dei principali problemi che bisogna affrontare in sede di utilizzo di un AG per

problemi ingegneristici complessi, vale a dire la presenza dei vincoli sulle variabili dipendenti (variabili di stato). Mentre i vincoli sulle variabili di progetto sono automaticamente rispettati dal codice in sede di generazione delle singole configurazioni (in quanto le configurazioni che non rispettano tali restrizioni sono impedite), quelli sulle variabili di stato rappresentano un aspetto cruciale. Quello che si fa quando si utilizza un AG è rappresentare tali vincoli tramite funzioni di penalità. Una buona funzione di penalità può essere costruita a partire dal cosiddetto *costo di completamento stimato*, vale a dire il costo necessario (stimato) per far diventare valida una configurazione che non lo è.

5.6 I vincoli utilizzati

In questo studio abbiamo dovuto affrontare vincoli su due differenti variabili di stato:

- Campo magnetico massimo;
- Campo magnetico sul percorso del fascio.

Il primo vincolo si rende necessario per evitare che i conduttori subiscano una transizione dallo stato superconduttore a quello normale, infatti le condizioni di supercondutivitá dipendono da due fattori: temperatura e campo magnetico. La prima condizione viene garantita dal sistema di raffreddamento del magnete, per quanto riguarda invece la seconda, bisogna evitare che i conduttori si trovino immersi in un campo magnetico di eccessiva intensitá. Nelle specifiche di progetto tale limite é stato fissato in 6 T, tale valore tiene giá conto di un opportuno coefficiente di sicurezza.

Il secondo vincolo é invece dovuto alla curvatura che deve essere impressa al fascio, legata al campo dall'equazione 2.1, tenendo conto delle altre variabili in gioco, il valore del campo lungo la traiettoria del fascio é stato posto uguale a 4 T.

Per tener conto di tali vincoli abbiamo fatto ricorso ad una funzione di penalitá consistente in un semplice fattore moltiplicativo proporzionale al superamento del limite.

5.7 Le variabili di progetto

L'utilizzo dell'algoritmo genetico ci ha consentito di considerare un numero di variabili di progetto che con un tradizionale algoritmo di ottimizzazione non avremmo potuto considerare. Ho così deciso di prendere in considerazione tutte le densità di corrente (6 variabili) e tutti i parametri che definiscono la posizione delle bobine (10 variabili), per un totale di 16 variabili di progetto. Le dimensioni della sezione dei singoli conduttori saranno oggetto di una successiva ottimizzazione.

La tabella elenca le variabili di progetto considerate per l'ottimizzazione.

1	Curr1	Densitá di corrente bobina n. 1
2	Curr2	Densitá di corrente bobina n. 2
3	Curr3	Densitá di corrente bobina n. 3
4	Curr4	Densitá di corrente bobina n. 4
5	Curr5	Densitá di corrente bobina n. 5
6	Curr6	Densitá di corrente bobina n. 6
7	Coox1	Distanza della bobina n. 1 dall'asse y
8	Coox2	Distanza della bobina n. 2 dall'asse y
9	Coox3	Distanza della bobina n. 3 dall'asse y
10	Coox4	Distanza della bobina n. 4 dall'asse y
11	Coox5	Distanza della bobina n. 5 dall'asse y
12	Coox6	Distanza della bobina n. 6 dall'asse y
13	Cooy3	Distanza della bobina n. 3 dall'asse x
14	Cooy4	Distanza della bobina n. 4 dall'asse x
15	Cooy5	Distanza della bobina n. 5 dall'asse x
16	Cooy6	Distanza della bobina n. 6 dall'asse x

Tab 5.2 *Variabili di progetto considerate per l'ottimizzazione.*

Per il lancio dell'algoritmo si é scelta una popolazione di 50 individui, una probabilitá di crossover pari a 0.5, probabilitá di mutazione 0.05 e 100 generazioni. L'andamento della fitness media della popolazione e del miglior individuo (miglior configurazione del magnete) sono mostrate dalle figure 5.7 e 5.8.

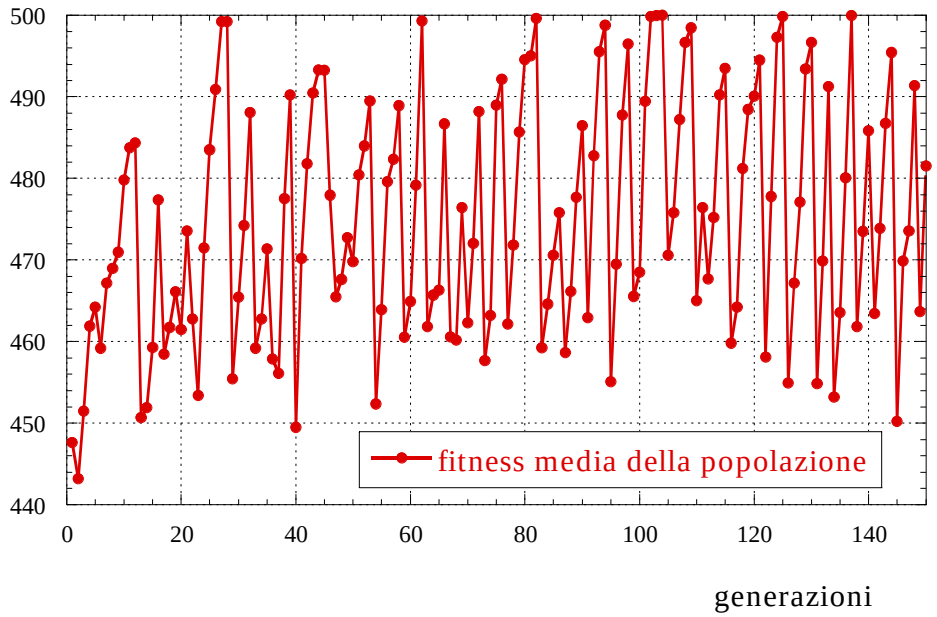


Fig 5.7 *Fitness media della popolazione.*

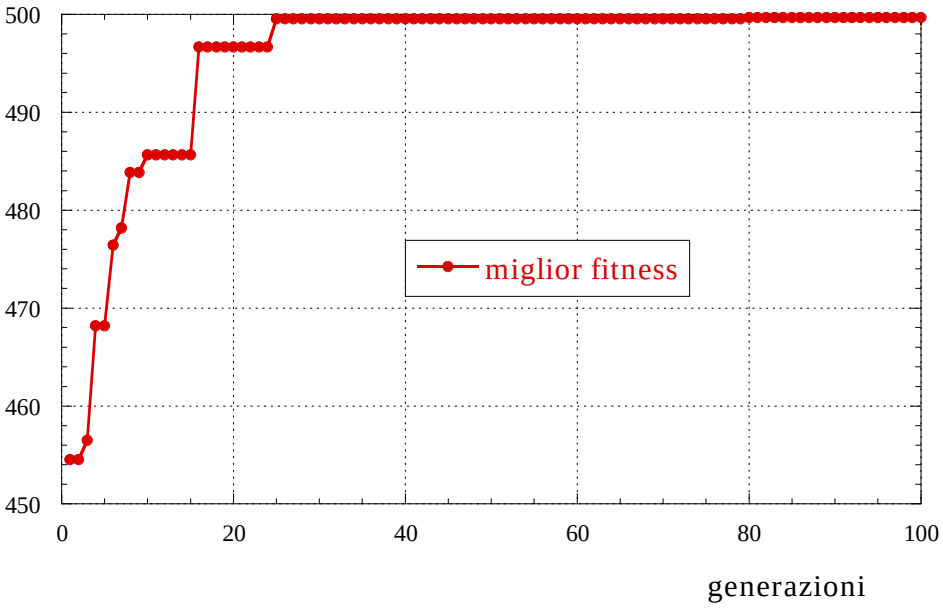


Fig 5.8 *Miglior fitness.*

5.8 Risultati dell'ottimizzazione

La fitness della miglior configurazione é giunta a convergenza entro le 100 generazioni assegnate, come visibile dalla figura 5.8.

I valori numerici delle variabili di progetto corrispondenti alla configurazione migliore sono riassunti nella tabella 5.3 (per la numerazione si faccia riferimento alla figura 3.3). L'uniformitá di campo magnetico raggiunta con questa configurazione é pari al 2.5‰.

1	Curr1	$0.947 \cdot 10^8$ A / m^2
2	Curr2	$1.336 \cdot 10^8$ A / m^2
3	Curr3	$2.459 \cdot 10^8$ A / m^2
4	Curr4	$2.717 \cdot 10^8$ A / m^2
5	Curr5	$2.419 \cdot 10^8$ A / m^2
6	Curr6	$2.223 \cdot 10^8$ A / m^2
7	Coox1	0.218 m
8	Coox2	0.428 m
9	Coox3	0.247 m
10	Coox4	0.214 m
11	Coox5	0.172 m
12	Coox6	0.115 m
13	Cooy3	0.082 m
14	Cooy4	0.148 m
15	Cooy5	0.187 m
16	Cooy6	0.247 m

Tab 5.3 *Valori numerici delle variabili di progetto corrispondenti alla configurazione ottimizzata.*

I corrispondenti valori delle variabili di stato risultano:

campo massimo : 5.8 T

campo sul percorso del fascio : 4 T

Tali valori numerici soddisfano appieno le specifiche, quindi le funzioni di penalitá introdotte sono risultate efficienti.

Come é stato detto nel paragrafo 5.4, l'algoritmo genetico non sfrutta le informazioni fornite dal gradiente, quindi la configurazione ottenuta potrá essere ulteriormente migliorata effettuando un'analisi gradientale. Questo affinamento verrá eseguito con la routine di ANSYS denominata *First Order*, e sará descritto nel capitolo 6.

6

L’ottimizzazione con ANSYS

6.1 Aspetti generali

La procedura di ottimizzazione è parte integrante del codice ANSYS e può venire utilizzata ogni qualvolta sia da determinare un *design ottimo.*

Per determinare un design ottimo, il codice produce una serie ciclica di analisi-valutazioni-modifiche. Questo significa che dapprima viene effettuata un’analisi del progetto di partenza, quindi sono valutati i risultati confrontandoli con specifici criteri di design e, di conseguenza, il progetto iniziale è modificato in direzione opportuna: il tutto ripetuto ciclicamente sino al raggiungimento di un progetto ottimo.

6.2 Impostazione matematica

In termini matematici un problema di ottimizzazione può essere definito come minimizzazione di una funzione

$$f = f(x_1, x_2, ..., x_3) \tag{6.1}$$

tenendo conto dei vincoli:

$$\underline{x}_i \le x_i \le \overline{x}_i \qquad \text{con } i = 1,2,.....n$$

$$\underline{g}_j \le g_j(x_1, x_2, ... , x_m) \le \overline{g}_j \quad \text{con } i = 1,2,.....m$$

ove:

- f = funzione obiettivo
- x_i = variabile di progetto i-esima
- n = numero variabili di progetto
- g_j = variabile di stato j-esima
- m = numero variabili di stato

Tali equazioni definiscono pertanto un problema di ricerca di minimo vincolato, vincolato in quanto viene limitato il campo di variabilità (di ammissibilità per i valori) delle variabili di progetto e delle variabili di stato.

Lo spazio di progetto si definisce mediante il vettore delle variabili di progetto:

$$\mathrm{X} = [x_1, x_2, \dots, x_n]$$

indicando un particolare set delle variabili di progetto come:

$$\mathrm{X}^* = [\, x_1^*, x_2^*, \dots, x_n^* \,]$$

Si può affermare che tale configurazione è accettabile solo se:

$$g_i^* = g_i(\mathrm{X}^*) \le \overline{g}_i + \alpha_i \qquad \text{con } i = 1,2,\dots,m_1$$

$$\underline{g}_i^{\,*} = h_i(\mathrm{X}^*) \ge \underline{h}_i - \beta_i \qquad \text{con } i = 1,2,\dots,m_2$$

$$\underline{w}_i - \gamma_i \le w_i^* = w_i(\mathrm{X}^*) \le \overline{w}_i + \gamma_i \qquad \text{con } i = 1,2,\dots,m_3$$

dove α_i , β_i e γ_i rappresentano le tolleranze rispettivamente sulle variabili di stato g_i , h_i , w_i .

La specificazione del numero di variabili di stato pari a tre è indicativa e ha il solo scopo di mostrare le tre possibili differenti modalità di limitazione.

In modo del tutto analogo è possibile introdurre delle tolleranze anche per gli estremi di variazione delle variabili di progetto.

Gli algoritmi di ottimizzazione di ANSYS convertono il problema vincolato in uno equivalente non vincolato (campo di ammissibilità per le variabili di stato e di progetto da $-\infty$ a $+\infty$) tramite opportune funzioni di penalità, in quanto i metodi di minimizzazione per tali problemi sono generalmente più efficaci di quelli relativi a funzioni vincolate.

6.3 Le routine di ANSYS

Il codice ANSYS mette a disposizione numerosi tecniche di ottimizzazione, che divide in *metodi* e *strumenti*. I metodi utilizzabili sono tre:

- **Subproblem** : è un metodo di ordine zero che utilizza una approssimazione ai minimi quadrati della funzione obiettivo e delle variabili di stato.
- **First Order** : è un metodo di tipo gradientale che esegue la ricerca del minimo sulla superficie di risposta reale.
- **User-supplied method** : viene utilizzato un algoritmo di ottimizzazione definito dall'utente.

Oltre a questi tre metodi ANSYS consente di utilizzare i seguenti strumenti:

- **Single Loop Run** : questo strumento esegue un solo ciclo e produce una sola analisi alla volta. Adatto per le analisi *what if*.
- **Random Design Generation** : il codice esegue più cicli, generando i set di variabili di progetto in modo del tutto casuale. E' uno strumento utile

per una vagliatura preliminare dell'intero spazio delle variabili di progetto da far precedere ad un successivo ciclo di ottimizzazione.

- **Sweep Generation** : questo strumento genera diversi set di variabili di progetto, a partire da un set di riferimento, spazzando l'intero campo di variazione di ogni variabile, mantenendo costante il valore delle restanti.
- **Factorial Evaluation** : questo è uno strumento statistico che vaglia le combinazioni delle variabili agli estremi dei loro intervalli di variazione, secondo la teoria dei progetti fattoriali (DOE) di ordine 2, generando 2^k set in corrispondenza dei quali calcolare la funzione obiettivo. La funzione principale di questo strumento è quella di valutare la significatività delle singole variabili di progetto calcolando gli *effetti principali* e le *interazioni a due* e *a tre* secondo la teoria dell'analisi della varianza. Possono essere eseguiti progetti fattoriali completi o frazionati.
- **Gradient Evaluation** : in corrispondenza di una determinata configurazione del sistema il codice calcola la derivata della funzione obiettivo e delle variabili di stato rispetto a tutte le variabili di progetto. E' uno strumento utile per analisi di sensitività nell'intorno dell'ottimo.
- **User-supplied Design Tool** : un algoritmo fornito dall'utente che bypassa la logica di ANSYS.

E' importante ricordare che i *metodi* e gli *strumenti* appena descritti possono essere ancora più efficaci se utilizzati in cascata.

6.3.1 METODO DEL SUBPROBLEM

Questo metodo può essere definito come un metodo di ordine zero per il fatto che esso richiede la valutazione del solo valore delle variabili, dipendenti e indipendenti, ma non delle loro derivate.

Il metodo del Subproblem esegue un campionamento dello spazio delle variabili di progetto sino alla determinazione di un numero di set (pari al numero

delle variabili di progetto più due) sufficiente a determinare una approssimazione della superficie di risposta mediante il metodo dei minimi quadrati.

Determinata la forma quadratica che adatta la superficie vera, il problema di minimo vincolato viene convertito in un problema non vincolato introducendo delle funzioni di penalità.

Ad ogni iterazione si minimizza la funzione approssimante penalizzata sino alla convergenza del metodo.

6.3.1.1 Approssimazione delle variabili dipendenti

Il primo passo è rappresentare ogni variabile dipendente con una approssimazione, indicata di seguito con la sopra scrittura " ^ ".

Per la funzione obiettivo:

$$\hat{f}(x) = f(x) + errore \tag{6.2}$$

In modo simile per le variabili di stato:

$$\hat{g}(x) = g(x) + errore$$
$$\hat{h}(x) = h(x) + errore$$
$$\hat{w}(x) = w(x) + errore$$

La forma utilizzata da ANSYS per l'approssimazione è rappresentata da una forma quadratica completa con termini misti. Riferendoci, per esempio, alla funzione obiettivo f, tale forma risulta del tipo:

$$\hat{f} = a_0 + \sum_{i=1}^{n} a_i \cdot x_i + \sum_{i=1}^{n} \sum_{j=1}^{n} b_{ij} \cdot x_i \cdot y_j \tag{6.3}$$

L'utente può scegliere il tipo di approssimazione ai minimi quadrati, sia delle funzioni obiettivo sia delle variabili di stato, le diverse possibilità sono:

- lineare
- quadratica
- quadratica con termini misti

Il programma adotta una scelta per difetto, ma tale modo di procedere può, in taluni casi, risultare sconveniente. Per default, l'approssimazione della funzione obiettivo è quadratica con termini misti, mentre è quadratica senza termini misti per le variabili di stato. I coefficienti a_i e b_{ij} che compaiono nell'equazione (6.3) sono calcolati mediante una tecnica ai minimi quadrati pesati minimizzando lo scarto quadratico medio.

La norma dell'errore fornito dai minimi quadrati pesati, per la funzione obiettivo, assume la forma:

$$E^2 = \sum_{i=1}^{n} \phi^{(j)} \cdot (f^{(j)} - \hat{f}^{(j)})^2 \tag{6.4}$$

Ove:

$\phi^{(i)}$ = peso associato alla configurazione i-esima

n = numero dei cicli di ottimizzazione effettuati

simili norme E^2 vengono calcolate per ognuna delle funzioni di stato definite.

I pesi ϕ, attribuiti all'errore su $\hat{f}$, sono calcolati in uno dei seguenti modi:

- secondo la distanza nello spazio delle variabili di progetto: i progetti più vicini come differenza tra variabili di progetto ricevono pesatura più alta.
- secondo il valore della funzione obiettivo

- secondo l'accettabilità o meno del progetto
- mediante una combinazione dei tre modi descritti sopra
- tutti i pesi sono unitari: $\phi^{(j)} = 1$ per ogni j

L'utente può specificare il tipo di approssimazione (lineare, quadratica, quadratica con termini misti) e il criterio con il quale calcolare i pesi attraverso il comando **OPEQN**.

Affinchè possa venire costruita una approssimazione è necessario che esista già un certo numero di configurazioni in corrispondenza delle quali calcolare la funzione obiettivo e le variabili di stato, questo numero è pari al numero delle variabili di progetto più due.

6.3.1.2 Minimizzazione dell'approssimazione del Subproblem

Una volta definita la funzione approssimante, il problema di minimo vincolato è esprimibile come minimizzazione della funzione

$$\hat{f} = \hat{f}(x)$$

soggetta agli stessi vincoli a cui è soggetta la funzione approssimata.

Il passo successivo è la trasformazione del problema da vincolato a problema non vincolato mediante funzioni di penalità. La funzione da minimizzare diventa:

$$F(\mathrm{x}, p_k) = \hat{f} + f_0 \cdot p_k \cdot \left(\sum_{i=1}^{n} X(x_i) + \sum_{i=1}^{m_1} G(\hat{g}_i) + \sum_{i=1}^{m_2} H(\hat{h}_i) + \sum_{i=1}^{m_3} W(\hat{w}_i) \right) \qquad (6.5)$$

In questa espressione X è la funzione di penalità relativa ai vincoli sulle variabili di progetto, G, H, W sono le funzioni di penalità relative ai vincoli sulle variabili di stato. La quantità p_k è detta 'parametro della superficie di risposta',

l'indice *k* compare in quanto all'interno della soluzione vengono utilizzate sotto iterazioni durante le quali p_k viene incrementato.

Il valore di riferimento f_0 per la funzione obiettivo viene introdotto per avere omogeneità delle unità di misura.

Per la soluzione dell'equazione (6.5) viene utilizzata un tecnica di minimizzazione non vincolata sequenziale (algoritmo SUMT).

Vicino al limite superiore della variabile di progetto, la funzione di penalità $X(x_i)$ è del tipo:

$$\frac{c_1 + c_2}{\overline{x}_i - x_i} \qquad \text{se} \qquad x_i < \overline{x}_i - \varepsilon\,(\overline{x}_i - \underline{x}_i) \tag{6.6}$$

$$\frac{c_3 + c_4}{\overline{x}_i - x_i} \qquad \text{se} \qquad x_i \geq \overline{x}_i - \varepsilon\,(\overline{x}_i - \underline{x}_i) \tag{6.7}$$

con $i = 1, \ldots, n$ e c_1, c_2, c_3 e c_4 opportune costanti calcolate dal codice, ε quantità infinitesima maggiore di zero.

Per le variabili di stato la forma delle funzioni di penalità è simile, per esempio la funzione di penalità $W(w_i)$ vale:

$$\frac{d_1 + d_2}{\overline{w}_i - \hat{w}_i} \qquad \text{se} \qquad \hat{w}_i < \overline{w}_i - \varepsilon\,(\overline{w}_i - \underline{w}_i) \tag{6.8}$$

$$\frac{d_3 + d_4}{\overline{w}_i - \hat{w}_i} \qquad \text{se} \qquad \hat{w}_i \geq \overline{w}_i - \varepsilon\,(\overline{w}_i - \underline{w}_i) \tag{6.9}$$

con $i = 1, \ldots, n$ e d_1, d_2, d_3 e d_4 costanti calcolate dal codice.

In modo silile si definiscono le funzioni di penalità per le funzioni di stato *G* ed *H*.

L'algoritmo SUMT calcola iterativamente il valore delle variabili di progetto $x^{(j)}$ in corrispondenza del minimo della funzione obiettivo non vincolata $F^{(j)}$, cioè ad ogni iterazione:

$$x^{(j)} \rightarrow \tilde{x}^{(j)} = F^{(j)} \rightarrow \tilde{F}^{(j)}$$

ove $\tilde{x}^{(j)}$ è il vettore delle variabili di progetto corrispondenti alla $\tilde{F}^{(j)}$.

Al termine di ciascun ciclo viene determinato il vettore delle variabili di progetto $x^{(j+1)}$ da utilizzare per la successiva iterazione (j+1). Il vetore è ottenuto secondo la seguente equazione:

$$x^{(j+1)} = x^{(b)} + c^{*} \cdot (\tilde{x}^{(j)} - x^{(b)}) \qquad (6.10)$$

ove c^{*} è una costante scelta dall'algoritmo, basata sul numero di soluzioni non fattibili, ed assume valori non compresi tra 0 e 1 e $x^{(b)}$ rappresenta la migliore configurazione delle variabili di progetto (best set), determinata secondo una delle seguenti condizioni:

1. se esistono già una o più configurazioni accettabili, la migliore è quella con il valore minore della funzione obiettivo.
2. se tutte le configurazioni prodotte sino a quel momento non sono accettabili, la migliore è quella più prossima a diventare accettabile, indipendentemente dal valore della funzione obiettivo.

6.3.1.3 Convergenza

La ripetizione dei cicli può arrestarsi per uno dei seguenti motivi:

1. è stato raggiunto il numero massimo di cicli stabiliti dall'utente
2. è stata raggiunta la convergenza dell'algoritmo.

La convergenza si ritiene raggiunta quando sia il set attuale $x^{(j)}$, sia il precedente $x^{(j-1)}$, sia il migliore $x^{(b)}$ sono fattibili, e quando una delle seguenti condizioni è soddisfatta:

- il modulo della differenza fra l'attuale valore della funzione obiettivo ed il valore corrispondente al miglior progetto accettabile, risulta minore di una tolleranza τ specificata dall'utente o assunta per default dal programma: $\left| f^{(j)} - f^{(b)} \right| \leq \tau$
- il modulo della differenza tra gli ultimi due valori della funzione obiettivo è minore della tolleranza τ: $\left| f^{(j)} - f^{(j-1)} \right| \leq \tau$
- le differenze tra le variabili di progetto, prese in valore assoluto, fra il migliore set accettabile e l'attuale, risultano minori delle rispettive tolleranze ρ_i scelte dall'utente o assunte per default dal codice:

$$\left| \mathrm{x}_i^{(j)} - \mathrm{x}_i^{(b)} \right| \leq \rho_i \qquad \text{con} \quad i = 1,2,\ldots,\mathrm{n}$$

- le differenze fra le variabili di progetto, prese in valore assoluto, fra gli ultimi due set risultano minori delle rispettive tolleranze:

$$\left| \mathrm{x}_i^{(j)} - \mathrm{x}_i^{(j-1)} \right| \leq \rho_i \qquad \text{con} \quad i = 1,2,\ldots,\mathrm{n}$$

6.3.2 METODO DEL FIRST ORDER

Questo metodo di ottimizzazione fa uso delle informazioni fornite dalle derivate. Il problema di minimo vincolato è ancora trasformato in problema di minimo non vincolato, mediante funzioni di penalità.

Le derivate sono valutate per le funzioni di penalità della funzione obiettivo e delle variabili di stato, in modo da avere delle informazioni circa l'orientamento da seguire nello spazio delle variabili di progetto. Ad ogni iterazione, sino a che la convergenza non è raggiunta, vengono eseguite ricerche nelle varie direzioni. Ad ogni iterazione sono svolte un certo numero di sotto-iterazioni che includono la ricerca della direzione di indagine e il calcolo del gradiente.

Confrontato con il metodo del *Subproblem*, il metodo del *First Order* richiede sicuramente una maggiore quantità di calcoli, risultando però anche più accurato.

6.3.2.1 La funzione obiettivo non vincolata

Il problema non vincolato è posto nella forma:

$$Q(\mathrm{x},q)=\frac{f}{f_0}+\sum_{i=1}^{n}P_x(x_i)+q\cdot\left(\sum_{i=1}^{m_1}P_g(g_i)+\sum_{i=1}^{m_2}P_h(h_i)+\sum_{i=1}^{m_3}P_w(w_i)\right) \tag{6.11}$$

ove :

- Q è la funzione obiettivo non vincolata adimensionale
- P_x , P_g , P_h , P_w sono le penalità applicate alle variabili di progetto e di stato.
- f_0 è la funzione obiettivo di riferimento, scelta tra il gruppo di set presenti al momento.

La rispondenza dei vincoli è controllata dal parametro q della superficie di risposta.

Una funzione di penalità esterna P_x è applicabile alle variabili di progetto. I vincoli sulle variabili di stato $\left(P_g\ ,P_h,P_w\right)$ sono invece rappresentati da funzioni di penalità prolungate per continuità all'interno.

Per esempio, per una variabile di stato vincolata da un estremo superiore, la funzione di penalità è scritta come:

$$P_g(g_i)=\left(\frac{g_i}{g_i+\alpha_i}\right)^{2\lambda} \tag{6.12}$$

ove λ è un numero intero sufficientemente grande in modo che la funzione assuma valori elevati qualora il vincolo venga violato.

Poichè la variazione di una singola variabile di progetto equivale ad un incremento assunto in una delle direzioni dello spazio ad *n* dimensioni in cui sono definite le variabili di progetto, un certo beneficio in termini di calcoli può essere ottenuto se la funzione Q viene riscritta come somma di due funzioni.

Definita:

$$Q_f(\mathrm{x}) = \frac{f}{f_0} \tag{6.13}$$

e:

$$\sum_{i=1}^{n} P_x(x_i) + q \cdot \left(\sum_{i=1}^{m_1} P_g(g_i) + \sum_{i=1}^{m_2} P_h(h_i) + \sum_{i=1}^{m_3} P_w(w_i) \right) \tag{6.14}$$

l'equazione (6.11) diviene:

$$Q(\mathrm{x}, q) = Q_f(\mathrm{x}) + Q_p(\mathrm{x}, q) \tag{6.15}$$

6.3.2.2 La direzione di indagine

Ad ogni iterazione (j) viene costruito il vettore $\mathrm{d}^{(j)}$ relativo alla direzione di indagine. La successiva iterazione è ottenuta dalla seguente equazione:

$$\mathrm{x}^{(j+1)} = \mathrm{x}^{(j)} + s_j \cdot \mathrm{d}^{(\mathrm{j})} \tag{6.16}$$

Il parametro di ricerca lineare $s_{(j)}$ misurato a partire da $\mathrm{x}^{(j)}$ corrisponde al minimo valore di Q nella direzione $\mathrm{d}^{(j)}$.

L'intervallo di definizione per $s_{(j)}$ è:

$$0 \le s_j \le \frac{s_{\max}}{100} \cdot \tilde{s}_j \tag{6.17}$$

ove :

- $\tilde{s}_j$ rappresenta il massimo incremento possibile per la ricerca lineare dell'interazione corrente, calcolato internamente;
- $s_{\max}$ rappresenta il massimo incremento percentuale della ricerca lineare, dato di input al programma;

La chiave per la minimizzazione dell'equazione (6.15) risiede nella generazione in sequenza delle direzioni di indagine e su aggiustamenti al parametro q della superficie di risposta.

Alla prima iterazione (j=0) la direzione di indagine è assunta come valore negativo del gradiente della funzione obiettivo non vincolata:

$$\mathrm{d}^{(0)} = -\nabla Q(\mathrm{x}^{(0)}, q) = \mathrm{d}_f^{(0)} + \mathrm{d}_p^{(0)} \tag{6.18}$$

in cui:

$q = 1$

$\mathrm{d}_\mathrm{f}^{(0)} = -\nabla Q_f(\mathrm{x}^{(0)})$

$\mathrm{d}_\mathrm{p}^{(0)} = -\nabla Q_p(\mathrm{x}^{(0)}, q)$

Chiaramente per la prima iterazione il metodo di ricerca è del tipo più rapido, mentre per le successive iterazioni vengono generate delle direzioni coniugate, in accordo con la formula di ricorsione di Polak-Ribiere:

$$\mathrm{d}^{(j)} = -\nabla Q(\mathrm{x}^{(j)}, q_k) + r_{j-1} \cdot \mathrm{d}^{(\mathrm{j-1})} \tag{6.19}$$

ove:

$$r_{j-1} = \frac{\left[\nabla Q(\mathrm{x}^{(j)}) - \nabla Q(\mathrm{x}^{(j-1)}, q)\right]^T \cdot \nabla Q(\mathrm{x}^{(j)}, q)}{\left|\nabla Q(\mathrm{x}^{(j-1)}, q)\right|^2} \tag{6.20}$$

Da rilevare che quando tutti i vincoli sulle variabili di progetto sono soddisfatti risulta $P_x(x_i) = 0$, questo significa che q può essere fattorizzato al di fuori dell'espressione di Q_p, ovvero:

$$Q_p(\mathrm{x}^{(j)}, q) = q \cdot Q_p(\mathrm{x}^{(j)}) \qquad \text{se} \quad \underline{x}_\mathrm{i} \le x_i \le \overline{x}_\mathrm{i} \qquad \text{con } i = 1, \ldots, \mathrm{n} \tag{6.21}$$

Piccole correzioni su q, da un'iterazione all'altra, sono effettuate in modo tale da non alterare la forma della (6.20) Modificando q si ottiene il controllo sulle variabili di progetto da parte del codice, per spingere, di quanto necessario, le variabili di stato verso i loro valori limite, a convergenza raggiunta.

Questo risulta più evidente se si separa la (6.20) in due vettori di direzione:

$$\mathrm{d}^{(j)} = \mathrm{d}_\mathrm{f}^{(\mathrm{j})} + \mathrm{d}_p^{(j)} \tag{6.22}$$

ove ogni vettore ha la propria direzione ricorsiva:

$$\mathrm{d}_f^{(j)} = -\nabla Q_f(\mathrm{x}^{(j)}) + r_{j-1} \cdot \mathrm{d}_f^{(j-1)} \tag{6.23}$$

$$\mathrm{d}_p^{(j)} = -q \cdot \nabla Q_p(\mathrm{x}^{(j)}) + r_{j-1} \cdot \mathrm{d}_p^{(j-1)} \tag{6.24}$$

E' necessario che sia disponibile il vettore gradiente; esso è calcolato usando la seguente approssimazione:

$$\frac{\partial Q}{\partial x_i}(\mathrm{x}^{(j)}) = \frac{Q(\mathrm{x}^{(j)} + \Delta \mathrm{x}_\mathrm{i} \cdot \mathrm{e}) - Q(\mathrm{x}^{(j)})}{\Delta \mathrm{x}_\mathrm{i}} \tag{6.25}$$

ove:

- e è un vettore con componenti tutte nulle tranne la i-esima pari ad 1;
- $\Delta \mathrm{x}_i = \Delta \mathrm{D} \cdot (\overline{\mathrm{x}}_i - \underline{\mathrm{x}}_i)/100$
- $\Delta \mathrm{D}$ è la differenza percentuale dell'incremento. E' un dato di input.

6.3.2.3 Convergenza

La convergenza è controllata tra il set corrente (j), il precedente (j-1) ed il migliore (b) ed è ottenuta se:

- $\left| f^{(i)} - f^{(j-1)} \right| \leq \tau$
- $\left| f^{(i)} - f^{(b)} \right| \leq \tau$

con τ tolleranza sulla funzione obiettivo.

E' inoltre un requisito che l'iterazione finale sia stata effettuata usando una ricerca di percorso più ripido (steepest descent), altrimenti vengono effettuate ulteriori iterazioni. In altre parole viene forzata una ricerca steepest descent e viene ricontrollata la convergenza.

6.4 Passi da seguire con il codice ANSYS

I cinque passi fondamentali su cui è basato il processo ciclico di ottimizzazione del codice ANSYS sono:

1. Definizione del problema in maniera parametrica: questa fase comprende la costruzione del modello agli elementi finiti ed è effettuata in fase di preprocessing.
2. Inizializzazione dei parametri che costituiscono le variabili di progetto: I valori forniti definiscono la configurazione iniziale (al ciclo zero) della struttura da ottimizzare.
3. Esecuzione dell'analisi relativa al modello iniziale (fase di soluzione).
4. Assegnazione dei valori numerici dei risultati ai parametri che costituiscono le variabili di stato e la funzione obiettivo: questo avviene in fase di *postprocessing*.
5. Costruzione dell'approssimazione (nel caso del *Subproblem*) della funzione obiettivo e delle funzioni di stato e calcolo delle variabili di progetto da utilizzare nel ciclo successivo (che riparte dal punto 2); questo compito è svolto dalla routine di ottimizzazione OPT.

Il comando OPEXE dà inizio alla riesecuzione ciclica dei comandi.

I diversi set delle variabili di progetto sono memorizzati sul FILE.OPT, il quale viene aggiornato ad ogni iterazione ed ogni volta che termina la routine di ottimizzazione con il comando FINISH.

Quando si utilizza il metodo del *Subproblem*, affinchè la prima approssimazione (di tipo lineare) della funzione obiettivo e delle variabili di stato possa essere costruita, il programma deve disporre di un numero di set uguale al numero delle variabili di progetto più due. Se i progetti iniziali non sono in numero sufficiente, l'algoritmo ne genera alcuni casuali ed esegue le analisi ad essi relative fino al raggiungimento del numero minimo necessario.

Una volta disponibile un numero sufficiente di set di variabili, la ricerca del minimo può essere avviata. Nuovi progetti non più casuali vengono così generati ed analizzati sino al raggiungimento della convergenza o del numero massimo di iterazioni previsto. Si entra così nel secondo ciclo in cui vengono eseguite le operazioni che vanno dal punto B al punto F.

Una volta raggiunta la convergenza il processo può essere riavviato con una procedura di *restart* se l'ottimo ottenuto non è soddisfacente. In tal caso, si possono migliorare le approssimazioni della funzione obiettivo e delle variabili di stato selezionando un certo numero di set e scartando i rimanenti, che non verranno quindi utilizzati per l'interpolazione. Il comando OPSEL consente la selezione di un numero massimo di 130 set. Le configurazioni selezionate sono valutate in base al criterio di pesatura stabilito con il comando OPEQN.

Il restart può essere ripetuto più volte, ed è anche possibile variare le tolleranze di convergenza, utilizzando opportuni parametri.

L'utente può scegliere anche la tolleranza di accettabilità per le variabili di stato, viceversa il valore assunto per default è pari ad un centesimo del corrispondente intervallo.

6.4.1 Single-Loop Analisys Tool

E' uno strumento molto semplice e diretto per vagliare lo spazio delle variabili di progetto. Calcolare il valore delle variabili di stato o della funzione obiettivo non è necessario ma può essere utile. Tutte le variabili di progetto sono esplicitamente definite dall'utente. Un single-loop è equivalente ad una analisi completa. Questo strumento viene attivato con il comando **OPTYPE**,RUN.

Prima di ogni iterazione l'utente definisce i valori di tutte le variabili di progetto ed esegue una sola iterazione, nel caso in cui siano state definite la funzione obiettivo e/o le variabili di stato, i rispettivi valori verranno calcolati e visualizzati.

6.4.2 Random Tool

Con questo strumento i set di variabili di progetto vengono generati in modo del tutto casuale ad ogni iterazione. Non è necessario definire una funzione obiettivo variabili di stato, ma può essere utile farlo nel caso vengano eseguiti più cicli di ottimizzazione in serie. Questo strumento viene selezionato mediante il comando **OPTYPE**,RAND.

La generazione di set casuali continua sino a quando una delle seguenti condizioni è soddisfatta:

- Viene raggiunto il numero di iterazioni specificato (NITR)
- Viene raggiunto il numero di feasible sets specificato (NFEAS)

La sintassi è: **OPRAND**,NITR,NFEAS

6.4.3 Sweep Tool

Lo strumento Sweep viene utilizzato per un'indagine complessiva dello spazio delle variabili di progetto centrato su un particolare set di variabili di riferimento. Questo metodo viene selezionato con il comando **OPTYPE**,SWEEP.

Ad ogni ciclo viene spazzato il campo di variazione di una variabile di progetto

mantenendo fisse tutte le altre. Ad ogni passo le variabili di stato e la funzione obiettivo vengono calcolate ed i rispettivi valori vengono memorizzati.
Un ciclo sweep produce n_s design sets ove

$$n_s = n \cdot N_s \tag{6.26}$$

con:

n numero di variabili di progetto

N_s numero di valutazioni per ogni variabile (specificato come NSPS nel comando **OPSWEEP**)

Per esempio, consideriamo una porzione di sweep eseguita per la variabile di progetto k. Per semplicità siano i design sets risultanti numerati come m+1, m+2, ecc., dove m è il set esistente prima di questa parte dello sweep. Le variabili di progetto di un dato set sono espresse da:

$$\mathrm{x}^{(m+i)} = \mathrm{x}^{(r)} + (i-1) \cdot \Delta \mathrm{x}_k \, \mathrm{e}^{(k)} \qquad \text{con } i = 1, \ldots, N_s \tag{6.27}$$

ove

$\mathrm{x}^{(r)}$ variabile di progetto di riferimento avente $\underline{\mathrm{x}}_k$ come k-esima componente, e valori di riferimento per le altre componenti. Il set di riferimento r-esimo è dato come input con l'opzione DSET del comando OPSWEEP.

$\mathrm{e}^{(k)}$ vettore avente la k-esima componente pari a 1 e nulle tutte le altre.

L'incremento di ogni variabile di progetto in ogni passo del ciclo di *sweep* è:

$$\Delta \mathrm{x}_k = (\overline{\mathrm{x}}_k - \underline{\mathrm{x}}_k) / (N_s - 1)$$

6.4.4 Gradient Tool

Questo strumento calcola il gradiente della funzione obiettivo e delle variabili di stato rispetto alle singole variabili di progetto. Il *Gradient Tool* viene selezionato con il comando **OPTYPE**,GRAD.

Con il campo DSET del comando **OPGRAD** viene definito il set di variabili di progetto come punto in corrispondenza del quale calcolare il gradiente.

Considerando ad esempio la funzione obiettivo;

sia

$$f_r(\mathrm{x}) = f(\mathrm{x}^{(\mathrm{r})})$$

la funzione obiettivo calcolata in corrispondenza del set di riferimento $\mathrm{x}^{(r)}$; il gradiente rispetto alle variabili di progetto è dato da:

$$\nabla f_r = \left[\frac{\partial f_r}{\partial x_1}, \frac{\partial f_r}{\partial x_2}, \frac{\partial f_r}{\partial x_3}, \ldots, \frac{\partial f_r}{\partial x_n} \right]$$

La derivata rispetto a ciascuna variabile di progetto x_i viene approssimata con la seguente espressione:

$$\frac{\partial f_r}{\partial x_i} = \frac{f_r(\mathrm{x} + \Delta \mathrm{x}_i \cdot \mathrm{e}) - f_r(\mathrm{x})}{\Delta \mathrm{x}_i}$$

ove:

- e è un vettore con componenti tutte nulle tranne la i-esima pari ad 1;
- $\Delta \mathrm{x}_i = \Delta \mathrm{D} \cdot (\overline{\mathrm{x}}_i - \underline{\mathrm{x}}_i)/100$
- $\Delta \mathrm{D}$ è la differenza percentuale dell'incremento. Viene specificato con il campo DELTA del comando **OPGRAD**.

6.4.5 Factorial Tool

Il *Factorial Tool* è un metodo di tipo statistico che fa riferimento alla teoria dei progetti fattoriali di Box e Hunter, più conosciuta con il suo acronimo DOE, Design Of Experiment. Questo metodo esegue un progetto fattoriale completo o frazionato (da ½ sino a 1/64) del tipo 2^k, cioè valutando due livelli per ogni fattore. Il codice esegue le 2^k prove (2^{k-p} nel caso di progetto frazionato). Attraverso il comando **OPLFA** viene rappresentato sotto forma di istogramma l'effetto principale dei fattori (MAIN), l'effetto delle interazioni a due a due (2FAC) ed anche l'effetto delle interazioni a tre (3FAC). I risultati in forma tabulare si ottengono con il comando **OPRFA**.

6.5 L'affinamento finale con il metodo del gradiente

Come é stato anticipato nel capitolo 5, una qualsiasi ottimizzazione ottenuta mediante algoritmi genetici necessita di essere affinata utilizzando le informazioni fornite dal gradiente. A partire dalla configurazione ottenuta con l'algoritmo genetico é stato utilizzato il metodo del *First Order* di ANSYS per ricercare eventuali ulteriori miglioramenti.

Si é deciso di effettuare due distinti calcoli di affinamento, uno riguardante le densitá di corrente ed uno relativo alle posizioni delle bobine. E' stata fatta questa scelta per evitare un'ottimizzazione con 16 variabili indipendenti, eccessivamente onerosa dal punto di vista del tempo macchina.

La tabella nella pagina seguente raccoglie i valori numerici delle complessive 16 variabili di progetto, ottenuti con le due analisi effettuate con il metodo del *First Order*.

L'uniformitá del campo magnetico raggiunta, nella regione di interesse, a seguito di questo affinamento, é passata dal **2.5‰** al **2.0‰**.

Nel capitolo 7, dedicato al modello 3D, sará verificata la validitá di questi risultati, ottenuti, lo ricordo, con un modello 2D assialsimmetrico.

1	Curr1	$0.9496 \cdot 10^8$ A / m^2
2	Curr2	$1.3358 \cdot 10^8$ A / m^2
3	Curr3	$2.4590 \cdot 10^8$ A / m^2
4	Curr4	$2.7173 \cdot 10^8$ A / m^2
5	Curr5	$2.4188 \cdot 10^8$ A / m^2
6	Curr6	$2.2230 \cdot 10^8$ A / m^2
7	Coox1	0.2179 m
8	Coox2	0.4276 m
9	Coox3	0.2473 m
10	Coox4	0.2142 m
11	Coox5	0.1722 m
12	Coox6	0.1149 m
13	Cooy3	0.0820 m
14	Cooy4	0.1482 m
15	Cooy5	0.1866 m
16	Cooy6	0.2470 m

Tab 6.1 *Valori numerici delle variabili di progetto ottenute con l'affinamento con il metodo del gradiente.*

I valori assunti dalle variabili di stato sono compatibili con i vincoli imposti e risultano:

campo magnetico massimo 5.9 T

campo sull'asse del magnete 4.0 T

L'obiettivo del nostro lavoro era quello di ottenere una configurazione delle bobine che ci permettesse di ottenere un campo magnetivo verticale il piú omogeneo possibile. La figura 6.1 visualizza le linee di flusso del campo, come si puó notare, nella regione all'interno delle bobine, le linee di flusso sono perfettamente parallele e perpendicolari rispetto al piano equatoriale.

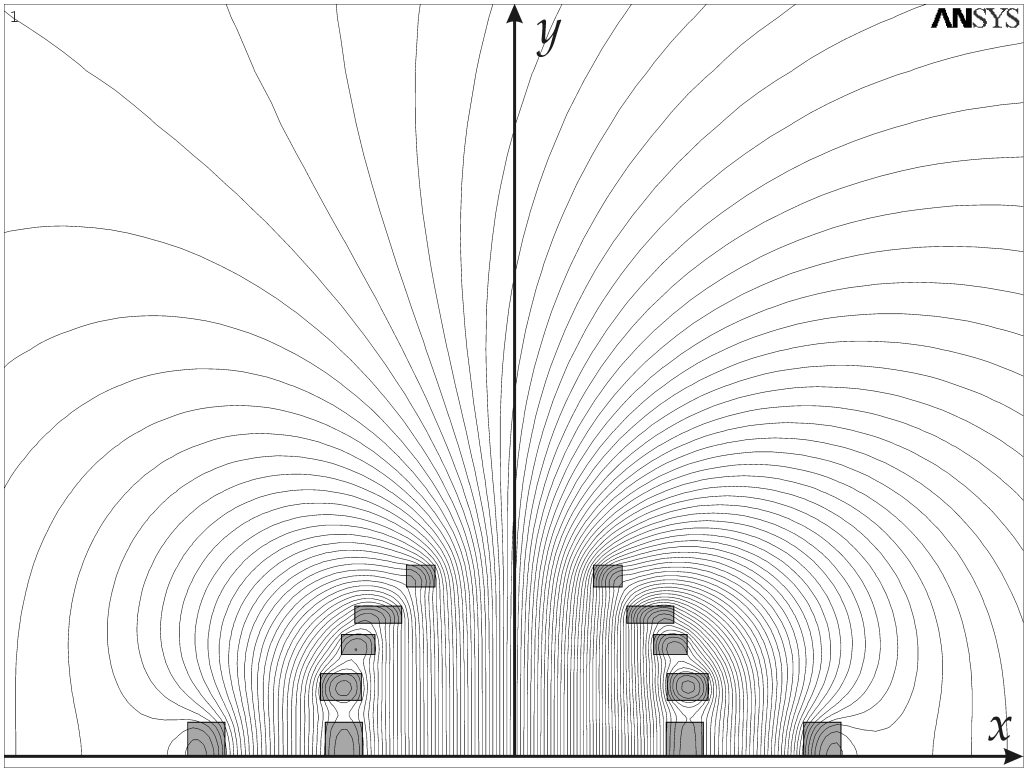


Fig 6.1 *Linee di flusso del campo magnetico.*

La figura 6.2 mostra invece i valori del campo magnetico, sotto forma di curve di livello, nella regione di interesse di 60x200 mm².

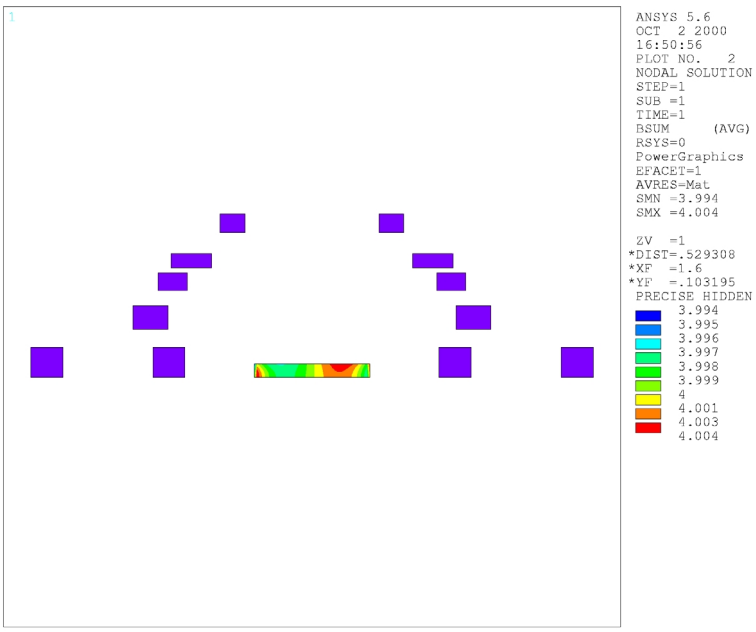


Fig 6.2 *Curve di livello del campo magnetico nella regione di interesse*

La figura 6.3 e mostra l'andamento del campo magnetico calcolato sul piano equatoriale del magnete, come si puó vedere, in prossimitá del centro del magnete, il campo risulta pressoché costante e pari a 4 T come da specifica; é necessario effettuare uno zoom (figura 6.4) per poterne apprezzare le variazioni.

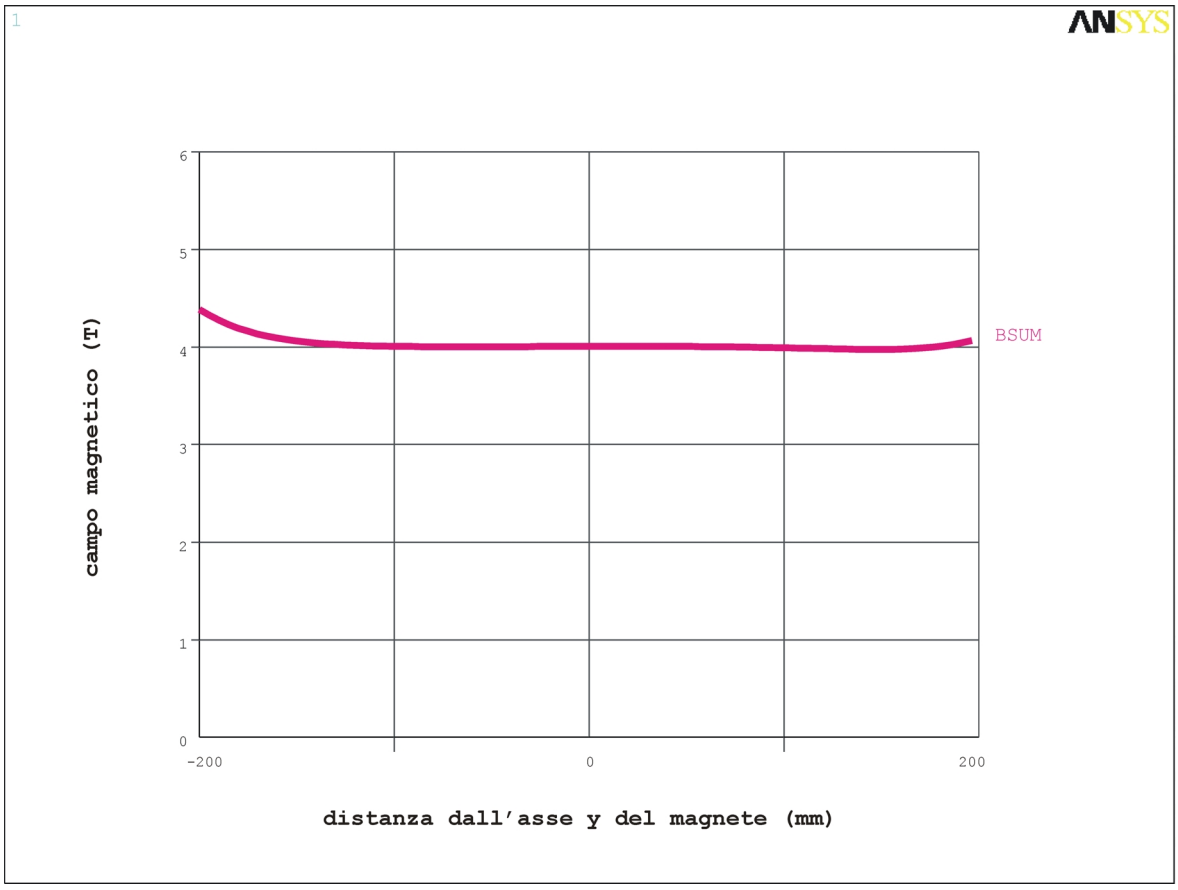


Fig 6.3 *Andamento del campo magnetico sul piano equatoriale del magnete*

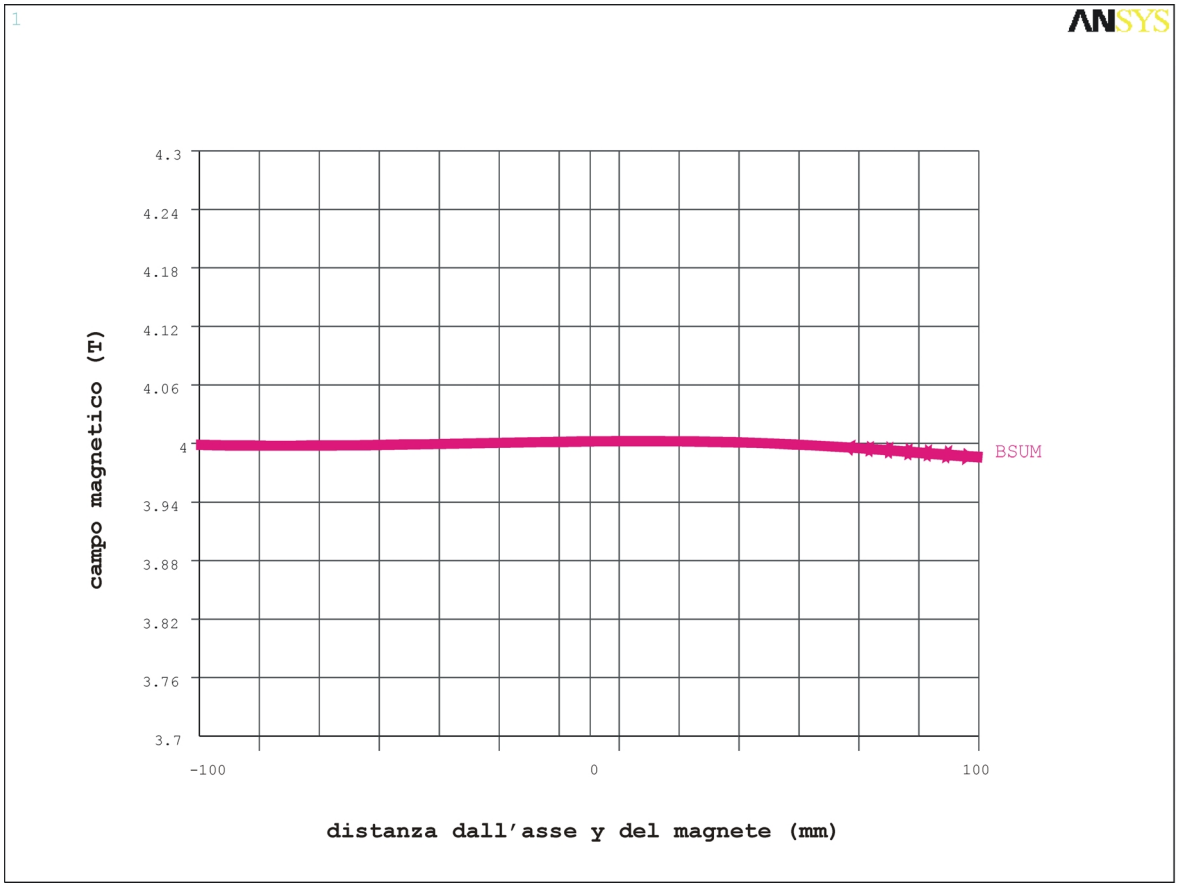


Fig 6.4 *Andamento del campo magnetico sul piano equatoriale del magnete nella regione di interesse.*

7

Il modello 3D

7.1 Scelta della formulazione

Il passaggio da un'analisi magnetica 2D a un'analisi magnetica 3D non é triviale, ma richiede la scelta a priori del tipo di formulazione che si vuole utilizzare. In ANSYS infatti esistono due metodi distinti per risolvere problemi magnetostatici: la *formulazione scalare*, che ha come unico grado di libertá il potenziale scalare, e la *formulazione vettoriale*, che ha come gradi di libertá le 3 componenti del potenziale vettore. Date le caratteristiche del nostro problema, che richiede un'analisi statica in assenza di regioni permeabili, la formulazione piú conveniente é sicuramente quella scalare. Essa presenta infatti due principali vantaggi. In primo luogo permette di minimizzare il numero di nodi del modello, in quanto le sorgenti di corrente non devono essere parte integrante della mesh, ma vengono modellate separatamente tramite le primitive dell'elemento SOURC36 e quindi sovrapposte alla mesh. Secondariamente il numero inferiore di gradi di libertá riduce drasticamente il tempo di CPU necessario alla soluzione.

7.2 I conduttori

Il secondo problema da affrontare é la scelta della struttura tridimensionale dei conduttori. Nei calcoli 2D assial-simmetrici infatti é come se ogni singolo conduttore fosse un anello chiuso, dando cosí origine, complessivamente, ad una sorta di toroide. Passando al modello 3D, che deve compiere, angolarmente, circa 90°, é necessario stabilire la struttura delle “teste”, ossia della richiusura di ogni singolo conduttore con il suo opposto. Il metodo piú semplice, mostrato in fig.7.1, é quello di chiudere le bobine tramite degli archi di circonferenza. Chiaramente i

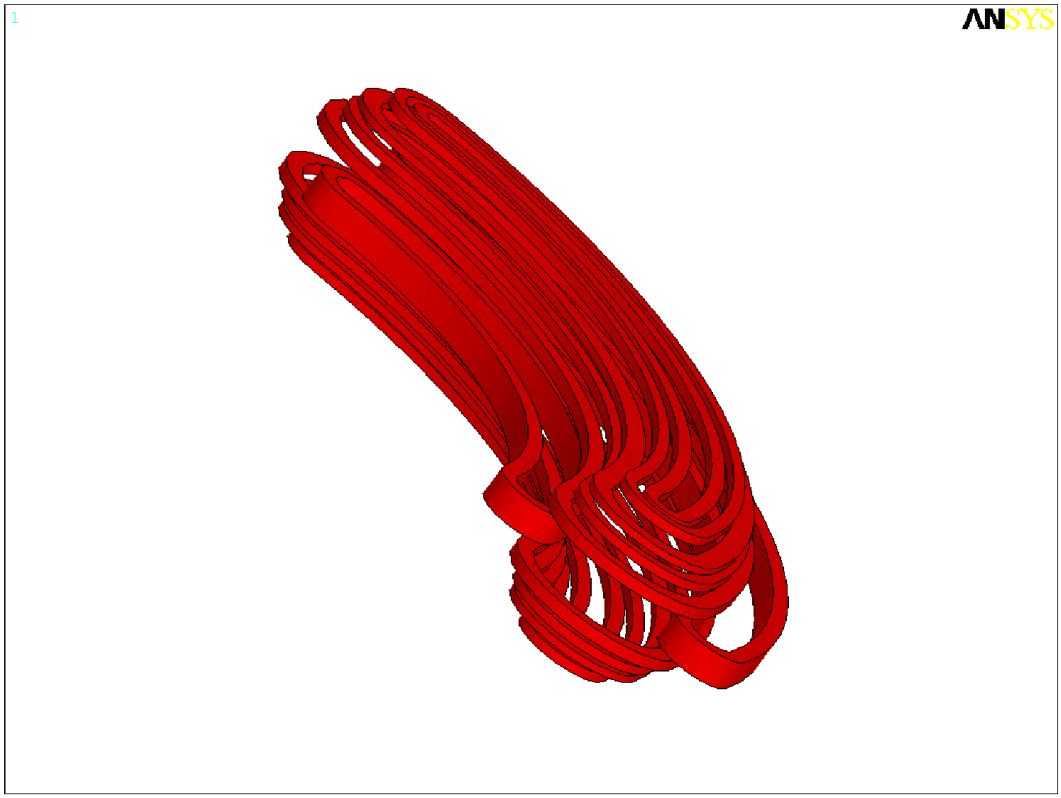


Fig.7.1 *Modello tridimensionale dei conduttori.*

risultati di questo primo approccio saranno da considerarsi preliminari, in quanto la struttura delle teste andrá successivamente studiata in modo da ottimizzare il percorso compiuto dal fascio di adroni, che deve ruotare di 90° restando collimato.

7.3 Risultati

In fig.7.2 é mostrato, in sezione, il campo magnetico risultante. L'accordo al centro del gantry con i risultati dell'analisi 2D é ottimo, sia in termini di distribuzione del campo magnetico che di uniformitá. L'unica differenza significativa é che i valori di campo magnetico risultano essere uniformemente piú elevati di circa il 5%; é ancora da chiarire se questo effetto sia dovuto ad una discretizzazione inevitabilmente piú larga del modello 3D rispetto al 2D, oppure all'influenza della richiusura dei conduttori.

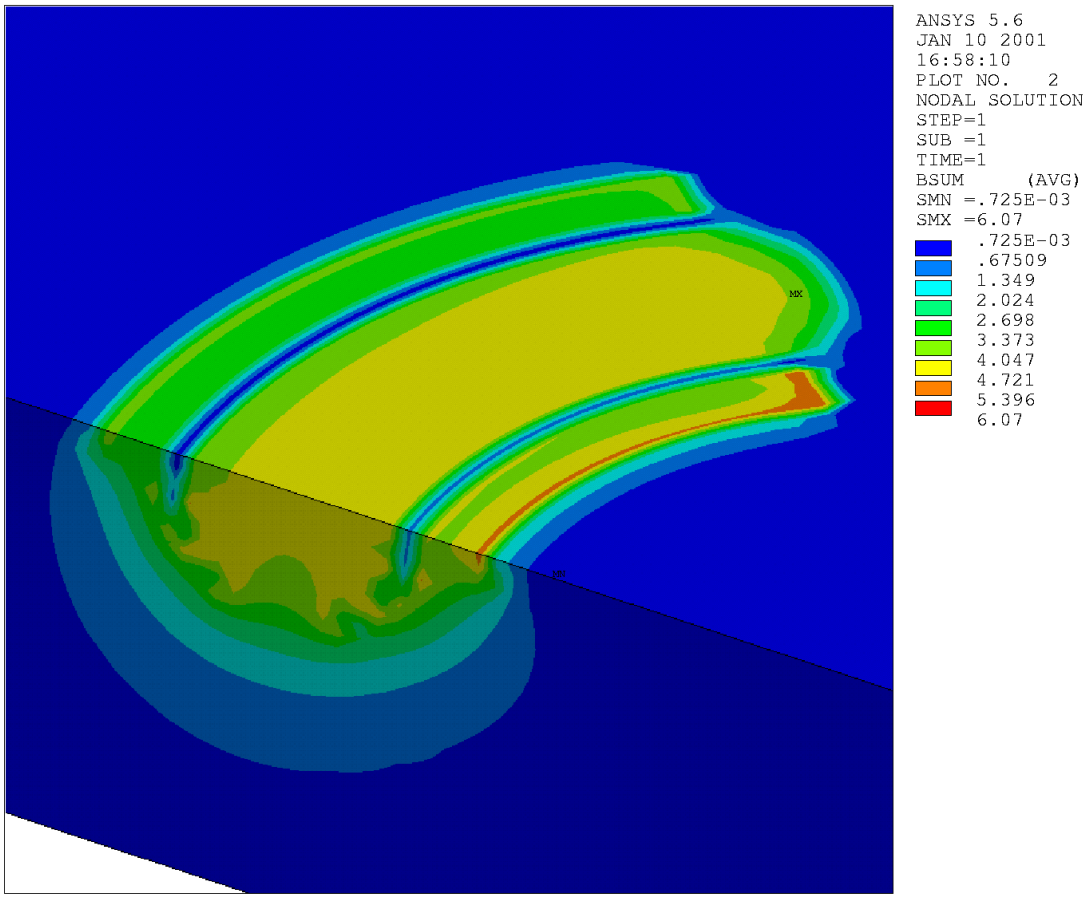


Fig.7.2 *Sezione del campo magnetico risultante.*

Per analizzare piú in dettaglio come la distibuzione di campo magnetico influisca sul percorso compiuto dal fascio di adroni, consideriamo l'ascissa curvilinea schematizzata in rosso in fig.7.3. Essa rappresenta il percorso idealmente compiuto

da particelle che entrino nel gantry in posizione centrale, e verrá adottata come variabile indipendente per i grafici successivi.

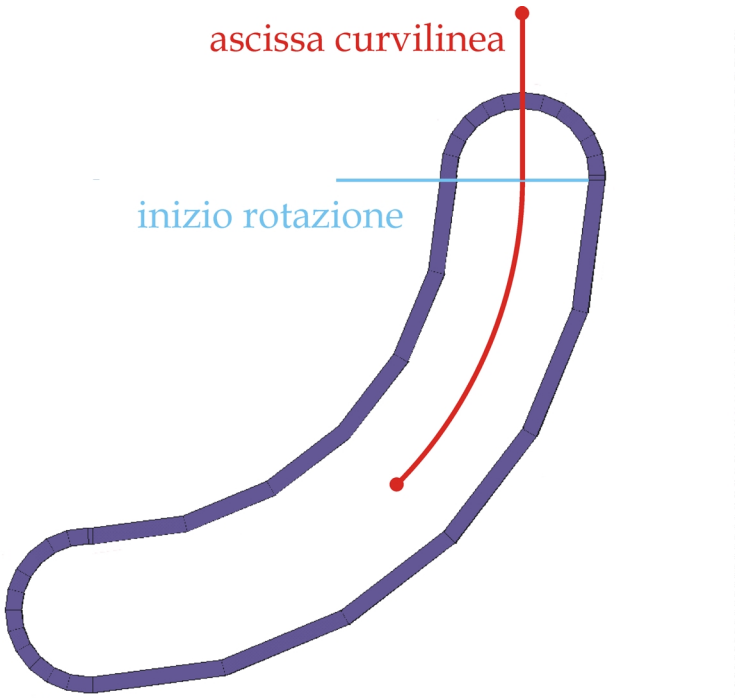


Fig.7.3 *Descrizione schematica dell'ascissa curvilinea utilizzata come variabile indipendente nelle figure successive.*

Consideriamo allora l'andamento del campo magnetico in funzione dell'ascissa curvilinea, mostrato in fig.7.4. Il profilo ideale é quello di una funzione a gradino; infatti, ricordando che il campo magnetico induce su particelle cariche una rotazione di raggio $r = \frac{mv}{qB}$, se il campo fosse nullo fino all'imbocco del gantry ed esattamente pari a 4 T lungo tutta la rotazione, ne seguirebbe che il fascio non subirebbe nessuna deviazione di traiettoria finché, entrato nel gantry, non comincerebbe a ruotare esattamente del raggio desiderato. Purtroppo, come si evince dalla figura, il profilo di campo magnetico risultante é ancora sostanzialmente dissimile dal profilo ideale. La conseguenza piú grave consiste nel fatto che, ancora prima di entrare fisicamente nella regione di rotazione, le particelle sono soggette a valori elevati di campo magnetico, e quindi assumono traiettorie circolari, di raggio anche inferiore a quello progettato. Il fascio quindi

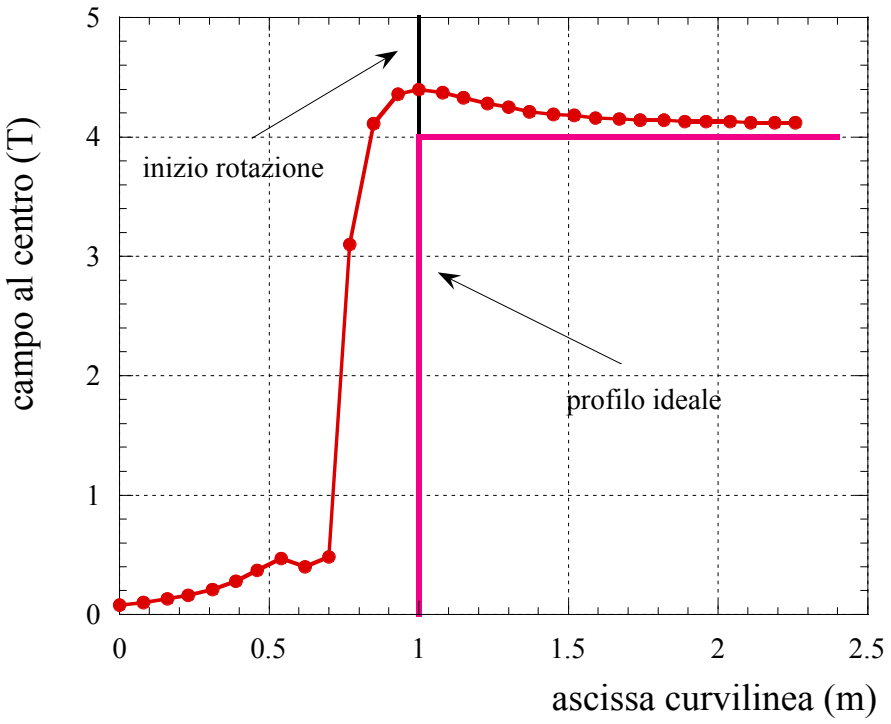


Fig.7.4 *Profilo del campo magnetico lungo il percorso descritto dall'ascissa curvilinea.*

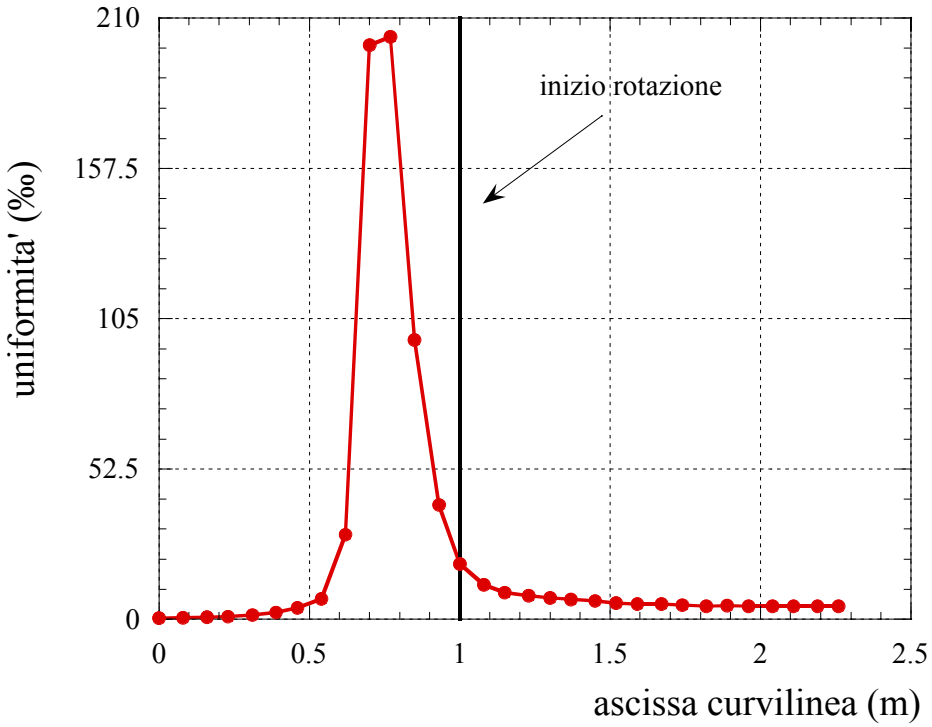


Fig.7.5 *Uniformitá del campo magnetico in sezioni 200×60 mm^2 perpendicolari al percorso descritto dall'ascissa curvilinea.*

entra nel gantry ben al di fuori della zona 200×60 mm^2 in cui é stata ottimizzata l'uniformitá, dando cosí origine a traiettorie non volute e imprevedibili. Fondamentale é quindi l'ottimizzazione della forma delle "teste", che devono accompagnare il fascio nella zona di rotazione senza perturbare troppo lo spazio circostante. Come si vede infatti dalla figura 7.5, che rappresenta l'uniformitá del campo magnetico in sezioni 200×60 mm^2 perpendicolari al percorso descritto dall'ascissa curvilinea, se si riuscisse a far entrare il fascio nel gantry nella posizione corretta, l'uniformitá diventerebbe quasi subito sufficiente a far percorrere al fascio la traiettoria voluta.

Conclusioni

Questo lavoro ha permesso di ottenere un miglioramento considerevole delle prestazioni del magnete in termini di uniformitá di campo magnetico, rispetto a quelle fornite dal progetto frutto dello studio preliminare condotto dal gruppo di lavoro dell'Istituto Nazionale di Fisica Nucleare.

Le specifiche di progetto richiedevano un'uniformitá del campo magnetico almeno del 2‰ in una regione di 200x60 mm^2.

Il progetto preliminare non consentiva il soddisfacimento delle specifiche sopracitate, per cui il lavoro é stato centrato sul loro raggiungimento.

Per lo svolgimento di questa tesi sono state utilizzate analisi agli elementi finiti pilotate da un algoritmo genetico. Il lavoro svolto mostra come l'algoritmo genetico combini l'elevata robustezza con la notevole semplicitá di implementazione e generalitá di impiego.

L'analisi 2D ha dimostrato che la nuova configurazione definita con l'ottimizzazione soddisfa tutte le specifiche richieste.

L'analisi 3D evidenzia un ottimo accordo con i risultati dell'analisi 2D al centro del magnete. I risultati in prossimitá delle richiusure delle bobine dovranno invece ritenersi preliminari, in quanto la geometria delle richiusure, non considerata nell'analisi 2D, andrá successivamente studiata in modo da ottimizzare il percorso compiuto dal fascio di particelle.

Appendice A

Listato del codice Fortran che implementa l'algoritmo genetico.

```
c
      program gafortran


c
c  This is version 1.7, last updated on 12/11/98.
c
c  Copyright David L. Carroll; this code may not be reproduced for sale
c  or for use in part of another code for sale without the express
c  written permission of David L. Carroll.
c
c
c  David L. Carroll
c  University of Illinois
c  306 Talbot Lab
c  104 S. Wright St.
c  Urbana, IL  61801
c
c  e-mail: carroll@uiuc.edu
c  Phone:  217-333-4741
c  fax:    217-244-0720
c
c  This genetic algorithm (GA) driver is free for public use.  My only
c  request is that the user reference and/or acknowledge the use of this
c  driver in any papers/reports/articles which have results obtained
c  from the use of this driver.  I would also appreciate a copy of such
c  papers/articles/reports, or at least an e-mail message with the
c  reference so I can get a copy.  Thanks.
c
c  This program is a FORTRAN version of a genetic algorithm driver.
c  This code initializes a random sample of individuals with different
c  parameters to be optimized using the genetic algorithm approach, i.e.
c  evolution via survival of the fittest.  The selection scheme used is
c  tournament selection with a shuffling technique for choosing random
c  pairs for mating.  The routine includes binary coding for the
c  individuals, jump mutation, creep mutation, and the option for
c  single-point or uniform crossover.  Niching (sharing) and an option
c  for the number of children per pair of parents has been added.
c  An option to use a micro-GA is also included.
c
c  For companies wishing to link this GA driver with an existing code,
c  I am available for some consulting work.  Regardless, I suggest
```

```
c  altering this code as little as possible to make future updates
c  easier to incorporate.
c
c  Any users new to the GA world are encouraged to read David Goldberg's
c  "Genetic Algorithms in Search, Optimization and Machine Learning,"
c  Addison-Wesley, 1989.
c
c  Other associated files are:  ga.inp
c                               ga.out
c                               ga.restart
c                               params.f
c                               ReadMe
c                               ga2.inp (w/ different namelist identifier)
c
c  I have provided a sample subroutine "func", but ultimately
c  the user must supply this subroutine "func" which should be your
c  cost function.  You should be able to run the code with the
c  sample subroutine "func" and the provided ga.inp file and obtain
c  the optimal function value of 1.0000 at generation 187 with the
c  uniform crossover micro-GA enabled (this is 935 function evaluations).
c
c  The code is presently set for a maximum population size of 200,
c  30 chromosomes (binary bits) and 8 parameters.  These values can be
c  changed in params.f as appropriate for your problem.  Correspondingly
c  you will have to change a few 'write' and 'format' statements if you
c  change nchrome and/or nparam.  In particular, if you change nchrome
c  and/or nparam, then you should change the 'format' statement numbers
c  1050, 1075, 1275, and 1500 (see ReadMe file).
c
c  Please feel free to contact me with questions, comments, or errors
c  (hopefully none of latter).
c
c  Disclaimer:  this program is not guaranteed to be free of error
c  (although it is believed to be free of error), therefore it should
c  not be relied on for solving problems where an error could result in
c  injury or loss.  If this code is used for such solutions, it is
c  entirely at the user's risk and the author disclaims all liability.
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      implicit real*8 (a-h,o-z)
      save


c
      include 'params.f'
      dimension parent(nparmax,indmax),child(nparmax,indmax)
      dimension fitness(indmax),nposibl(nparmax),nichflg(nparmax)
      dimension iparent(nchrmax,indmax),ichild(nchrmax,indmax)
      dimension g0(nparmax),g1(nparmax),ig2(nparmax)
      dimension ibest(nchrmax)
      dimension parmax(nparmax),parmin(nparmax),pardel(nparmax)
      dimension geni(1000000),genavg(1000000),genmax(1000000)
c     real*4 cpu,cpu0,cpu1,tarray(2)
c
      common / ga1   / npopsiz,nowrite
      common / ga2   / nparam,nchrome
      common / ga3   / parent,iparent
```

```
      common / ga4   / fitness
      common / ga5   / g0,g1,ig2
      common / ga6   / parmax,parmin,pardel,nposibl
      common / ga7   / child,ichild
      common / ga8   / nichflg
      common /inputga/ pcross,pmutate,pcreep,maxgen,idum,irestrt,
     +                 itourny,ielite,icreep,iunifrm,iniche,
     +                 iskip,iend,nchild,microga,kountmx
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c  Input variable definitions:
c
c  icreep   = 0 for no creep mutations
c           = 1 for creep mutations; creep mutations are recommended.
c  idum     The initial random number seed for the GA run.  Must equal
c           a negative integer, e.g. idum=-1000.
c  ielite   = 0 for no elitism (best individual not necessarily
c              replicated from one generation to the next).
c           = 1 for elitism to be invoked (best individual replicated
c              into next generation); elitism is recommended.
c  iend        = 0 for normal GA run (this is standard).
c           = number of last population member to be looked at in a set
c             of individuals.  Setting iend-0 is only used for debugging
c             purposes and is commonly used in conjunction with iskip.
c  iniche   = 0 for no niching
c           = 1 for niching; niching is recommended.
c  irestrt  = 0 for a new GA run, or for a single function evaluation
c           = 1 for a restart continuation of a GA run.
c  iskip    = 0 for normal GA run (this is standard).
c           = number in population to look at a specific individual or
c             set of individuals.  Setting iskip-0 is only used for
c             debugging purposes.
c  itourny  No longer used.  The GA is presently set up for only
c           tournament selection.
c  iunifrm  = 0 for single-point crossover
c           = 1 for uniform crossover; uniform crossover is recommended.
c  kountmx  = the maximum value of kount before a new restart file is
c             written; presently set to write every fifth generation.
c             Increasing this value will reduce I/O time requirements
c             and reduce wear and tear on your storage device
c  maxgen   The maximum number of generations to run by the GA.
c           For a single function evaluation, set equal to 1.
c  microga  = 0 for normal conventional GA operation
c           = 1 for micro-GA operation (this will automatically reset
c             some of the other input flags).  I recommend using
c             npopsiz=5 when microga=1.
c  nchild   = 1 for one child per pair of parents (this is what I
c              typically use).
c           = 2 for two children per pair of parents (2 is more common
c              in GA work).
c  nichflg  = array of 1/0 flags for whether or not niching occurs on
c             a particular parameter.  Set to 0 for no niching on
c             a parameter, set to 1 for niching to operate on parameter.
c             The default value is 1, but the implementation of niching
c             is still controlled by the flag iniche.
c  nowrite  = 0 to write detailed mutation and parameter adjustments
c           = 1 to not write detailed mutation and parameter adjustments
```

```
c  nparam   Number of parameters (groups of bits) of each individual.
c          Make sure that nparam matches the number of values in the
c          parmin, parmax and nposibl input arrays.
c  npopsiz  The population size of a GA run (typically 100 works well).
c          For a single calculation, set equal to 1.
c  nposibl  = array of integer number of possibilities per parameter.
c           For optimal code efficiency set nposibl=2**n, i.e. 2, 4,
c           8, 16, 32, 64, etc.
c  parmax   = array of the maximum allowed values of the parameters
c  parmin   = array of the minimum allowed values of the parameters
c  pcreep   The creep mutation probability.  Typically set this
c          = (nchrome/nparam)/npopsiz.
c  pcross   The crossover probability.  For single-point crossover, a
c          value of 0.6 or 0.7 is recommended.  For uniform crossover,
c          a value of 0.5 is suggested.
c  pmutate  The jump mutation probability.  Typically set = 1/npopsiz.
c
c
c  For single function evaluations, set npopsiz=1, maxgen=1, & irestrt=0.
c
c  My favorite initial choices of GA parameters are:
c     microga=1, npopsiz=5, iunifrm=1, maxgen=200
c     microga=1, npopsiz=5, iunifrm=0, maxgen=200
c  I generally get good performance with both the uniform and single-
c  point crossover micro-GA.
c
c  For those wishing to use the more conventional GA techniques,
c  my old favorite choice of GA parameters was:
c     iunifrm=1, iniche=1, ielite=1, itourny=1, nchild=1
c  For most problems I have dealt with, I get good performance using
c     npopsiz=100, pcross=0.5, pmutate=0.01, pcreep=0.02, maxgen=26
c  or
c     npopsiz= 50, pcross=0.5, pmutate=0.02, pcreep=0.04, maxgen=51
c
c  Any negative integer for idum should work.  I typically arbitrarily
c  choose idum=-10000 or -20000.
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c  Code variable definitions (those not defined above):
c
c  best     = the best fitness of the generation
c  child    = the floating point parameter array of the children
c  cpu      = cpu time of the calculation
c  cpu0,cpu1= cpu times associated with 'etime' timing function
c  creep    = +1 or -1, indicates which direction parameter creeps
c  delta    = del/nparam
c  diffrac  = fraction of total number of bits which are different
c           between the best and the rest of the micro-GA population.
c           Population convergence arbitrarily set as diffrac<0.05.
c  evals    = number of function evaluations
c  fbar     = average fitness of population
c  fitness  = array of fitnesses of the parents
c  fitsum   = sum of the fitnesses of the parents
c  genavg   = array of average fitness values for each generation
c  geni     = generation array
c  genmax   = array of maximum fitness values for each generation
c  g0       = lower bound values of the parameter array to be optimized.
```

```
c           The number of parameters in the array should match the
c           dimension set in the above parameter statement.
c  g1       = the increment by which the parameter array is increased
c           from the lower bound values in the g0 array.  The minimum
c           parameter value is g0 and the maximum parameter value
c           equals g0+g1*(2**g2-1), i.e. g1 is the incremental value
c           between min and max.
c  ig2      = array of the number of bits per parameter, i.e. the number
c           of possible values per parameter.  For example, ig2=2 is
c           equivalent to 4 (=2**2) possibilities, ig2=4 is equivalent
c           to 16 (=2**4) possibilities.
c  ig2sum   = sum of the number of possibilities of ig2 array
c  ibest    = binary array of chromosomes of the best individual
c  ichild   = binary array of chromosomes of the children
c  icount   = counter of number of different bits between best
c           individual and other members of micro-GA population
c  icross   = the crossover point in single-point crossover
c  indmax   = maximum # of individuals allowed, i.e. max population size
c  iparent  = binary array of chromosomes of the parents
c  istart   = the generation to be started from
c  jbest    = the member in the population with the best fitness
c  jelite   = a counter which tracks the number of bits of an individual
c           which match those of the best individual
c  jend     = used in conjunction with iend for debugging
c  jstart   = used in conjunction with iskip for debugging
c  kount    = a counter which controls how frequently the restart
c           file is written
c  kelite   = kelite set to unity when jelite=nchrome, indicates that
c           the best parent was replicated amongst the children
c  mate1    = the number of the population member chosen as mate1
c  mate2    = the number of the population member chosen as mate2
c  nchrmax  = maximum # of chromosomes (binary bits) per individual
c  nchrome  = number of chromosomes (binary bits) of each individual
c  ncreep   = # of creep mutations which occurred during reproduction
c  nmutate  = # of jump mutations which occurred during reproduction
c  nparmax  = maximum # of parameters which the chromosomes make up
c  paramav  = the average of each parameter in the population
c  paramsm  = the sum of each parameter in the population
c  parent   = the floating point parameter array of the parents
c  pardel   = array of the difference between parmax and parmin
c  rand     = the value of the current random number
c  npossum  = sum of the number of possible values of all parameters
c  tarray   = time array used with 'etime' timing function
c  time0    = clock time at start of run
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c  Subroutines:
c  ____________
c
c  code     = Codes floating point value to binary string.
c  crosovr  = Performs crossover (single-point or uniform).
c  decode   = Decodes binary string to floating point value.
c  evalout  = Evaluates the fitness of each individual and outputs
c           generational information to the 'ga.out' file.
c  func     = The function which is being evaluated.
c  gamicro  = Implements the micro-GA technique.
c  input    = Inputs information from the 'ga.inp' file.
```

```
c  initial  = Program initialization and inputs information from the
c            'ga.restart' file.
c  mutate   = Performs mutation (jump and/or creep).
c  newgen   = Writes child array back into parent array for new
c            generation; also checks to see if best individual was
c            replicated (elitism).
c  niche    = Performs niching (sharing) on population.
c  possibl  = Checks to see if decoded binary string falls within
c            specified range of parmin and parmax.
c  ran3     = The random number generator.
c  restart  = Writes the 'ga.restart' file.
c  select   = A subroutine of 'selectn'.
c  selectn  = Performs selection; tournament selection is the only
c            option in this version of the code.
c  shuffle  = Shuffles the population randomly for selection.
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c     call etime(tarray)
c     write(6,*) tarray(1),tarray(2)
c     cpu0=tarray(1)
c


c  Call the input subroutine.
c     TIME0=SECNDS(0.0)
      call input
c
c  Perform necessary initialization and read the ga.restart file.
      call initial(istart,npossum,ig2sum)
c
c  $$$$$ Main generational processing loop. $$$$$
      kount=0
      do 20 i=istart,maxgen+istart-1
        write (6,1111) i
        write (24,1111) i
        write(24,1050)
c
c  Evaluate the population, assign fitness, establish the best
c  individual, and write output information.
        call evalout(iskip,iend,ibest,fbar,best)
        geni(i)=float(i)
        genavg(i)=fbar
        genmax(i)=best
        if(npopsiz.eq.1 .or. iskip.ne.0) then
          close(24)
          stop
        endif
c
c  Implement "niching".
        if (iniche.ne.0) call niche
c
c  Enter selection, crossover and mutation loop.
        ncross=0
        ipick=npopsiz
        do 45 j=1,npopsiz,nchild
c
c  Perform selection.
```

```
          call selectn(ipick,j,mate1,mate2)
c
c  Now perform crossover between the randomly selected pair.
          call crosovr(ncross,j,mate1,mate2)
 45      continue
        write(6,1225) ncross
        write(24,1225) ncross
c
c  Now perform random mutations.  If running micro-GA, skip mutation.
        if (microga.eq.0) call mutate
c
c  Write child array back into parent array for new generation.  Check
c  to see if the best parent was replicated.
        call newgen(ielite,npossum,ig2sum,ibest)
c
c  Implement micro-GA if enabled.
        if (microga.ne.0) call gamicro(i,npossum,ig2sum,ibest)
c
c  Write to restart file.
        call restart(i,istart,kount)
 20   continue
c  $$$$$ End of main generational processing loop. $$$$$
c 999  continue
           open(unit=1,file='summary.dat')
           write(1,3050)
      write(24,3000)
      do 100 i=istart,maxgen+istart-1
         evals=float(npopsiz)*geni(i)
         write(24,3100) geni(i),evals,genavg(i),genmax(i)
              unif=1000-genmax(i)
              write(1,3150) geni(i),genavg(i),genmax(i),unif

 100  continue
           close(unit=1)

c      call etime(tarray)
c      write(6,*) tarray(1),tarray(2)
c      cpu1=tarray(1)
c      cpu=(cpu1-cpu0)
c      write(6,1400) cpu,cpu/60.0
c      write(24,1400) cpu,cpu/60.0
      CLOSE (24)
c
 1050 format(1x,4x,'   Dv1     Dv2     Dv3     Dv4     Dv5     Dv6     D
     *v7     Dv8     Fitness')
 1111 format(//'################################  Generation',i5,'  ######
     *############################')
 1225 format(/'  Number of Crossovers      =',i5)
c 1400 format(2x,'CPU time for all generations=',e12.6,' sec'/
c     +       2x,'                            ',e12.6,' min')
 3000 format(2x//'Summary of Output'/
     +       2x,'Generation   Evaluations   Avg.Fitness   Best Fitness')
 3050    format(2x,'Generazione Avg.Fitness Best Fitness        Unif')
 3100 format(2x,3(e10.4,4x),e11.5)
 3150    format(2x,(f7.0,6x),2(f8.4,6x),f7.2)
c
      stop
      end
```

```
c
c#############################################################################
      subroutine input
c
c  This subroutine inputs information from the ga.inp (gafort.in) file.
c
      implicit real*8 (a-h,o-z)
      save
c
      include 'params.f'
      dimension nposibl(nparmax),nichflg(nparmax)
      dimension parmax(nparmax),parmin(nparmax),pardel(nparmax)
c
      common / ga1   / npopsiz,nowrite
      common / ga2   / nparam,nchrome
      common / ga6   / parmax,parmin,pardel,nposibl
      common / ga8   / nichflg
      common /inputga/ pcross,pmutate,pcreep,maxgen,idum,irestrt,
     +                 itourny,ielite,icreep,iunifrm,iniche,
     +                 iskip,iend,nchild,microga,kountmx
c
      namelist / ga    / irestrt,npopsiz,pmutate,maxgen,idum,pcross,
     +                   itourny,ielite,icreep,pcreep,iunifrm,iniche,
     +                   iskip,iend,nchild,nparam,parmin,parmax,nposibl,
     +                   nowrite,nichflg,microga,kountmx
c
      kountmx=5
      irestrt=0
      itourny=0
      ielite=0
      iunifrm=0
      iniche=0
      iskip=0
      iend=0
      nchild=1
      do 2 i=1,nparam
        nichflg(i)=1
 2    continue
      microga=0
c
      OPEN (UNIT=24, FILE='ga.out', STATUS='UNKNOWN')
      rewind 24
      OPEN (UNIT=23, FILE='ga.inp', STATUS='OLD')
      READ (23, NML = ga)
      CLOSE (23)
      itourny=1
c      if (itourny.eq.0) nchild=2
c
c  Check for array sizing errors.
      if (npopsiz.gt.indmax) then
        write(6,1600) npopsiz
        write(24,1600) npopsiz
        close(24)
        stop
      endif
      if (nparam.gt.nparmax) then
        write(6,1700) nparam
        write(24,1700) nparam
```

```
        close(24)
        stop
      endif
c
c  If using the microga option, reset some input variables
      if (microga.ne.0) then
        pmutate=0.0d0
        pcreep=0.0d0
        itourny=1
        ielite=1
        iniche=0
        nchild=1
        if (iunifrm.eq.0) then
          pcross=1.0d0
        else
          pcross=0.5d0
        endif
      endif
c
 1600 format(1x,'ERROR: npopsiz > indmax.  Set indmax = ',i6)
 1700 format(1x,'ERROR: nparam > nparmax.  Set nparmax = ',i6)
c
      return
      end
c
c*************************************************************************
      subroutine initial(istart,npossum,ig2sum)
c
c  This subroutine sets up the program by generating the g0, g1 and
c  ig2 arrays, and counting the number of chromosomes required for the
c  specified input.  The subroutine also initializes the random number
c  generator, parent and iparent arrays (reads the ga.restart file).
      implicit real*8 (a-h,o-z)
      save
c
      include 'params.f'
      dimension parent(nparmax,indmax),iparent(nchrmax,indmax)
      dimension nposibl(nparmax)
      dimension g0(nparmax),g1(nparmax),ig2(nparmax)
      dimension parmax(nparmax),parmin(nparmax),pardel(nparmax)
c
      common / ga1   / npopsiz,nowrite
      common / ga2   / nparam,nchrome
      common / ga3   / parent,iparent
      common / ga5   / g0,g1,ig2
      common / ga6   / parmax,parmin,pardel,nposibl
      common /inputga/ pcross,pmutate,pcreep,maxgen,idum,irestrt,
     +                 itourny,ielite,icreep,iunifrm,iniche,
     +                 iskip,iend,nchild,microga,kountmx
c
c
      do 3 i=1,nparam
        g0(i)=parmin(i)
        pardel(i)=parmax(i)-parmin(i)
        g1(i)=pardel(i)/dble(nposibl(i)-1)
 3    continue
      do 6 i=1,nparam
        do 7 j=1,30
```

```
          n2j=2**j
          if (n2j.ge.nposibl(i)) then
            ig2(i)=j
            goto 8
          endif
          if (j.ge.30) then
            write(6,2000)
            write(24,2000)
            close(24)
            stop
          endif
 7      continue
 8      continue
 6    continue
c
c  Count the total number of chromosomes (bits) required
      nchrome=0
      npossum=0
      ig2sum=0
      do 9 i=1,nparam
        nchrome=nchrome+ig2(i)
        npossum=npossum+nposibl(i)
        ig2sum=ig2sum+(2**ig2(i))
 9    continue
      if (nchrome.gt.nchrmax) then
        write(6,1800) nchrome
        write(24,1800) nchrome
        close(24)
        stop
      endif
c
      if (npossum.lt.ig2sum .and. microga.ne.0) then
        write(6,2100)
        write(24,2100)
      endif
c
c  Initialize random number generator
      call ran3(idum,rand)
c
      if(irestrt.eq.0) then
c  Initialize the random distribution of parameters in the individual
c  parents when irestrt=0.
        istart=1
        do 10 i=1,npopsiz
          do 15 j=1,nchrome
            call ran3(1,rand)
            iparent(j,i)=1
            if(rand.lt.0.5d0) iparent(j,i)=0
 15       continue
 10     continue
        if (npossum.lt.ig2sum) call possibl(parent,iparent)
      else
c  If irestrt.ne.0, read from restart file.
        OPEN (UNIT=25, FILE='ga.restart', STATUS='OLD')
        rewind 25
        read(25,*) istart,npopsiz
        do 1 j=1,npopsiz
          read(25,*) k,(iparent(l,j),l=1,nchrome)
```

```
 1      continue
        CLOSE (25)
      endif
c
      if(irestrt.ne.0) call ran3(idum-istart,rand)
c
 1800 format(1x,'ERROR: nchrome > nchrmax.  Set nchrmax = ',i6)
 2000 format(1x,'ERROR: You have a parameter with a number of '/
     +       1x,'   possibilities > 2**30!  If you really desire this,'/
     +       1x,'   change the DO loop 7 statement and recompile.'//
     +       1x,'   You may also need to alter the code to work with'/
     +       1x,'   REAL numbers rather than INTEGER numbers; Fortran'/
     +       1x,'   does not like to compute 2**j when j>30.')
 2100 format(1x,'WARNING: for some cases, a considerable performance'/
     +       1x,'   reduction has been observed when running a non-'/
     +       1x,'   optimal number of bits with the micro-GA.'/
     +       1x,'   If possible, use values for nposibl of 2**n,'/
     +       1x,'   e.g. 2, 4, 8, 16, 32, 64, etc.  See ReadMe file.')
c
      return
      end
c
c#############################################################################
      subroutine evalout(iskip,iend,ibest,fbar,best)
c
c  This subroutine evaluates the population, assigns fitness,
c  establishes the best individual, and outputs information.
      implicit real*8 (a-h,o-z)
      save
c
      include 'params.f'
      dimension parent(nparmax,indmax),iparent(nchrmax,indmax)
      dimension fitness(indmax)
      dimension paramsm(nparmax),paramav(nparmax),ibest(nchrmax)
c
      common / ga1   / npopsiz,nowrite
      common / ga2   / nparam,nchrome
      common / ga3   / parent,iparent
      common / ga4   / fitness
c
      fitsum=0.0d0
      best=-1.0d10
      do 29 n=1,nparam
        paramsm(n)=0.0d0
 29   continue
      jstart=1
      jend=npopsiz
      if(iskip.ne.0) jstart=iskip
      if(iend.ne.0) jend=iend
      do 30 j=jstart,jend
        call decode(j,parent,iparent)
        if(iskip.ne.0 .and. iend.ne.0 .and. iskip.eq.iend)
     +   write(6,1075) j,
     +                 (parent(kk,j),kk=1,nparam),0.0
c
c  Call function evaluator, write out individual and fitness, and add
c  to the summation for later averaging.
        call func(j,funcval)
```

```
        fitness(j)=funcval
        write(24,1075) j,
     +                 (parent(kk,j),kk=1,nparam),fitness(j)
        fitsum=fitsum+fitness(j)
        do 22 n=1,nparam
          paramsm(n)=paramsm(n)+parent(n,j)
 22     continue
c
c  Check to see if fitness of individual j is the best fitness.
        if (fitness(j).gt.best) then
          best=fitness(j)
          jbest=j
          do 24 k=1,nchrome
            ibest(k)=iparent(k,j)
 24       continue
        endif
 30   continue
c
c  Compute parameter and fitness averages.
      fbar=fitsum/dble(npopsiz)
      do 23 n=1,nparam
        paramav(n)=paramsm(n)/dble(npopsiz)
 23   continue
c
c  Write output information
      if (npopsiz.eq.1) then
        write(24,1075) 1,
     +                 (parent(k,1),k=1,nparam),fitness(1)
c         write(24,*) 'Average Values:'
        write(24,1275) (parent(k,1),k=1,nparam),fbar
      else
        write(24,1275) (paramav(k),k=1,nparam),fbar
      endif
      write(6,1100) fbar
      write(24,1100) fbar
      write(6,1200) best
      write(24,1200) best
c
 1075 format(i3,1x,8(1x,f7.4),3x,f8.2)
 1100 format(1x,' Average Function Value of Generation= ',f8.2)
 1200 format(1x,' Maximum Function Value              = ',f8.2/)
 1275 format(/'Average Values:',1x,8(1x,f7.4),2x,f8.2/)
      return
      end
c
c*************************************************************************
      subroutine niche
c
c  Implement "niching" through Goldberg's multidimensional phenotypic
c  sharing scheme with a triangular sharing function.  To find the
c  multidimensional distance from the best individual, normalize all
c  parameter differences.
c
      implicit real*8 (a-h,o-z)
      save
c
      include 'params.f'
      dimension parent(nparmax,indmax),iparent(nchrmax,indmax)
```

```
      dimension fitness(indmax),nposibl(nparmax),nichflg(nparmax)
      dimension parmax(nparmax),parmin(nparmax),pardel(nparmax)
c
      common / ga1   / npopsiz,nowrite
      common / ga2   / nparam,nchrome
      common / ga3   / parent,iparent
      common / ga4   / fitness
      common / ga6   / parmax,parmin,pardel,nposibl
      common / ga8   / nichflg
c
c   Variable definitions:
c
c  alpha   = power law exponent for sharing function; typically = 1.0
c  del     = normalized multidimensional distance between ii and all
c            other members of the population
c            (equals the square root of del2)
c  del2    = sum of the squares of the normalized multidimensional
c            distance between member ii and all other members of
c            the population
c  nniche  = number of niched parameters
c  sigshar = normalized distance to be compared with del; in some sense,
c            1/sigshar can be viewed as the number of regions over which
c            the sharing function should focus, e.g. with sigshar=0.1,
c            the sharing function will try to clump in ten distinct
c            regions of the phase space.  A value of sigshar on the
c            order of 0.1 seems to work best.
c  share   = sharing function between individual ii and j
c  sumshar = sum of the sharing functions for individual ii
c
c     alpha=1.0
      sigshar=0.1d0
      nniche=0
      do 33 jj=1,nparam
        nniche=nniche+nichflg(jj)
 33   continue
      if (nniche.eq.0) then
        write(6,1900)
        write(24,1900)
        close(24)
        stop
      endif
      do 34 ii=1,npopsiz
        sumshar=0.0d0
        do 35 j=1,npopsiz
          del2=0.0d0
          do 36 k=1,nparam
            if (nichflg(k).ne.0) then
              del2=del2+((parent(k,j)-parent(k,ii))/pardel(k))**2
            endif
 36         continue
          del=(dsqrt(del2))/dble(nniche)
          if (del.lt.sigshar) then
c             share=1.0-((del/sigshar)**alpha)
            share=1.0d0-(del/sigshar)
          else
            share=0.0d0
          endif
          sumshar=sumshar+share/dble(npopsiz)
```

```
 35      continue
         if (sumshar.ne.0.0d0) fitness(ii)=fitness(ii)/sumshar
 34    continue
c
 1900 format(1x,'ERROR: iniche=1 and all values in nichflg array = 0'/
     +       1x,'       Do you want to niche or not?')
c
      return
      end
c
c#######################################################################
      subroutine selectn(ipick,j,mate1,mate2)
c
c  Subroutine for selection operator.  Presently, tournament selection
c  is the only option available.
c
      implicit real*8 (a-h,o-z)
      save
c
      include 'params.f'
      dimension parent(nparmax,indmax),child(nparmax,indmax)
      dimension fitness(indmax)
      dimension iparent(nchrmax,indmax),ichild(nchrmax,indmax)
c
      common / ga1   / npopsiz,nowrite
      common / ga2   / nparam,nchrome
      common / ga3   / parent,iparent
      common / ga4   / fitness
      common / ga7   / child,ichild
      common /inputga/ pcross,pmutate,pcreep,maxgen,idum,irestrt,
     +                 itourny,ielite,icreep,iunifrm,iniche,
     +                 iskip,iend,nchild,microga,kountmx
c
c  If tournament selection is chosen (i.e. itourny=1), then
c  implement "tournament" selection for selection of new population.
      if(itourny.eq.1) then
         call select(mate1,ipick)
         call select(mate2,ipick)
c        write(3,*) mate1,mate2,fitness(mate1),fitness(mate2)
         do 46 n=1,nchrome
            ichild(n,j)=iparent(n,mate1)
            if(nchild.eq.2) ichild(n,j+1)=iparent(n,mate2)
 46      continue
      endif
c
      return
      end
c
c#######################################################################
      subroutine crosovr(ncross,j,mate1,mate2)
c
c  Subroutine for crossover between the randomly selected pair.
      implicit real*8 (a-h,o-z)
      save
c
      include 'params.f'
      dimension parent(nparmax,indmax),child(nparmax,indmax)
      dimension iparent(nchrmax,indmax),ichild(nchrmax,indmax)
```

```
c
      common / ga2   / nparam,nchrome
      common / ga3   / parent,iparent
      common / ga7   / child,ichild
      common /inputga/ pcross,pmutate,pcreep,maxgen,idum,irestrt,
     +                 itourny,ielite,icreep,iunifrm,iniche,
     +                 iskip,iend,nchild,microga,kountmx
c
      if (iunifrm.eq.0) then
c  Single-point crossover at a random chromosome point.
        call ran3(1,rand)
        if(rand.gt.pcross) goto 69
        ncross=ncross+1
        call ran3(1,rand)
        icross=2+dint(dble(nchrome-1)*rand)
        do 50 n=icross,nchrome
          ichild(n,j)=iparent(n,mate2)
          if(nchild.eq.2) ichild(n,j+1)=iparent(n,mate1)
 50     continue
      else
c  Perform uniform crossover between the randomly selected pair.
        do 60 n=1,nchrome
          call ran3(1,rand)
          if(rand.le.pcross) then
            ncross=ncross+1
            ichild(n,j)=iparent(n,mate2)
            if(nchild.eq.2) ichild(n,j+1)=iparent(n,mate1)
          endif
 60     continue
      endif
 69   continue
c
      return
      end
c
c#########################################################################
      subroutine mutate
c
      implicit real*8 (a-h,o-z)
      save
c
      include 'params.f'
      dimension nposibl(nparmax)
      dimension child(nparmax,indmax),ichild(nchrmax,indmax)
      dimension g0(nparmax),g1(nparmax),ig2(nparmax)
      dimension parmax(nparmax),parmin(nparmax),pardel(nparmax)
c
      common / ga1   / npopsiz,nowrite
      common / ga2   / nparam,nchrome
      common / ga5   / g0,g1,ig2
      common / ga6   / parmax,parmin,pardel,nposibl
      common / ga7   / child,ichild
      common /inputga/ pcross,pmutate,pcreep,maxgen,idum,irestrt,
     +                 itourny,ielite,icreep,iunifrm,iniche,
     +                 iskip,iend,nchild,microga,kountmx
c
c  This subroutine performs mutations on the children generation.
c  Perform random jump mutation if a random number is less than pmutate.
```

```
c  Perform random creep mutation if a different random number is less
c  than pcreep.
      nmutate=0
      ncreep=0
      do 70 j=1,npopsiz
        do 75 k=1,nchrome
c  Jump mutation
          call ran3(1,rand)
          if (rand.le.pmutate) then
            nmutate=nmutate+1
            if(ichild(k,j).eq.0) then
              ichild(k,j)=1
            else
              ichild(k,j)=0
            endif
            if (nowrite.eq.0) write(6,1300) j,k
            if (nowrite.eq.0) write(24,1300) j,k
          endif
 75     continue
c  Creep mutation (one discrete position away).
        if (icreep.ne.0) then
          do 76 k=1,nparam
            call ran3(1,rand)
            if(rand.le.pcreep) then
              call decode(j,child,ichild)
              ncreep=ncreep+1
              creep=1.0d0
              call ran3(1,rand)
              if (rand.lt.0.5d0) creep=-1.0d0
              child(k,j)=child(k,j)+g1(k)*creep
              if (child(k,j).gt.parmax(k)) then
                child(k,j)=parmax(k)-1.0d0*g1(k)
              elseif (child(k,j).lt.parmin(k)) then
                child(k,j)=parmin(k)+1.0d0*g1(k)
              endif
              call code(j,k,child,ichild)
              if (nowrite.eq.0) write(6,1350) j,k
              if (nowrite.eq.0) write(24,1350) j,k
            endif
 76       continue
        endif
 70   continue
      write(6,1250) nmutate,ncreep
      write(24,1250) nmutate,ncreep
c
 1250 format(/'  Number of Jump Mutations  =',i5/
     +        '  Number of Creep Mutations =',i5)
 1300 format('*** Jump mutation performed on individual  ',i4,
     +       ', chromosome ',i3,' ***')
 1350 format('*** Creep mutation performed on individual ',i4,
     +       ', parameter  ',i3,' ***')
c
      return
      end
c
c#########################################################################
      subroutine newgen(ielite,npossum,ig2sum,ibest)
c
```

```
c  Write child array back into parent array for new generation.  Check
c  to see if the best parent was replicated; if not, and if ielite=1,
c  then reproduce the best parent into a random slot.
c
      implicit real*8 (a-h,o-z)
      save
c
      include 'params.f'
      dimension parent(nparmax,indmax),child(nparmax,indmax)
      dimension iparent(nchrmax,indmax),ichild(nchrmax,indmax)
      dimension ibest(nchrmax)
c
      common / ga1   / npopsiz,nowrite
      common / ga2   / nparam,nchrome
      common / ga3   / parent,iparent
      common / ga7   / child,ichild
c
      if (npossum.lt.ig2sum) call possibl(child,ichild)
      kelite=0
      do 94 j=1,npopsiz
        jelite=0
        do 95 n=1,nchrome
          iparent(n,j)=ichild(n,j)
          if (iparent(n,j).eq.ibest(n)) jelite=jelite+1
          if (jelite.eq.nchrome) kelite=1
 95     continue
 94   continue
      if (ielite.ne.0 .and. kelite.eq.0) then
        call ran3(1,rand)
        irand=1d0+dint(dble(npopsiz)*rand)
        do 96 n=1,nchrome
          iparent(n,irand)=ibest(n)
 96     continue
        write(24,1260) irand
      endif
c
 1260 format('  Elitist Reproduction on Individual ',i4)
c
      return
      end
c
c*******************************************************************************
      subroutine gamicro(i,npossum,ig2sum,ibest)
c
c  Micro-GA implementation subroutine
c
      implicit real*8 (a-h,o-z)
      save
c
      include 'params.f'
      dimension parent(nparmax,indmax),iparent(nchrmax,indmax)
      dimension ibest(nchrmax)
c
      common / ga1   / npopsiz,nowrite
      common / ga2   / nparam,nchrome
      common / ga3   / parent,iparent
c
c  First, check for convergence of micro population.
```

```
c  If converged, start a new generation with best individual and fill
c  the remainder of the population with new randomly generated parents.
c
c  Count number of different bits from best member in micro-population
      icount=0
      do 81 j=1,npopsiz
        do 82 n=1,nchrome
          if(iparent(n,j).ne.ibest(n)) icount=icount+1
 82     continue
 81   continue
c
c  If icount less than 5% of number of bits, then consider population
c  to be converged.  Restart with best individual and random others.
      diffrac=dble(icount)/dble((npopsiz-1)*nchrome)
      if (diffrac.lt.0.05d0) then
      do 87 n=1,nchrome
        iparent(n,1)=ibest(n)
 87   continue
      do 88 j=2,npopsiz
        do 89 n=1,nchrome
          call ran3(1,rand)
          iparent(n,j)=1
          if(rand.lt.0.5d0) iparent(n,j)=0
 89     continue
 88   continue
      if (npossum.lt.ig2sum) call possibl(parent,iparent)
      write(6,1375) i
      write(24,1375) i
      endif
c
 1375 format(//'%%%%%%%  Restart micro-population at generation',
     +       i5,'  %%%%%%%')
c
      return
      end
c
c##############################################################################
      subroutine select(mate,ipick)
c
c  This routine selects the better of two possible parents for mating.
c
      implicit real*8 (a-h,o-z)
      save
c
      include 'params.f'
      common / ga1   / npopsiz,nowrite
      common / ga2   / nparam,nchrome
      common / ga3   / parent,iparent
      common / ga4   / fitness
      dimension parent(nparmax,indmax),iparent(nchrmax,indmax)
      dimension fitness(indmax)
c
      if(ipick+1.gt.npopsiz) call shuffle(ipick)
      ifirst=ipick
      isecond=ipick+1
      ipick=ipick+2
      if(fitness(ifirst).gt.fitness(isecond)) then
        mate=ifirst
```

```
      else
        mate=isecond
      endif
c     write(3,*)'select',ifirst,isecond,fitness(ifirst),fitness(isecond)
c
      return
      end
c
c*************************************************************************
      subroutine shuffle(ipick)
c
c  This routine shuffles the parent array and its corresponding fitness
c
      implicit real*8 (a-h,o-z)
      save
c
      include 'params.f'
      common / ga1   / npopsiz,nowrite
      common / ga2   / nparam,nchrome
      common / ga3   / parent,iparent
      common / ga4   / fitness
      dimension parent(nparmax,indmax),iparent(nchrmax,indmax)
      dimension fitness(indmax)
c
      ipick=1
      do 10 j=1,npopsiz-1
        call ran3(1,rand)
        iother=j+1+dint(dble(npopsiz-j)*rand)
        do 20 n=1,nchrome
          itemp=iparent(n,iother)
          iparent(n,iother)=iparent(n,j)
          iparent(n,j)=itemp
 20     continue
        temp=fitness(iother)
        fitness(iother)=fitness(j)
        fitness(j)=temp
 10   continue
c
      return
      end
c
c*************************************************************************
      subroutine decode(i,array,iarray)
c
c  This routine decodes a binary string to a real number.
c
      implicit real*8 (a-h,o-z)
      save
c
      include 'params.f'
      common / ga2   / nparam,nchrome
      common / ga5   / g0,g1,ig2
      dimension array(nparmax,indmax),iarray(nchrmax,indmax)
      dimension g0(nparmax),g1(nparmax),ig2(nparmax)
c
      l=1
      do 10 k=1,nparam
        iparam=0
```

```
        m=l
        do 20 j=m,m+ig2(k)-1
          l=l+1
          iparam=iparam+iarray(j,i)*(2**(m+ig2(k)-1-j))
 20      continue
        array(k,i)=g0(k)+g1(k)*dble(iparam)
 10   continue
c
      return
      end
c
c#########################################################################
      subroutine code(j,k,array,iarray)
c
c  This routine codes a parameter into a binary string.
c
      implicit real*8 (a-h,o-z)
      save
c
      include 'params.f'
      common / ga2   / nparam,nchrome
      common / ga5   / g0,g1,ig2
      dimension array(nparmax,indmax),iarray(nchrmax,indmax)
      dimension g0(nparmax),g1(nparmax),ig2(nparmax)
c
c  First, establish the beginning location of the parameter string of
c  interest.
      istart=1
      do 10 i=1,k-1
        istart=istart+ig2(i)
 10   continue
c
c  Find the equivalent coded parameter value, and back out the binary
c  string by factors of two.
      m=ig2(k)-1
      if (g1(k).eq.0.0d0) return
      iparam=nint((array(k,j)-g0(k))/g1(k))
      do 20 i=istart,istart+ig2(k)-1
        iarray(i,j)=0
        if ((iparam+1).gt.(2**m)) then
          iarray(i,j)=1
          iparam=iparam-2**m
        endif
        m=m-1
 20   continue
c     write(3,*)array(k,j),iparam,(iarray(i,j),i=istart,istart+ig2(k)-1)
c
      return
      end
c
c#########################################################################
c
      subroutine possibl(array,iarray)
c
c  This subroutine determines whether or not all parameters are within
c  the specified range of possibility.  If not, the parameter is
c  randomly reassigned within the range.  This subroutine is only
c  necessary when the number of possibilities per parameter is not
```

```
c  optimized to be 2**n, i.e. if npossum < ig2sum.
c
      implicit real*8 (a-h,o-z)
      save
c
      include 'params.f'
      common / ga1   / npopsiz,nowrite
      common / ga2   / nparam,nchrome
      common / ga5   / g0,g1,ig2
      common / ga6   / parmax,parmin,pardel,nposibl
      dimension array(nparmax,indmax),iarray(nchrmax,indmax)
      dimension g0(nparmax),g1(nparmax),ig2(nparmax),nposibl(nparmax)
      dimension parmax(nparmax),parmin(nparmax),pardel(nparmax)
c
      do 10 i=1,npopsiz
        call decode(i,array,iarray)
        do 20 j=1,nparam
          n2ig2j=2**ig2(j)
          if(nposibl(j).ne.n2ig2j .and. array(j,i).gt.parmax(j)) then
            call ran3(1,rand)
            irand=dint(dble(nposibl(j))*rand)
            array(j,i)=g0(j)+dble(irand)*g1(j)
            call code(i,j,array,iarray)
            if (nowrite.eq.0) write(6,1000) i,j
            if (nowrite.eq.0) write(24,1000) i,j
          endif
 20     continue
 10   continue
c
 1000 format('*** Parameter adjustment to individual     ',i4,
     +       ', parameter  ',i3,' ***')
c
      return
      end
c
c##############################################################################
      subroutine restart(i,istart,kount)
c
c  This subroutine writes restart information to the ga.restart file.
c
      implicit real*8 (a-h,o-z)
      save
c
      include 'params.f'
      common / ga1   / npopsiz,nowrite
      common / ga2   / nparam,nchrome
      common / ga3   / parent,iparent
      dimension parent(nparmax,indmax),iparent(nchrmax,indmax)
      common /inputga/ pcross,pmutate,pcreep,maxgen,idum,irestrt,
     +                 itourny,ielite,icreep,iunifrm,iniche,
     +                 iskip,iend,nchild,microga,kountmx

      kount=kount+1
      if(i.eq.maxgen+istart-1 .or. kount.eq.kountmx) then
        OPEN (UNIT=25, FILE='ga.restart', STATUS='OLD')
        rewind 25
        write(25,*) i+1,npopsiz
        do 80 j=1,npopsiz
```

```
          write(25,1500) j,(iparent(l,j),l=1,nchrome)
 80      continue
        CLOSE (25)
        kount=0
      endif
c
 1500 format(i5,3x,120i2)
c
      return
      end
c
c*******************************************************************************
      subroutine ran3(idum,rand)
c
c  Returns a uniform random deviate between 0.0 and 1.0.  Set idum to
c  any negative value to initialize or reinitialize the sequence.
c  This function is taken from W.H. Press', "Numerical Recipes" p. 199.
c
      implicit real*8 (a-h,m,o-z)
      save
c     implicit real*4(m)
      parameter (mbig=4000000.,mseed=1618033.,mz=0.,fac=1./mbig)
c     parameter (mbig=1000000000,mseed=161803398,mz=0,fac=1./mbig)
c
c  According to Knuth, any large mbig, and any smaller (but still large)
c  mseed can be substituted for the above values.
      dimension ma(55)
      data iff /0/
      if (idum.lt.0 .or. iff.eq.0) then
        iff=1
        mj=mseed-dble(iabs(idum))
        mj=dmod(mj,mbig)
        ma(55)=mj
        mk=1
        do 11 i=1,54
          ii=mod(21*i,55)
          ma(ii)=mk
          mk=mj-mk
          if(mk.lt.mz) mk=mk+mbig
          mj=ma(ii)
 11     continue
        do 13 k=1,4
          do 12 i=1,55
            ma(i)=ma(i)-ma(1+mod(i+30,55))
            if(ma(i).lt.mz) ma(i)=ma(i)+mbig
 12       continue
 13     continue
        inext=0
        inextp=31
        idum=1
      endif
      inext=inext+1
      if(inext.eq.56) inext=1
      inextp=inextp+1
      if(inextp.eq.56) inextp=1
      mj=ma(inext)-ma(inextp)
      if(mj.lt.mz) mj=mj+mbig
      ma(inext)=mj
```

```
      rand=mj*fac
      return
      end
c
c#############################################################################
c
      subroutine func(j,funcval)
      USE PORTLIB

c
      implicit real*8 (a-h,o-z)
      save
c
      include 'params.f'
      INTEGER(4) I, errnum
      dimension parent(nparmax,indmax)
      dimension iparent(nchrmax,indmax)
c     dimension parent2(indmax,nparmax),iparent2(indmax,nchrmax)
c
      common / ga2   / nparam,nchrome
      common / ga3   / parent,iparent

10    format(e12.6)

            open(unit=1,file='inpansy.dat')
            do i=1,nparam
            write(1,10)         parent(i,j)
            enddo
            close(unit=1)

      I = SYSTEM("c:\ansys55\bin\intel\ansys55.exe -m 80 -p ANSYSRF -i g
            *enetic.mac -o out.txt ")
      If (I .eq. -1) then
            errnum = ierrno()
        print *, 'Error ', errnum
      end if

            funcval=0.
            open(unit=1,file='outansy.dat')
            read(1,*) funcval
            close(unit=1)

      return
      end
c#############################################################################
```

Appendice B

Listato del file di input di ANSYS che implementa l'algoritmo genetico.

```
/units,si
/filname,genetic

/PREP7

et,1,plane13,,,1
emunits,mks

mp,murx,1,1
aria=1
mp,murx,2,1
bobina=2

*dim,corr,,8
*dim,f,,1

box=2.5
ff =5.e-3*1.5
ri =1.6

fact=1.0      ! fattore moltiplicativo densita' correnti
              ! (trasla la curva del campo)
```

```
/input,inpansy,dat

curr1 = dv1*1e8
curr2 = dv2*1e8
curr3 = dv3*1e8
curr4 = dv4*1e8
curr5 = dv5*1e8
curr6 = dv6*1e8

curr7 =0



coox1= dv7
coox2= dv8

coox3=0.24728
coox4=0.2071
coox5=0.1671
coox6=0.1179
coox7=0.0369

cooy1=0
cooy2=0
cooy3=0.082
cooy4=0.14817
cooy5=0.1989
cooy6=0.2532
cooy7=cooy6

dx1=0.055
```

```
dx2=dx1      !
dx3=0.06
dx4=0.05
dx5=0.07
dx6=0.043
dx7=0.05

*dim,dx,,7
dx(1)=dx1,dx1,dx3,dx4,dx5,dx6,dx7

dy1=0.051
dy2=dy1
dy3=0.04
dy4=0.03
dy5=0.025
dy6=0.032
dy7=0.032

*dim,dy,,7
dy(1)=dy1,dy2,dy3,dy4,dy5,dy6,dy7

*dim,coox,,7
coox(1)=coox1
coox(2)=coox2    ! posizione bobina esterna
coox(3)=coox3
coox(4)=coox4
coox(5)=coox5
coox(6)=coox6
coox(7)=coox7

*dim,cooy,,7
cooy(1)=0
```

```
cooy(2)=0
cooy(3)=cooy3
cooy(4)=cooy4
cooy(5)=cooy5
cooy(6)=cooy6
cooy(7)=cooy6

local,11,0,ri,0
csys,11
wpcsys,-1,11

*do,i,1,7
  rectng,coox(i),coox(i)+dx(i),cooy(i),cooy(i)+dy(i)
*enddo

rectng,0,coox(1),0,dy1
rectng,coox(1)+dx1,coox(2),0,dy1
rectng,0,coox(2)+dx(2),dy1,cooy(3)

*do,i,3,5
  rectng,0,coox(i),cooy(i),cooy(i)+dy(i)
*enddo

rectng,0,coox(7),cooy(7),cooy(7)+dy(7)
rectng,coox(7)+dx(7),coox(6),cooy(6),cooy(6)+dy(6)

*do,i,3,6
  rectng,coox(i)+dx(i),coox(2)+dx(2),cooy(i),cooy(i)+dy(i)
*enddo

*do,i,3,5
  rectng,0,coox(2)+dx(2),cooy(i)+dy(i),cooy(i+1)
```

```
*enddo

aglue,all
nummrg,kp
numcmp,kp
numcmp,line
numcmp,area

mshkey,1
mshape,0,2d
esize,ff

mat,bobina
amesh,1,7

*do,i,2,5
  lsel,s,loc,y,cooy(i)+dy(i)
  lccat,all
  lsel,all
*enddo

*do,i,3,6
  lsel,s,loc,y,cooy(i)
  lccat,all
  lsel,all
*enddo

mat,aria
amesh,8,22

lsel,s,lcca
ldele,all
```

```
lsel,all

rectng,0,coox(2)+dx(2),cooy(7)+dy(7),box    !
rectng,coox(2)+dx(2),box,0,cooy(7)+dy(7)    ! Aree Esterne
rectng,coox(2)+dx(2),box,cooy(7)+dy(7),box  !

allsel
aglue,all


lsel,s,loc,y,cooy(7)+dy(7)
lsel,r,loc,x,0,coox(2)+dx(2)
lccat,all

lsel,all

lsel,s,loc,x,coox(2)+dx(2)
lsel,r,loc,y,0,cooy(7)+dy(7)
lccat,all

lsel,all

lsel,s,line,,80,81
lesize,all,10*ff,,,10,1

lsel,s,line,,84,85
lesize,all,10*ff,,,10,1
lsel,all

mat,aria
amesh,26,28
```

allsel

asel,s,loc,x,0,coox(2)+dx(2)

arsym,x,all
allsel
nummrg,node,1e-6
nummrg,kp
numcmp,node
numcmp,kp
numcmp,line
numcmp,area

lsel,s,lcca
ldele,all
lsel,all

rectng,-ri,-(coox(2)+dx1),0,cooy(7)+dy(7)
rectng,-ri,-(coox(2)+dx1),cooy(7)+dy(7),box
aglue,all

lsel,s,line,,144
lesize,all,10*ff,,,10,1

lsel,s,line,,147
lesize,all,10*ff,,,10,1

lsel,s,loc,x,-(coox(2)+dx1)
lsel,r,loc,y,0,cooy(7)+dy(7)
lccat,all
lsel,all

mat,aria

```
amesh,51,52


csys,0
wpcsys,-1,0

/pnum,mat,1
/num,1

finish


/SOLU !           *** SOLU ***

nsel,s,loc,x,box+ri
nsel,a,loc,y,box
d,all,az,0
nsel,all

*dim,currdi,,7
currdi(1)=curr1,curr2,curr3,curr4,curr5,curr6,curr7

*do,i,3,7
  asel,s,area,,i
  esla,s
  bfe,all,js,3,currdi(i)
*enddo

allsel

*do,i,4,7
  asel,s,area,,25+i
```

```
  esla,s
  bfe,all,js,3,-currdi(i)
*enddo

asel,s,area,,25
esla,s
bfe,all,js,3,-curr3

asel,s,area,,23          !
esla,s                !
bfe,all,js,3,-curr2      !
asel,s,area,,24          ! bobina sinistra
esla,s                !
bfe,all,js,3,curr2       !

asel,s,area,,1           !
esla,s                !
bfe,all,js,3,curr1       ! bobina destra
asel,s,area,,2           !
esla,s                !
bfe,all,js,3,-curr1      !


allsel
magsolv,0
finish

/post1 !          *** POST1 ***

nnnx  =ri
deltax=10.e-2
deltay=3.e-2
```

```
nsel,s,loc,x,nnnx-deltax,nnnx+deltax
nsel,r,loc,y,0,deltay
nsort,b,sum

*get,bmax,sort,0,max
*get,bmin,sort,0,min

csys,11
nsel,s,loc,x,0
nsel,r,loc,y,0
*get,nnode,node,0,num,max
*get,campo,node,nnode,b,sum
csys,0

unif=(bmax-bmin)/campo
unif=unif+abs(campo-4)/campo
allsel

nsort,b,sum
*get,cmax,sort,0,max
allsel

*if,cmax,gt,6,then       ! funzione di penalita'
  penalty=cmax-5        ! per il vincolo sul campo max
  unif=unif*penalty
*endif

f(1)=1/unif

/input,uscita,inp
```

```
finish

/exit,nosa
```

Bibliografia

[1] Gen M., Cheng R., *Genetic algorithms and engineering design*, New York Wiley, 1997.

[2] Goldberg D., Genetic algorithms in search, optimization, and machine learning, Addison-Wesley , 1989.

[3] Mitchel M., *Introduzione agli Algoritmi Genetici*, Apogeo, 1999.

[4] Mosca R., Giribone P., *Metodi gradientali per la ricerca dell'ottimo negli esperimenti progettati di sistemi fisici*, Quaderni di gestione degli impianti industriali, Istituto di tecnologie e impianti meccanici, 1984, Genova.

[5] Mosca R., Giribone P., *Teoria degli esperimenti e simulazione*, Quaderni di gestione degli impianti industriali, Istituto di tecnologie e impianti meccanici, 1985, Genova.

[6] Sushil J. L., *Genetic algorithms as a computational tool for design*, Ph.D. thesis, Department of Computer Science, Indiana University.