Generazione di ampiezze di Feynman con Mathematica: progetto finale per il corso di Metodi Computazionali Per la Fisica

Cesare Carlo Mella

25 giugno 2020

1 Introduzione

La generazione di *grafi* è un problema ricorrente di alcuni ambiti della Fisica teorica. In questo progetto abbiamo affrontato il problema adattandolo ai processi di scattering di particelle. Il formalismo della Teoria Quantistica dei Campi, QFT, e la tecnica degli sviluppi perturbativi sono facilmente applicabili nel momento in cui si è in grado di *disegnare* tutti i possibili grafi soddisfacenti vincoli imposti dalla teoria. All'aumentare dell'ordine perturbativo e del numero di particelle iniziali e finali, la quantità di diagrammi possibili aumenta fattorialmente. Diventa rapidamente non pratico cercare tutte queste possibilità senza l'ausilio di uno strumento informatico. Il software Mathematica si presta alla manipolazione simbolica e permette una certa generalità.

Abbiamo sviluppato un codice in grado di:

- Generare tutti i possibili grafi soggetti ai diversi vincoli topologici imposti dalla teoria;
- Vestire i grafi, assegnando proprietà fisiche alle sue componenti;
- Manipolare gli oggetti ottenuti rispettando le leggi fisiche di conservazione e le così dette *regole di Feynman*;
- Generare le ampiezze di probabilità per un processo d'urto per due teorie, la QED a livello albero e la Scalar QED fino al primo loop;

2 Generazione dei grafi

2.1 Preliminari

Riportiamo due definizioni fondamentali, quella di grafo e di grafo connesso:

Definizione 2.1. *Si dice grafo una coppia $G = \{V, E\}$ di insiemi, dove V è detto insieme dei vertici ed E è l'insieme degli archi. Un arco è una coppia di elementi di V.*

Definizione 2.2. *Un grafo è connesso se per ogni coppia di vertici esiste sempre un percorso, ovvero una sequenza di archi, che li connette.*

Diamo due esempi e una loro rappresentazione grafica:

$G_1 = \{\{1, 2, 3, 4, 5, 6\}, \{(1, 2), (3, 4), (1, 5)\}\}$;

$G_2 = \{\{a, b, c, d, e\}, \{(a, b), (b, d), (a, e), (a, c)\}\}$.

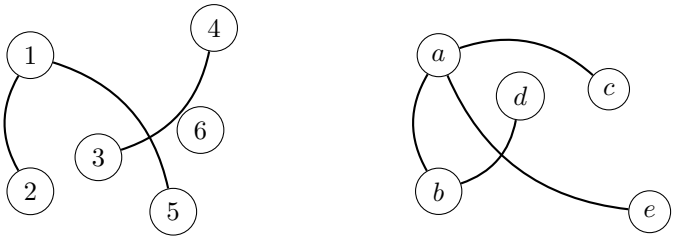


Figura 1: Esempio di un grafo non connesso e di un grafo connesso.

2.2 Generazione dei grafi con Mathematica

Per i nostri scopi possiamo limitarci a considerare solamente i grafi connessi. In questi casi tutta l'informazione è contenuta nella lista degli archi. Omettiamo dunque la lista dei vertici. Inoltre, accadono le seguenti due:

- Un vertice ha un solo arco che lo connette al resto del grafo: corrisponde a uno stato iniziale o finale (particella entrante o uscente) e chiamiamo tale arco *linea esterna*. Individuiamo i vertici esterni con gli interi positivi;
- Le teorie che abbiamo sviluppato prevedono che i vertici non corrispondenti a stati iniziali e finali abbiano 3 o 4 archi entranti. Di questi, quelli che non sono *linee esterne* li chiamiamo *propagatori*. Vertici interni vengono individuati da interi negativi.

In questo modo, gli archi saranno coppie di interi che possono essere entrambi negativi (propagatori) o uno negativo e l'altro positivo (stati iniziali e finali). La convenzione che abbiamo adottato è quella di mettere sempre l'intero positivo al primo posto. Questo permette di individuare con molta facilità tutte le linee esterne. Allo scopo di generare tutti i diagrammi abbiamo sfruttato la possibilità, in Mathematica, di agire su una lista esistente generandone una nuova ed evitando di implementare esplicitamente un algoritmo di tipo "*ciclo for*". Il codice dispone di una funzione in grado di generare inserzioni di linee esterne creando nuovi vertici con tre linee o completando un vertice con tre linee a uno con quattro linee. La generazione di un nuovo vertice a tre è sempre possibile. Un vertice già completato a quattro linee entranti non potrà più essere modificato. Il conteggio del numero di linee evita che ne vengano aggiunte altre.
Più nel dettaglio, abbiamo definito un grafo di partenza,

$$G_3 = \{\{1,-1\},\{2,-1\},\{3,-1\}\}$$

sul quale vengono applicate le inserzioni.
Nel caso in cui si vogliano considerarieinserzioni di loop, il grafo di partenza è il *tadpole*,

$$G_{tadpole} = \{\{1,-1\},\{-1,-1\}\}$$

Generare grafi ad un loop inserendo vertici e completando quelli esistenti è ancora sufficiente ma bisogna prestare attenzione ad eventuali grafi uguali che vanno eliminati. Per esempio, si consideri il diagramma

$$\{\{1,-1\},\{-1,-2\},\{-1,-2\},\{2,-1\}\}$$

in cui compaiono due linee uguali. Nel tentativo di raggiungere la situazione con tre linee esterne l'algoritmo incontrerà questo grafo di esempio e tenterà di aggiungere linee esterne in tutti i modi possibili, tra i quali l'inserzione di un vertice a tre sulla linea interna $\{-1,-2\}$. Questo accadrà due volte, generando due grafi indistinguibili. Uno dei due è da scartare.
In fig. 2.2 un esempio di generazione: dal tadpole a due linee esterne.

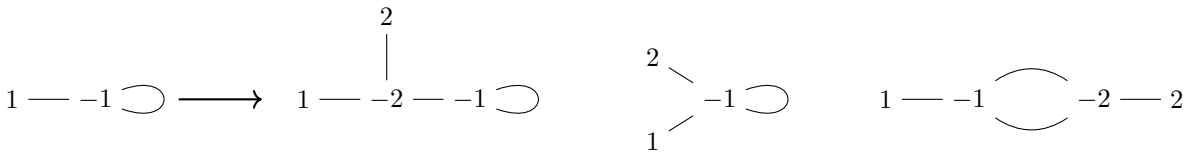


Figura 2: Esempio di costruzione grafi a partire dal tadpole.

E' possibile verificare l'affidabilità del codice generando manualmente i grafi con quattro e cinque linee esterne. Nel primo caso si ottengono 4 diagrammi, nel secondo caso 25. Come si può osservare in fig. 3, andare oltre diviene rapidamente impraticabile.

Una verifica indiretta dei tempi di generazione dei grafi sembra essere compatibile con una legge fattoriale, fig. 4.

Linee esterne	3	4	5	6	7	8	9
Diagrammi	1	4	25	220	2485	343000	559405

Figura 3: Numero di diagrammi al crescere delle linee esterne

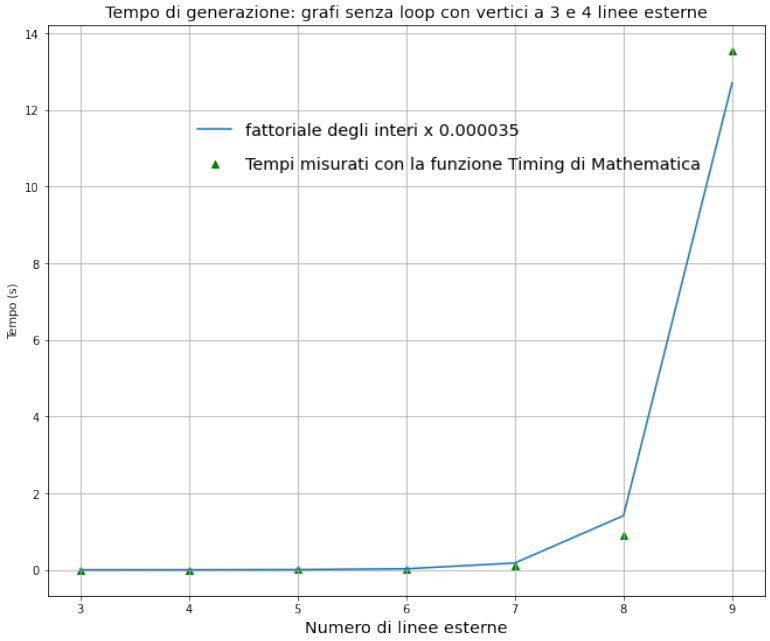


Figura 4: Tempi di generazione dei diagrammi

3 Vestire i grafi: assegnazione delle proprietà fisiche

3.1 Procedura generale e risultati

Il risultato della procedura precedente è una lista di tutti i possibili grafi con il corretto numero di linee esterne. Il passo successivo è quello di assegnare a ogni elemento di ciascun grafo tutte le proprietà fisiche rilevanti. Chiameremo il risultato di questa operazione *grafo vestito*. Mathematica permette di essere del tutto generali, dunque abbiamo assegnato tutte le proprietà che potenzialmente entrano in gioco nella generazione di un ampiezza: carica, massa, impulso, indici di Lorentz. In un secondo momento è possibile ignorare quelle che non servono. Inoltre, vengono applicate leggi di conservazione opportune (impulso, carica, ecc.). Di nuovo, ci basta definire una funzione che compia questo lavoro su un grafo generico e poi *mapparla* sull'intera lista di diagrammi.

Prima di procedere con una descrizione più dettagliata dell'algoritmo, riporto un esempio di quello che si ottiene. Consideriamo $g = \{\{1,-1\},\{2,-1\},\{3,-2\},\{4,-2\},\{-1,-2\}\}$, la cui rappresentazione è data dalla fig. 5. Separiamo le sue parti nella seguente lista:

- due vertici, indicati da -1 e -2;
- un propagatore $\{-1,-2\}$;
- quattro linee esterne $\{\{1,-1\},\{2,-1\},\{3,-2\},\{4,-2\}\}$.

La ragione per cui è utile separare queste componenti risiede nella successiva applicazione delle regole di Feynman.

Riportiamo, per brevità, il risultato dell'operazione di vestizione solo per alcuni di questi:

$$\{1,-1\} \quad \longrightarrow \quad \text{lineaext}\,[1,-1]\,[l\,[1]\,,m\,[1]\,,p\,[1]\,,Q\,[1]]$$

$$\{-1,-2\} \quad \longrightarrow \quad \text{prop}\,[-1,-2]\,[\{l\,[-1]\,,l\,[-2]\},m\,[-1,-2]\,,p\,[1]+p\,[2]\,,Q\,[1]+Q\,[2]]$$

$$\begin{aligned}\text{vertice} \quad -1 \quad \longrightarrow \quad & \text{vertice}\,[-1]\,\big[\{l\,[-2]\,,-m\,[-1,-2]\,,-p\,[1]-p\,[2]\,,-Q\,[1]-Q\,[2]\},\\ & \{l\,[1]\,,m\,[1]\,,p\,[1]\,,Q\,[1]\},\{l\,[2]\,,m\,[2]\,,p\,[2]\,,Q\,[2]\}\big]\end{aligned}$$

A questo livello non siamo interessati alla leggibilità dell'oggetto manipolato, essendo un risultato intermedio. Convenzione sui vertici: tutte le linee connesse a un vertice sono considerate *entranti*, nel senso che le quantità sono trasportate "verso" il vertice. Dunque, la conservazione della carica e dell'impulso in un vertice sono verificate sommando tutti i termini che vi compaiono.

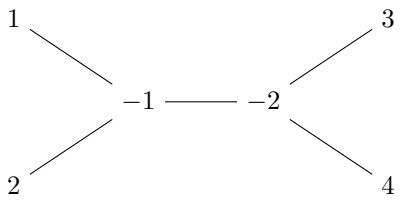


Figura 5: Rappresentazione grafica del diagramma di esempio g

3.2 La funzione *Vestigrafo* e l'identificazione dei loop

La funzione *Vestigrafo* si occupa di organizzare la struttura del grafo vestito. A questo scopo genera le tre liste per vertici, propagatori e linee esterne. Generando queste liste vengono applicate alle componenti considerate le proprietà richieste, utilizzando funzioni specifiche implementate. Prima di commentarle, un altro dettaglio di notazione: per assenza di ambiguità, tutte le proprietà che riguardano le linee esterne, e dunque un'etichetta intera positiva, vengono riferite al solo intero positivo. E' anche necessario applicare le leggi di conservazione. Concentriamoci su queste e su come abbiamo trattato gli impulsi non soggetti a conservazione.
Per individuare il problema, è opportuno descrivere il calcolo delle leggi di conservazione. Di nuovo, Mathematica offre strumenti utilissimi. In particolare la possibilità di riconoscere dei *patterns* specifici. Risulta così estremamente immediato "raccogliere" le incognite e costruire le leggi di conservazione: si impone la conservazione a ogni vertice con l'aggiunta della conservazione globale e si risolve il sistema usando come incognite le quantità interne (associate ai propagatori). Non è necessario scrivere le leggi per tutte le quantità fisiche. La legge di conservazione del quadrimpulso avrà la medesima struttura di quella della carica: possiamo sostituire le variabili associate all'impulso con quelle di carica.
D'altra parte, sappiamo che all'interno di un loop circola impulso non vincolato da alcuna legge, fig. 6. Il sistema di equazioni lineari che si andrebbe a scrivere non sarebbe dunque risolvibile, per la presenza di troppe incognite. Il modo in cui raccogliamo le quantità è riconoscendo la loro lettera associata (la *Head* in Mathematica). Dunque, per l'impulso, sarà un pattern che presenta la lettera p in testa. Si è dunque presentato il problema di cambiare nome all'impulso che circola nel loop, così che non venga inserito tra le incognite (dovrà comunque essere considerato nelle equazioni di conservazione). La soluzione del problema, che ora esporremo concettualmente, è stata implementato a livello di assegnazione della proprietà, non di scrittura della legge di conservazione.

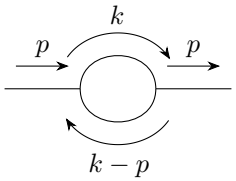


Figura 6: Loop in cui circola impulso k

Abbiamo implementato un algoritmo che fosse in grado di individuare il loop, così che potessimo applicare la modifica richiesta. Innanzitutto, una semplificiazione notevole: per costruzione, indipendentemente da quante inserzioni di linee esterne vengono eseguite sugli archi del loop, il vertice -1 deve appartenervi. Però non sappiamo quale, delle possibili linee collegate al vertice -1, appartiene al loop. L'algoritmo di individuazione è il seguente:

- Si eliminano dal grafo tutte le linee esterne (intero positivo al primo posto), che, sicuramente, non fanno parte del loop;
- Si eliminano dalle coppie rimaste tutti gli archi per cui *almeno uno* dei due indici *non* compare *almeno due volte* in tutta la lista;
- Si procede con il punto precedente fino a quando la successione non raggiunge un punto fisso: la lista degli archi che appartengono al loop;
- Sapendo che l'indice -1 compare sicuramente, si modifica uno qualsiasi degli archi che lo interessano (saranno due, dal momento che un loop è chiuso su se stesso) cambiando nome alle quantità fisiche.

Per convincersi che questo funziona è utile, per prima cosa, dare una rappresentazione grafica di quello che sta succedendo, fig. 7. Al primo passaggio vengono eliminate le linee esterne. Poi si eliminano tutte le linee in

cui un vertice ha solo un collegamento. Ma è chiaro che esse rappredentano le linee esterne del nuovo grafo. L'espediente del contare i collegamenti è necessario per il solo motivo che il nuovo grafo non ha gli indici positivi a segnalare le linee esterne. A questo punto appare ovvio che vengono eliminati tutti gli archi tranne quelli che costituiscono il loop. Infatti, chiudendosi su se stesso, non è mai possibile avere meno di due conteggi per ogni vertice.

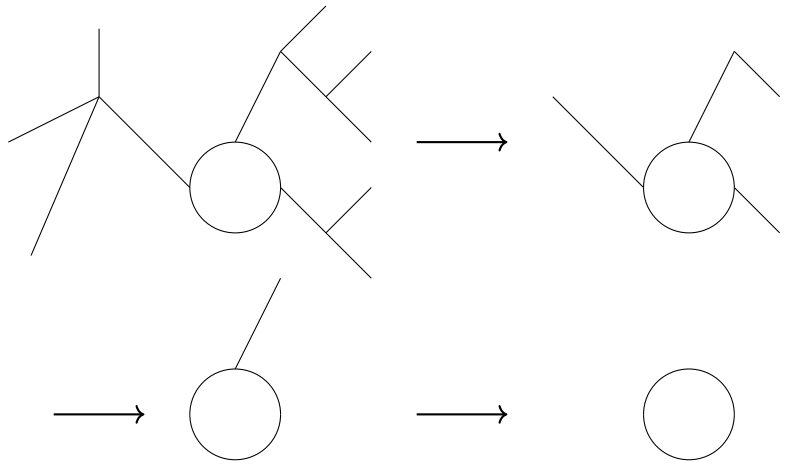


Figura 7: Rappresentazione grafica dell'algoritmo di ricerca del loop

Un commento sull'ordinamento delle proprietà. L'algoritmo costruisce le liste obbedendo all'ordinamento lessicografico. Poichè nel seguito del codice si accede alle variabili attraverso la loro posizione, è necessario prestare attenzione nel caso di modifiche e sviluppi del codice. In particolare, cercando di mantenere le posizioni attuali con nomi opportuni, non si dovrebbe incorrere in errori.

4 Dall'assegnazione dei processi all'ampiezza di Feynman

Il diverso formalismo matematico da riservare alle due teorie considerate ci ha portati a trattarle separatamente. In una teoria scalare come la SQED non siamo interessati all'ordinamento dei campi. Questo risulta in una notevole semplificazione, che ci ha permesso di sviluppare facilmente anche il calcolo del primo contributo di loop. Nel caso della QED siamo costretti ad affrontare il problema dell'ordinamento dei campi fermionici.

4.1 Scalar Quantum ElectroDynamics

In fig. 4.1 le regole di Feynman per la teoria. Sono ammessi tre vertici differenti: il vertice a tre linee con $2e^{\pm}$ e 1γ, e due vetrtici a quattro linee, uno con 2γ e $2e^{\pm}$ e uno con $2e^{-}$ e $2e^{+}$. Con e indichiamo la carica adimensionale della particella scalare. La costante di accoppiamento sarà $\alpha = \frac{e^2}{4\pi}$. Per quanto riguarda il vertice a quattro particelle scalari, la costante di accoppiamento è indicata da λ.

La procedura di assegnazione del processo si divide in due momenti:

- Fissati gli stati asintotici si procede alla sostituzione dei valori di carica e impulso. Nel caso dei loop si prova ad assegnare tutti i possibili valori di carica ammessi dalla teoria (gli impulsi andranno integrati, invece). Una funzione specifica passa in rassegna tutti i diagrammi proposti ed elimina quelli non fisici, ovvero in cui compaiono vertici non ammessi dalla teoria.
- Assegnazione delle regole di Feynman.

Un controllo iniziale Gli stati iniziali e finali sono di input al codice, così che è facile commettere errori assegnando stati fisicamente non sensati.
Un banale controllo di conservazione globale è posto all'inizio della procedura.

Assegnate le regole di Feynman rimane una semplificazione formale da fare: molte quantità presentano indici ripetuti che possono essere eliminati introducendo un'espressione formale per il prodotto scalare nello spazio di Minkowski, oo indichiamo con SP. A questo scopo abbiamo implementato una funzione che manipola gli indici dell'espressione, semplificandoli il più possibile.
Il risultato di questa operazione è l'espressione simbolica dell'ampiezza di Feyman. L'ultimo operazione da

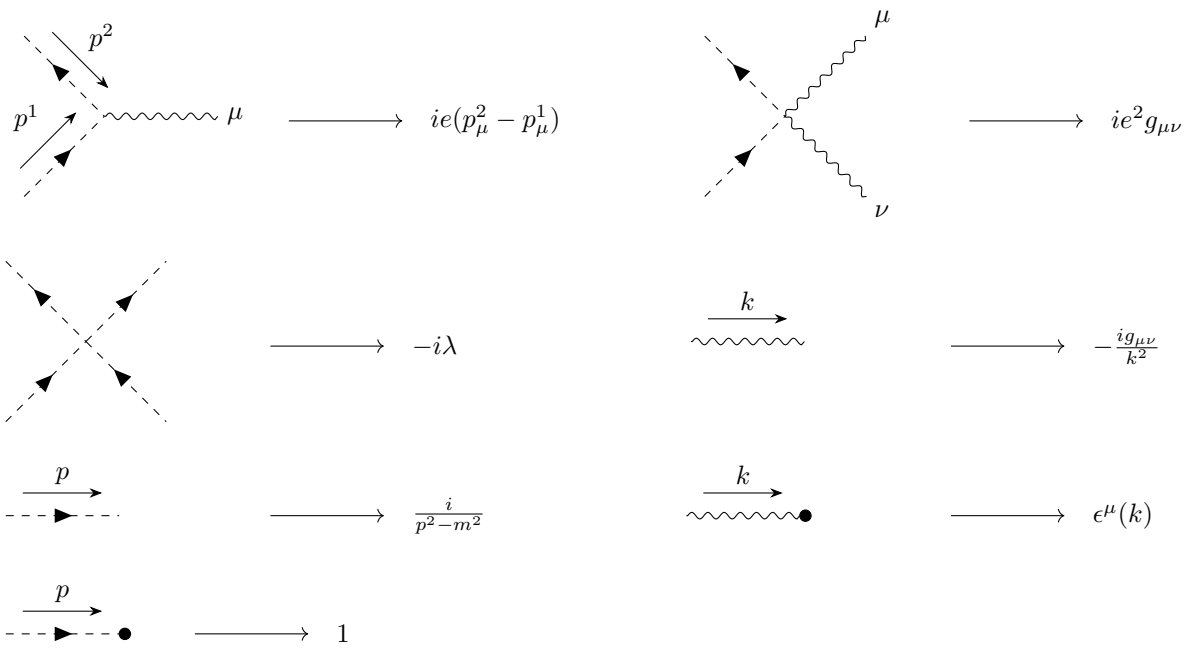


Figura 8: Regole di Feynman per la SQED

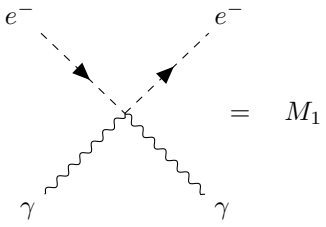
eseguire è il modulo quadro delle ampiezze. Il nostro codice è preparato per studiare processi non polarizzati. Sommando sulle polarizzazioni perdiamo i vettori di polarizzazione dei fotoni iniziali e finale e guadagniamo un'ulteriore contrazione di Lorentz. La procedura consiste di sostituzioni di patterns, dunque non merita ulteriori spiegazioni.

4.2 Controllo dei risultati per un caso semplice: Scattering $e^-\gamma \to e^-\gamma$

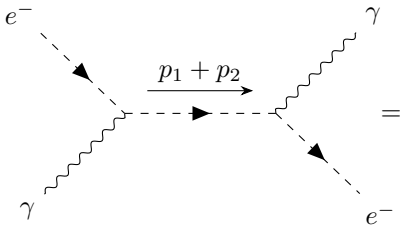
Allo scopo di verificare l'affidabilità del codice ho svolto la verifica diretta per alcuni processi a livello albero. Ne riporto uno di esempio:

$$e^-(p_2) + \gamma(p_1) \longrightarrow e^-(p_1 + p_2 - p_3) + \gamma(p_3)$$

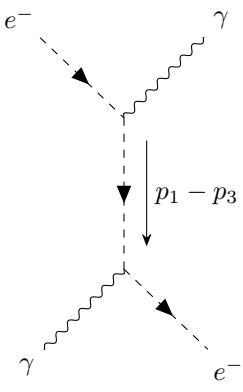
All'ordine più basso $M = M_1 + M_2 + M_3$, dove le tre ampiezze corrispondono ai tre diagrammi:



$= \quad M_1$



$= \quad M_2$



$= \quad M_3$

L'espressione esplicita delle ampiezze è data da:

$$M_1 = i\ e\ g^{\mu\nu}\epsilon_\mu(p_1)\epsilon^*_\nu(-p_3)$$

$$M_2 = -e^2\ (2p_2+p_1)^\mu\epsilon_\mu(p_1)\ (2p_1+2p_1-p_3)^\nu\epsilon^*_\nu(-p_3)\ \frac{1}{(p_1+p_1)^2-m^2}$$

$$M_3 = -e^2\ (p_3-2p_2)^\mu\epsilon^*_\mu(-p_3)\ (2p_3-2p_2-p_1)^\nu\epsilon_\nu(p_1)\ \frac{1}{(p_2-p_3)^2-m^2}$$

Le ampiezze si possono scrivere come $M = \sum_{i=1}^{3}\epsilon_\mu(p_1)\epsilon^*_\nu(-p_3)M_i^{\mu\nu}$. Il modulo quadro per un processo non polarizzato è dato da:

$$X = \frac{1}{2}\sum_{i,j=1}^{3} M_i^{\mu\nu}M^*_{j\ \mu\nu}$$

Per esempio

$$i=j=1 \qquad X_{11} = 2e^4$$

$$i=1, j=2 \qquad X_{12} = -\frac{e^4}{2}(2p_2+p_1)^\mu(2p_1+2p_2-p_3)_\mu\frac{1}{(p_1+p_2)^2-m^2}$$

e altri 7 termini. Il risultato fornito dal nostro codice coincide esattamente. I primi due termini in output sono proprio (osserviamo che c'è un fattore 2 davanti al secondo addendo, in effetti si sommano due termini uguali):

```
                    4
             4     e  SP[p[1] + 2 p[2], 2 p[1] + 2 p[2] - p[3]]
Out[2]= 2 e   - ------------------------------------------- -
                                   2                    2
                          -m[-1, -2]  + (p[1] + p[2])
```

Problemi incontrati La gestione dei prodotti complessi è una delle parti più strutturate. Particolarmente utile è il comando *Outer* che permette di utilizzare il calcolo esterno per organizzare i prodotti. Uno sviluppo futuro di questo progetto potrebbe partire dal chiedersi se è possibile snellire questa parte, che richiede, nel caso del conto a un loop, molto tempo (qualche decina di secondi già per 4 gambe esterne). Per esempio, nel calcolo al primo loop abbiamo diviso in tre classi i prodotti complessi, in modo da gestire meglio l'inserimento dei corretti integrali sulle variabili impulso: si potrebbe cercare di includere un "flag" che intichi la presenza di un integrale da inserire già in precedenza, così da non dover suddividere i prodotti complessi.

4.3 Quantum ElectroDynamics

In fig. 9 le regole di Feynman per la teoria. E' concesso solo il vertice con tre linee, due fermioniche e una bosonica. Usiamo le stesse notazioni della sezione precedente. Introduciamo le matrici *gamma di Dirac*, definite come set qualsiasi di matrici 4x4, $\{\gamma^\mu, \mu = 0, ..., 3\}$, che soddisfano la seguente relazione definitoria[1]: $\{\gamma^\mu, \gamma^\nu\} = 2g^{\mu\nu}I$

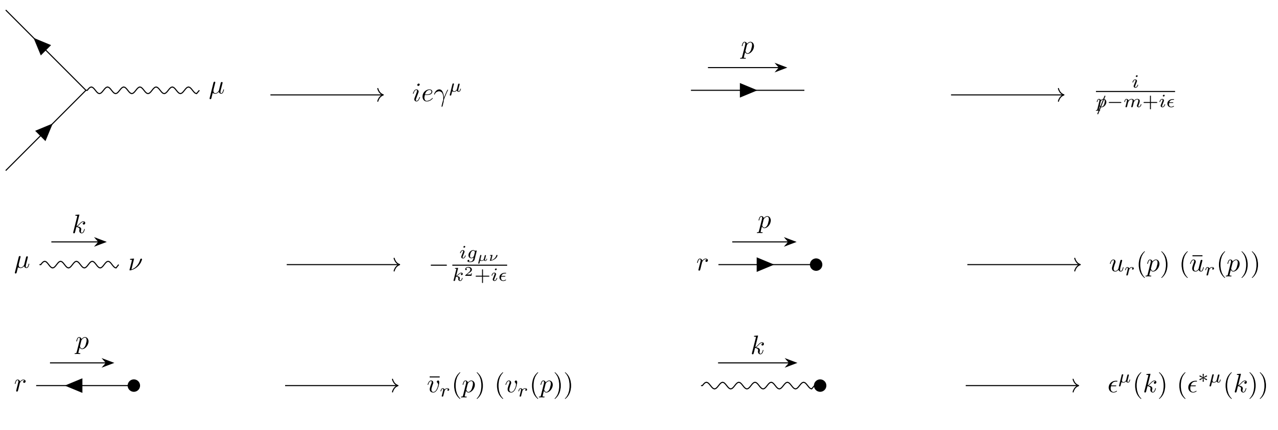


Figura 9: Regole di Feynman per la QED. In parentesi il caso di particella uscente.

[1]Stiamo definendo un'algebra di Clifford

Anche per la QED, l'assegnazione del processo può essere divisa nei due momenti sopra indicitati. La struttura del primo step (sostituzione di carica e impulsi, controllo dei diagrammi) non cambia. Le difficoltà di implementazione della QED si incontrano cercando di applicare le regole di Feynman. Gli oggetti che stiamo manipolando non sono più scalari, bensì spinori e matrici. Per preservare l'ordinamento degli oggetti abbiamo introdotto una funzione *FermionChain*. Infatti, poichè di default le funzioni preservano l'ordinamento dei loro argomenti, è possibile gestire facilmente le varie parti senza che vengano scambiati spinori e matrici. Le catene di fermioni e propagatori fermionici vengono costruite seguendo a ritroso il percorso di ogni freccia, aggiungendo spinori e matrici gamma opportunamente. La catena si chiude quando si incontra una linea esterna. Di seguito vengono aggiunti propagatori e stati asintotici di fotoni.
Come già anticipato, abiamo trattato solo il tree level: non essendoci loop è sufficiente passare in rassegna le linee esterne per essere sicuri di non perdere alcuna catena di fermioni. Per esempio, consideriamo uno dei due diagrammi che contribuiscono allo scattering Compton all'ordine più basso, fig. 10 (polarizzazioni e impulsi non sono rilevanti e vengono omessi).

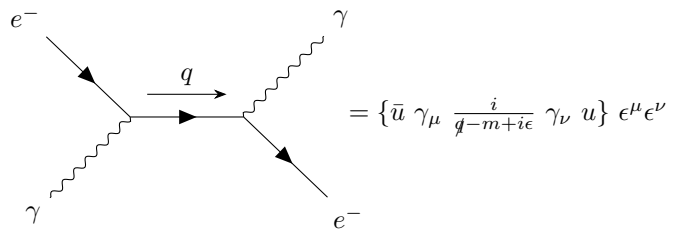


Figura 10: Contributo di canale s allo scattering Compton

Partendo dall'elettrone finale e seguendo a ritroso la freccia, si ricostruisce la catena correttamente. In seguito si aggiungono i vettori di polarizzazione dei fotoni.

4.4 Ordinamento Normale dei campi fermionici

Più sottile è il problema dell'ordinamento normale. Suddiviamolo in due sottoproblemi:

- Portare tutti i gli operatori di creazione a sinistra di quelli di annichilazione;
- Ordinare i due gruppi in modo da ricostruire gli stati sullo spazio di Hilbert sempre nello stesso ordine;

Per prima cosa, ricordiamo che il vertice di QED è

$$-e\ \bar{\psi} \not{A} \psi$$

Possiamo ignorare il campo del fotone, che comuta con tutto, e ci interessiamo a quelli fermionici. Ci riferiremo ad essi, per semplicità, con le lettere maiuscole dell'alfabeto. Eseguiamo la seguente procedura:

1. Se la fermion chain considerata inizia e termina nello stesso gruppo di stati (iniziale o finale), allora riguarda solo operatori di costruzione o di distruzione. Non tengo segno di alcuno scambio. Comincio a riempire una lista nell'ordine in cui compaiono gli stati iniziali e finali, utilizzando l'intero positivo delle linee esterne come segnaposto.
2. Se la fermion chain considerata inizia nel primo gruppo (positrone entrante) ma termina in un altro gruppo (positrone uscente, necessariamente), allora guadagno un segno. Continuo a riempire la lista degli stati iniziali e finali nell'ordine in cui compaiono.

Per capire perchè questo funziona osserviamo che, in generale, è sempre possibile spostare i campi fermionici a coppie in modo tale che la sequenza di campi è della forma seguente:

$$(A\ ContractionList\ B)(C\ ContractionList\ D)(E\ ContractionList\ F)...$$

Questo significa che nel riordinare le catene tra di loro non compare alcun segno di scambio. Dunque, l'ordine delle catene sarà quello in cui compaiono durante l'esecuzione del codice, qualunque esso sia. Il punto 1 dovrebbe ora essere ovvio: un fermion chain che parte e termina nello stesso gruppo di stati ha, come campi agli estremi, due costruttori o due distruttori. Nel primo caso immagino di spostarli sulla sinistra *in coda* ad altri eventualmente presenti, nel secondo caso immagino di spostarli sulla destra, *in testa* ad altri eventualmente presenti.
Consideriamo ora il caso di una catena che parte dagli stati iniziali, attraversa il grafo, e termina con uno stato finale. Ricordandoci che costruiamo le catene seguendo a ritroso le frecce, questo può accadere solo come:

positrone entrante - propagatori - positrone uscente. In termini di campi si tratta di avere un costruttore sulla destra e un distruttore sulla sinistra. E' necessario eseguire uno scambio (punto 2). In seguito, si sposta sulla sinistra, in coda agli altri operatori di creazione, l'operatore che crea il positrone, mentre quello che lo distrugge viene mandato sulla destra, in testa. Nel caso in cui si abbia un elettrone entrante e uno uscente (se si vuole, una catena di positroni con direzione del tempo invertita), non c'è alcuno scambio da fare.
Dunque, abbiamo tenuto conto degli scambi necessari per posizionare costruttori e distruttori in ordinamento normale e abbiamo costruito una lista dell'ordine in cui compaiono. Ricordiamo che abbiamo utilizzato gli interi positivi che caratterizzano (univocamente!) le linee esterne per costruire le liste. Ci riferiamo a un ordinamento di riferimento, quello canonico $\{1, 2, ..., n\}$ e calcoliamo la segnatura delle permutazioni. Facciamo lo stesso per ogni ampiezza. Il segno che otteniamo è esattamente quello che stiamo cercando. Infatti: Supponiamo che le ampiezze abbiano ordinamento identificato dai seguenti elementi del gruppo delle permutazioni:

$$\{\sigma_1, \sigma_2, ..., \sigma_N\}$$

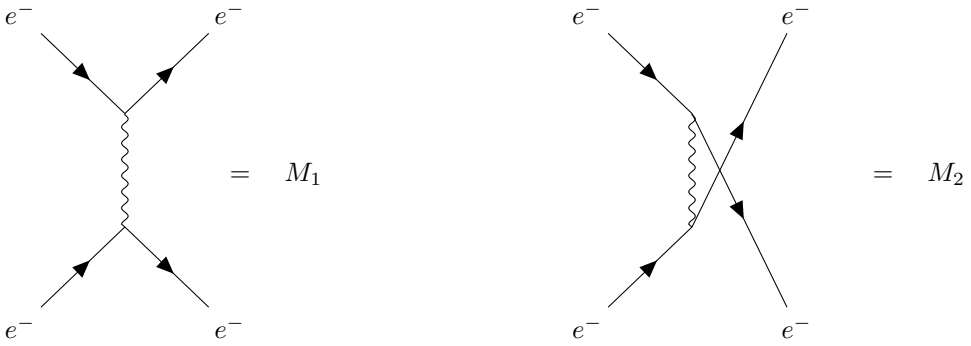
Siamo interessati al segno che ci permette di andare da una permutazione a un'altra. Ordiniamo la prima ampiezza canonicamente e scegliamola come riferimento; allora, è sufficiente riordinare tutto canonicamente tenendo conto del segno della permutazione[2].

4.5 Controllo dei risultati: Scattering di Møller

Il caso dello scattering di elettroni permette di controllare che compaia il corretto segno relativo tra le ampiezze. Consideriamo dunque il processo

$$e^-(p_1) + e^-(p_2) \longrightarrow e^-(p_3) + e^-(p_1 + p_2 - p_3)$$

Al *tree level* contribuiscono due diagrammi, quello di canale t e quello di canale u, che differiscono per lo scambio dei due elettroni finali. Di seguito i diagrammi (omettiamo impulsi e polarizzazioni) e le espressioni associate:



$$M_1 = -ie^2 \left(\bar{u}(\vec{p}_3)\gamma^\mu u(\vec{p}_1)\right) \frac{g_{\mu\nu}}{(p_1 - p_3)^2} \left(\bar{u}(\vec{p}_1 + \vec{p}_2 - \vec{p}_3)\gamma^\nu u(\vec{p}_2)\right)$$

$$M_2 = ie^2 \left(\bar{u}(\vec{p}_1 + \vec{p}_2 - \vec{p}_3)\gamma^\mu u(\vec{p}_1)\right) \frac{g_{\mu\nu}}{(p_2 - p_3)^2} \left(\bar{u}(\vec{p}_3)\gamma^\nu u(\vec{p}_2)\right)$$

da confrontare con l'output fornito dal nostro codice:

```
                    2
Out[11]= {(-I e  FermionChain[Bar[SpinorU[p[1] + p[2] - p[3], Me]],
>        DiracMatrix[l[-1]], SpinorU[p[1], Me]]
>       FermionChain[Bar[SpinorU[p[3], Me]], DiracMatrix[l[-2]],
                                                                  2
>        SpinorU[p[2], Me]] g[l[-2], l[-1]]) / (p[2] - p[3])  ,
          2
>    (I e  FermionChain[Bar[SpinorU[p[1] + p[2] - p[3], Me]],
>        DiracMatrix[l[-1]], SpinorU[p[2], Me]]
>       FermionChain[Bar[SpinorU[p[3], Me]], DiracMatrix[l[-2]],
                                                                  2
>        SpinorU[p[1], Me]] g[l[-2], l[-1]]) / (p[1] - p[3])  }
```

[2]anche per la prima ampiezza è necessario calcolare il corretto segno

Osserviamo che compare un segno overall, dovuto alla scelta di ordinare la prima ampiezza di riferimento canonicamente.

Per completezza mostriamo la correttezza del modulo quadro per un termine:

$$X_{11} = \frac{1}{4}\sum_{spin}|M_1|^2 = \frac{e^4}{4}\frac{1}{(p_1-p_3)^4}\ Tr\Big(\frac{\not{p}_3+m_e}{2m_e}\gamma^\mu\frac{\not{p}_1+m_e}{2m_e}\gamma^\nu\Big)\ Tr\Big(\frac{\not{p}_1+\not{p}_2-\not{p}_3+m_e}{2m_e}\gamma_\mu\frac{\not{p}_2+m_e}{2m_e}\gamma_\nu\Big)$$

da confrontarsi con:

```
                  4         Me + DiracSlash[p[1] + p[2] - p[3]]
Out[12]= (e  Traccia[-----------------------------------, DiracMatrix[l[-1]],
                                    2 Me

          Me + DiracSlash[p[2]]
>         ---------------------, DiracMatrix[l[-2]]]
                  2 Me

                 Me + DiracSlash[p[3]]
>        Traccia[---------------------, DiracMatrix[l[-1]],
                        2 Me

          Me + DiracSlash[p[1]]                                      4
>         ---------------------, DiracMatrix[l[-2]]]) / (4 (p[1] - p[3]) ) +
                  2 Me
```

Controllo delle masse Un controllo ulteriore è quello svolto sulle masse: la QED non cambia i sapori, dunque dobbiamo stare attenti a fare in modo che non compaiano vertici in cui cambia la massa del leptone. In realtà questo controllo agisce in due modi: permette sia di segnalare un errore nell'inserimento dei dati, sia di eliminare i diagrammi non fisici nel caso in cui si voglia considerare un processo in cui le masse degli stati iniziali e finali cambiano. Per esempio, nella produzione di coppie $\mu^+\mu^-$ in annichilazione e^+e^- viene eliminato il canale t, non fisico. Abbiamo testato il corretto funzionamento del nostro software in questi casi.

5 Conclusioni

Come anticipato al termine della sezione riguardante la Scalar QED, è richiesto uno *speed up* del codice, in particolare sulla parte che tratta i loop. Per quanto riguarda la QED potrebbe essere utile aggiungere controlli per testare la correttezza del codice. Nonostante la QED porti a inevitabili complicazioni di manipolazione simbolica rispetto alla teoria scalare, la presenza di un solo vertice permette di accelerare il codice e migliorarne le performance, rendendole simili al caso scalare. Per quanto riguarda sviluppi futuri della parte di QED, al primo loop e oltre, è possibile che il codice necessiti di una parziale riscrittura. Alcune procedure fallirebbero sicuramente.

La correttezza dei risultati è sicuramente verificata laddove è possibile fornire una soluzione autonomamente ed eseguire un cronfonto diretto.

Infine, potrebbe essere opportuno migliorare la leggibilità del risultato, già a un livello accettabile.