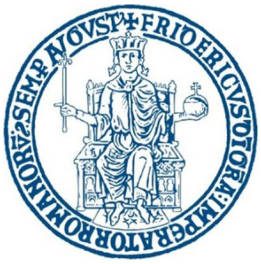
UNIVERSITÀ DEGLI STUDI DI NAPOLI
"FEDERICO II"



Scuola Politecnica e delle Scienze di Base

Area Didattica di Scienze Matematiche Fisiche e Naturali

Dipartimento di Fisica "Ettore Pancini"

Laurea Triennale in Fisica

Sull'impiego di algoritmi genetici per lo sviluppo perturbativo in teoria dei campi

Relatori:
Prof. Francesco Tramontano
Dott. Luca Buonocore

Candidato:
Roberta Viscardi
Matr. N85000841

Anno Accademico 2018/2019

Indice

Introduzione **i**

1 Calcolo perturbativo in TQC **1**
1.1 L'ampiezza di scattering 1
1.2 Il Modello Standard 3
1.3 La fisica di precisione 4
1.4 Il problema di calcolo 6

2 Algoritmi Genetici **8**
2.1 Preparazione 9
2.2 Selezione 9
2.3 Crossover 10
2.4 Mutazione 11
2.5 Cenni teorici 12

3 Impiego e risultati **14**
3.1 Il sistema di equazioni 14
3.2 La funzione fitness 15
3.3 L'applicazione 16
3.3.1 CICLI EFFETTUATI (CE) vs PERCENTUALE (%) . . 17
3.3.2 CICLI EFFETTUATI (CE) vs POOL INIZIALE (NI) . . 20
3.3.3 CICLI EFFETTUATI (CE) vs BIT (L) 21
3.3.4 Il successo a dodici variabili 23
3.3.5 Un problema NP-hard. 24

A SIMPLE 1 e SIMPLE 2 **27**

Bibliografia **29**

Introduzione

Nel seguente elaborato di tesi si propone un approccio di tipo numerico per la risoluzione di un particolare problema matematico nell'ambito della *teoria quantistica dei campi*. Il fenomeno fisico a cui si fa riferimento è il *decadimento* di un bosone di Higgs nella coppia del quark bottom (b) e della sua antiparticella ($\bar{\text{b}}$), mentre lo strumento di indagine impiegato è un *algoritmo genetico*. Come discusso nel primo capitolo, quando si dispone di dati sperimentali estremamente precisi per un determinato processo di interazione tra particelle elementari, allora, per confrontare la teoria con l'esperimento, ci si trova ad affrontare problemi matematici caratterizzati da un elevato grado di complessità. L'approccio numerico proposto nel presente lavoro costituisce un tentativo di costruire le soluzioni cercate aggirando la difficoltà strettamente matematica prendendo spunto dalle dinamiche casuali di ambiti anche lontani da quello strettamente fisico. È questo il caso degli algoritmi genetici che intraprendono la ricerca del minimo di una funzione, simulando la riproduzione biologica degli organismi viventi sessuati.
Nel Capitolo 2 viene presentata la struttura di un algoritmo genetico e sono dati i dettagli del codice realizzato per il particolare caso in esame.
Infine, nel Capitolo 3, vengono elaborati e commentati i risultati ottenuti. Lo studio preliminare contenuto nel presente lavoro dimostra che l'approccio numerico, consentendo di ricostruire la soluzione del problema posto in maniera altamente parallelizzabile, rappresenta un'alternativa ai metodi già in uso.

Capitolo 1

Calcolo perturbativo in TQC

1.1 L'ampiezza di scattering

Nella prima metà del secolo scorso, le irrisolvibili inconsistenze della meccanica quantistica spinsero affermati fisici come Paul Dirac, Wolfgang Pauli, Sin-Itiro Tomonaga, Julian Schwinger, Richard P. Feynman, Freeman Dyson, a formulare una nuova descrizione che conservasse, tuttavia, le idee rivoluzionarie del modello quantistico: la teoria quantistica dei campi (TQC).
In quest'ottica, l'oggetto di studio non è la particella che si muove soggetta all'azione di eventuali potenziali, ma sono i campi che permeano lo spazio che conferiscono ad esso particolari proprietà, e le cui fluttuazioni portano alla produzione o assorbimento delle particelle.
Analogamente al modello quantistico classico, anche in questa teoria è fondamentale per lo studio dei fenomeni di interazione tra particelle determinare l'*ampiezza di scattering*. Valutando il modulo quadro di questa quantità, infatti, si ottiene la sezione d'urto differenziale del processo, termine necessario per calcolare la probabilità che il processo avvenga.
Quasi sempre si arriva a scrivere una relazione per l'ampiezza di scattering che, però, non si sa risolvere esattamente e pertanto si ricorre al calcolo perturbativo che consente di sviluppare in serie la funzione rispetto ad un piccolo parametro. Quest'ultimo, nel caso di interazione tra particelle, è la costante di accoppiamento della forza.
Per fare un esempio, pensiamo all'interazione elettromagnetica per cui la constante di interazione è la costante di struttura fine α che vale $\frac{1}{137}$ e, quindi, si capisce che già al second'ordine il parametro di sviluppo è piuttosto piccolo.
Il particolare processo che si intende analizzare in questo elaborato è il decadimento di un **bosone di Higgs** nel quark pesante b **(bottom)** e la sua antiparticella $\bar{\mathrm{b}}$.
Senza entrare nei dettagli tecnici della teoria che sta alla base della seguente schematizzazione, ma per voler dare un profilo concettuale al problema che si sta presentando, ci serviamo della rappresentazione pittorica dei diagrammi di Feynamn per descrivere l'interazione.
Fissati i due stati, iniziale e finale, il processo può avvenire secondo diversi percorsi, con l'introduzione di particelle virtuali che si interpongono in numerosis-

simi modi tra le particelle reali. I percorsi chiusi che si creano sono detti loop e in base al numero di vertici che presentano vengono classificati in topologie.

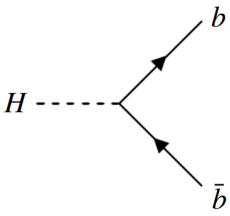


Figura 1.1: Decadimento dell'Higgs in b $\bar{b}$ all'ordine più basso della teoria delle perturbazioni.

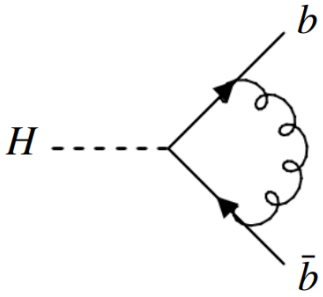


Figura 1.2: Esempio di correzione ad un loop con scambio di un gluone virtuale.

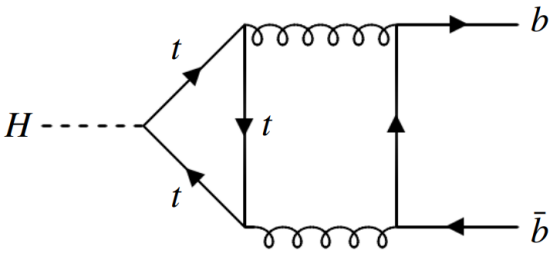


Figura 1.3: Al secondo ordine della teoria perturbativa (due loop), anche il quark pesante top partecipa alla reazione.

L'ordine di sviluppo perturbativo è indicato anche dal numero di loop preso in considerazione. I problemi in cui lo sviluppo si arresta ad un solo loop possono dirsi completamente risolti e risolvibili. La difficoltà di calcolo, però, si manifesta già pesantemente all'ordine successivo (due loop). Ciò è dovuto al fatto che le particelle virtuali possono assumere un qualunque valore di impulso e per tener conto della loro presenza, bisogna considerare i contributi di ciascun loro stato possibile, sovrapponendoli. Quest'operazione porta a formulare l'ampiezza di scattering come una sommatoria di integrali e ricordare che la grandezza di interesse è il suo modulo quadro basta per comprendere la complessità del

problema in analisi.
Prima di proseguire con l'analisi matematica, è doveroso chiarire l'importanza fisica del problema e investigare sui motivi che convincono a spingersi oltre i risultati già ottenuti, verso un miglioramento che si rivela essere non mero perfezionismo.

1.2 Il Modello Standard

Il **Modello Standard** (SM) è la teoria attualmente riconosciuta dalla comunità scientifica per la descrizione dei componenti elementari della materia e delle loro interazioni: le particelle.
La teoria ha preso forma a partire dalla seconda metà del secolo scorso con il susseguirsi di rivelazioni di particelle e con la realizzazione di collisori sempre più prestanti. Di pari passo, la TQC è stata elaborata e perfezionata, portando a predizioni teoriche che hanno accompagnato e sostenuto la sperimentazione.
Nel modello, l'interazione è interpretata come lo scambio di informazioni tra le particelle che ha luogo grazie all'azione dei *mediatori*, ciascuno caratteristico della singola forza.
La conoscenza in questo campo raggiunta sino ad oggi ha portato alla costruzione di una *tavola* di dodici particelle fondamentali, basata sulla classificazione in **leptoni** e **quark**, ai quali si aggiungono i mediatori delle forze, i **bosoni** (vedi Figura 1.4).

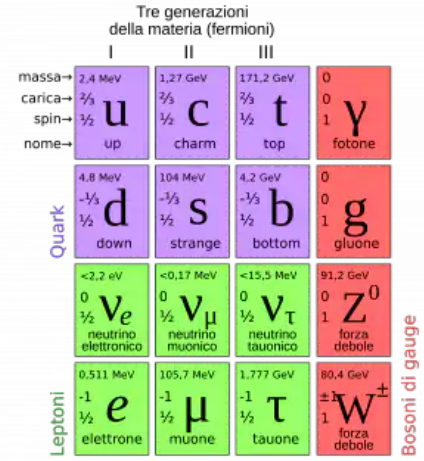


Figura 1.4: Tavola delle particelle elementari.

Le *forze fondamentali* incluse nel MS sono le seguenti:

- la **forza elettromagnetica**, per l'interazione tra particelle cariche e il cui mediatore è il *fotone*;
- la **forza debole**, per l'interazione tra leptoni e quark, tra soli leptoni e tra soli quark e i cui mediatori sono W e Z;

- la **forza forte**, per l'interazione tra quark e il cui mediatore è il *gluone*.

A queste bisognerebbe aggiungere la *forza gravitazionale* la cui attuale descrizione relativistica, però, non riesce ad entrare in accordo con la formulazione della TQC e per questo resta ancora un punto interrogativo della teoria.
Questo insieme alle evidenze cosmologiche di materia oscura e alla asimmetria materia-antimateria, rappresenta un fronte aperto che rende la fondatezza del modello ancora incerta e lo pone al centro di un annoso dibattito scientifico.
Gli atteggiamenti dei fisici contemporanei rispetto a questi interrogativi, sono diversi: da un lato si lavora per il miglioramento del modello nel tentativo di completarlo, dall'altro c'è un appassionato tentativo di formulare una nuova teoria del tutto.

Nel prossimo paragrafo segue un trattazione di livello piuttosto divulgativo di quel tipo di ricerca che aspira al progresso scientifico attraverso il miglioramento dei risultati teorici parallelamente ad un perfezionamento degli apparati sperimentali; tale è la *fisica di precisione*.

1.3 La fisica di precisione

Una teoria può essere confutata nel momento in cui le sue previsioni presentano una discrepanza consistente dai risultati sperimentali che si ottengono.
E' necessario, però, mettersi nelle condizioni per cui questo confronto tra teoria e sperimentazione sia fondato o altrimenti detto, bisogna assicurarsi che l'accuratezza degli esperimenti e la precisione del modello teorico siano sulla stessa scala.
Al giorno d'oggi, il mondo scientifico si trova in un momento in cui l'avanzamento tecnologico che sta alla base degli esperimenti in funzione, mette a disposizione dati con un'ottima precisione che spesso supera i limiti teorici.
Avanzamento teorico, d'altra parte, vuol dire approcciare a problemi matematici di difficile risoluzione per i quali bisogna rielaborare i metodi già esistenti o escogitare nuove strategie *ad hoc*. Le procedure, inoltre, devono presentare le caratteristiche di realizzabilità nei limiti delle capacità di calcolo attualmente esistenti ed efficienza nei tempi di convergenza alla soluzione.

Una volta migliorata la precisione dei risultati, trovare un'incompatibilità con i dati sperimentali potrebbe essere la prova indiretta della presenza di qualcosa di cui ancora non si è tenuto conto, offrendo, così, nuovi orizzonti alla ricerca scientifica. In particolare, l'entità della discrepanza tra i risultati di natura teorica e quelli sperimentali, darebbe un'importante indicazione sulle scale di energie a cui bisognerebbe indagare per la ricerca delle quantità ignote che causano il disaccordo.

A titolo d'esempio, si guardi la figura 1.5 [1]. Quello riportato è un grafico molto diffuso in letteratura, poiché rappresenta una verifica dell'unificazione elettrodebole prevista dal MS, secondo cui le forze elettromagnetica e debole sono due

aspetti di un'unica forza.
Il grafico confronta la misura sperimentale e le predizioni teoriche nel piano individuato dalla massa del bosone W e quella del quark top, a fissati valori della massa dell'Higgs.
I risultati delle misure dirette si possono considerare indipendenti dalla descrizione del MS e sono rappresentati in figura dall'ellissi verde.

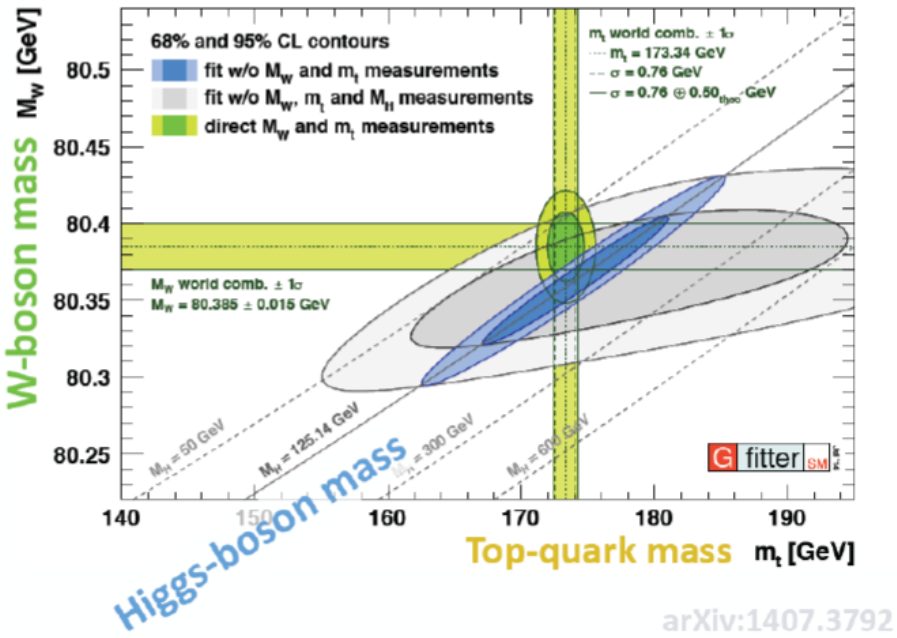


Figura 1.5: Confronto tra misura diretta (verde) e risultato predetto prima (grigio) e dopo (blu) la scoperta dell'Higgs, nel piano individuato dalle masse m_W e m_t, a fissati valori della massa dell'Higgs.

L'ellissi blu nasce, invece, dai fit elettrodeboli che si sono potuti effettuare soltanto grazie alle correzioni radiative che hanno prodotto una stima, anche piuttosto stringente, per le due masse m_t e m_W. Tali risultati sono stati ottenuti tenendo conto del valore misurato della massa dell'Higgs ($m_H = 125,14 GeV$), diversamente dal caso dell'ellisse grigia per cui la stima di m_H resta indeterminata nell'intervallo $[50, 600] GeV$.
Seppure si registra una deviazione tra i centri, la sovrapposizione che si ha tra le due figure entro le bande d'errore è un chiaro accordo tra i risultati diretti ed indiretti.
Le operazioni di riduzione dell'errore sperimentale e di aumento della precisione teorica renderebbero possibili diversi scenari, ad esempio potrebbe verificarsi una diminuzione della sovrapposizione a causa di un allontanamento tra i due centri o per un restringimento delle bande d'errore, o contrariamente la zona blu potrebbe spostarsi ulteriormente verso quella verde.
In questo secondo caso, ci si troverebbe davanti ad un'importante conferma dell'attendibilità e validità del Modello Standard.

Viceversa, il primo scenario proverebbe l'inadeguatezza dell'attuale descrizione e porterebbe ad una rivoluzione di questo "trionfo della fisica moderna" [2], superandolo.

1.4 Il problema di calcolo

Per affrontare il calcolo di un termine ad un dato ordine perturbativo per un problema di interazione, si usa seguire la seguente prassi [3] :

1. generare tutti i possibili diagrammi di Feynman;
2. ridurre gli integrali tensoriali a integrali scalari (algebra di Dirac);
3. scomporre gli integrali di Feynman in integrali master;
4. calcolare gli integrali master.

Gli step 1 e 2 sono eseguibili per ogni processo e ad ogni ordine dello sviluppo; per un approfondimento si consigliano le letture [4] e [5].
Gli integrali scalari di Feynman sono tra loro legati tramite delle identità che conducono alla formulazione di un sistema lineare omogeneo di un numero di equazioni $N \sim 10^3 - 10^5$ del tipo seguente :

$$a_{i1} I_1 + a_{i2} I_2 + ... + a_{in} I_n = 0 \tag{1.1}$$

dove $i = 1, .., N$ è l'indice che individua un'equazione del sistema, I_j ($j = 1, ..., n$) sono gli integrali scalari. Gli integrali I_j e i coefficienti a_{ij} dipendono dagli $\{\alpha_j\}$ parametri fisici del processo in esame (masse, energie caratteristiche). Molte di queste sono dipendenti e possono, quindi, essere scartate dal sistema, riducendo la dimensione da N ad l.
Un importante teorema di TQC [6], assicura che si abbia sempre un numero n di incognite maggiore del numero l di equazioni. Si scelgono, allora, $s = n - l$ integrali da trattare come parametri, detti **integrali master**, e si procede alla risoluzione del sistema. Un qualsiasi integrale scalare può, quindi, essere scomposto in combinazione lineare di un numero finito di master. Il suddetto teorema è, però, un teorema non costruttivo, nel senso che non offre un criterio pratico per eseguire le scomposizioni.
Per fare un'analogia, questi integrali possono essere pensati come gli elementi di base di una spazio vettoriale, le cui combinazioni lineari generano tutti gli integrali-vettori che vivono nello spazio. È comunque importante tenere ben chiaro che non è propriamente questo il caso, poiché gli integrali che si stanno considerando non rispettano le proprietà necessarie per costituire uno spazio vettoriale.
Quindi il j-esimo integrale I_j può essere rappresentato come :

$$I_j = b_{j1} M_1 + b_{j2} M_2 + ... + b_{js} M_s \tag{1.2}$$

Per la loro forma analitica, gli integrali master presentano l'importantissima proprietà per cui una loro derivata, rispetto ad una qualsiasi delle variabili da cui

dipendono, è in generale una combinazione lineare di integrali della famiglia e, quindi, per quanto asserisce il suddetto teorema, se $M_i = M_i(\alpha_1, ..., \alpha_r)$, allora:

$$\begin{aligned}\partial_{\alpha_j} M_i(\alpha_1, ..., \alpha_r) &= c_{i1} I_1(\alpha_1, ..., \alpha_r) + ... + c_{ip} I_p(\alpha_1, ..., \alpha_r) \\ &= c_{i1}(b_{11} M_1 + ... + b_{1s} M_s) + ... + c_{ip}(b_{p1} M_1 + ... + b_{ps} M_s) \\ &= d_{i1} M_1(\alpha_1, ..., \alpha_r) + ... + d_{ir} M_r(\alpha_1, ..., \alpha_r)\end{aligned} \tag{1.3}$$

con $j \in [1, r]$, $i \in [1, s]$ e dove in ciascun d_{ij} sono raccolti tutti i coefficienti dell'M_j.
Dalla fisica del problema, inoltre, si ricava che i d_{ij} sono funzioni razionali fratte nelle variabili α_j che possono facilmente essere condotti a polinomi calcolando il minimo comune multiplo; in tal modo, anche la derivata acquista un coefficiente:

$$d_{i1} M_1(\alpha_1, ..., \alpha_r) + ... + d_{ir} M_r(\alpha_1, ..., \alpha_r) + d_{ir+1} \partial_{\alpha_j} M_i(\alpha_1, ..., \alpha_r) = 0 \tag{1.4}$$

Per $j \in [1, r]$ e per $i \in [1, s]$, le equazioni 1.4 costituiscono un sistema lineare alle derivate parziale al primo ordine, chiuso, ossia con numero sufficiente di equazioni per essere risolto. Imposte le opportune condizioni al contorno, la risoluzione del sistema permette di valutare gli integrali master in qualunque punto $\{\alpha_j\}$ e in questo si manifesta la potenza di questo metodo.
Fissati i valori delle variabili α_j in opportune regioni dello spazio dei parametri, è possibile calcolare, mediante integrazione numerica diretta, i valori dei master e delle loro derivate parziali nel dato punto; in tal modo, manca da determinare soltanto i coefficienti che individuano i polinomi $\{d_{ij}\}$. È importante sottolineare che l'integrazione numerica dei master risulta efficace ed accurata solamente in regioni confinate dello spazio dei parametri che di solito risultano distanti da quelle di interesse fisico. In queste ultime, infatti, gli integrandi possono mostrare un comportamento altamente oscillante che rende inefficiente, e in molti casi di fatto impraticabile, l'integrazione numerica diretta. Per questo motivo, il metodo delle equazioni differenziali è da preferirsi.
La determinazione del sistema di equazioni differenziali, per quanto sopra discusso, necessita la risoluzione del sistema lineare 1.1, ovvero ridurre completamente tutti gli integrali della famiglia ai master. Il problema algebrico resta, d'altra parte, non affatto banale, dato il consistente numero di equazioni coinvolte nel sistema.
In questo lavoro, si studia una strategia risolutiva alternativa di tipo numerico che tramite un'operazione di *fitting*, punti alla determinazione del sistema di equazioni differenziali dei master, ovvero della matrice di coefficienti $\{d_{ij}\}$, senza passare per la completa risoluzione del sistema 1.1.
Nello specifico, si è voluto testare l'efficacia di un particolare algoritmo esplorativo, l'**algoritmo genetico**, per la ricerca del vettore di coefficienti che risolvesse separatamente le equazioni del sistema.
I dettagli sulla struttura e il funzionamento dell'algoritmo sono trattati nel Capitolo 2, mentre nel Capitolo 3 sono presentati i casi studiati e i risultati ottenuti per essi.

Capitolo 2

Algoritmi Genetici

A partire dalla pubblicazione della monografia "Adaptation in Natural and Artificial Systems" [7] del matematico e sociologo J.H. Holland, la diffusione degli algoritmi genetici come metodi euristici per la risoluzione di problemi di ottimizzazione, subì un notevole incremento. Gli algoritmi genetici rientrano nella classe degli algoritmi evolutivi e, come suggerisce il nome, prendono a modello le leggi che regolano le dinamiche di riproduzione naturale. In particolare, la logica su cui si basano trova ispirazione nella teoria della selezione darwiniana, la cui idea di fondo è che in una popolazione eterogenea di specie, quella che presenta un maggior numero di caratteristiche conformi all'ambiente circostante, sopravvive diffondendosi. A questo si aggiunge la visione per cui gli esemplari della specie sono nient'altro che "macchine da sopravvivenza" - come descrive R. Dawkins in "The selfish gene" [8] - "programmati ciecamente per preservare quelle molecole egoiste note sotto il nome di geni".
Un algoritmo genetico vuole trattare le soluzioni di una funzione multivariata come le diverse specie di una popolazione e si propone l'obbiettivo di individuare il punto di ottimo disponendo ad esso un ambiente che lo favorisca, facendolo emergere.
Nella schematizzazione dell'algoritmo, ogni soluzione è detta *cromosoma*. Un cromosoma conserva e trasporta i *geni*, ovvero le singole variabili da cui dipende la funzione. L'informazione genetica che rappresenta le caratteristiche di una specie, si può individuare nel valore e nella posizione del gene all'interno del vettore soluzione (il cromosoma). La bontà di un cromosoma è determinata dalla *fitness*, una funzione test che si trova in corrispondenza con la funzione da minimizzare e che guida l'algoritmo nella ricerca. Dunque, un cromosoma si definisce più *adatto* in base al valore che la fitness assume quando viene calcolata in esso e conserva il primato finché non si incorra in un nuovo cromosoma più prestante, grazie all'azione degli operatori genetici. Questi ultimi permettono l'esplorazione del dominio della funzione di interesse, testando e modificando la popolazione.
L'algoritmo presenta uno schema fisso, componendosi di tre fasi: **selezione**, **crossover**, **mutazione**. Ciascuna di queste operazioni contribuisce da un lato alla conservazione dei cromosomi che si sono mostrati più adatti, dall'altro all'esplorazione di tutto lo spazio delle configurazioni.
Segue un'analisi dettagliata delle tre fasi componenti, dopo aver premesso una

descrizione della fase preparatoria per l'implementazione dell'algoritmo.

2.1 Preparazione

Per quanto detto, si intende che i cromosomi sono l'oggetto che il programma elabora e restituisce in output. È pertanto necessario scegliere una loro rappresentazione che consenta l'applicazione di ciascun operatore genetico. A tal fine, costruiamo un cromosoma come un array i cui elementi, i geni, sono le variabili da cui dipende la funzione in analisi e, quindi, la fitness. I geni, a loro volta, necessitano anch'essi di una codifica che si presti al tipo di operazioni che il programma dovrà eseguire; tale è la rappresentazione binaria. Nel caso specifico preso in analisi, i geni sono numeri relativi di **Z**, rappresentati da un numero l di bit, scelto in base al modulo del numero massimo che compare nel dominio che vogliamo esplorare.
Si riporta una breve descrizione del metodo di codifica binaria di un intero con segno. Dato un numero l di bit, si dice bit più significativo quello che occupa la posizione più a sinistra, sia la posizione l. Il bit alla posizione 1 è, allora, il bit meno significativo. Sia il bit l riservato al segno e tale che il valore 1 indichi il segno $+$ e lo 0 il segno $-$. Se si vuole rappresentare in binario un numero il cui valore assoluto sia X, saranno necessari l bit tali che $X \leqslant 2^{l-1} - 1$. Si procede calcolando il resto della divisione di X per 2^i, per i che corre da 0 ad $l-1$, sostituendo ad X volta per volta il risultato della divisione precedente, cioè per 2^{i-1}, e assegnando tale valore al bit di posizione $i+1$. Ad esempio, sia $X = 8$: la sua rappresentazione in binario con 5 bit è 11000, mentre $X = -8$ è rappresentato da 01000 .
La codifica binaria si rivela essere fortemente adeguata poiché permette di applicare gli operatori genetici ai bit che compongono i geni del cromosoma, andando ad alterare o conservare direttamente i componenti elementari del cromosoma.
Il programma presenta un primo momento di creazione della gene pool, in cui viene generato un numero *ni* di cromosomi di partenza, i cui geni sono costruiti in maniera randomica, assegnando ai bit i valori 0 e 1. Dell'initial gene pool viene, però, conservata solo la miglior metà, scelta dopo aver ordinato i cromosomi in base al valore della fitness calcolata in corrispondenza di essi.
La trattazione procede con la definizione degli operatori genetici con cui l'algoritmo lavora e l'analisi in dettaglio del loro funzionamento.

2.2 Selezione

La prima operazione da effettuare è scegliere i cromosomi che devono riprodursi. Un criterio applicabile può essere quello dei "tornei a coppia"[9]: scelta una coppia, il cromosoma vincente è quello che presenta una fitness migliore, aggiudicandosi, così, un posto nella parent pool. Prima di effettuare i tornei, però, bisogna preoccuparsi di disordinare la gene pool così da limitare, da un lato, la perdita di cromosomi con buona fitness e conservare, dall'altro, la casualità degli eventi.

In questa applicazione, l'operatore di *mixing* elaborato scambia il cromosoma $i-esimo$ con quello alla posizione h, con un ciclo che parte dal cromosoma $n-esimo$ retrocedendo fino al secondo. L'indice h è pescato casualmente nell'intervallo $[1, i]$, in modo da non coinvolgere più i cromosomi già scambiati.
Segue il codice per la subroutine utilizzata per generare la parent pool:

```
h=1
do i=1,int(n*0.5)
        if(f(h)<f(h+1)) then
            f(i)=f(h)
            do j=1,m
               do k=1,l
                    cromosoma(i,j,k)= cromosoma(h,j,k)
                end do
                dcromosoma(i,j)=dcromosoma(h,j)
            end do
        else
            f(i)=f(h+1)
            do j=1,m
                do k=1,l

                    cromosoma(i,j,k)= cromosoma(h+1,j,k)
                end do
                dcromosoma(i,j)=dcromosoma(h+1,j)
            end do
        end if
        h=h+2
end do
```

dove *cromosoma* al variare dell'indice *i* individua un cromosoma della pool, *dcromosoma* è la rappresentazione in decimale del cromosoma $i-esimo$, *f* è l'array che contiene i valori della fitness in corrispondenza dei cromosomi.

Un metodo di selezione alternativo a quello presentato è il criterio di *selezione proporzionata alla fitness* (anche detto *Roulette Wheel Selection*), in base al quale il cromosoma i-esimo è scelto come genitore con probabilità

$$P_{selection} = \frac{f(i)}{\sum_i f(i)}$$

dove $f(i)$ è il valore della fitness valutata per il cromosoma i .

2.3 Crossover

Ottenuto un insieme di genitori, bisogna che questi si riproducano e tale compito è assegnato all'operatore di *crossover*, ovvero il processo per cui i cromosomi figli vengono costruiti componendo parti di due cromosomi genitori. In particolare, il metodo utilizzato nell'algoritmo elaborato in questo lavoro è quello del

crossover a due punti. Presi due cromosomi e scelti due punti casuali in cui effettuare il taglio, ogni gene viene diviso in tre parti: i geni $i-esimi$ dei cromosomi genitori vengono composti per formare i geni dei due cromosomi figli.
Per chiarire il procedimento, si osservi la schematizzazione in Figura 2.1. Sia la stringa in alto a sinistra la rappresentazione binaria del gene 1 del genitore 1 e la stringa in alto a destra il gene 1 del genitore 2. Le caselle evidenziate in rosso sono i bit coinvolti nel crossover. I genitori si scambiano materiale genetico ad incrocio generando due nuove stringhe che vanno a formare i geni 1 di due cromosomi figli.

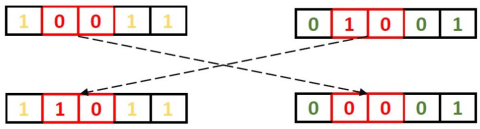


Figura 2.1: Crossover a due punti di taglio.

Questa procedura va effettuata per ogni gene dei genitori, per ottenere due figli completi. Alla fine dell'intero processo, si riotterrà una pool di n cromosomi, metà genitori scelti, metà figli creati.

2.4 Mutazione

La nuova generazione di cromosomi è ora soggetta all'ultima operazione genetica: la trasformazione di alcuni bit che compongono i geni, al loro valore complementare, ovvero da 1 a 0 e viceversa. Questo processo è del tutto casuale e coinvolge solo una percentuale di bit sul totale dell'intera pool. Il valore della percentuale deve essere tale da velocizzare il processo di esplorazione, senza risultare tanto stravolgente da causare la perdita di informazioni sui migliori cromosomi trovati fino a quel punto. Una strategia mirata a contenere questo effetto, può essere rendere alcuni cromosomi immutabili, ad esempio i primi in ordine per fitness.
Per capire l'importanza di quest'operatore, si può pensare al caso in cui l'algoritmo incorra in un minimo locale. Per come si struttura la sequenza dei numeri binari, si capisce che il crossover ha una debole capacità esplorativa o, comunque, piuttosto localizzata, per questo è necessario l'intervento di un altro processo che aiuti il programma a svincolarsi dalla valle di minimo e saltare in una zona completamente diversa che potrebbe contenere cromosomi migliori. Risulta, inoltre, utile rendere la percentuale un parametro dinamico, permettendo al programma di incrementarla quando, dopo un certo numero di cicli dell'algoritmo, esso si trovi bloccato ad uno stesso valore di fitness che, però, non rispetta le caratteristiche richieste per poter essere la soluzione del problema.
Impostata la percentuale di bit da mutare, il programma sceglie cromosoma, gene e bit su cui operare pescando un numero casuale tra $[0, n]$, uno in $[0, m]$ ed uno in $[0, l]$, se n, m ed l sono il numero di cromosomi nella pool, di geni

in un cromosoma e di bit per un ciascun gene rispettivamente. Bisogna, però, tener conto dei cromosomi che sono stati scelti come immutabili, aggiungendo un controllo al ciclo tale che l'operazione venga impedita per gli indici di cromosoma coincidenti con quelli degli immutabili.
Segue il codice utilizzato:

```
i=1
mutati=int(percentuale*n*l*m/100)
do while(i<=mutati)
   i=i+1
   do
        call random_number(t)
        v=int(t*n+1)
    if(v>immutati) exit
    end do
    call random_number(p)
    q=int(p*m+1)
    call random_number(r)
    s=int(r*l+1)
    if (cromosoma(v,q,s)==1) then
    cromosoma(v,q,s)=0
    else
    cromosoma(v,q,s)=1
    end if
end do
```

dove *immutati* indica il numero di cromosomi che non sono sottoposti all'operazione di mutazione.

2.5 Cenni teorici

Come detto all'inizio di questo capitolo, il lavoro di Holland del 1975 è riuscito a porre l'algoritmo genetico sotto i riflettori di uno scenario molto ampio che spazia dall'ambito più strettamente scientifico a quello economico o ancora a quello sociale, seppur dopo qualche anno dalla pubblicazione della prima edizione.
Già nella prefazione della seconda edizione del 1992, lo stesso Holland riconobbe il ruolo pionieristico che il proprio lavoro aveva rappresentato nella diffusione di questo schema di calcolo, soffermandosi sull'osservazione di come "si iniziò a vedere gli algoritmi genetici come uno strumento teorico per investigare i fenomeni originati da sistemi complessi adattivi", letteralmente dal testo: "genetic algorithms began to be seen as a theoretical tool for investigating the phenomena generated by complex adaptive systems. " (cit.).
I sistemi complessi adattivi (CAS) sono sistemi dinamici (evolvono deterministicamente nel tempo), auto-organizzati, ovvero che modificano il proprio comportamento in base a fattori esterni che dipendono dalle variazioni dell'ambiente in cui vivono che essi stessi influenzano. Un algoritmo genetico non fa altro che

trattare i problemi che gli vengono sottoposti come CAS, valorizzando l'autogestibilità di questi e facendola diventare il proprio punto di forza.
Per problemi complicati, non è quasi mai possibile affidarsi unicamente alla potenza evolutiva del CAS, poiché si verifica spesso che l'esplorazione dello spazio delle configurazioni incontri zone a "collo di bottiglia" ("bottleneck") in cui si registra un miglioramento dei risultati, ma non la convergenza del problema alla soluzione. Accorre in soccorso a questa falla del metodo il teorema degli schemi. Gli schemi ("building blocks" o "schemata") sono strutture ricorrenti con cui è possibile campionare lo spazio da esplorare, effettuando una partizione in iperpiani, e il teorema propone una metodologia per sfruttarli al fine di ottimizzare l'algoritmo.
I dettagli del suddetto teorema non verranno trattati in questo lavoro di tesi.
Tuttavia, in generale i fondamenti teorici che sostengono gli algoritmi genetici sono forti nella definizione del metodo, ma producono risultati ancora deboli nella capacità predittiva degli esiti di un'esecuzione, lasciando in sospeso la valutazione della sua effettiva efficienza.

Capitolo 3

Impiego e risultati

Il programma realizzato è un codice in linguaggio `Fortran95` strutturato seguendo la linea guida esposta nel *Capitolo 11-"Genetic algorithm and programming"* del testo *"An Introduction to Computational Physics, 2thEdition"* del fisico Tao Pang [9]. I dettagli del codice sono stati riportati e discussi nel **Capitolo 2** del corrente elaborato.
Analizzato il problema fisico e l'entità della sua portata e presentata la struttura generica di un algoritmo genetico, in questo terzo capitolo verranno riportati i dettagli dell'applicazione ed una discussione dei risultati ottenuti.

3.1 Il sistema di equazioni

Le equazioni che si vogliono risolvere sono quelle del sistema 1.4. I valori dei master $M_j(\alpha_1, ..., \alpha_r)$ e delle loro derivate $\partial_{\alpha_j} M_i(\alpha_1, ..., \alpha_r)$ in un dato punto $(\alpha_1, ..., \alpha_r)$ sono stati calcolati con il programma numerico `PySecDec` [10][1].
Le incognite $\{d_{ij}\}$ sono polinomi delle $\{\alpha_j\}$, di cui bisongna determinare sia il grado sia i coefficienti, il che, in tutta generalità, rende il problema piuttosto complesso.
Iniziando a caratterizzare le funzioni in analisi, le variabili $\{\alpha_j\}$ sono i quadrati delle tre masse in gioco nel processo fisico considerato, la m_H dell'Higgs, la m_b del quark bottom e la m_t del quark top. Si può, allora, scrivere che in generale

$$\partial_{\alpha_j} M_i = \partial_{\alpha_j} M_i(m_H, m_b, m_t)$$

$$M_j = M_j(m_H, m_b, m_t)$$

Quello trattato è, però, un problema fisico e questo porta a delle condizioni con cui si ottiene una piacevole semplificazione delle equazioni.
La prima semplificazione è che le derivate possono essere indipendenti da qualche variabile α_j; allora, in tutti i polinomi-coefficienti i coefficienti che moltiplicano tale variabile a qualsiasi grado, sono subito determinati e sono nulli.
Si può sapere, inoltre, per ciascuna equazione, quali integrali compaiono con

[1] Il programma è stato opportunamente modificato per consentire la valutazione delle derivate dei master.

coefficiente nullo nella combinazione lineare con cui si esprime la derivata, snellendo notevolmente il sistema.
Il grado dei polinomi-coefficienti è parzialmente determinato operando un'analisi dimensionale; fissato arbitrariamente il grado del polinomio-coefficiente della derivata, si determinano facilmente i gradi di tutti gli altri polinomi.Gli esempi in Appendice A rendono chiara questa procedura.
In base al grado cambia il numero di coefficienti da determinare. Ad esempio, polinomi lineari contribuiranno con un numero n=2 parametri, quelli quadratici con n=3 parametri, i cubici con n=6 e così procedendo.
Se N è il numero totale di coefficienti, si avranno delle equazioni del tipo

$$y(\{a[j]\}) = 0 \tag{3.1}$$

con j che va da 0 ad $N - 1$.
In realtà, al calcolo del numero di coefficienti da determinare contribuisce anche un altro fattore, la dimensione d. Questa quantità è in stretta relazione con il parametro con cui si regolarizzano le divergenze degli integrali, secondo una procedura di sviluppo in serie di Laurent tipica in TQC.

3.2 La funzione fitness

Il programma opera su una sola equazione 3.1 del sistema. Scelta questa, la *fitness* è costruita come scarto quadratico tra i valori della funzione e il suo valore atteso, lo zero:

$$\mathtt{fitness} = |y(\{a[j]\}) - 0|^2 \tag{3.2}$$

In questo modo, cercando il vettore che minimizza la fitness, il programma starà ricercando proprio la soluzione di coefficienti che verifica l'equazione $y(\{d_{ij}\}) = 0$ del sistema 1.4, risolvendo il problema in questione.

PRECISIONE
In realtà, la valutazione dei master nel punto $\{\alpha_j\}$ è ottenuta in maniera approssimata e pertanto presenta un errore numerico, sia ϵ, che anche se piccolo non può essere trascurato. Quest'ultimo indica, quindi, l'ordine di precisione con cui si conoscono i master e deve essere propagato per ogni operazione in cui essi sono coinvolti. Per questo, anche nel punto di ottimo, il risultato che si ottiene non è esattamente uno $0 = 0$, ma il valore minimo per la fitness risulta dell'ordine dell'errore ϵ.
Sapere la precisione con cui si ottiene il risultato è di fondamentale importanza per la determinazione della condizione di uscita dal programma: la ricerca, infatti, può interrompersi o perché è stato effettuato il numero massimo di cicli richiesto oppure perché la fitness ha raggiunto un valore dell'ordine di ϵ, che, quindi, può ragionevolmente essere considerato il minimo.

CAMPIONAMENTO
La fitness può essere migliorata per aumentare l'affidabilità dei risultati: campionando la $y(\{d_{ij}\})$ per diversi punti (m_H, m_b, m_t), si può scrivere la fitness

come somma delle fitness per ciascuna valutazione; in questo modo, si guadagna la condizione per cui un vettore che minimizza la fitness totale deve minimizzare contemporaneamente tutto il campione e ciò dà buone speranze che quello trovato sia effettivamente un minimo globale.

SPAZIO DELLE SOLUZIONI
Come è stato precedentemente detto, i polinomi-coefficienti $\{d_{ij}\}$ sono polinomi a coefficienti interi; ciò significa che gli $a\{j\}$ sono array di interi che vivono in uno spazio che costringe l'algoritmo a muoversi a passi discreti. Può succedere che se il minimo globale è molto "profondo", l'algoritmo, non muovendosi con continuità, potrebbe oltrepassarlo senza caderci dentro. Un modo per "allargare" la zona intorno ad un minimo di questo tipo può essere elevare la fitness ad una potenza positiva minore di 1, così da "addolcire" la discesa della buca. Detta α questa potenza, essa rappresenta un ulteriore parametro di cui tenere conto nella preparazione del programma.

Il programma è stato impiegato per due diverse equazioni del sistema **SIMPLE1** e **SIMPLE2**, delle quali sono già noti i polinomi-coefficienti $\{d_{ij}\}$ soluzione, cosicché è stato possibile testare le capacità risolutive dell'algoritmo.
I valori degli integrali master e delle loro derivate sono stati trovati con un programma di calcolo che, però, come anticipato nel Capitolo 1, restituisce i risultati in tempi ragionevoli soltanto per regioni non fisiche dello spazio.
Alcuni dettagli riguardanti le due equazioni studiate sono riportati in Appendice A.

3.3 L'applicazione

Gli studi effettuati sono consistiti principalmente nel valutare la capacità risolutiva dell'algoritmo al variare dei parametri da cui esso dipende, al fine di conoscerne gli andamenti generali o eventuali indipendenze.
Prima di riportare i risultati ottenuti, è necessario presentare i parametri e chiarire un po' di nomenclatura che verrà adoperata in seguito.

- **bit (L)**: indica il numero di bit con cui i singoli geni vengono rappresentati;
- **geni (M)**: è il numero di geni che compone un cromosoma;
- **cromosomi iniziali (NI)**: indica l'entità della popolazione iniziale;
- **percentuale (%)**: indica la percentuale di bit sull'intera pool che subisce l'operazione di *mutazione* ad ogni ciclo;
- **immutati**: è il numero di cromosomi che non può subire *mutazione*;
- **cicli max**: è il numero massimo di cicli da effettuare prima dell'arresto del programma;

- **cicli effettuati (CE)**: è il numero di cicli che il programma ha compiuto prima di restituire un risultato. Se coincide con *cicli max*, allora il programma ha eseguito tutti i cicli a disposizione senza riuscire a trovare la soluzione.

Il numero M di geni dà una dimensionalità al problema e per questo da esso ne dipende la complessità. E' intuitivo capire che all'aumentare della dimensione M, lo spazio di ricerca si amplia e ciò comporta un rallentamento nell'esecuzione e una diminuzione dei successi di convergenza. Per controllare questi effetti ed ottimizzare l'efficienza dell'algoritmo, diventa decisivo effettuare una scelta consapevole per i valori dei suddetti parametri e ciò può avvenire soltanto se si possiede una sufficiente conoscenza del comportamento del programma al variare di essi.
Lavorare a diversi M con la stessa funzione è stato possibile impostando manualmente i valori di $(r - M)$ geni, se r è il numero totale di variabili della fitness.
L'analisi è stata effettuata per casi semplici, ovvero per piccoli valori di M, in modo tale da lavorare in regimi in cui l'algoritmo si mostrava soddisfacentemente prestante.
Ciascun dato riportato nelle tabelle seguenti è stato ottenuto come media su dieci esecuzioni a fissata scelta dei parametri; tale stima è stata effettuata nel tentativo di contenere l'effetto delle inevitabili fluttuazioni degli eventi statistici.
Tutti i grafici seguenti sono stati costruiti con dati normalizzati al valore ottenuto al primo punto, al fine di migliorare la leggibilità degli andamenti risultanti.

3.3.1 CICLI EFFETTUATI (CE) vs PERCENTUALE (%)

La Tabella 3.1 e la Figura 3.1 sono relative alla funzione SIMPLE 1.
I parametri rimanenti sono stati fissati come segue:
$NI = 200$, $L = 4$, $immutati = 1$, $ciclimax = 30000$.

	$M = 2$	$M = 3$	$M = 4$	$M = 5$	$M = 6$
%	CICLI ESEGUITI				
1	1822,8	8107,6	12473,6	13142,2	20490,1
2	562,0	4701,7	13618,4	9012,9	12330,3
3	59,1	4404,3	7046,4	7650,2	11056,9
4	72,9	645,0	2562,9	3493,5	6081,3
5	1,4	621,5	946,4	1142,4	3634,6
7	1,1	10,6	426,3	939,2	3894,0
10	2,7	11,7	159,3	933,7	3588,7
14	2,1	17,2	140,1	650,7	7027,0
30	1,2	25,2	262,1	1963,8	8167,9
50	1,7	20,8	189,6	960,6	9229,3
70	1,2	15,2	424,3	1418,6	9085,2

Tabella 3.1: SIMPLE 1, CICLI VS PERCENTUALE

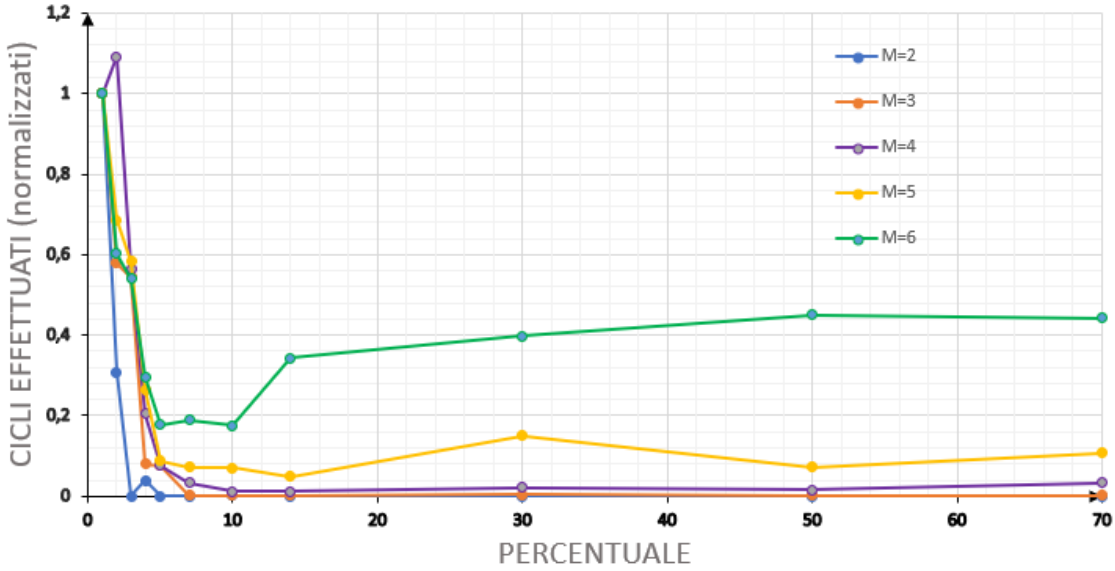


Figura 3.1: SIMPLE 1, CICLI VS PERCENTUALE

Si inizi con il considerare la zona di piccole percentuali: per valori inferiori a 4%, tutte le curve suggeriscono una scarsa efficienza dell'algoritmo, in quanto, sebbene si raggiunga la soluzione, il numero di cicli impiegato non è ottimizzato. La zona di grandi percentuali, invece, sembra mostrare il raggiungimento di una saturazione, per cui l'algoritmo si comporta in maniera piuttosto indifferente al valore scelto per il parametro.
La curva corrispondente a $M = 6$ (verde) presenta una valle di minimo a cui corrispondono i migliori valori di percentuale.
La risalita, in realtà, si verifica anche per $M = 5$ (gialla), se si pensa di poter considerare il punto in $\% = 30$ un'infelice fluttuazione.
Per $M = 4$ (viola) potrebbe sembrare essere una forzatura riconoscere un'analoga risalita che, comunque, si manifesta in $\% = 70$.
È evidente, in ogni caso, che il comportamento dell'algoritmo si modifica con un andamento funzionale rispetto alla dimensione del problema e ciò trova una prova anche nel fatto che le curve normalizzate si dispongono in maniera ordinata per M crescente.

È interessante, a questo punto, confrontare questi risultati con quelli ottenuti per una diversa funzione, la SIMPLE 2.
Anche in questo caso è stato assegnato un valore fisso ai parametri rimanenti : $NI = 200$, $L = 5$, $immutati = 1$, $ciclimax = 30000$.

In Tabella 3.2, per il caso di $M = 5$, i puntini stanno ad indicare che non sono stati presi i dati corrispondenti a causa dei tempi di esecuzione troppo lunghi[2]. D'altra parte, si può con certezza affermare che il numero di cicli debba essere almeno uguale all'ultimo dato rilevato, ovvero $17728,0$ ottenuto per $\% = 14$.
Il problema dei tempi motiva anche l'assenza dello studio del caso $M = 6$, di-

[2]Stesso significato assumono i puntini in Tabella 3.4 più avanti.

	M = 2	M = 3	M = 4	M = 5
%		CICLI ESEGUITI		
1	48, 6	5222, 4	8992, 7	23499, 8
2	2, 6	2943, 5	6757, 0	21322, 4
3	3, 4	54, 9	1219, 6	13686, 3
4	3, 2	69, 3	443, 2	9838, 7
5	3, 1	81, 3	334, 2	9378, 3
7	3, 9	30, 4	297, 5	6253, 4
10	4, 8	41, 4	368, 9	17012, 5
14	1, 8	56, 0	506, 8	17728, 0
30	2, 5	91, 6	2444, 4	...
50	4, 3	68, 5	2313, 1	...
70	2, 8	57, 1	4328, 9	...

Tabella 3.2: SIMPLE 2, CICLI VS PERCENTUALE

versamente da quanto fatto per la funzione SIMPLE 1.
I dati ottenuti danno comunque un risultato interessante: l'insieme di curve, infatti, presenta un comportamento analogo al caso della funzione SIMPLE 1, confermando l'ipotesi dell'esistenza di una precisa dipendenza funzionale dal parametro percentuale.
Infatti, per piccoli valori della percentuale, tutte le curve si trovano a grandi valori del numero di cicli, per poi scendere rapidamente verso un minimo e, infine, segue una risalita che si fa sempre più ripida con l'aumentare della dimensione.
La presenza di una valle di minimo è ancora più evidente in questo esempio, così come la sua tendenza a diventare sempre più profonda e stretta per M crescenti.

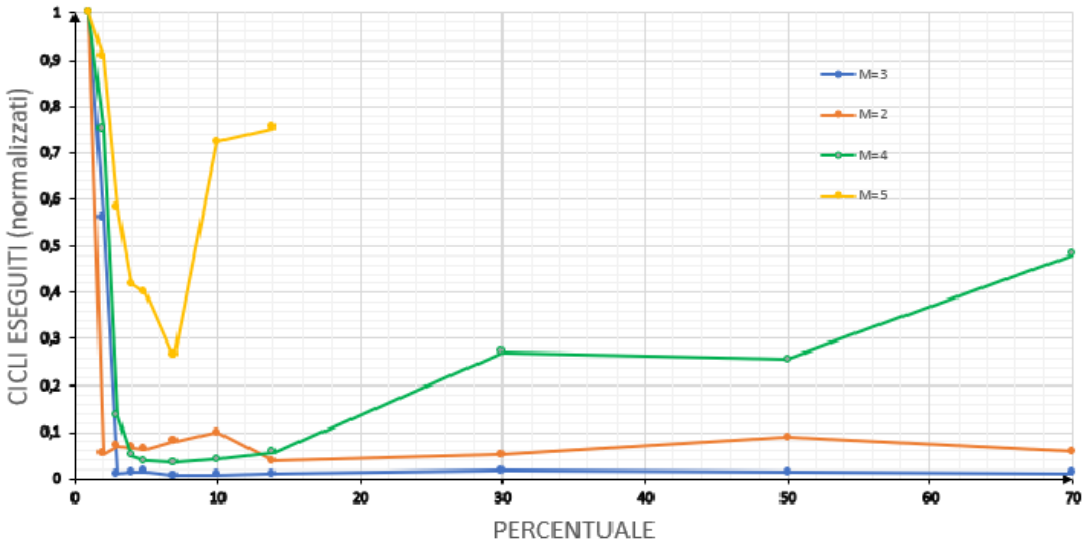


Figura 3.2: SIMPLE 2, CICLI VS PERCENTUALE

3.3.2 CICLI EFFETTUATI (CE) vs POOL INIZIALE (NI)

Il secondo parametro che si è fatto variare è il numero iniziale di cromosomi, impostando i rimanenti ai valori :
$\% = 5$, $\mathrm{L} = 4$, $\mathit{immutati} = 1$, $\mathit{ciclimax} = 20000$.
Per quanto detto nella sezione 2.1, la popolazione iniziale viene generata in maniera del tutto casuale e di essa, dopo aver valutato la fitness per ciascun cromosoma, si conserva soltanto la metà migliore con cui il programma lavora.
Ci si aspetta che all'aumentare di questo parametro, l'algoritmo converga alla soluzione in un numero minore di cicli, proprio perché ad ogni ciclo ci sono più possibilità di incorrere nel cromosoma giusto.
Quest'ipotesi è confermata dagli andamenti registrati in Figura 3.3, in cui le curve tendono a decrescere stabilizzandosi. La curva per $\mathrm{M} = 4$ (arancione) è piuttosto verosimilmente affetta da fluttuazioni statistiche che potrebbero essere attenuate migliorando la precisione delle stime.

	$\mathrm{M}=2$	$\mathrm{M}=3$	$\mathrm{M}=4$	$\mathrm{M}=5$	$\mathrm{M}=6$
NI	CICLI ESEGUITI				
200	1,5	77,4	218,2	1231,0	5379,6
320	1,1	20,2	228,7	957,8	4192,7
400	1,1	8,0	309,0	1130,5	2728,5
800	1,0	3,0	102,3	546,2	1046,8
1200	1,0	3,9	320,3	332,4	1258,3
2000	1,0	1,9	25,1	195,8	734,6

Tabella 3.3: SIMPLE 1, CICLI VS POPOLAZIONE INIZIALE

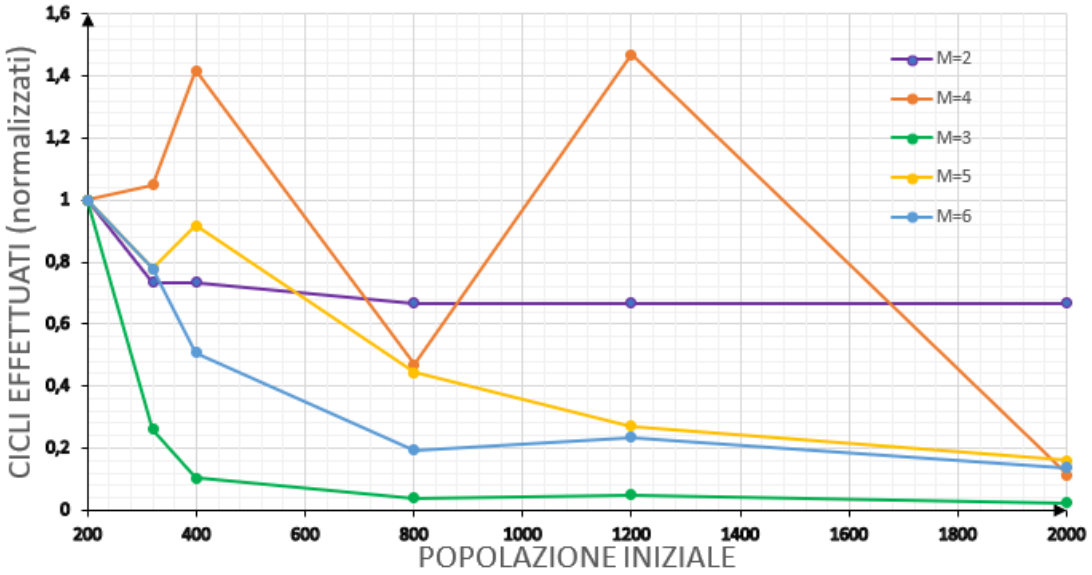


Figura 3.3: SIMPLE 1, CICLI VS POPOLAZIONE INIZIALE

Dall'analisi effettuata non è possibile trarre nessun tipo di conclusione particolarmente utile ai fini dell'ottimizzazione dell'algoritmo; piuttosto potrebbe essere interessante capire come cambiano i tempi di esecuzione di un ciclo al variare del parametro NI. Infatti, per una pool più popolosa aumentano le operazioni che ad ogni ciclo il programma deve compiere e ciò comporta una dilatazione dei tempi di esecuzione. Sapere di quanto tempo necessita un ciclo per essere compiuto è importante per poter stimare l'attesa massima di elaborazione del programma.

3.3.3 CICLI EFFETTUATI (CE) vs BIT (L)

Scegliere il numero di bit con cui rappresentare i geni vuol dire porre i confini allo spazio da esplorare. Infatti, dato un numero L di bit di cui uno riservato al segno, il numero decimale massimo rappresentabile è

$$N_{max} = \sum_{i=0}^{L-2} 2^i$$

mentre i cromosomi ottenibili sono

$$C_{max} = (2N_{max} - 1)^M$$

ricordando che M è il numero di geni.

	$M = 3$	$M = 4$	$M = 5$	$M = 6$
L	CICLI ESEGUITI			
4	3,9	61,6	487,0	3967,8
5	9,8	55,2	600,4	6184,6
6	30,4	253,6	5516,2	...
7	63,6	1837,8	...	...

Tabella 3.4: SIMPLE 1, CICLI VS BIT

La Figura 3.4 mostra come estendendo lo spazio di ricerca, la complessità del problema cresce rapidamente e questo effetto è sempre più accentuato all'aumentare della dimensione. La scelta di estrapolare quattro dati per i casi $M = 3$ e $M = 4$, tre per il caso $M = 5$ e soli due per $M = 6$, è stata forzata proprio dai lunghi tempi di attesa dell'esecuzione che, per di più, non sempre portava alla soluzione.
L'analisi del parametro L ha fatto emergere un interessante fenomeno: la convergenza a nuovi minimi. Fintanto che l'algoritmo investigasse in spazi piccoli (per $L = 4$), la condizione di uscita dal programma si mostrava ben adatta al caso, poiché evidentemente l'unico punto che la soddisfacesse era soltanto la soluzione esatta. L'introduzione di molti nuovi cromosomi ha determinato l'inclusione di molti nuovi minimi dello stesso ordine della soluzione.
In Tabella 3.5 sono riportati i valori dei nuovi minimi e in ultimo quello del vero minimo.

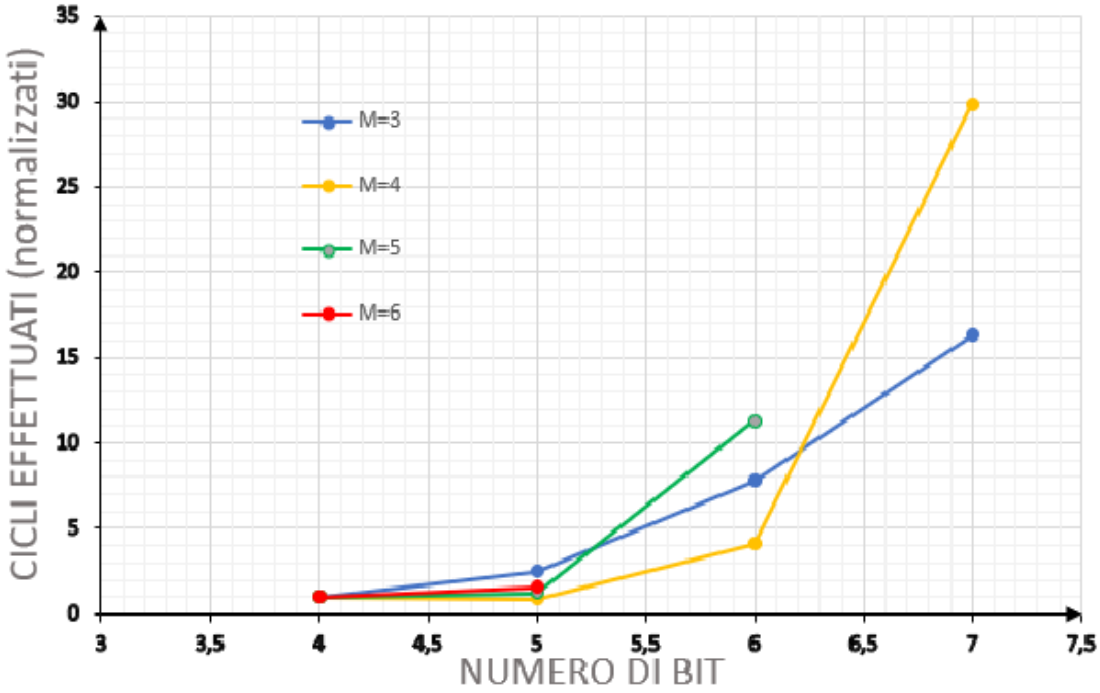


Figura 3.4: SIMPLE 1, CICLI VS NUMERO DI BIT

Il fatto che il secondo ed il terzo valore siano uguali ma relativi a cromosomi con segni invertiti è una diretta conseguenza del fatto che la fitness è costruita come un modulo quadro. Si può pensare di sfruttare questa simmetria per semplificare il problema: fissando il segno del gene i-esimo, si sta scartando l'insieme di cromosomi che ha per quel gene il segno opposto e questo porterebbe ad una "divisione dei mondi", quello con gene i-esimo positivo e quello con gene i-esimo negativo. Se, ad esempio, si sceglie di tenere il mondo positivo, lo spazio totale di esplorazione viene effettivamente dimezzato, riducendo i tempi di ricerca.

FITNESS	CROMOSOMA
$1,46001156\mathrm{E}-07$	$(0,0,-6,0,0,-12,0,3,0,0,0,0,12,0,-3,0,0,0,0,0,0,0)$
$6,48894058\mathrm{E}-08$	$(0,0,-4,0,0,-8,0,2,0,0,0,0,8,0,-2,0,0,0,0,0,0,0)$
$6,48894058\mathrm{E}-08$	$(0,0,4,0,0,8,0,-2,0,0,0,0,-8,0,2,0,0,0,0,0,0,0)$
$1,62223515\mathrm{E}-08$	$(0,0,2,0,0,4,0,-1,0,0,0,0,-4,0,1,0,0,0,0,0,0,0)$

Tabella 3.5: Minimi e relativi cromosomi nuovi (prime tre righe), minimo e cromosoma veri (ultima riga).

Dovendo tener conto del caso in cui il gene i-esimo selezionato sia nullo, la condizione da imporre è che esso sia, ad esempio, *maggiore o uguale a zero*.
In questo caso, i parametri fissi sono stati portati ai seguenti valori:
$\mathrm{NI} = 800$, $\% = 5$, $\mathtt{immutati} = 1$, $\mathtt{ciclimax} = 30000$.

3.3.4 Il successo a dodici variabili

Il caso più complesso sottoposto all'algoritmo prevedeva la ricerca dei "dodici" coefficienti della funzione SIMPLE 1 (vedi Appendice A).
I parametri sono stati fissati come segue :
$\% = 5, \mathrm{L} = 4, \mathrm{NI} = 400, \mathtt{ciclimax} = 2000000, \mathtt{immutati} = 1.$
Come risulta dalla tabella 3.6, su quattordici run soltanto due hanno restituito la soluzione.

C.E.	FITNESS	CROMOSOMA
2000000	1.7498826127848588E − 004	$(-6, 0, -6, 0, 0, 0, 0, 0, -1, 0, 6, 0)$
277543	4.5451452024281025E − 007	$(0, 2, 0, 4, 0, -1, 0, 0, 0, -4, 1, 0)$
2000000	7.1544361999258399E − 003	$(3, -4, 3, 4, 0, -4, 0, 0, 0, -4, -1, -1)$
2000000	1.7498826127848588E − 004	$(6, 0, 6, 0, 0, 0, 0, 0, 1, 0, -6, 0)$
2000000	1.7498826127848588E − 004	$(-6, 0, -6, 0, 0, 0, 0, 0, -1, 0, 6, 0)$
2000000	1.7498826127848588E − 004	$(-6, 0, -6, 0, 0, 0, 0, 0, -1, 0, 6, 0)$
2000000	7.1544361999258399E − 003	$(3, -4, 3, 4, 0, -4, 0, 0, 0, -4, -1, -1)$
720203	4.5451452024281025E − 007	$(0, 2, 0, 4, 0, -1, 0, 0, 0, -4, 1, 0)$
2000000	1.7498826127848588E − 004	$(-6, 0, -6, 0, 0, 0, 0, 0, -1, 0, 6, 0)$
2000000	1.7498826127848588E − 004	$(-6, 0, -6, 0, 0, 0, 0, 0, -1, 0, 6, 0)$
2000000	1.7498826127848588E − 004	$(6, 0, 6, 0, 0, 0, 0, 0, 1, 0, -6, 0)$
2000000	1.7498826127848588E − 004	$(6, 0, 6, 0, 0, 0, 0, 0, 1, 0, -6, 0)$
2000000	6.6838762722909451E − 003	$(0, -4, 0, -2, 0, -1, 0, 0, 0, 2, 0, 0)$
2000000	1.7498826127848588E − 004	$(6, 0, 6, 0, 0, 0, 0, 0, 1, 0, -6, 0)$

Tabella 3.6: SIMPLE 1 a dodici geni; C.E. sta per Cicli Effettuati.

Questo test rappresenta comunque un importante successo, in quanto rende evidente la capacità esplorativa dell'algoritmo genetico e dà motivo di credere che, migliorando le condizioni di lavoro, esso possa condurre a risultanti di un maggiore livello di interesse.

3.3.5 Un problema NP-hard.

Eseguendo un fit esponenziale (Figura3.5) è stato possibile determinare il grado di complessità del problema.

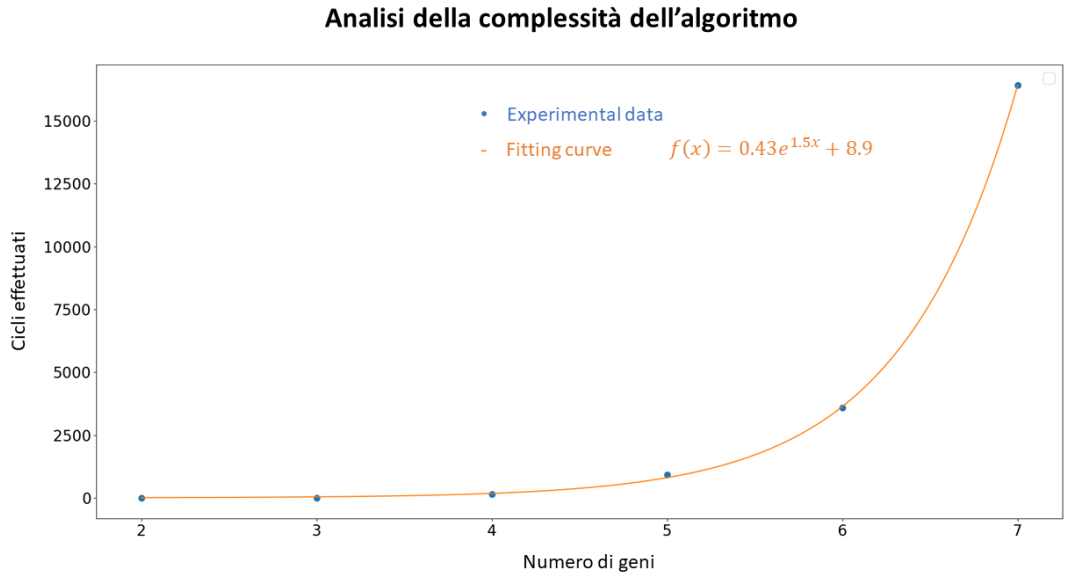


Figura 3.5: Fit esponenziale per la determinazione della complessità dell'algoritmo.

L'andamento del numero di cicli necessari per l'esecuzione in funzione del numero di geni M (la dimensione del problema) si adatta perfettamente all'andamento esponenziale. Ciò consente di classificare il problema affrontato nella classe degli **NP-hard (nondetermistic polynomial-time hard problem)**.
I punti scelti sono i valori che compaiono in Tabella 3.1 corrispondenti al parametro $\% = 10$, relativi alla funzione SIMPLE 1; il fit è stato eseguito in ambiente Python.

Conclusioni

La problematica affrontata nel presente lavoro di tesi è se una procedura come un algoritmo genetico sia appropriata nella ricerca della soluzione di sistemi lineari che si incontrano nel calcolo perturbativo della teoria quantistica dei campi.
Il risultato più interessante sotto questo punto di vista è quello ottenuto dallo studio della minimizzazione della funzione SIMPLE 1 a 12 geni: i successi ottenuti lasciano credere che con un miglioramento delle condizioni di lavoro da un lato e un perfezionamento dell'algoritmo dall'altro, si possa essere in grado di risolvere casi anche più complessi con un carico computazionale e una tempistica ragionevolmente soddisfacenti. Secondo la nostra analisi, i miglioramenti dovrebbero riguardare due aspetti: l'ottimizzazione nella scelta dei parametri richiesti dal programma e lo sviluppo di una infrastruttura distribuita di calcolo che sfrutti appieno le proprietà di parallelismo dell'algoritmo genetico.
Il primo può essere ottenuto a partire da una più approfondita analisi dell'algoritmo, in particolare tramite la trattazione dei seguenti punti:

1. migliorare le stime dei dati presi, mediando su un maggior numero di esecuzioni;

2. spingere lo studio degli andamenti funzionali a dimensioni successive ($M > 6$);

3. migliorare l'analisi della dipendenza dal numero iniziale di cromosomi, eseguendo uno studio del tempo di esecuzione in funzione di NI.

Perfezionare l'algoritmo, invece, vuol dire modificare il codice introducendo nuove strategie che rendano l'algoritmo più efficiente, come, ad esempio, le due seguenti:

1. operare la "divisione dei mondi" esposta nel Capitolo 3. Ci si aspetta che, mettendo l'algoritmo nelle condizioni di lavorare su una sola metà dell'intero spazio delle configurazioni, i tempi di esecuzioni vengano significativamente ridotti;

2. rendere continuo lo spazio da esplorare, in modo da infittire la ricerca ed evitare che l'algoritmo superi la soluzione senza individuarla.

La trattazione del paragrafo 1.3 intende comunicare il peso che la precisione nel calcolo teorico assume nella ricerca scientifica in quanto contribuisce attivamente al progresso. Migliorare le predizioni teoriche vuol dire in maniera concreta

setacciare le ipotesi e le idee che nascono nel tentativo di descrivere la realtà fisica, scartando quelle che si rivelano essere non idonee a tale scopo.
Se un algoritmo genetico fosse in grado di trovare i coefficienti delle equazioni differenziali negli integrali master, un passo importante si compirebbe nel calcolo dei diagrammi che appaiono al secondo ordine della teoria perturbativa.
Il presente lavoro di tesi, seppur ancora lontano da questo obbiettivo, rappresenta il primo tentativo di formulare il problema del calcolo delle correzioni radiative mediante l'impiego di algoritmi di tipo genetico, e conduce a pensare che un opportuno approfondimento dell'approccio proposto potrebbe portare ad interessanti risultati.

Appendice A

SIMPLE 1 e SIMPLE 2

SIMPLE 1

La SIMPLE 1 è relativa all'equazione differenziale più semplice del sistema, ovvero :

$$\frac{\partial I_9}{\partial m_t} = \left[-2(-5+d)(m_H - 4m_t)^{-1}\right] I_9 + \left[\frac{1}{2}(-4+d) m_t^{-1}(m_H - 4m_t)^{-1}\right] I_2$$

Il numero di parametri della soluzione di prova è in gran parte determinato dall'analisi dimensionale. Si consideri come riferimento la dimensionalità dell'I_9, che verrà indicata come $[I_9] = dim$. Quindi, $\frac{\partial I_9}{\partial m_t}$ ha dimensione $dim \times [M]^{-2}$.
Operando il minimo comune multiplo, il termine $(m_H - 4m_t)$[1] diviene il polinomio-coefficiente per la derivata, a cui si assegna $grado = 2$, per cui sono necessari 3 variabili. Il termine d contribuisce con un fattore 2, per un totale di 6 variabili per la derivata.
Di conseguenza, I_9 dovrà avere un polinomio-coefficiente lineare, aggiungendo altre 2 variabili, raddoppiate da d, quindi 4 variabili.
L'I_2 ha già dimensione $dim \times [M]$, quindi il polinomio corrispondente é costante (due coefficienti considerando l'effetto di d).
In totale si hanno $6 + 4 + 2 = 12$ variabili.
In generale, quest'equazione presenterebbe anche un termine in I_1 che, però, ha coefficiente nullo nella combinazione. Esso ha la stessa dimensione di I_2, dunque tenendone conto, il numero di variabili salirebbe a 14.
Nel lavoro svolto, si è usata una fitness dipendente da 22 parametri. Questo perché si è voluto tener conto della dipendenza da m_b per mantenersi nella massima generalità.

Il cromosoma soluzione è :

$$a(22) = (0, 0, 2, 0, 0, 4, 0, -1, 0, 0, 0, 0, -4, 0, 1, 0, 0, 0, 0, 0, 0, 0)$$

a cui corrisponde un valore di fitness: $1,62223515E-08$.

[1] Si ricordi che m_H e m_t sono in realtà quadrati di masse.

SIMPLE 2

La seconda equazione studiata è la SIMPLE 2 :

$$\frac{\partial I_3}{\partial m_t} = \left[-\frac{1}{2}m_t^{-1}(4m_H - m_H d + 8dm_t - 36m_t)(m_H - 4m_t)^{-1}\right] I_3 + \\ + \left[\frac{1}{2}(-4+d)m_t^{-1}(m_H - 4m_t)^{-1}\right] I_1$$

Operando il minimo comune multiplo, il coefficiente della derivata diventa quadratico, il coefficiente di I_3 lineare, mentre quello di I_1 è costante. Il conteggio dei coefficienti è analogo al caso precedente, ottenendo così un totale di 22 variabili da determinare.
Il cromosoma soluzione è :

$$a(22) = (0,0,2,-1,0,8,0,0,0,-1,0,0,-4,0,1,0,0,0,0,0,0,0)$$

a cui corrisponde un valore di fitness: $1,45114734E-06$.

Bibliografia

[1] Ian Shipsey. *Vision and Outlook: The Future of Particle Physics*. 2017. arXiv: 1707.03711 [hep-ex].

[2] Mark Thomson. *Modern Particle Physics*. Cambridge University Press, 2013. DOI: 10.1017/CBO9781139525367.

[3] A. G. Grozin. *Integration by parts: An introduction*. 2011. arXiv: 1104.3993 [hep-ph].

[4] V. A. Smirnov. *Feynman integral calculus*. 2006.

[5] E. Jurcisinova e M. Jurcisin. «A general formula for analytic reduction of multi-loop tensor Feynman integrals». In: *Phys. Lett.* B692 (2010), pp. 57–60. DOI: 10.1016/j.physletb.2010.07.018.

[6] A. V. Smirnov e A. V. Petukhov. *The number of master integrals is finite*. 2010. arXiv: 1004.4199 [hep-th].

[7] John H. Holland. *Adaptation in Natural and Artificial Systems*. second edition, 1992. Ann Arbor, MI: University of Michigan Press, 1975.

[8] R Dawkins. *The Selfish Gene*. Oxford University Press, Oxford, UK, 1976.

[9] Tao Pang. *An Introduction to Computational Physics*. 2ª ed. Cambridge University Press, 2006. DOI: 10.1017/CBO9780511800870.

[10] S. Borowka et al. «pySecDec: a toolbox for the numerical evaluation of multi-scale integrals». In: *Comput. Phys. Commun.* 222 (2018), pp. 313–326. DOI: 10.1016/j.cpc.2017.09.015. arXiv: 1703.09692 [hep-ph].