UNIVERSITÀ DI PISA
FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI
Dipartimento di Fisica

# SVILUPPO DI UNO STRUMENTO AD ALTO LIVELLO PER DIAGNOSTICA, CONTROLLO E FEEDBACK DI UN ACCELERATORE GENERICO

Elaborato di Laurea Specialistica

Francesco Guatieri


UNIVERSITÀ DI PISA

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI

Corso di Laurea Specialistica in
FISICA DELLE INTERAZIONI FONDAMENTALI


# SVILUPPO DI UNO STRUMENTO AD ALTO LIVELLO PER DIAGNOSTICA, CONTROLLO E FEEDBACK DI UN ACCELERATORE GENERICO


Candidato:

Francesco Guatieri

**Relatore interno:**

Prof. Giovanni Batignani .....

**Relatori esterni:**

Dott.ssa Catia Milardi .....

Prof. Luigi Rolandi .....

**Candidato:**

Francesco Guatieri .....



Sessione di Laurea 24 ottobre 2013
Anno Accademico 2012-2013

*a Davide Orsucci*

*catalizzatore*
*della mia produttività*

*che mi ascolta e annuisce*
*anche quando non ha capito*
*poi mi chiede di rispiegare*
*finché non ho capito io*

*senza di lui non potrei*
*la metà di quello che faccio*

# Indice

**Introduzione** **3**

**1 Corolla** **6**
1.1 Obiettivi di design . . . . . 6
1.1.1 Astrazione e praticità . . . . . 6
1.1.2 Flessibilità e portabilità . . . . . 7
1.1.3 Semplicità di utilizzo . . . . . 8
1.2 Workbenching a più livelli . . . . . 8
1.3 Lo standard di comunicazione . . . . . 10
1.3.1 Il protocollo a *matrioska* . . . . . 10
1.3.2 Lo standard di tipo 0 . . . . . 15
1.3.3 Lo standard di tipo 1 . . . . . 16
1.4 Astrazione delle interfacce di macchina . . . . . 16
1.4.1 L'acceleratore di particelle generico . . . . . 17
1.4.2 Sistema a Tag . . . . . 18
1.4.3 L'interfaccia di macchina . . . . . 19
1.4.4 Modelling online . . . . . 20
1.5 Sistema semaforico ad albero . . . . . 21

**2 Le matrici di risposta** **23**
2.1 Caratteristiche della scatola nera . . . . . 23
2.1.1 L'evoluzione temporale spontanea . . . . . 23
2.1.2 Le isteresi . . . . . 25
2.2 Definizione della matrice di risposta . . . . . 26
2.3 Tecniche di misura delle matrici di risposta . . . . . 28
2.3.1 Misurazione diretta . . . . . 28
2.3.2 Misurazione mediante Monte Carlo . . . . . 29
2.3.3 Campionamento su singole direzioni . . . . . 31
2.4 Errore sperimentale . . . . . 32
2.5 Implementazione . . . . . 34

**3 Beam Base Alignment** **37**
3.1 Introduzione . . . . . 37
3.1.1 Individuazione di errori macroscopici . . . . . 38
3.1.2 Riduzione dei disallineamenti fisiologici . . . . . 39
3.1.3 Correzione del modello di macchina . . . . . 39
3.2 Beam Base Alignment efficace . . . . . 39
3.3 Effetti del disallineamento . . . . . 40
3.3.1 Sviluppo in multipoli . . . . . 40
3.3.2 Classificazione dei disallineamenti . . . . . 41
3.3.3 Effetti dei disallineamenti sulle funzioni ottiche . . . . . 42
3.4 Effetti dei disallineamenti sulle matrici di risposta . . . . . 46
3.5 Tecnica algoritmica . . . . . 48
3.5.1 Il tensore di risposta . . . . . 48
3.5.2 Misura del tensore di risposta . . . . . 50
3.5.3 Beam Base Alignment mediante tensore di risposta . . . 51
3.5.4 La fattorizzazione SVD . . . . . 51
3.6 Verifica del funzionamento . . . . . 54
3.6.1 Disallineamento lungo l'asse $\hat{x}$ . . . . . 55
3.6.2 Scostamenti lungo l'asse $\hat{y}$ . . . . . 58
3.6.3 Accuratezza della stima degli errori . . . . . 58
3.7 Prospettive di miglioramento . . . . . 60

**Bibliografia** **62**

# Introduzione

> *Gli dei* [...] *hanno ritenuto, a buon titolo, che non esista punizione più spaventosa di una fatica inutile e priva di significato.*
>
> – Albert Camus, *Il Mito di Sisifo*

C'è un sottile filo etimologico che segue lo sviluppo della moderna automazione. Nell'ottocento le macchine a vapore venivano dotate di meccanismi a retroazione che, regolando il flusso di vapore, mantenevano costante la velocità, la potenza o il momento torcente sprigionati dalla macchina. All'epoca tali meccanismi venivano chiamati *governor*, parola inglese che indica colui che dirige o conduce e che deriva dal latino *gubernator*, timoniere. Nella metà dell'ottocento, la nascente disciplina ingegneristica elettronica aprì la strada allo sviluppo di nuovi strumenti di regolazione che, basandosi sulla più flessibile arte della direzione di correnti, permettevano la realizzazione di meccanismi di controllo di più facile messa a punto e di maggiore affidabilità. Nasce così la disciplina cui nel 1948 verrà dato il nome di *cibernetica*, dal greco *kybernetes*, timoniere. Ancora un balzo attendeva di essere compiuto quando, nella seconda metà del novecento, l'avvento dell'era dei computer permise di introdurre il concetto di periferica virtuale, un oggetto informatico il quale implementava l'immagine astratta che un utente poteva avere di una periferica. Il compito di concretizzare l'azione di controllo, in questa nuova ottica spettò ad un software di traduzione che assunse il nome di *driver*, parola che in inglese, ancora una volta, indica il guidatore. Realizzati con dispositivi meccanici, elettronici o informatici, capaci di controllo a retroazione, di interfaccia con l'utente o di entrambe le cose, i sistemi di controllo costituiscono la nervatura centrale dell'intera tecnologia moderna.

In quest'ultimo secolo la disciplina del disegno e dell'organizzazione dei sistemi di controllo ha dovuto affrontare numerose e notevoli sfide, non ultima il design su grande scala di sistemi di regolazione atti a controllare meccanismi estremamente complessi, dotati di centinaia o addirittura migliaia di parametri. La prassi più comune per la progettazione di tali sistemi di controllo consiste nello sviluppo di un'architettura gerarchica, schematizzabile in una struttura ad albero capovolto, nella quale si evidenziano tre livelli chiave: un

livello di controllo *basso*, costituito da feedback locali con funzioni relativamente semplici ed inconsapevoli le une delle altre; un livello *intermedio*, volto a raccogliere e ridistribuire informazioni tra le singole periferiche appartenenti al sistema; quindi, un livello *alto* che, consapevole del quadro complessivo ottenuto incrociando le informazioni provenienti dalla totalità delle periferiche, dirige complessivamente l'azione della macchina[1]. Questo tipo di organizzazione non è prerogativa solo delle opere ingegneristiche umane, ma trova riscontro immediato nella strutturazione del sistema nervoso di quasi tutti gli animali superiori. Tra le macchine più complesse fino ad oggi costruite dall'uomo spiccano gli acceleratori di particelle; non sorprenderà quindi scoprire che, superata una dimensione minima, tutti i moderni acceleratori debbano giocoforza essere dotati di un sistema di controllo gerarchizzato.

Lo sviluppo dei livelli di controllo *basso* e *intermedio* di un acceleratore di particelle è un lavoro sensibilmente diverso dallo sviluppo del sistema di controllo ad *alto* livello. Affrontare lo sviluppo di un sistema di controllo locale comporta, infatti, principalmente difficoltà di carattere tecnico e attività di ottimizzazione. Si tratta di un puzzle nel quale è ben chiaro l'obbiettivo da raggiungere; resta da determinare quale sia il modo migliore per raggiungerlo. Lo stesso si può dire del livello di controllo *intermedio*. Quando si affronta lo sviluppo del livello di controllo *alto*, invece, oltre ad affrontare le consuete difficoltà tecniche è necessario tenere conto di una visione di insieme che descriva quali compiti si desidera che la macchina sia in grado di espletare. Per questo motivo è nello sviluppo del livello di controllo *alto* che la fisica di macchina entra maggiormente, mentre essa è presente in misura minore nei sistemi a *basso* livello e quasi del tutto assente nei sistemi *intermedi*.

Dal momento che lo sviluppo di una visione complessiva, articolata ed esaustiva della gamma completa di operazioni necessarie per il funzionamento di un acceleratore è per lo sviluppatore di sistemi di controllo un compito arduo e spesso lontano dalle proprie competenze, lo sviluppo del livello *alto* rischia spesso di ridursi all'implementazione di una infinità di piccoli applicativi destinati, ciascuno indipendentemente dall'altro, all'esecuzione puntuale di specifiche operazioni di misura o calibrazione. L'integrazione tra applicativi in tale scenario è spesso difficile quando non impossibile e costringe gli operatori a procedure lunghe e farraginose. Il costo di tale inefficienza è un dispendio di forza lavoro all'atto dell'operazione della macchina ed una fonte di distrazione che riduce la possibilità per gli operatori di concentrarsi sui risultati delle procedure in atto e sulla fisica di macchina a esse sottostante.

[1] Esiste, a dire il vero, anche la possibilità di basarsi sull'approccio inverso, sfruttando la cosiddetta *intelligenza di sciame*. Sebbene sperimentazioni di questo genere abbiano dato risultati promettenti, ad oggi nessuna è stata ancora in grado di dimostrare benefici sufficienti a giustificare la sostituzione massiccia dell'approccio tradizionale con sistemi di controllo distribuiti.

La mancata integrazione degli strumenti, inoltre, può in taluni casi arrivare a rendere impossibile l'esecuzione di specifiche procedure. Infine, la forza lavoro richiesta per l'integrazione di sistemi non nativamente pensati come integrati, spesso rende proibitiva l'introduzione di potenziali migliorie.

Nella prima parte di questo lavoro verrà descritto il mio tentativo di risolvere i sopraelencati problemi mediante lo sviluppo di un sistema di controllo ad alto livello altamente integrato ed utilizzabile su macchine differenti, a cui ho dato il nome di *Corolla*. L'implementazione di tale sistema è stata da me effettuata nell'arco di otto mesi di lavoro. Il design complessivo del sistema è stato reso possibile dalla profonda conoscenza della chiarissima dottoressa Catia Milardi in materia di sviluppo e operazione di acceleratori di particelle. Oltre all'implementazione del nucleo del sistema *Corolla*, ho proceduto allo sviluppo di alcune funzionalità di base del sistema di controllo, quindi di un sistema di misura del *Beam Base Alignment* basata sull'impiego di matrici di risposta. Nella seconda parte del presente lavoro verranno trattati i principi di funzionamento di tale strumento, le tecniche impiegate nel suo sviluppo e verrà discussa la fisica di macchina ad esso sottostante. Infine verranno tratte conclusioni derivanti dall'impiego pratico di tali strumento su di una simulazione dell'anello di accumulazione DAΦNE[1][2] del Laboratorio Nazionale di Frascati.

# Capitolo 1

# Corolla

> *Ma come può esistere un essere necessario totalmente intessuto di possibile?* [...] *Affermare l'assoluta onnipotenza di Dio* [...] *non equivale a dimostrare che Dio non esiste?*
>
> – Umberto Eco, *Il Nome della Rosa*

## 1.1 Obiettivi di design

Prima di procedere nell'analisi della strutturazione del sistema *Corolla*, è bene chiarire quali siano i punti fondamentali che ne hanno regolato la concezione, in modo da poter meglio comprendere e giustificare le scelte di design. Vediamo, quindi, quali devono essere le caratteristiche fondamentali di un sistema di controllo ad alto livello per collisori che aspiri a diventare uno strumento di utilità generale.

### 1.1.1 Astrazione e praticità

Il progetto *Corolla* nasce con l'ambizione di essere uno strumento di immediata utilità pratica e di essere, allo stesso tempo, un sistema adattabile ad una vasta gamma di esigenze specifiche, progettato in modo da garantire un'estesa possibilità di ampliamento e sviluppo prima di giungere ai limiti generati della sua complessità intrinseca.

Passando in rassegna i programmi impiegati comunemente nei laboratori per eseguire sperimentazione scientifica non è difficile imbattersi in software scritto sull'onda di necessità contingenti ed impellenti. In media questi pacchetti vengono sviluppati con un numero di opzioni molto limitato e con documentazione carente. Terminata la loro immediata utilità e sorte nuove necessità (il che solitamente avviene con celerità insospettata) essi difficilmente si prestano a soddisfarle senza che lo sforzo necessario a modificarli

risulti paragonabile a quello necessario allo sviluppo tout-court di un nuovo software.

D'altro canto non è raro imbattersi in progetti ambiziosi la cui aspirazione di produrre strumenti universali e flessibili non si è mai alla fine concretizzata in alcunché di utilità pratica: in essi l'errore più comune sta nell'esagerare nel voler costruire uno strumento che *potenzialmente* possa fare qualsiasi cosa. Un'analisi più attenta rivela come uno strumento che *potenzialmente* possa fare qualunque cosa sia inevitabilmente destinato a diventare uno strumento che non è in grado di fare niente: infatti scrivere un programma che *potenzialmente* possa essere adattato a qualunque scopo si riduce per forza di cose alla produzione di articolate riformulazioni di un file di sorgenti vuoto.

Il nostro primo obiettivo dovrà essere, pertanto, cercare di mantenerci costantemente in equilibrio tra la tentazione di impiegare scorciatoie che, riducendo la flessibilità degli oggetti implementati, permettano di giungere più rapidamente ad un risultato concreto e la tentazione di lasciare liberi ed indeterminati, in nome di fantomatici potenziali utilizzi futuri, parametri che possiamo invece tranquillamente permetterci di fissare. Per scongiurare ulteriormente il rischio di costruire una piattaforma dotata di potenzialità ma priva di concretezza non avremo timore a sviluppare e testare anche numerosi strumenti in grado di girare all'interno di questa piattaforma, concretizzando la flessibilità nella possibilità da parte dell'utente di affiancare a questi altri strumenti, sviluppati secondo le sue specifiche necessità.

### 1.1.2 Flessibilità e portabilità

Come affermato sin dall'inizio, il progetto *Corolla* vuole fornire uno strumento per la gestione ad alto livello di un acceleratore di particelle generico. Date le caratteristiche peculiari di ciascun acceleratore non sarà possibile adottare la strategia utilizzata, per esempio, nell'industria dei sistemi operativi, la quale per permettere di utilizzare un gran numero di periferiche diverse fornisce grosse librerie di driver tra le quali il software sceglie il codice richiesto nella specifica circostanza. Nel nostro caso, invece, dovrà essere il software stesso ad adattarsi alle caratteristiche della macchina sottostante.

Inoltre vista la limitata estensione del bacino di potenziali utilizzatori di tale software, sarà di primaria importanza garantire la possibilità di impiegarlo sul più vasto numero possibile di macchine e sistemi. La richiesta in termini di sistemi e librerie dovrà, quindi, venire accuratamente dimensionata e ridotta al minimo, prediligendo, laddove possibile, quelle soluzioni che garantiscano la massima portabilità e compatibilità. In questo, *Corolla* si differenzia da tutti gli strumenti sviluppati su di un *framework* commerciale quale è, ad esempio, MATLAB Accelerator Toolbox [3]; tutte le librerie su

cui si basa *Corolla* sono disponibili liberamente senza costi aggiuntivi per la sperimentazione che decida di adoperarlo.

L'unica e più grande limitazione che porremo in termini di compatibilità sarà assumere che *Corolla* giri su un sistema avente kernel Unix dotato delle interfacce API definite nello standard POSIX[4]. Non darò per scontato che le macchine sulle quali viene eseguito il software costituiente *Corolla* siano dotate di schermo, tuttavia procederò nondimeno a sviluppare strumenti dotati di interfaccia grafica. Qualora si desideri impiegare tali strumenti darò per assunta la presenza di un server grafico X11 e di una versione base della libreria OpenGL[5].

### 1.1.3 Semplicità di utilizzo

È irragionevole supporre che una qualsiasi interfaccia di amministrazione di un acceleratore di particelle possa venire impiegata senza una minima preparazione specifica; tra i miei propositi non rientra, quindi, la progettazione di architetture per i non addetti ai lavori. Cionondimeno la semplicità di utilizzo dell'interfaccia non deve venire assolutamente trascurata. Mi sono posto, a tal fine, due importanti obiettivi;

- **Limitare l'addestramento:** qualunque utente che si troverà ad utilizzare il sistema da me costruito, dovrà essere messo in condizione di poterlo fare senza dover imparare più dello stretto indispensabile per l'esecuzione del suo lavoro. Quindi, mentre il tecnico che deve installare *Corolla* e renderlo operativo o lo sviluppatore che deve estenderne le parti dovranno possedere una conoscenza approfondita di del sistema, l'operatore che lo utilizza per lavorare sulla macchina deve poter restare completamente inconsapevole in merito ai dettagli della sua implementazione.

- **Non distogliere l'attenzione:** il layout finale di *Corolla* dovrà essere tale da richiedere il minimo livello di concentrazione possibile per venire impiegato. Questo vuol dire che, a qualunque livello il sistema venga adoperato, l'utente dovrà essere messo in condizione di utilizzare la propria concentrazione primariamente nell'interpretazione della fisica di macchina e non disperderla nella comprensione dell'interfaccia.

## 1.2 Workbenching a più livelli

Giunti a questo punto è bene definire cosa, a livello informatico, debba essere concretamente *Corolla*. *Corolla* è prima di tutto una file structure (una cartella contenente file e sottocartelle), la quale contiene file di configurazione

e file sorgenti. Il risultato della compilazione dei sorgenti si traduce in una rosa di applicativi in grado di lavorare in sinergia per fornire l'ambiente di controllo descritto nel paragrafo precedente. E, cosa assai più fondamentale, dovendo strutturare questo lavoro sinergico, *Corolla* è di conseguenza anche la definizione di uno standard di comunicazione tra applicazioni: standard che viene impiegato dalla rosa di applicativi che lo costituiscono e standard al quale eventuali strumenti sviluppati dall'utente devono uniformarsi.

Dal momento che, per sua costituzione, il progetto *Corolla* aspira ad essere adattabile ed estensibile nella maniera più diretta e naturale possibile, anche i sorgenti che lo costituiscono devono essere facilmente impiegabili dall'utente. La loro strutturazione deve prevedere l'implementazione di una gamma completa di funzioni atte a garantire un utilizzo semplice e diretto dello standard di comunicazione tra applicativi definito nel progetto, così come una vasta gamma di classi che permettano la manipolazione ad alto livello dei parametri di controllo della macchina, delle letture effettuate sulla sua diagnostica e dei risultati di operazioni di simulazione eseguiti con pacchetti software esterni. La strutturazione di *Corolla*, come sarà possibile constatare in seguito, incoraggia l'utente ad integrare software di simulazione di terze parti all'interno del sistema e fornisce gli strumenti necessari a facilitare al massimo tale operazione.

Ciò detto, l'utente che si avvicina al sistema *Corolla* ha (almeno) tre potenziali modalità di lavoro:

- L'utente può utilizzare il sistema *Corolla* al livello più alto, limitandosi ad impiegare l'interfaccia grafica ed i file di configurazione per eseguire operazioni di lettura e scrittura dei parametri di funzionamento della macchina, impiegare gli strumenti presenti all'interno della suite per eseguire tutte le operazioni di cui essi sono capaci ed, eventualmente, affidarsi a software di terze parti per operare analisi e controllo sfruttando gli strumenti di importazione/esportazione via file onnipresenti all'interno del sistema *Corolla*.

- In aggiunta alla modalità precedente l'utente può decidere di sviluppare i propri personali strumenti di diagnostica e di controllo, a patto di scrivere applicativi compatibili con lo standard di intercomunicazione *Corolla*. Tale standard, che impiega i file descriptor 0 e 1, altresì noti come *standard in* e *standard out* è stato definito in modo da essere allo stesso tempo flessibile e di facile implementazione anche da parte di un programmatore alle prime armi. L'utente che si trova nella necessità di eseguire analisi complesse che comportano l'uso di strumenti matematici di difficile implementazione può, lavorando in questo modo, usare il linguaggio di programmazione a lui più familiare, così come il compila-

tore e le librerie di sua preferenza, ed integrare il proprio eseguibile in maniera naturale ed armonica all'interno del sistema di controllo.

- Come terza alternativa, l'utente più esperto può decidere di impiegare il linguaggio in cui è stato scritto il sistema *Corolla* (attualmente c++), includendo parte degli header di *Corolla* all'interno del suo progetto. Per coloro che decidessero di lavorare in questo modo sarà sì necessaria una conoscenza più approfondita degli oggetti costituenti il sistema, ma potrano d'altro lato disinteressarsi sia degli standard di intercomunicazione impiegati in *Corolla*, sia dei formati file impiegati, dal momento che tali oggetti possiedono un'implementazione completa e naturale degli standard pertinenti. In aggiunta a ciò, includendo ulteriori header, sarà loro possibile includere nei propri strumenti l'implementazione degli oggetti grafici impiegati per costruire le interfacce utente, presente in *Corolla*.

Così organizzato, *Corolla* offre quello che in inglese definiremmo *Multilevel workbenching*: la possibilità di lavorare su diversi livelli in un ambiente dotato di potenti strumenti di sviluppo forniti direttamente dal sistema, scgliendo di volta in volta il livello più consono al lavoro necessario.

## 1.3 Lo standard di comunicazione

Dovendo mettere a punto uno standard di comunicazione tra le applicazioni, mi sono concentrato sul delinarne uno che fosse al tempo stesso leggibile, di facile implementazione e agevolmente estensibile in futuro. Per farlo ho deciso di definire un protocollo costruito su una struttura che chiamo a *matrioska*.

### 1.3.1 Il protocollo a *matrioska*

Si parte adottando come base del protocollo di comunicazione la trasmissione di singoli pacchetti, ciascuno dei quali preposto a veicolare una struttura dati. Le tipologie dei pacchetti sono sempre in numero finito e la definizione dello standard di comunicazione elenca esaustivamente tutte le tipologie di pacchetto usabili nella trasmissione e le costanti numeriche impiegate per identificarle. Revisioni successive dello standard potranno prevedere l'introduzione di nuove tipologie di pacchetti: qualunque revisione dello standard dovrà però, obbligatoriamente, venire numerata progressivamente e mantenere invariata la definizione di tutti i pacchetti presenti nello standard immediatamente precedente.

Per venire incontro a esigenze (anche tecnologiche) future, sarà possibile ridefinire nel protocollo anche la codifica impiegata nel trasmettere i dati.

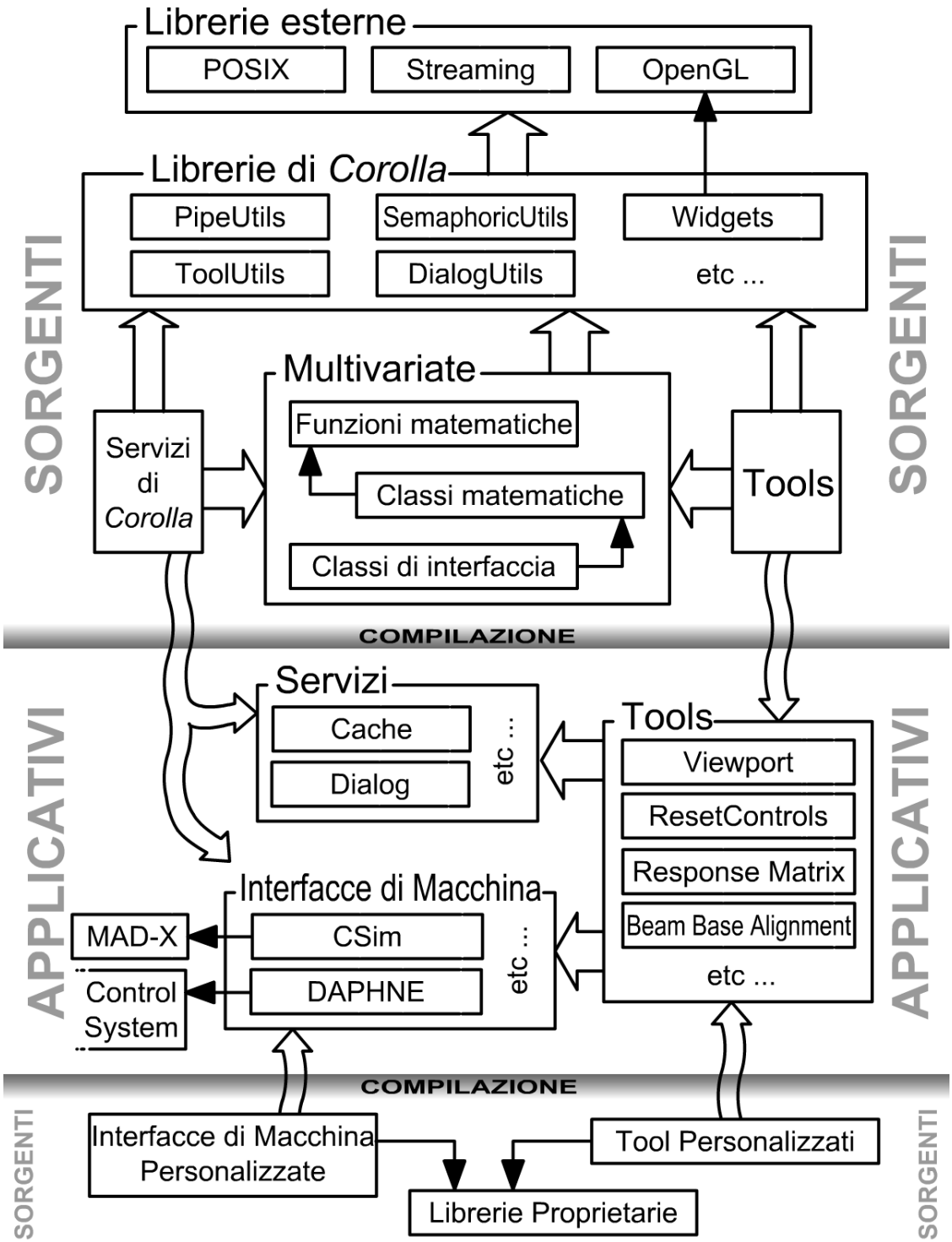





**Figura 1.1:** Diagramma delle componenti di Corolla

**Librerie di *Corolla***

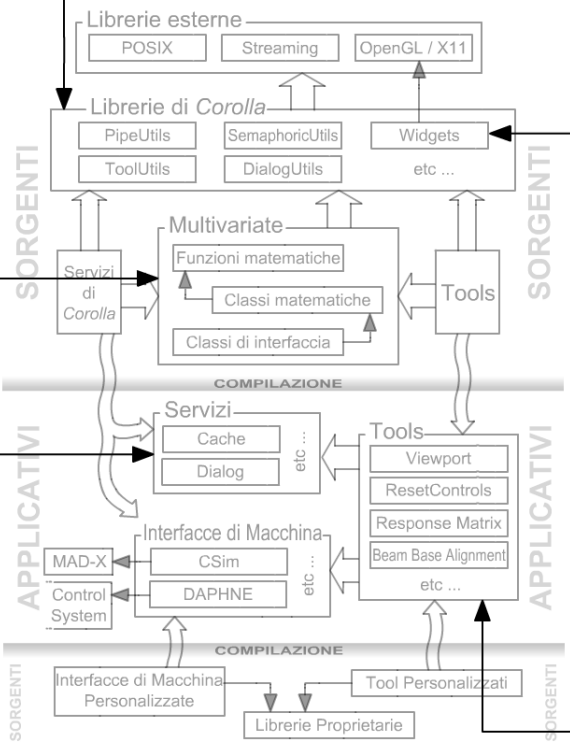
I sorgenti di *Corolla* comprendono un esteso set di librerie che implementano operazioni frequentemete utilizzate, gli standard di comunicazione, il sistema semaforico ed ogni altra funzione che richieda di venire effettuata coerentemente dalle diverse componenti del sistema. La centralizzazione di queste funzioni permette da un lato un maggiore efficienza nella stesura del codice riducendone la ridondanza, dall'altro garantisce che ogni aggionamento delle librerie venga propagato automaticamente a tutte le componenti che vi fanno riferimento.

**Multivariate**

La libreria *Multivariate*, sviluppata contestualmente a *Corolla* fornisce un set di strumenti di analisi e regressione lineare implementati mediante classi in grado di comunicare direttamente con le interfacce di macchina mediante lo standard *Corolla*

**Servizi**

Il sistema di servizi di *Corolla* è costituito da un set di applicativi che forniscono funzioni di generale utilità mediante le interfacce di comunicazione del sistema. Il sistema per ora comprende un serivizio di *Caching* centralizzato ed un servizio di generazione di finestre di dialogo che permette ai Tools di non comprendere un'interfaccia grafica al loro interno.




**Componentistica personalizzata**

La struttua di *Corolla* prevede che ai *Tools* già presenti nel sistema ne vengano affiancati altri personalizzati che vengano incontro alle specifiche esigenze di ogni sperimentazione. Tali strumenti possono essere sviluppati in un ambiente di compilazione aompletamente separato da quello nel quale è stato scritto *Corolla*.

**Widgets**

I *Widgets* sviluppati per il sistema *Corolla* sono versatili classi che implementano la gestione delle interfacce grafiche. Essi comunicano con un server *X11* e gestiscono automaticamente la cattura e l'elaborazione degli eventi generati dall'utente, così come l'aggiornameno in Realtime della propria rappresentazione grafica. Il sistema *Corolla* fornisce *Widgets* che implementano tutti i controlli utente più comuni e supporta la gestione di server grafici remoti per venire incontro alle sperimentazioni che abbiano deciso di virtualizzare le macchine di controllo.

**Tools**

Chiamiamo *Tools* gli applicativi che effettivamente vanno ad eseguire le operazioni permesse dal sitema Corolla.

**Figura 1.2:** Un breve commento su alcune componenti di Corolla

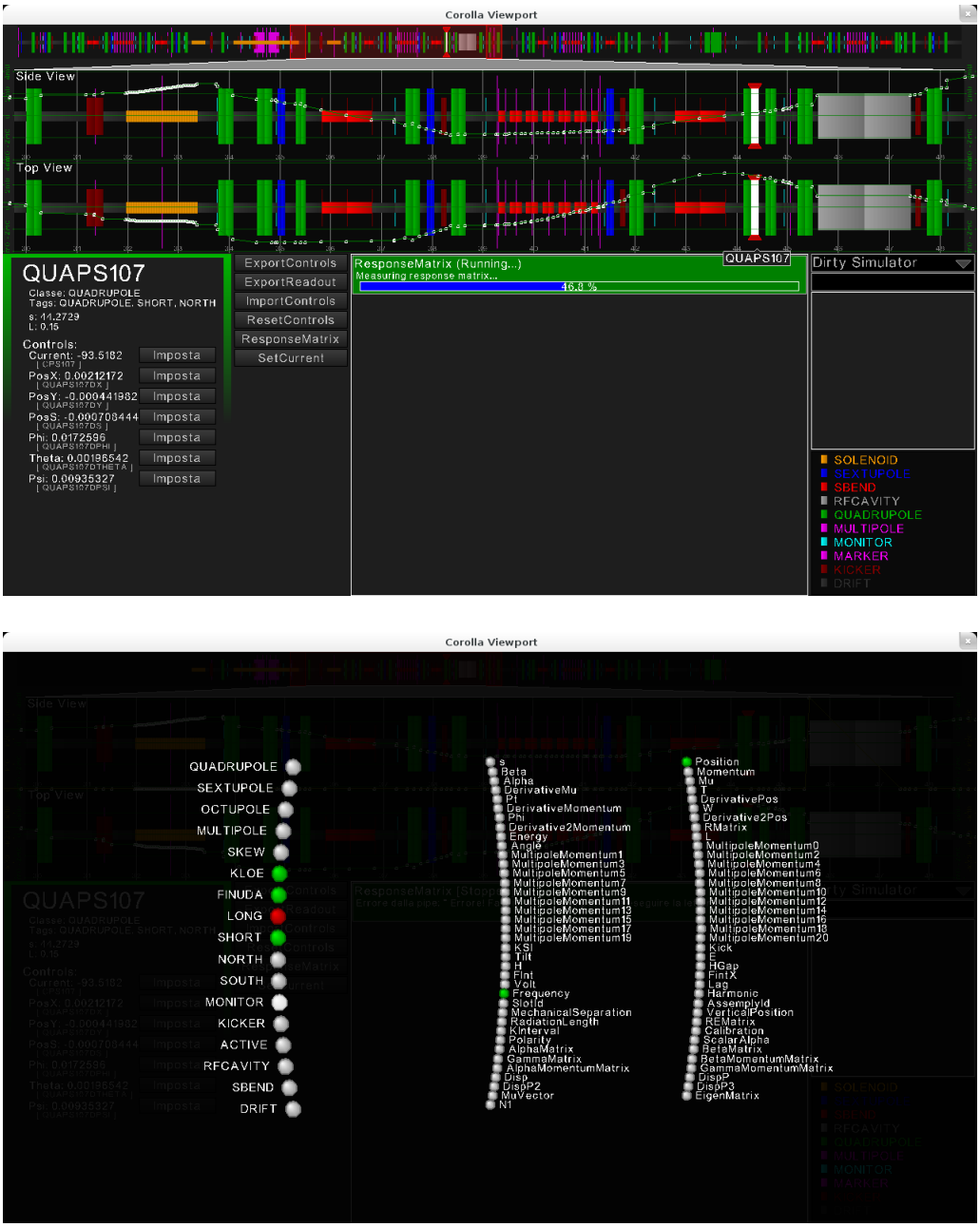






**Figura 1.3:** Due *Screenshots* dello strumento *Viewport* disegnato per acquisire la composizione dell'acceleratore da un'interfaccia di macchina e darne una rappresentazione grafica sulla quale può, poi, visualizzare l'evoluzione lungo l'orbita di caratteristiche acquisite direttamente dall'interfaccia di macchina quali la posizione del fascio o l'ampiezza delle oscillazioni di betatrone. Nel secondo screenshot si vede come il sistema a *Tags* venga utilizzato per permettere all'utente di nascondere o evidenziare sottoinsiemi della componentistica di macchina.


**Figura 1.4:** *Screenshot* di una finestra di dialogo generata dal servizio *Dialog* incluso in *Corolla*. L'utilizzo del sistema centralizzato di finestre di dialogo permette la standardizzazione dell'aspetto dei controlli rendendo più intuitiva l'interfaccia per l'utente. Il servizio *Dialog* gestisce automaticamente anche la funzione di salvataggio di configurazioni predefinite per gli strumenti aumentandone la rapidità di utilizzo e riducendo la probabilità di errore.

Anche in questo caso, comunque, è necessario che qualunque nuova codifica venga numerata progressivamente impiegando lo stesso contatore adoperato per le revisioni delle definizioni dei pacchetti e che ogni ridefinizione dello standard di codifica venga accompagnata dall'implementazione di un traduttore bidirezionale tra il vecchio ed il nuovo formato.

L'unica parte del protocollo che, per compatibilità, non deve essere soggetta a traduzioni è lo *handshake*. Nell'implementare una componente di *Corolla* che debba interagire con un'altra, lo sviluppatore, conoscendo quali tipologie di pacchetti gli sia necessario trasmettere e ricevere per portare a termine il lavoro, è in grado di stabilire quale sia la più bassa revisione dello standard di comunicazione necessario a scrivere l'applicazione. Parimenti, a seconda della revisione della libreria di comunicazione, egli è altresì a conoscenza di quale sia la massima revisione compatibile con l'applicativo da lui sviluppato (al limite si tratterà del più recente protocollo definito al momento dello sviluppo). Nel corso dello *handshake* le due applicazioni si comunicano gli intervalli numerici costituiti dal limite inferiore e superiore delle revisioni supportate dello standard di comunicazione. Nel caso i due intervalli possiedano un'intersezione non nulla, la comunicazione avverrà utilizzando come standard quello avente numerazione pari all'estremo superiore dell'intersezione degli intervalli ammissibili, nel caso gli intervalli non si accavallino, le applicazioni saranno incompatibili e l'operazione dovrà essere abortita.

### 1.3.2 Lo standard di tipo 0

Lo standard di tipo 0 è uno standard molto limitato concepito per mettere a disposizione un'interfaccia in riga di comando del tutto intuitiva. La comunicazione avviene mediante la trasmissione di uno stream di caratteri stampabili, suddiviso in pacchetti impiegando come separatore il carattere *newline*. Ogni pacchetto contiene una direttiva, eventualmente seguita da un vettore di parametri. Se presente il vettore di parametri deve essere racchiuso entro parentesi quadre, ed i suoi elementi devono essere separati mediante virgole. I parametri possono a loro volta essere ulteriori direttive: in questo caso le virgolette doppie forzano l'interpretazione letterale del parametro. Il protocollo di tipo 0 viene impiegato come protocollo di default per qualunque comunicazione e tramite esso avviene lo *handshake*.

Per facilitare le operazioni di *scripting*, si assume che nella riga di comando con cui viene lanciata una qualunque componente di *Corolla* sia possibile specificare, come parametri del comando shell, delle direttive scritte secondo lo standard 0.

### 1.3.3 Lo standard di tipo 1

Lo standard di tipo 1 è definito sulla falsa riga dei protocolli FTP e POP3[6]. La comunicazione avviene mediante la trasmissione di uno stream di caratteri stampabili, suddiviso in pacchetti impiegando come simbolo di separazione degli stessi il carattere *newline*. Ogni pacchetto comincia con quattro caratteri identificativi, costituiti da tre caratteri numerici seguiti da uno spazio. Il contenuto del pacchetto a seguire dipende dai tre caratteri numerici identificativi.

L'informazione trasportata da un pacchetto viene codificata sia mediante il contenuto del pacchetto sia mediante il tipo di pacchetto inviato. Alcuni numeri identificativi, infatti, non prevedono alcun contenuto per il pacchetto al di fuori dell'identificazione: in tal caso l'informazione trasportata è data dalla presenza e dal tipo di pacchetto. Facciamo alcuni esempi di pacchetti presenti nello standard di tipo 1:

- **000** Termine delle comunicazioni, segue un pacchetto vuoto.
- **001** Errore. Il pacchetto contiene una stringa in formato leggibile contenente una descizione dell'errore verificatosi.
- **002** Informazione. Il pacchetto contiene una stringa in formato leggibile umano contenente un generico messsaggio per l'utente. È previsto che tali pacchetti vengano ignorati qualora la comunicazione sia tra due applicativi.
- **003** Attesa. L'applicativo attende input sul canale di comunicazione, segue un pacchetto vuoto.
- **004** Stringa. Il pacchetto contiene una stringa.
- **005** / **006** Floating point number / Integer number. Il pacchetto contiene un numero in virgola mobile / un numero intero scritti in notazione decimale.
- **007** Void. Il pacchetto non contiene nulla.

## 1.4 Astrazione delle interfacce di macchina

La sfida più grande dello standard *Corolla* è quella di fornire strumenti di diagnostica e controllo che abbiano la pretesa di funzionare su un qualsiasi acceleratore di particelle. Per fare questo, ho definito una entità che identificheremo con il nome di **interfaccia di macchina**. Tale entità rappresenta

virtualmente la macchina, implementando un concetto astratto di acceleratore di particelle, concetto definito in modo da poter essere applicato ad un acceleratore qualsiasi.

Una o più interfacce di macchina, ogni volta che *Corolla* viene installato sui computer di una sala controllo, fungeranno da portale di comunicazione tra il sistema e l'acceleratore retrostante o gli eventuali strumenti di simulazione di terze parti. Ogni componente di *Corolla*, nel comunicare con la macchina, farà sempre esclusivamente riferimento a una di queste interfacce. Allo stesso modo qualunque configurazione venga eseguita per la quale sia necessaria la conoscenza di caratteristiche specifiche della macchina, verrà eseguita interpellando un'interfaccia di macchina.

### 1.4.1 L'acceleratore di particelle generico

La rappresentazione più generica possibile di un acceleratore di particelle è quella di una scatola nera la quale accetta un certo numero di parametri in ingresso e fornisce un certo numero di letture in uscita. Tale rappresentazione è intrinsecamente generica e non si presta, per i motivi già discussi in precedenza, allo sviluppo pratico di alcun progetto vantaggiosso.

Ho preferito, perciò, contrarre la raffigurazione della macchina forzandola all'interno di uno schema nel quale le caratteristiche che la descrivono vengono raggruppate in quattro insiemi di elementi rappresentativi della sua struttura e delle sue funzionalità. Sorvolando sulla tattica da impiegarsi per descriverle e sul modo in cui le informazioni che vi vengono immagazzinate sono strutturate, mi limiterò a dare una descrizione di cosa questi insiemi rappresentano dal punto di vista della comprensione umana e di come essi si intrecciano con i compiti che un'interfaccia di macchina deve essere in grado di compiere.

- **Parti:** è la lista di tutti gli elementi attivi e passivi costituenti la macchina. Essa deve obbligatoriamente contenere tutte le componenti magnetiche, le cavità a radio frequenza, i dispositivi di feedback e di diagnostica, gli elementi passivi come gli elettrodi e tutti gli elementi virtuali, come ad esempio i marcatori, utilizzati per evidenziare posizioni specifiche. Nella lista può essere elencata anche qualunque altra componente della macchina della quale si voglia tenere traccia. La caratteristica fondamentale degli elementi presenti all'interno dell'elenco delle parti è di possedere una collocazione spaziale nel percorso compiuto dal fascio di particelle.

- **Campi** è la lista di tutti i valori che è possibile leggere per ciascun elemento contenuto nell'elenco *Parti* (non necessariamente tutti gli elementi dell'elenco dell'elenco *Parti* deve essere in grado di fornire un valore per ogni voce dell'elenco *Campi*). Ogni elemento dell'elenco *Campi*

è possiede un certo numero di sottocampi, tutti costituiti dallo stesso tipo dati scelto tra numero in virgola mobile, numero in virgola fissa, stringa o booleano.

- **Canali** è la lista di tutti i parametri che possono essere impiegati per controllare l'acceleratore. Alcuni elementi dell'elenco *Canali* potrebbero essere legati ad un elemento dell'elenco *Parti*.

- **Caratteristiche** è la lista di tutti i valori che possono essere letti dalla macchina e che non sono legati a specifiche componenti della macchina stessa.

Può sembrare superfluo il fatto di aver voluto separare i canali di lettura in due famiglie, l'una legata alla componentistica, l'altra legata alla macchina nel suo complesso, mentre i canali di controllo si sono raccolti in un'unica famiglia e legando solo aluni di essi a parti specifiche. Il motivo di tale scelta è dato dal fatto che alcuni tipi di lettura (quali, ad esempio, quelle relative ad una misura di orbita o di ampiezza di oscillazioni di betatrone del fascio) sono costituite da serie di valori che evolvono in maniera correlata lungo il tracciato della macchina, mentre la stessa necessità non si manifesta per i canali di controllo, per i quali si è preferito, quindi, utilizzare l'approccio di maggiore semplicità.

### 1.4.2 Sistema a Tag

Una delle esigenze maggiori che emergono nel momento in cui vuole operare un acceleratore di particelle, è quella di poter gestire nella maniera più spedita ed intuitiva possibile sottoinsiemi di volta in volta diversi di elementi che costituiscono la macchina. Nel caso di una macchina specifica l'esigenza viene assolta programmando direttamente gli strumenti in modo che si colleghino di volta in volta alle parti (o ai canali di lettura/controllo) opportune. Nel nostro caso dovremo affrontare il problema generale.

Esistono due approcci possibili per gestire famiglie di elementi: o per ogni famiglia si stila la lista degli elementi ad essa appartenenti oppure per ogni elemento si elencano le famiglie cui esso appartiene. Il primo approccio privilegia la rapidità di accesso (è massima la velocità nello stilare liste di elementi appartenti o meno ad una o più famiglie), il secondo garantisce la massima facilità nella creazione e mantenimento delle liste di elementi (la struttura che descrive il singolo elemento trasporta con sé tutta l'informazione relativa all'appartenenza dell'elemento ad eventuali famiglie, per cui il riordino e la modifica di una lista non richiedono l'aggiornamento di strutture accessorie). Per *Corolla* ho scelto il secondo approccio.

Ogni elemento appartenente ad una qualsiasi delle sopracitate categorie conterrà, perciò, una lista di identificatori che chiameremo *tag*, consistenti ciascuno in una stringa alfabetica, ed un identificatore univoco che chiameremo *nome*, che si comporterà a tutti gli effetti come un tag esclusivo del singolo elemento e il cui obiettivo sarà identificarlo tra tutti gli elementi della lista. Affinché ciò avvenga univocamente, è necessario l'impiego di nomi distinti per elementi distinti.

Sul sistema di *tag* operano le *tag condition*, le quali selezioneranno sottoinsiemi di elementi dotati di *tag*. Una *tag condition* è costituita da un elenco di *tag* separati dagli operatori logici **and** (&) **or** (—) e **not** (!). All'atto della valutazione, ogni *tag* viene sostituito dalla valutazione logica della proposizione *"L'oggetto contiene tale tag?"*, quindi l'espressione viene valutata secondo algebra: se il risultato è **true** l'elemento fa parte della selezione, se il risultato è **false** l'elemento non fa parte della selezione.

Quindi, per esempio, una *tag condition* costituita dal solo *nome* di un elemento seleziona tale elemento, con una *tag condition* costituita da un solo tag si ottengono tutti gli oggetti che lo possiedono. La maniera in cui sono strutturate le *tag condition* fa sì che sia estremamente semplice, data una lista di elementi da selezionare, scrivere la *tag condition* che seleziona tale lista, come, per esempio, esprimere condizioni quali "tutti i quadrupoli tranne questo e quello", espressioni che sono ricorrenti nell'operazione di acceleratori di particelle.

Dal punto di vista dell'implementazione, il sistema *Corolla* offre una classe parziale denominata *FGTaggedObject* che permette ad uno sviluppatore di *Corolla* di far ereditare alle proprie classi la corretta gestione a *tag* e tutte le funzioni di filtraggio implementate nativamente all'interno del sistema.

### 1.4.3 L'interfaccia di macchina

Un'interfaccia di macchina è quindi un applicativo che implementa una gestione delle liste precedentemente elencate (e che nel nostro modello concorrono alla descrizione di un acceleratore di particelle) e, nella logica complessiva con la quale è stato progettato l'intero sistema, funge da portale tra il singolo componente di *Corolla* e l'acceleratore vero e proprio (o una sua simulazione). Un'interfaccia di macchina necessita dello standard 0 per ricevere comandi e dello standard 1 per trasmettere le risposte.

Le direttive accettate da un'interfaccia di macchina sono molteplici ed è previsto che lo standard possa venire esteso. La compatibilità verrà assicurata richiedendo a chiunque comunichi con un'interfaccia di macchina di intercettare gli eventuali messaggi di errore che segnalano la mancata interpretazione di una direttiva da parte dell'interfaccia.

Elenco di seguito, a titolo di esempio, alcune direttive accettate da un'interfaccia di macchina:

- $PARTS[\{< tagcondition > \{, ...\}\}]$ Elenca i nomi di tutti gli elementi della lista *parti* compatibili con la *tag condition* specificata. Nel caso vengano specificate più *tag conditions* elenca i risultati concatenandoli uno dopo l'altro.
- $CONTROLS[\{< tagcondition > \{, ...\}\}]$ Funziona similmente a *PARTS* solo relativamente ai canali di controllo, simi comandi sono definiti anche per le altre liste.
- $PARTSTAGS[\{< tagcondition > \{, ...\}\}]$ Funziona similmente a *PARTS*, solo che restituisce, unitamente al nome, l'intera lista di *tags* di ciascuno degli oggetti identificati.
- $CONTROLSVALUES[\{< tagcondition > \{, ...\}\}]$ Restituisce il valore corrente dei canali di controllo selezionati.
- $SETCONTROLS[\{< tagcondition >\}, < value >]$ Imposta tutti i controlli selezionati al valore specificato.
- $READOUT[\{< tagcondition >\}, \{< tagcondition >\}]$ Acquisisce canali di lettura legati a parti di macchina. La prima *tag condition* filtra le parti sulle quali eseguire la lettura, la seconda determina quali canali si desidera leggere.

### 1.4.4 Modelling online

Come si è detto un'interfaccia di macchina non deve essere necessariamente legata ad un hardware fisico, infatti può anche essere legata ad un software di simulazione. *Corolla* non fornisce direttamente alcuno strumento di simulazione per acceleratori di particelle, tuttavia incoraggia l'integrazione di pacchetti indipendenti all'interno del sistema. A questo fine, unitamente al pachetto *Corolla*, ho realizzato una semplice interfaccia di macchina con il pacchetto di simulazione *MadX (Methodical Accelerator Design X)* sviluppato al *CERN*[7].

La strutturazione a interfacce di macchina di *Corolla* permette di far girare in maniera naturale su di un simulatore tutti gli strumenti impiegati per gestire l'acceleratore reale senza necessità di applicar loro alcuna modifica per adattarli alla situazione. Questo permette di sviluppare e verificare ogni nuovo strumento in un ambiente sicuro e di impiegarli sulla macchina reale solo quando tutte le caratteristiche critiche siano state testate garantendo un'elevato livello di sicurezza nella gestione dell'acceleratore.

Predisponendo alcune interfacce di simulazione artificialmente alterate, dotate di modelli le cui caratteristiche sono state modificate usando numeri casuali generati con distribuzioni di probabilità atte a simulare le imperfezioni della macchina reale, è possibile testare la stabilità degli strumenti di controllo; contestualmente viene saggiata la capacità degli strumenti di diagnostica nell'individuare e ricostruire tali imperfezioni.

Infine sfruttando la possibilità di agganciare un'applicazione contemporaneamente a più di un'interfaccia di macchina è possibile sviluppare strumenti che, una volta misurate determinate caratteristiche della macchina reale, cerchino di ricostruire i parametri non direttamente misurabili eseguendo un *matching* tra i parametri misurati sulla macchina reale e quelli ottenuti nella simulazione. È il caso del *Beam Base Alignment*, che verrà trattato nel prossimo capitolo e che costituirà una delle prime applicazioni sviluppate nell'ambito dell'infrastruttura *Corolla*.

## 1.5 Sistema semaforico ad albero

Dopo aver descritto il funzionamento di un'interfaccia di macchina, è necessario chiarire come funziona l'accesso alla stessa. Come anticipato, la comunicazione tra i processi appartenenti a *Corolla* avviene mediante le interfacce *standard in* e *standard out*. L'avvio di un qualsiasi componente di *Corolla* può avvenire sia da parte dell'utente che da parte di uno degli altri strumenti di cui è composto *Corolla*. Se nel primo caso gli strumenti di *Corolla* non verranno a conoscenza della loro reciproca esistenza, nel secondo caso essi saranno classificati come processi legati da un rapporto di parentela genitore-figlio all'interno della strutturazione *POSIX* e il processo genitore entrerà naturalmente in possesso dei descrittori di file corrispondenti alle pipe *standard in* e *standard out* del processo figlio.

Questo rapporto di parentela fa sì che avviando strumenti di *Corolla* da altri strumenti si crei in maniera naturale una strutturazione ad albero dell'intero sistema. Sebbene sia possibile avviare anche solo singole componenti di *Corolla* e tale attività sia incoraggiata dalla struttura del protocollo di comunicazione di tipo 0, è d'obbligo mettere in guardia l'operatore dall'utilizzare contemporaneamente componenti non collegate ad uno stesso albero. Purtroppo, la necessità per alcuni strumenti di possedere un accesso esclusivo in scrittura per eseguire determinate operazioni (si veda ad esempio il caso del misuratore di matrici di risposta), rende altamente sconsigliabile tale scelta. Oltre a ciò determinate interfacce di macchina potrebbero rispondere erroneamente nel caso in cui ne venissero istanziate più sessioni contemporaneamente (ad esempio un'interfaccia di simulazione potrebbe soffrire di conflitti nell'accesso ai propri file di configurazione, mentre un'interfaccia con una macchina

reale potrebbe non essere in grado di condividere risorse con una altra istanza di sé stessa).

Per risolvere sistematicamente il problema dei conflitti di accesso alle interfacce di macchina ho deciso di utilizzare un sistema centralizzato della gestione degli accessi alle stesse. Quando uno strumento di *Corolla* viene avviato direttamente dall'utente questo assume il ruolo di *Arbiter Centrale* e gestisce una tabella nella quale viene segnalato lo stato di utilizzo delle varie interfacce di macchina, bloccandole ogniqualvolta questo fosse necessario per portare a termine le proprie operazioni. Quando, invece, uno strumento di *Corolla* viene avviato come figlio di un altro processo, dovrà all'occorrenza chiedere al suo processo genitore l'accesso all'interfaccia di macchina. Se poi questi, a sua volta, è un processo figlio, ripeterà la richiesta al proprio genitore e così via fino a che tale richiesta non sarà giunta all'*Arbiter Centrale*, il quale bloccherà il *thread* che ha richiesto l'accesso all'interfaccia fino a quando l'interfaccia non si sia liberata. Al termine delle operazioni (o in ogni caso alla morte del processo richiedente) viene propagata una richiesta di rilascio dell'interfaccia.

Per come è stato strutturato, questo sistema di arbitraggio è intrinsecamente limitato a girare su di una singola macchina. Un'estensione naturale di questo protocollo che sia in grado di girare in maniera distribuita in realtà esiste ed è realizzabile mediante l'implementazione di un *Arbiter Centrale* confezionato *Ad-hoc*. Limiti di risorse mi hanno, putroppo, costretto a circoscrivere il mio lavoro all'elaborazione della versione locale. Lo sviluppo dell'estensione rimane, ciononostante, uno dei miei prossimi obiettivi.

# Capitolo 2

# Le matrici di risposta

*Ogni scienza esatta è dominata dall'idea di approssimazione.*
– Bertrand Russell, *The Scientific Outlook*

## 2.1 Caratteristiche della scatola nera

Avevo suggerito, all'interno del capitolo precedente, come la più semplice descrizione che si potesse dare di un acceleratore di particelle generico fosse quella di una scatola nera che accetta un certo numero di parametri in ingresso e che restituisce un certo numero di parametri in uscita. Torniamo per un momento ad utilizzare questa schematizzazione e proviamo a analizzare alcune proprietà di questa scatola nera.

Prima di tutto concentriamoci sui parametri in uscita e chiediamoci se essi siano una funzione dei parametri in ingresso, la risposta è inevitabilmente: no. Con ciò non intendo dire che i valori rilevati da un acceleratore siano indipendenti dalla scelta dei valori assegnati ai suoi parametri di controllo, bensì che essi non dipendono unicamente da quelli. Esistono infatti due fenomeni ulteriori dei quali è necessario tenere conto: l'evoluzione temporale spontanea del collisore e l'eventuale presenza di isteresi.

### 2.1.1 L'evoluzione temporale spontanea

Poniamo di aver acceso un acceleratore di particelle fornendo in ingresso un set di parametri di controllo in grado di garantirne il funzionamento e di aver poi bruscamente modificato tale set in un altro anch'esso compatibile con il funzionamento della macchina. Ci chiediamo cosa vedremo mettendoci a questo punto ad osservare i parametri in uscita senza intervenire ulteriormente sul controllo.

Se la scelta dei due set viene effettuata in maniera del tutto casuale tra i punti di lavoro che danno origine ad un'orbita stabile è possibile che la transizione tra i due punti di lavoro porti alla perdita del fascio, quindi al brusco spegnimento dell'acceleratore. Assumiamo pertanto di aver scelto l'origine e la destinazione della transizione in modo da scongiurare questa eventualità.

Prima di tutto osserveremo un moto di rilassamento dei parametri in uscita man mano che l'acceleratore si adatta al nuovo punto di lavoro. In un acceleratore convenzionale questa transizione è principalmente dovuta al tempo di *ramping* dei magneti.

Immediatamente dopo osserveremo una oscillazione di tutti i parametri intorno ad un valore intermedio: l'oscillazione è dovuta al rumore presente negli apparati di misura, all'azione continua dei sistemi di correzione di basso livello ed alle operazioni periodiche di ottimizzazione del fascio, ad esempio il *refill* (che, avendo data per assunta la stabilità della macchina assumeremo completamente automatizzato). Campionando nel tempo i valori assunti dai parametri in uscita è possibile dedurre la distribuzione di probabilità di ciascuno. Ci aspettiamo che questa distribuzione di probabilità sia stabile su una mesoscala temporale.

Infine, se osservassimo nel corso di una lunga scala temporale tali distribuzioni di probabilità, ci accorgeremo che esse evolvono lentamente nel tempo. Questo è dovuto al naturale invecchiamento della componentistica di cui è costituito l'acceleratore, a mutazioni di condizioni esterne (prima fra tutti la temperatura) e alle alterazioni generate dalla continua sollecitazione cui sono sottoposti, nel corso dell'operazione, i materiali di costruzione. In realtà osservare questa aberrazione direttamente misurando le alterazioni del funzionamento di un acceleratore attivo è materialmente irrealizzabile: difficilmente si riesce a tenere acceso un acceleratore di particelle per un tempo così lungo senza interromperne il funzionamento per eseguire operazioni di manutenzione. Un esperimento così formulato probabilmente si concluderebbe nel momento in cui un malfunzionamento catastrofico obbligasse lo spegnimento dell'apparato.

Considerati i tre fenomeni precedentemente descritti possiamo dedurre che, sebbene le variabili direttamente leggibili evolvano nel tempo, trascorso un tempo di rilassamento opportuno i momenti delle distribuzioni relative a tali parametri evolvano nel tempo così lentamente da poterli considerare costanti per qualunque fine pratico. Quando a breve si andrà a definire il concetto di matrice di risposta e, trattandone la misura, mi riferirò a parametri rilevati dall'acceleratore, darò per scontato che si starà considerando la media della distribuzione raggiunta dopo l'opportuno tempo di rilassamento.

### 2.1.2 Le isteresi

Immaginiamo di operare un acceleratore di particelle mediante un set $A$ di parametri di ingresso e di misurare la distribuzione dei parametri in uscita. Andiamo, quindi, a sostituire tale set con un set $B$ di parametri ed attendiamo la stabilizzazione del sistema, quindi applichiamo di nuovo il set di parametri $A$, attendiamo il rilassamento e misuriamo di nuovo la distribuzione dei parametri in uscita. Assunto che l'intera operazione sia stata eseguita su una scala temporale piccola rispetto ai tempi di aberrazione lenta dell'acceleratore, ci chiediamo se le nuove distribuzioni con il set di parametri $A$ differiranno da quelle misurate precedentemente.

Adottando una risoluzione sufficientemente elevata nella misurazione dei parametri in uscita, la risposta sarà inevitabilmente sì. Infatti, anche ignorando l'aberrazione progressiva della macchina, i sistemi macroscopici facilmente immagazzinano informazioni sulla loro precedente evoluzione temporale e al livello di complessità raggiunto da un acceleratore di particelle la presenza di componentistica con caratteristiche isteretiche è inevitabile. Riformuliamo la domanda chiedendoci se le nuove distribuzioni siano *significativamente* differenti dalle precedenti. Così facendo troviamo tre possibili sorgenti di isteresi.

La prima è dovuta alla presenza di isteresi intrinseche nel sistemi di regolazione. Si prenda come esempio un alimentatore che possieda un sistema a retroazione che, confrontando la corrente in uscita con il valore richiesto dal sistema di controllo, alteri il suo funzionamento in modo da cercare di far corrispondere tali valori. Tale sistema di comparazione e feedback ha, inevitabilmente, una precisione finita, per cui il valore in corrente ottenuto in uscita non sarà esattamente pari al valore numerico richiesto in ingresso bensì sarà compreso in un intervallo centrato intorno a tale valore più il piedistallo del sistema di misura. Se il livello di rumore nel sistema è piccolo rispetto all'ampiezza di tale intervallo e il sistema di reazione è veloce rispetto al rapporto tra il fattore di ramping e l'ampiezza di tale intervallo, allora, al cambiare del valore di corrente richiesto, l'alimentatore conserverà memoria della precedente impostazione. Se fossero presenti nel sistema meccanismi dotati di tali caratteristiche e se la loro azione dovesse produrre effetti significativi, la definizione di matrice di risposta così come da me proposta nel paragrafo (2.2) sarebbe inapplicabile.

La seconda fonte di isteresi deriva dalla fisica di macchina, in particolare dalla possibilità che la transizione tra differenti impostazioni modifichi caratteritiche del fascio quali l'emittanza o la dispersione in energia. È ragionevole assumere che, al più attendendo il primo refill della macchina, qualunque alterazione di questo genere venga riassorbita. Lo stesso dicasi per eventuali alterazioni introdotte dalla presenza della camera a vuoto (se, ad esempio,

una delle due impostazioni impiegate prevede un fenomeno di *electron cloud* notevolmente superiore all'altra). Se per qualche motivo questa condizione non dovesse venire soddisfatta quantomeno in un intorno dell'impostazione $A$, come nel caso precedente la definizione di matrice di risposta così come da me proposta nel paragrafo (2.2) risulterebbe inapplicabile.

La terza fonte di isteresi è data dall'interazione con i sistemi di retroazione locali. Consideriamo un sistema che, letta una caratteristica del fascio, esegua una correzione repentina. Il fattore di amplificazione del *feedback* viene tarato in funzione del tempo di reazione del sistema; un fattore di amplificazione eccessivo, infatti può dare luogo ad un sistema oscillante, un fattore di amplificazione insufficiente dà invece luogo ad isteresi. Anche nel caso di sistema oscillante, l'ampiezza del moto conserva inevitabilmente memoria del modo in cui esso è stato innescato. Poiché la taratura di qualunque sistema non può essere mai perfetta, uno dei due fenomeni sarà inevitabilmente presente in una qualche misura in ogni feedback installato sull'acceleratore, anche se generalmente non produce effetti significativi. In caso contrario, come nelle circostanze precedenti, la definizione di matrice di risposta così come da me proposta nel paragrafo (2.2) diventerebbe inapplicabile.

## 2.2 Definizione della matrice di risposta

Poste le limitazioni descritte precedentemente possiamo considerare l'acceleratore di particelle come una funzione dei parametri in ingresso nei parametri in uscita. Non ha alcun senso chiedersi se questa funzione sia differenziabile, dal momento che la tecnologia umana maneggia parametri numerici limitati nella risoluzione. Ha senso chiederci, invece, se sia computazionalmente conveniente approssimare tale funzione mediante una funzione continua descritta da uno sviluppo in serie di Taylor intorno ad un'orbita stabile. Siano $x_i$ con $1 \leq i \leq n$ i parametri in ingresso e $y_j = F_j(x_1, \ \ldots \ , x_n)$ con $1 \leq j \leq m$ i parametri in uscita. Lo sviluppo della funzione risulta:

$$\begin{aligned} \vec{F}(\vec{x}) &= \vec{F}(\vec{x}_0) + \sum_{j=1}^{m} \hat{y}_j \sum_{k=1}^{\infty} \sum_{1 \leq \alpha_{1\ldots k} \leq n} K_{j,\alpha_1,\ldots,\alpha_k} \prod_{z=1}^{k} (x_{\alpha_z} - x_{0\alpha_z}) \\ &= \vec{F}(\vec{x}_0) + \sum_{j=1}^{m} \sum_{k=1}^{n} \hat{y}_j K_{j,k} (x_k - x_{0k}) + \ldots \end{aligned} \tag{2.1}$$

Condensando i coefficienti $K_{j,k}$ in un'unica matrice $\hat{K}$ possiamo scrivere:

$$\vec{F}(\vec{x}) = \vec{F}(\vec{x}_0) + \hat{K} \cdot (\vec{x} - \vec{x}_0) + \ldots \tag{2.2}$$

Ora, se volessimo eseguire un vero sviluppo in serie di Taylor, i coefficienti $K_{j,k}$ sarebbero pari alle derivate parziali di $\vec{F}(\vec{x})$, ovvero:

$$K_{i,j} = \frac{\partial F_i(\vec{x}_0)}{\partial \hat{x_j}} \tag{2.3}$$

La scelta di impiegare coefficienti così determinati si rivela sconveniente per due motivi. Il primo è dovuto al fatto che questa definizione si basa sull'assunto che esista una funzione continua e differenziabile sottostante al funzionamento dell'acceleratore di particelle, della quale il comportamento della macchina è una discretizzazione; tale funzione è frutto di una astrazione che rende più comprensibile la fisica di macchina, ma non esiste nella realtà dei fatti. In secondo luogo lo scopo precipuo per cui desideriamo eseguire tale sviluppo è troncarlo al termine lineare ed utilizzare la versione troncata per descrivere il comportamento della macchina nell'intorno di un punto. Assumendo questo come obiettivo è più opportuno definire la matrice di risposta come quella matrice $\hat{K}$ i cui coefficienti rendono lo sviluppo troncato il più aderente possibile al comportamento della macchina.

Definiamo la matrice di risposta come quella matrice $\hat{K}$ che minimizza:

$$\sum_i \left| \hat{K} \cdot (\vec{x}_i - \vec{x}_0) - \vec{F}(\vec{x}_i) + \vec{F}(\vec{x}_0) \right|^2 \tag{2.4}$$

il che equivale a richiedere che la matrice sia tale da ottimizzare lo sviluppo (2.2) troncato al termine lineare. Si potrebbe argomentare sul fatto che una differente scelta del termine lineare potrebbe portare ad ottimizzare ulteriormente tale sviluppo. Questo è certamente corretto; tuttavia gran parte degli impieghi della matrice di risposta sono penalizzati da uno sviluppo in cui $\vec{F}'(\vec{x}_0) \neq \vec{F}(\vec{x}_0)$.

Si noti come in questa definizione della matrice di risposta entri anche la scelta non solo di un punto intorno al quale eseguire lo sviluppo, ma anche di un intorno di tale punto. La scelta di tale intorno deve essere effettuata in base al punto di lavoro selezionato e all'utilizzo che si desidera fare dello sviluppo così ottenuto. La scelta delle dimensioni di tale intorno può influenzare molto pesantemente il risultato ottenuto dalla misura della matrice di risposta, ma, se ben effettuata, essa restituisce l'oggetto matematico più adatto ad eseguire le analisi desiderate. Questo fatto da solo è indicativo di quanto la definizione basata sulle derivate parziali possa essere estremamente fuorviante.

La definizione di matrice di risposta come miglior approssimazione del comportamento della macchina si differenzia sostanzialmente dalla definizione più comune in letteratura che, invece, si basa sull'utilizzo di un gradiente

approssimato. Per comprendere la motivazione di una simile scelta, andiamo a considerare una funzione come quella in figura (2.1): è evidente come lo sviluppo di Taylor troncato eseguito in prossimità dell'origine non fornisca una approssimazione soddisfacente nell'intervallo visualizzato. Le due approssimazioni eseguite mediante fit su di una selezione di punti raccolti entro intervalli ben definiti, invece, offrono buone approssimazioni del comportamento della funzione entro tali intervalli.

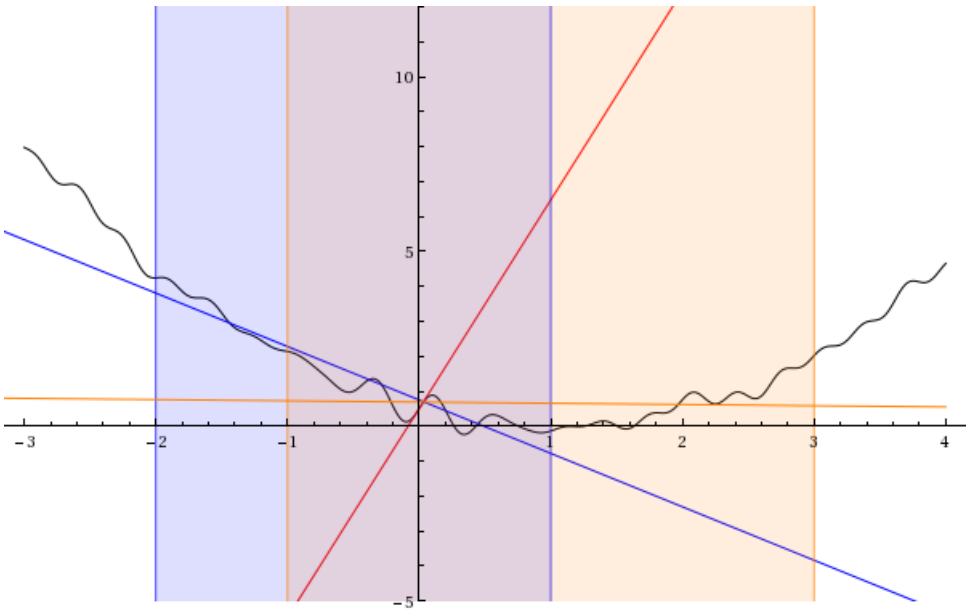



**Figura 2.1:** Esempio di differenti approssimazioni lineari di una stessa funzione. In rosso l'approssimazione data dallo sviluppo in Taylor troncato al primo termine, in blu l'approssimazione ottenuta mediante fit sull'intervallo $\{-2, 1\}$, in arancione l'approssimazione ottenuta mediante fit sull'intervallo $\{-1, 3\}$. Includere la scelta di un intorno nella definizione della matrice di risposta permette di incrementare la sua efficacia come approssimazione della risposta reale della macchina.

## 2.3 Tecniche di misura delle matrici di risposta

### 2.3.1 Misurazione diretta

Volendo applicare pedissequamente la definizione fornita in (2.4), la misura della matrice di risposta dovrebbe partire dalla definizione di un intorno di un punto di lavoro. Definiamo tale intorno mediante il suo centro $\vec{x}_0$ e un vettore

di scostamenti $\vec{\Delta x}$ come l'insieme di tutti i punti tali che $\forall i | x_i - x_{0i}| < \Delta x_i$ . Detto in altri termini un intorno sferico ottenuto mediante la norma infinito indotta dalla metrica diagonale, avente elementi di matrice pari agli inversi degli elementi di $\vec{\Delta x}$. Sebbene siano possibili altre definizioni di tale intervallo, non le considereremo rilevanti ai fini della nostra trattazione.

La misurazione diretta consistere nella misura della risposta dell'acceleratore su tutti i punti appartenenti a questo intorno, quindi nel fit lineare della risposta così ottenuta. Il tempo impiegato per eseguire questo tipo di misura è sostanzialmente pari al tempo di rilassamento che è necessario attendere per misurare la risposta moltiplicato per il numero di punti da campionare, pertanto il costo di una misurazione eseguita in questo modo è pari a:

$$C_{direct} \; = \; \prod_{i=1}^{n} (S_i + 1) \; \approx \; (< S > +1)^n \tag{2.5}$$

ove $S_i$ è il numero di passi in cui è stata suddivisa la scansione dell'i-esima dimensione dello spazio dei parametri. Si noti come, anche in nel caso banale in cui si sia eseguita una sola suddivisione e sia necessario campionare due soli punti per dimensione, l'utilizzo di più di 30 parametri in ingresso richiede il campionamento di miliardi di punti e risulta, pertanto, proibitivo. È, quindi, necessario un approccio statistico alla misura.

### 2.3.2 Misurazione mediante Monte Carlo

La misurazione mediante Monte Carlo si basa sulla raccolta di punti appartenenti all'intorno nel quale si desidera eseguire la misura e nel fit dei risultati. Il vantaggio di questa tecnica è il fatto che il costo computazionale non cresce al crescere delle dimensioni. Vale infatti la relazione:

$$< rms > \; \propto \; \frac{1}{\sqrt{n}} \tag{2.6}$$

in cui $< rms >$ è l'errore ottenuto su di un singolo parametro ed il fattore di proporzionalità è un O-grande di uno nel numero di dimensioni. Invertendo la relazione si ottiene:

$$n \; < \; \frac{K_{max}}{< rms >^2} \tag{2.7}$$

ove $K_{max}$ è un maggiorante della dipendenza del fattore di proporzionalità dal numero di dimensioni. Si noti come tale formula costituisce una stima

pessimistica del numero di misurazioni necessarie. Infatti, un comportamento altamente lineare della macchina nell'intorno del punto nel quale viene effettuata la misura incrementa notevolmente la precisione del risultato ottenuto. Nel caso limite in cui la macchina rispondesse in maniera perfettamente lineare, si otterrebbe la matrice esatta dopo $n+1$ campionamenti, dove $n$ è il numero di canali di controllo.

Il grafico in figura (2.2) mostra la velocità di convergenza del metodo Monte Carlo in un caso simulato. Esso è stato ottenuto simulando la misura di una matrice di risposta mediante il software *MAD-X*, impiegando il modello di macchina dell'acceleratore DAΦNE aggiornato ad Agosto 2013. Sono stati campionati 670 punti in un intorno di ampiezza 100mA del punto di lavoro avente i 62 *kicker* installati sulla macchina spenti. L'acquisizione è stata effettuata misurando virtualmente la posizione del fascio mediante i 47 *Beam Position Monitor* di cui è dotato l'anello di DAΦNE, ottenendo una matrice $62 \times 94$. Dopo ogni acquisizione è stato ricalcolato il fit lineare sui punti raccolti, al termine dell'acquisizione è stato calcolato la distanza euclidea tra il fit parziale e la matrice ottenuta dal fit complessivo delle 670 acquisizioni. Il risultato è stato normalizzato dividendo tale distanza per la norma euclidea della matrice risultante dal fit complessivo.

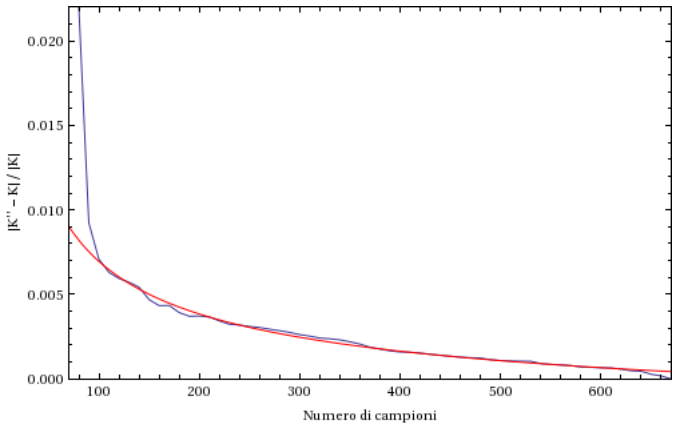



**Figura 2.2:** Velocità di convergenza della misura della matrice di risposta, effettuata mediante tecnica Monte Carlo, in funzione del numero di campioni raccolti. Sulle ascisse è riportata la distanza euclidea tra la matrice data dal fit complessivo e la matrice data dal fit parziale dei punti raccolti. È stato evidenziato l'andamento tracciando in rosso il fit effettuato mediante la funzione $y = \frac{A}{\sqrt{x}} + B$

La tecnica di misura mediante Monte Carlo è certamente molto più conve-

niente della misurazione diretta in quanto permette di determinare la matrice di risposta con un grado di accuratezza molto elevato in tempi molto inferiori.

### 2.3.3 Campionamento su singole direzioni

La tecnica di campionamento su singole direzioni consiste nell'eseguire un campionamento della risposta della macchina spostando un singolo parametro in ingresso per volta e quindi nell'eseguire il fit dei risultati ottenuti. L'assunto su cui si basa questa tecnica è assumere che l'approssimazione della risposta data da (2.4) descriva accuratamente il comportamento della macchina in un intorno del punto di lavoro e che, quindi, la misura di punti ottenuti mediante la variazione simultanea di più parametri non aggiunga alcuna informazione.

Esistono due modi in cui si può procedere per eseguire il fit dei parametri in uscita: si può decidere di eseguire un fit complessivo multidimensionale, andando, cioè, a cercare la matrice di risposta che minimizza:

$$\sum_i \left| \vec{F}(\vec{x}_i) - \vec{F}(\vec{x}_0) - \hat{K} \cdot (\vec{x}_i - \vec{x}_0) \right|^2 \tag{2.8}$$

In alternativa possiamo andare ad eseguire il fit di ogni singolo dataset ottenuto variando uno dei parametri in ingresso. In tal caso la misura si riduce ad una serie di fit lineari consistenti nella minimizzazione di:

$$\sum_i \left| \vec{F}(\vec{x}_0 + \Delta x_i \cdot \hat{x}_i) - \vec{F}(\vec{x}_0) - \hat{K} \cdot \vec{\Delta x_i} \right|^2 \tag{2.9}$$

scelto un opportuno insieme di valori per i $\Delta x_i$.

Questa ultima versione è decisamente la tecnica più popolare a causa della sua semplicità di implementazione. In particolare, quando viene eseguita campionando solo due punti per ogni parametro in ingresso, essa si riduce alla computazione di un semplice rapporto incrementale. A dispetto della sua popolarità, però, questa tecnica restituisce risultati dotati di bias.

Il grafico sottostante è stato ottenuto simulando la misura di una matrice di risposta mediante il simulatore *MAD-X* impiegando il modello di macchina dell'acceleratore DAΦNE aggiornato ad Agosto 2013. La matrice di risposta è stata misurata mediante la tecnica dei rapporti incrementali, quindi è stata eseguita una misura della stessa matrice di risposta mediante Monte Carlo. Al crescere del numero di campioni raccolti per eseguire la misura mediante Monte Carlo è stata computata la distanza euclidea tra le matrici ottenute con queste due tecniche. Nel grafico viene presentata tale distanza divisa per la norma matriciale della matrice calcolata mediante rapporti incrementali.

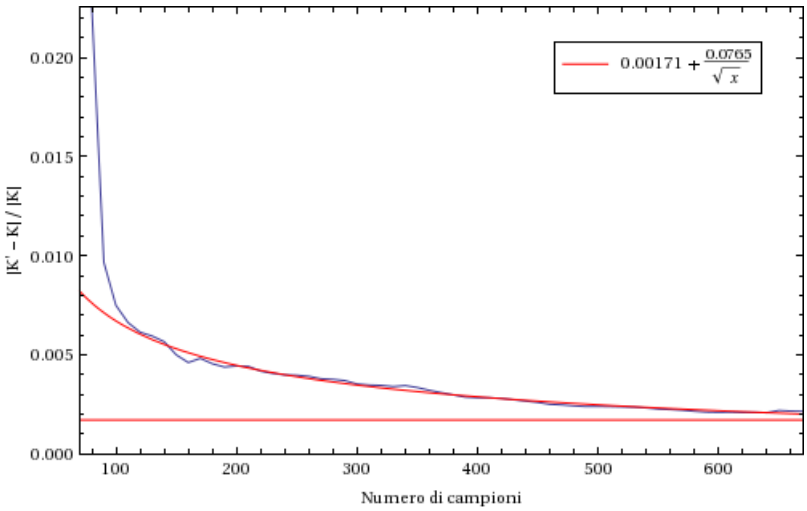



**Figura 2.3:** La distanza euclidea tra la matrice misurata mediante rapporti incrementali e la matrice misurata mediante Monte Carlo in funzione del numero di campioni impiegati per eseguire il Monte Carlo.

Come si può notare dal risultato del fit, la misura mediante rapporti incrementali introduce un errore sistematico intorno al due per mille. Vedremo nel capitolo 4 come l'alta precisione nella misura delle matrici di risposta risulti essenziale in alcune loro applicazioni. Ci possiamo chiedere se, a parità di misure necessarie, questa tecnica sia più efficace del Monte Carlo. Per ottenere i risultati in figura (2.3) sono stati eseguiti 124 campionamenti; dalla curva di convergenza del Monte Carlo si vede come con 124 campionamenti tale metodo approssimi la matrice di risposta a meno del 6 per mille. Analizzando una varietà di casi simili ho potuto riscontrare come in generale la tecnica dell'approssimazione mediante rapporti incrementali permetta di eseguire una misura sì più rapida, però meno precisa della matrice di risposta rispetto al metodo Monte Carlo.

## 2.4 Errore sperimentale

Come ogni misura sperimentale la misura delle matrici di risposta non possiede alcun significato se non corredata di un'incertezza sperimentale. L'incertezza sperimentale sulla misura delle matrici di risposta deriva da due sorgenti: la finita precisione con cui la strumentazione di macchina regola le correnti e rileva le caratteristiche del fascio e il numero limitato di campioni impiegati per eseguire il fit della matrice di risposta.

L'incertezza derivante dalla finita precisione della strumentazione di macchina può essere determinata mediante una simulazione che, utilizzando una modellizzazione delle incertezze della strumentazione, generi una serie di misure fittizie sommando alla misura reale un errore generato secondo le distribuzioni attese. Eseguendo il fit di queste misure alterate è possibile determinare la distribuzione d'errore delle entrate della matrice di risposta misurata. Questa tecnica richiede che la scala su cui varia la precisione della strumentazione di macchina sia grande rispetto alla precisione della stessa, assunto che è ragionevole nella quasi totalità delle circostanze. Eseguendo la propagazione degli errori mediante simulazione è possibile, qualora lo si ritenga rilevante, tenere conto della correlazione tra le distribuzioni d'errore che descrivono la precisione della strumentazione di macchina, così come è possibile estrapolare dal risultato dei fit ripetuti la matrice di covarianza relativa alla linearizzazione della matrice di risposta.

L'incertezza derivante dal numero limitato di misure eseguite, invece, presenta delle difficoltà ulteriori causa principale delle quali è l'impossibilità di impiegare un approccio analitico al problema. Una possibile strategia per stimare l'incertezza dovuta al Monte Carlo potrebbe essere quella di ripetere numerose volte la misurazione valutando la varianza tra i risultati ottenuti. Questa tecnica ha il difetto di mal impiegare il gran numero di misure da essa richieste. Se infatti per misurare una matrice di risposta si decidesse di raccogliere una quantità $N$ di punti per determinare l'incertezza sulla misura sarebbe necessario ripetere $M$ volte il procedimento raccogliendo ogni volta un campione di punti differente. Al termine della procedura saranno stati raccolti $N \times M$ punti: se questi punti venissero impiegati per eseguire un unico fit della matrice di risposta tale matrice potrebbe venire determinata con una precisione molto superiore a quella ottenuta con $N$ misurazioni. Tale tecnica quindi, sebbene corretta in linea di principio, è da evitarsi per questioni di economia.

Una tecnica alternativa potrebbe essere quella di raccogliere un campione di $N \times M$ punti ed eseguire sull'intero sample il fit della matrice di risposta; eseguire, quindi, il fit di $M$ sottoinsiemi disgiunti del sample complessivo costituiti ciascuno da $N$ punti, infine di stimare la varianza sul membro $K_{i,j}$ come:

$$\sigma\left(K_{i,j}\right) \approx \frac{1}{\sqrt{M}} \sigma\left(\tilde{K}^n_{i,j}\right) \tag{2.10}$$

In cui $\tilde{K}^n$ è la matrice di risposta derivante dall'n-esimo fit parziale e $\sigma$ indica lo scarto quadratico medio. Questa tecnica di misura non stima direttamente l'errore relativo al fit complessivo, bensì l'errore relativo alla media dei fit.

Non essendo questa stima dell'errore con cui è conosciuta la media dei fit una sovrastima, qualora la media dei fit si possa confondere con il fit eseguito sull'intero campione di punti raccolti la stima dell'errore sarà valida anche per il fit complessivo. Al crescere del campione, la misura della matrice di risposta eseguita mediante fit complessivo tende allo stesso limite della misura eseguita mediando i fit. In tutti i casi da me analizzati ho riscontrato che, preso $N$ pari ad almeno il doppio del numero di canali di controllo e $M$ non inferiore a 10 la distanza tra le matrici determinate con le due tecniche risulta molto inferiore all'errore sperimentale misurato; è pertanto possibile utilizzare la formula (2.10) come estimatore dell'incertezza sulla matrice di risposta.

Nella figura (2.4) presento la distribuzione della differenza tra la matrice di risposta determinata mediante fit complessivo e quella determinata mediante media di fit successivi. Ogni differenza è stata normalizzata dividendola per l'errore corrispondente, stimato come da formula (2.10). Come atteso, la media della distribuzione è compatibile con 0, mentre la varianza è largamente inferiore a 1, dimostrazione che la matrice ottenuta mediante il fit integrale è molto più vicina alla matrice media di quanto previsto dagli intervalli di confidenza. A titolo di osservazione, ho aggiunto all'istogramma il plot della distribuzione di Cauchy avente parametro di localizzazione pari alla media della distribuzione e parametro di scala pari ad un terzo dello scarto quadratico medio dei punti raccolti; non azzarderò, tuttavia, alcuna speculazione sulla ragione di tale apparente corrispondenza.

Si noti come questa tecnica di misurazione dell'incertezza includa naturalmente nel valore di incertezza calcolato l'errore statistico sulle misure eseguite dalla strumentazione di macchina che, quindi, non avrà bisogno di essere misurato.

## 2.5 Implementazione

Lo strumento di misura delle matrici di controllo è uno dei primi strumenti da me implementati per essere distribuiti insieme al sistema *Corolla*. L'implementazione ha seguito le linee guida del sistema *Corolla* che prescrivono la massima riutilizzabilità, flessibilità e facilità d'uso del codice.

L'oggetto chiave impiegato per eseguire la misurazione delle matrici di risposta è la classe *FGMVResponseMatrix*, inclusa all'interno del pacchetto di strumenti di algebra variazionale *Multivariate* fornito all'interno della suite di *Corolla*. Questa classe contiene al suo interno altri tre oggetti provenienti dal pacchetto variazionale: un *FGMVInputVector*, un *FGMVOutputVector* ed una *FGMVMatrix*, i primi due gestiscono, rispettivamente, i canali di lettura ed i canali di controllo impiegati nella misurazione della matrice di risposta, mentre il terzo contiene la matrice di risposta misurata. La classe fornisce,

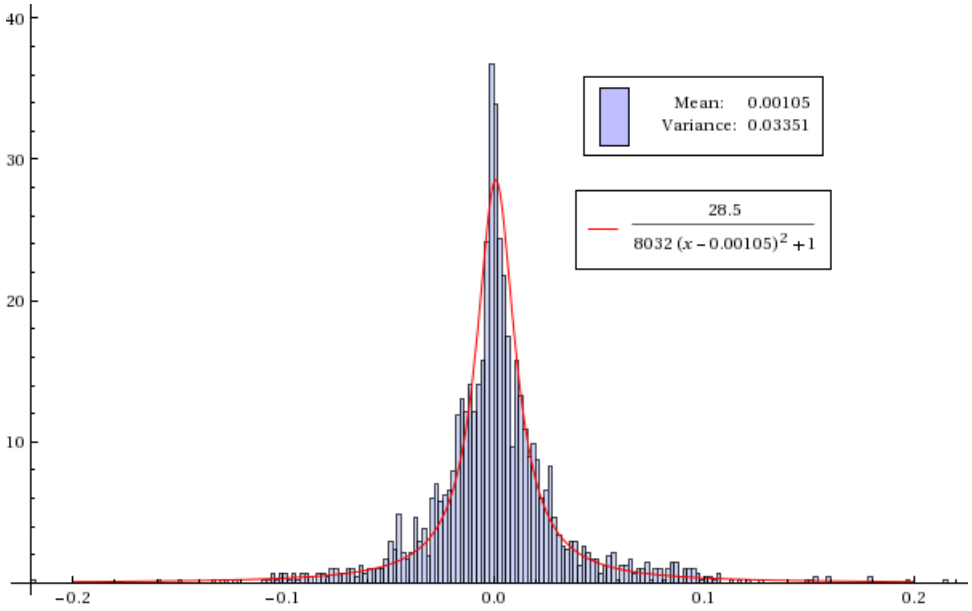



**Figura 2.4:** Distribuzione della differenza tra i termini della matrice di risposta misurata mediante singolo fit e quelli della matrice di risposta determinata come media di fit successivi, normalizzata mediante l'errore determinato mediante RMS dei fit successivi. Il campione è stato costruito mediante il simulatore MAD-X utilizzando il modello di macchina dell'acceleratore DAΦNE aggiornato al Luglio 2013. Il campione è costituito da $N = 50 \times M = 10$ punti.

inoltre, numerosi metodi che permettono la misura della matrice di risposta mediante le tecniche finora descritte, così come l'esportazione o l'importazione da file.

Dal punto di vista del programmatore la misura di una matrice di risposta all'interno di *Corolla* è facilmente eseguibile utilizzando il pacchetto variazionale *Multivariate* presente tra le librerie del progetto. Questo pacchetto contiene una varietà di funzioni in grado di eseguire le più comuni operazioni di algebra lineare, di classsi che rappresentano oggetti matematici interfacciate naturalmente con le funzioni matematiche e di classi *wrapper* che permettono di collegare facilmente questi oggetti con le interfacce di macchina di *Corolla*. Per rappresentare una matrice di risposta è predisposta la classe *FGMVResponseMatrix*, la quale include una classe *FGMVMatrix*, che rapprersenta la matrice di risposta misurata, e due classi di interfaccia: una *FGMVInputVector* ed una *FGMVOutputVector*. Le classi di interfaccia permettono di specificare mediante *Tag Conditions* quali canali di lettura e scrittura devono essere impiegati nella misurazione. Una volta specificati questi canali, la stessa classe *FGMVResponseMatrix* fornisce metodi nativi per eseguire la misura della matrice di risposta. Eseguita la misura, il risultato è facilmente accessibile e può venire manipolato mediante i metodi nativi presenti nella classe *FGMVMatrix* contenuta in *FGMVResponseMatrix*.

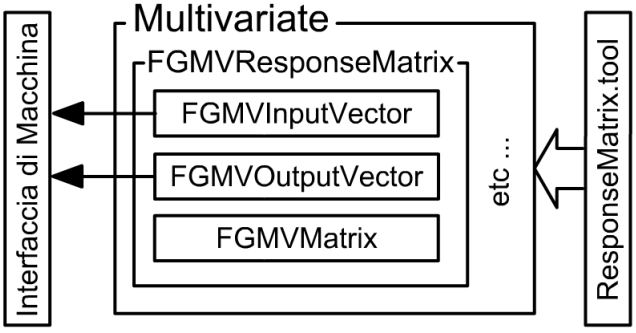



**Figura 2.5:** Implementazione del sistema di misurazione delle matrici di risposta all'interno del pacchetto *Multivariate*.

Tale codice, pensato per venire impiegato da sviluppatori di strumenti di *Corolla*, è stato dotato di un'interfaccia accessibile all'utente mediante l'implementazione del tool *ResponseMatrix.tool*, il quale espone una semplice interfaccia grafica basata sul servizio di finestre di dialogo incluso in *Corolla* che permette di eseguire la misura di una matrice di controllo e di esportarne su file il risultato.

# Capitolo 3

# Beam Base Alignment

*La perfezione dell'uomo consiste proprio nello scoprire le proprie imperfezioni.*

– Sant'Agostino di Ippona

## 3.1 Introduzione

I due anelli di DAΦNE sono dotati di centinaia di componenti attivi, la maggior parte dei quali sorgenti *2n*-polari, kicker e beam position monitor. Una frazione consistente di questi elementi è dotata di guide magnetiche aventi una massa compresa tra il quintale e la decina di tonnellate. Queste ingenti masse pongono pesanti vincoli ingegneristici nella progettazione delle strutture di supporto e, unitamente alla compattezza dell'acceleratore, limitano l'utilizzo di attrezzature di precisione per eseguire il posizionamento (l'esempio più comune è la limitata visibilità delle mire laser di cui sono dotate le componenti della macchina). Se a questo aggiungiamo le forze prodotte dall'interazione dei campi magnetici generati da differenti componenti e la dilatazione termica dell'acciaio impiegato per costruire le strutture di supporto che sostengono l'acceleratore di particelle, risulta evidente come sia in fase di design che in fase di operazione si debba tenere conto di una certa tolleranza nell'allineamento della componentistica di macchina.

Imperfezioni nel posizionamento degli elementi magnetici possono determinare alterazioni nella dinamica dei fasci tali da limitare le prestazioni dell'acceleratore stesso[8][9]. Questo è vero non solo nel caso di DAΦNE, ma anche riferito ad un acceleratore generico, anche se è necessario notare come l'influenza del disallineamento sulle prestazioni vari notevolmente tra le macchine. Nel caso di DAΦNE la tolleranza dell'allineamento meccanico è limitata a 100$\mu$m traslazionali, nel caso di acceleratori lineari o di *Free Electron*

*Lasers* in cui gli effetti del disallineamento sono più marcati si può arrivare ad eseguire l'allineamento con precisioni che raggiungono il singolo micrometro. Lo sforzo tecnologico impiegato per eseguire l'allineamento della componentistica di macchina è dimensionato in funzione della precisione richiesta e limitato da fattori di ordine economico e lascia, generalmente, un margine di miglioramento non irrilevante ai fini dell'esperimento in corso[10].

Lo strumento denominato *Beam Base Alignment*, sviluppato contestualmente al progetto *Corolla* persegue il fine di determinare il disallineamento delle varie componenti basandosi unicamente sulle misure che la strumentazione installata sulla macchina ed operabile on line è in grado di eseguire. Trattandosi di un tool incorporato nel complesso del progetto *Corolla*, esso è stato sviluppato cercando di garantirgli la massima generalità possibile; pertanto la natura dei canali di lettura (e, come vedremo, di controllo) impiegati non entra minimamente a far parte dell'implementazione. Nel caso di DAΦNE, che analizzeremo nel dettaglio, verranno impiegate le misure effettuate per mezzo dei 47 *Beam Position Monitor* installati su ciascuno dei due anelli: si tratta di strumenti in grado di misurare la posizione del fascio nel piano trasverso all'orbita.

La misura di *Beam Base Alignment* ha tre scopi precipui che discuteremo qui di seguito.

### 3.1.1 Individuazione di errori macroscopici

Le operazioni di assemblaggio di un collisore sono estremamente complesse. Come è noto, la probabilità di commettere errori cresce di pari passo con la complessità del progetto in corso di realizzazione. Al livello di complessità di un acceleratore di particelle moderno è sostanzialmente impossibile non commettere alcun errore di assemblaggio. Tenuto conto di ciò, è possibile che uno di questi errori si manifesti sotto forma di un disallineamento macroscopico (fuori dalle tolleranze previste) di una delle componenti di macchina.

Le cause principali di questi errori macroscopici di assemblaggio sono correlate ad errori di comunicazione (quote erroneamente riportate, correzioni nella geometria di progetto trasmesse in maniera inesatta, tolleranze macroscopiche non adeguatamente segnalate etc.) e ad errori procedurali (allineamenti non eseguiti, fissaggi temporanei non revisionati etc.).

Lo strumento di *Beam Base Alignment* si propone di individuare eventuali disallineamenti macroscopici e di segnalarli al team di controllo, mettendolo in tal modo in condizione di decidere quale strategia adottare per contrastarli.

### 3.1.2 Riduzione dei disallineamenti fisiologici

Come ho già detto un certo grado di disallineamento della componentistica di macchina è fisiologico all'interno di un acceleratore di particelle. A seconda della componentistica interessata, gli effetti possono risultare più o meno devastanti. Per esempio, se un dipòlo viene traslato o ruotato rispetto all'azimuth l'effetto più rilevante è un'alterazione del focheggiamento nel piano orizzontale; se lo stesso quadrupolo viene, invece, traslato smetterà di eseguire unicamente una funzione focheggiante e comincierà a distorcere anche la traiettoria dell'asse del fascio. Se un kicker viene traslato nel piano trasverso gli effetti di tale disallinemento saranno diffcilmente percepibili, se, invece, viene ruotato lungo l'asse del fascio inizierà ad introdurre un fattore di accoppiamento tra le componenti trasverse proporzionale alla sua intensità di correzione.

La mappatura di questi disallineamenti mediante il *Beam Base Alignment* permette di pianificare interventi di correzione meccanica in grado di eliminare gli effetti più deteriori, riducendo così il livello di disallineamento fisiologico dell'acceleratore.

### 3.1.3 Correzione del modello di macchina

Il progetto *Corolla* favorisce l'impiego del modelling online quale strumento da integrare all'interno del software di controllo. L'efficacia di tale strumento è pesantemente influenzata dall'accuratezza del modello impiegato; è quindi fondamentale introdurre al suo interno il maggior numero di informazioni disponibili sulla macchina e sulle sue imperfezioni.

Le misure effettuate mediante lo strumento di *Beam Base Alignment* possono in linea di principio essere anche usate per incrementare l'accuratezza del modello di macchina intervenendo sull'eliminazione della discrepanza tra le funzioni ottiche simulate e le funzioni ottiche reali.

## 3.2 Beam Base Alignment efficace

Come ho spiegato precedentemente, l'asse delle funzioni ottiche prodotte dagli elementi di un acceleratore di particelle possiede un certo grado di disallineamento rispetto alla posizione effettiva dell'asse del fascio. Tale disallineamento è indesiderabile sia perché, come vedremo, introduce nuove funzioni ottiche non previste nel design, sia perché, legando ad un singolo parametro di controllo funzioni ottiche di natura differente, rende più difficoltoso il controllo stesso.

Il disallineamento delle funzioni ottiche rispetto al fascio è dovuta a tre fattori:

- Il primo contributo è dovuto alla collocazione spaziale degli elementi di macchina
- Il secondo contributo è dovuto al fallace allineamento dell'asse magnetico rispetto all'involucro esterno della singola componente
- Il terzo contributo è dovuto al passaggio del fascio in una posizione differente rispetto a quella prevista dal design.

Quest'ultimo fattore esula dagli scopi del *Beam Base Alignment*, dal momento che l'aggiustamento della traiettoria di passaggio del fascio viene di solito effettuato mediante tecniche di ottimizzazione dell'orbita attuate per mezzo di correttori magnetici che agiscono indipendentemente su ciascuno dei due assi trasversi.

Il disallineamento delle funzioni ottiche generato dal secondo fattore è, generalmente, molto inferiore rispetto a quello generato dal primo tanto che, talvolta, ci si riferisce con il termine *Beam Base Alignment* alla misura o correzione unicamente dei disallineamenti dovuti al primo fattore.

Quando si esegue una misura (o una correzione) del disallineamento basata sul metodo meccanico della componentistica di macchina, quello che si misura è, in realtà, la posizione relativa dei dispositivi di misura (ad esempio i mirini per l'allineamento laser) installati sulle componenti e non la posizione delle funzioni ottiche generate dalle componenti stesse. Questo comporta che le aberrazioni dovute all'imperfetta realizzazione delle componenti di macchina sono inevitabilmente impossibili da rilevare e correggere. Eseguendo, invece, la misura del *Beam Base Alignment* basandosi sugli effetti delle funzioni ottiche sul fascio di particelle, essa sarà riferita in maniera naturale ai campi magnetici realmente presenti sull'asse del fascio.

## 3.3 Effetti del disallineamento

### 3.3.1 Sviluppo in multipoli

Andiamo ora ad analizzare nel dettaglio matematico alcuni degli effetti sull'ottica del fascio generati dal disallineamento di alcune componenti magnetiche.

Nel design di una componente ottica di un acceleratore di particelle due fattori entrano prepotentemente in gioco: la tolleranza sulla posizione del fascio e la sua dimensione. Con la parziale eccezione dei dipoli di curvatura, tutte le componenti ottiche vengono disegnate intorno ad un asse che in fase di assemblaggio dovrà coincidere con l'asse del fascio.

L'impiego di campi elettrici per focheggiare il fascio di particelle risulta inefficace al crescere dell'energia delle stesse, dato che il momento delle particelle entra nella forza di Lorentz solo nella componente magnetica. L'utilizzo di campi magnetici per deflettere i fasci di particelle si rende necessario già a partire da pochi keV, come testimoniato dal fatto che i tubi catodici con giogo capacitivo difficilmente superano un rapporto tra apertura e profondità e superiore a 0.3, mentre i CRT ultra sottili a giogo magnetico raggiungono valori sei volte più elevati.

In secondo luogo i campi impiegati per focalizzare e deflettere il fascio sono (idealmente) unicamente diretti lungo il piano trasverso. Campi collineari al fascio di particelle vengono utilizzati per scopi differenti. Nel caso di DAΦNE, ad esempio, un debole campo azimutale viene impiegato per ridurre il fenomeno dell'*electron cloud* all'interno dell'anello positronico.

Poste queste due condizioni, la descrizione del campo elettromagnetico si riduce da sei a due componenti, ossia le due componenti del campo magnetico ortogonali all'asse del fascio, che denomineremo $B_x$ e $B_y$ utilizzando il sistema di riferimento di Frenet-Serret[1]. Possiamo interpretare questi due campi come le componenti di un campo complesso. Andiamo, quindi, a definire:

$$\tilde{B} = B_x + iB_y \tag{3.1}$$

Tale campo si può scrivere sviluppato in multipoli come:

$$\begin{gathered}\tilde{B} = \sum_{n=1}^{\infty}(\alpha_n + i\beta_n)\cdot(x+iy)^{n-1} = \sum_{n=1}^{\infty} C_n \cdot z^{n-1} \\ con \quad C_n \equiv (\alpha_n + i\beta_n) \quad e \quad z \equiv x + iy\end{gathered} \tag{3.2}$$

ove il coefficiente complesso $C_n$ viene definito *momento magnetico 2(n+1)-polare*. La crescente potenza di $z$ al crescere di $n$ fa sì che per $z$ di modulo piccolo i termini di maggior influenza nello sviluppo siano i primi. Assumendo (correttamente) che l'asse effettivo del fascio poco si discosti da tale asse e che le dimensioni trasverse del fascio siano sufficientemente piccole, risulta efficace sviluppare in multipoli il campo magnetico intorno all'asse di design ed ignorare i momenti di multipolo superiori al primo non nullo.

### 3.3.2 Classificazione dei disallineamenti

Lo spazio delle fasi dei possibili disallineamenti possiede sei dimensioni: tre spaziali e tre angolari. Chiameremo $\Delta x$, $\Delta y$ e $\Delta s$ i disallineamenti lungo le

[1] Definito relativamente ad un luogo nell'orbita come una terna di assi cartesiani avente origine in un punto appartenente al luogo dell'orbita ideale, con un asse denominato $\hat{s}$ diretto lungo tale orbita nella direzione del momento delle particelle, un asse $\hat{x}$ equiverso e collineare alla curvatura macroscopica della traiettoria ideale e un asse $\hat{y} = \hat{s} \times \hat{x}$

tre direzioni spaziali definite nel sistema di riferimento di Frenet-Serret, $\Delta\theta$ una rotazione destrorsa intorno all'asse $\hat{y}$, $\Delta\phi$ una rotazione destrorsa lungo l'asse $\hat{x}$ e $\Delta\psi$ una rotazione destrorsa lungo l'asse $\hat{s}$.

Dal momento che nel corso della nostra trattazione matematica (così come, ad esempio, nell'implementazione eseguita all'interno del pacchetto di simulazione MAD-X) assumeremo gli errori di allineamento essere piccoli, possiamo eseguire l'approssimazione di piccole rotazioni ed assumerle commutanti. Non precisiamo, quindi, un ordine di applicazione delle definizioni precedentemente fornite.

### 3.3.3 Effetti dei disallineamenti sulle funzioni ottiche

Assumiamo che una sorgente magnetica venga spostata di una piccola quantità lungo una o più delle direzioni di disallineamento sopraelencate e ci chiediamo quale sia la relazione tra i coefficienti dello sviluppo in multipoli della sorgente originaria e quelli della sorgente disallineata.

#### Traslazione lungo $\widehat{s}$

In questo caso lo sviluppo multipolare del campo magnetico generato dall'elemento non cambia. L'effetto del disallineamento si riscontra nell'alterazione delle matrici di trasporto relative ai tratti di beam pipe che precedono e seguono l'elemento.

#### Traslazione lungo $\widehat{x}$ e $\widehat{y}$

L'effetto della traslazione sul piano trasverso è quella di traslare il campo di una quantità $(\Delta x, \Delta y)$. Ritorniamo allo sviluppo presentato in (3.3) traslando l'origine dello sviluppo di una quantità $\Delta z \equiv \Delta x + i\Delta y$

$$\begin{aligned}
\tilde{B} &= \sum_{n=1}^{\infty}(\alpha_n + i\beta_n)\cdot(z-\Delta z)^{n-1} \\
&= \sum_{n=1}^{\infty}(\alpha_n + i\beta_n)\cdot\left[z^{n-1} - (n-1)z^{n-2}\Delta z + ... + (-\Delta z)^{n-1}\right] \\
&\approx \sum_{n=1}^{\infty}(\alpha_n + i\beta_n)\cdot(z^{n-1} - (n-1)z^{n-2}\Delta z) \qquad (3.3)
\end{aligned}$$

Come si può vedere, ogni momento va ad influenzare i momenti di ordine inferiore, con effetto massimo sul momento precedente nello sviluppo. Andiamo a riscrivere la sommatoria:

$$
\begin{aligned}
\tilde{B} &\approx \sum_{n=1}^{\infty}(\alpha_n + i\beta_n)\cdot(z^{n-1} - (n-1)z^{n-2}\Delta z) \\
&\approx \sum_{n=1}^{N}(\alpha_n + i\beta_n)\cdot z^{n-1} - \sum_{n=1}^{N+1}(\alpha_n + i\beta_n)(n-1)\ z^{n-2}\ \Delta z \\
&= \sum_{n=1}^{N}(\alpha_n + i\beta_n)\cdot z^{n-1} - \sum_{n=0}^{N}(\alpha_{n+1} + i\beta_{n+1})\ n\ z^{n-1}\ \Delta z \\
&= \sum_{n=1}^{N}\Big[(\alpha_n + i\beta_n) - n(\alpha_{n+1} + i\beta_{n+1})\ \Delta z\Big]\ z^{n-1} \\
&= \sum_{n=1}^{N}\Big[\ \ \alpha_n\ -\ n\ \alpha_{n+1}\Delta x\ -\ n\ \beta_{n+1}\Delta y \\
&\qquad\qquad i\ \big(\beta_n\ +\ n\ \beta_{n+1}\Delta x\ -\ n\ \alpha_{n+1}\Delta y\big)\ \Big]\ z^{n-1}
\end{aligned}
\tag{3.4}
$$

Da cui la trasformazione dei coefficienti multipolari:

$$
\begin{aligned}
\tilde{\alpha}_n &= \alpha_n - n\ \alpha_{n+1}\ \Delta x - n\ \beta_{n+1}\ \Delta y \\
\tilde{\beta}_n &= \beta_n + n\ \beta_{n+1}\ \Delta x - n\ \alpha_{n+1}\ \Delta y
\end{aligned}
\tag{3.5}
$$

che, scritti con numeri complessi risulta:

$$
\tilde{c}_n = c_n - i\,n\,c_{n+1}\Delta z \tag{3.6}
$$

**Rotazione lungo $\widehat{\psi}$**

L'applicazione di una rotazione positiva di un angolo $\Delta\psi$ alla sorgente del campo magnetico ruota il campo magnetico di un uguale angolo. Similmente a quanto fatto per il caso traslazionale presentato nel punto precedente, ruotiamo le coordinate di un angolo $-\Delta\psi$:

$$
\begin{aligned}
\tilde{B} &= \sum_{n=1}^{\infty}(\alpha_n + i\beta_n)\cdot(ze^{-i\ \Delta\psi})^{n-1} \\
&= \sum_{n=1}^{\infty}e^{-i(n-1)\Delta\psi}(\alpha_n + i\beta_n)\cdot z^{n-1}
\end{aligned}
\tag{3.7}
$$

Volendo una dipendenza lineare, andiamo a sviluppare $e^{-i(n-1)\Delta\psi}$ per $(n-1)\Delta\psi \ll 1$

$$\begin{aligned} e^{-i(n-1)\Delta\psi} &= cos\,((n-1)\Delta\phi) \;-\; i\;sin\,((n-1)\Delta\phi) \\ &\approx 1 - i\;(n-1)\Delta\phi \end{aligned} \tag{3.8}$$

$$\begin{aligned} \tilde{B} \;\approx\; \sum_{n=1}^{N} \Big( \quad & \alpha_n \;+\; \beta_n(n-1)\Delta\psi \;+ \\ & i(\beta_n \;-\; \alpha_n(n-1)\Delta\psi) \quad \Big) \cdot z^{n-1} \end{aligned} \tag{3.9}$$

Si noti che lo sviluppo è stato troncato ad un numero finito di termini, infatti la condizione $(n-1)\Delta\psi \ll 1$ viene falsificata per $n \to \infty$ con $\Delta\psi \neq 0$. La trasformazione dei coefficienti risulta:

$$\begin{aligned} \tilde{\alpha}_n &= \alpha_n \;+\; \beta_n(n-1)\Delta\psi \\ \tilde{\beta}_n &= \beta_n \;-\; \alpha_n(n-1)\Delta\psi \end{aligned} \tag{3.10}$$

**Rotazione lungo $\hat{\theta}$ e $\hat{\phi}$**

Similmente al caso della traslazione lungo l'asse $\hat{z}$, gli effetti di queste rotazioni non possono essere descritti unicamente mediante una variazione dei coefficienti dello sviluppo in multipoli. Per eseguire una trattazione di tali rotazioni dovremo considerare, almeno temporaneamente, la profondità della sorgente. Finora abbiamo assunto una sorgente ideale non sottile, ovvero una sorgente dotata di profondità finita $L$, avente campo magnetico nullo per $s \leq 0 \vee L \leq s$ e costante rispetto ad $s$ per $0 \leq s \leq L$.

Ora, il raggio di curvatura indotto da un campo magnetico $\vec{B}$ sulla traiettoria di una particella avente momento $\vec{p}$ e carica $q$ è pari a:

$$\rho \;=\; \frac{|\vec{p}|}{|\vec{B}|\;q} \tag{3.11}$$

Nel caso in cui $\rho \ll L \cdot |\vec{p}| \cdot |\vec{p} \cdot \hat{s}|^{-1}$ nella regione dello spazio delle fasi accessibile alle particelle del fascio, si può eseguire l'approssimazione di sorgente sottile, nella quale si approssima la deflessione dal multipolo esercitata su di una particella di impulso $\vec{p}$ in coordinate $(x, y, 0)$ come:

$$\Delta\vec{p} \;\approx\; L\; q\; \frac{\vec{p}\times\vec{B}}{|\vec{p}\cdot\hat{s}|} \tag{3.12}$$

Nel caso di un acceleratore di particelle possiamo assumere che il momento trasverso sia molto più piccolo del momento totale. Utilizzando l'approssimazione $\vec{p} \;\approx\; |\,\vec{p}\,|\cdot\hat{s}$ otteniamo:

$$\Delta\vec{p} \;\approx\; L\; q\; \frac{|\,\vec{p}\,|\hat{s}\times\vec{B}}{|\,\vec{p}\,|} \;=\; L\; q\; \left(\hat{s}\times\vec{B}\right) \tag{3.13}$$

Tale approssimazione può venire impiegata anche per descrivere elementi dallo spessore non trascurabile. Per fare ciò è necessario dividere l'elemento spesso in elementi sottili, scrivere la funzione di trasporto che mappa lo spazio delle fasi prima dell'attraversamento di ciascuno di questi elementi sottili, nello spazio delle fasi dopo l'attraversamento e comporre tali funzioni tra di loro.

Nel momento in cui si va ruotare l'elemento lungo gli assi $\hat{x}$ e $\hat{y}$ la base viene trasformata mediante la matrice di rotazione:

$$\begin{aligned}
\begin{pmatrix} x' \\ y' \\ s' \end{pmatrix} \;&=\; \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\Delta\phi & sin\Delta\phi \\ 0 & -sin\Delta\phi & cos\Delta\phi \end{pmatrix} \begin{pmatrix} \cos\Delta\theta & 0 & -\sin\Delta\theta \\ 0 & 1 & 0 \\ \sin\Delta\theta & 0 & \cos\Delta\theta \end{pmatrix} \begin{pmatrix} x \\ y \\ s \end{pmatrix} \\
&\approx\; \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & \Delta\phi \\ 0 & -\Delta\phi & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -\Delta\theta \\ 0 & 1 & 0 \\ \Delta\theta & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ s \end{pmatrix} \\
&\approx\; \begin{pmatrix} x \;-\; s\Delta\theta \\ y \;+\; s\Delta\phi \\ s \;+\; x\Delta\theta \;-\; y\Delta\phi \end{pmatrix}
\end{aligned} \tag{3.14}$$

Andiamo ad inserire questa trasformazione nella distorsione data dall'approssimazione di lente sottile. Per prima cosa riscriviamo la (3.13) in funzione del campo complesso $\tilde{B}$:

$$\Delta\vec{p} \;\approx\; L\; q\; \left(\hat{s}\times\vec{B}\right) \;=\; L\; q\; i\; \tilde{B} \tag{3.15}$$

Eseguendo il cambio di coordinate:

$$\Delta\vec{p}\,' \;\approx\; L\; q\; i\; \tilde{B}(x',y') \;=\; L\; q\; i\; \tilde{B}(x - s\Delta\phi, y + s\Delta\theta) \tag{3.16}$$

Ma, poiché l'elemento è sottile, i prodotti $s\Delta\theta$ e $s\Delta\phi$ sono correzioni del secondo ordine, per cui ruotare gli elementi lungo $\hat{x}$ o $\hat{y}$ non produce alcun effetto al primo ordine.

## 3.4 Effetti dei disallineamenti sulle matrici di risposta

Nel paragrafo (3.2) abbiamo fornito due valide motivazioni per voler correggere il disallineamento della componentistica di macchinache si possono riassumere in un unico concetto: rendere le caratteristiche dell'acceleratore reale il più possibile simili a quelli di progetto.

Il fisico dedito allo studio dell'ottica di un acceleratore di particelle è a conoscenza della funzione specifica per cui viene utilizzato ogni singolo multipolo: i dipoli servono a deflettere il fascio e a definire l'orbita di riferimento, i quadrupoli a focalizzarlo, i quadrupoli skew a correggere gli accoppiamenti tra i moti trasversi, i sestupoli a correggere la cromaticità, e i multipoli successivi a correggere effetti di non linearità. Individuare la presenza di una funzione ottica indesiderata può in determinate circostanze essere facile, in altre molto difficoltoso. Anche nelle casistiche nelle quali sia possibile individuare funzioni ottiche spurie, non sempre è possibile effettuarne una misura quantitativa, così come individuarne la natura e la collocazione lungo l'orbita. Consideriamo il caso di un quadrupolo leggermente traslato sul piano trasverso: dalla trattazione precedente prevediamo che il suo momento di dipolo smetta di essere nullo e che si aggiunga, quindi, una deflessione parassita del fascio di particelle. È possibile che l'esistenza di tale deflessione sia rilevabile, ad esempio mediante *beam position monitors*; tuttavia non è possibile, unicamente mediante questa misura, determinare se sia il quadrupolo ad essere disallineato o se non lo siano, invece, i rivelatori. Eseguire il *Beam Base Alignment* sfruttando la misura diretta delle funzioni ottiche spurie causate dal disallineamento della componentistica di macchina è reso impossibile anche dalla penuria di misure: se un *beam position monitor* è in grado di rilevare solo due quantità, ogni singola componente montata sulla macchina possiede potenzialmente ben sei direzioni di disallineamento possibili.

L'unica nostra speranza di eseguire la misura dei disallineamenti sarà, quindi, sfruttare l'altro effetto nocivo generato da tali imperfezioni, ovvero l'interferenza con il controllo della macchina. La funzione ben definita di ogni sorgente $2n - polare$ è il motivo per cui le componenti di macchina sono disegnate in modo da possedere uno o più momenti multipolari prevalenti sugli altri, in modo che l'azione di controllo possa, agendo sulla corrente fornita, determinare un effetto facilmente riscontrabile. La comparsa di un momento multipolare parassita è facilmente individuabile mappando gli effetti della variazione della corrente fornita ad un elemento magnetico. La sua collocazione e la sua intensità risulteranno, pure, misurabili. Il *Beam Base Alignment* si baserà, pertanto, sugli effetti indotti dal disallineamento degli elementi magnetici sulle matrici di risposta.

Gli ingredienti del *Beam Base Alignment* saranno:

- Un simulatore in grado di prevedere il comportamento del fascio nel caso in cui l'acceleratore contenga degli elementi magnetici disallineati.
- Una matrice di risposta misurata sull'acceleratore reale.

Il compito del *Beam Base Alignment* sarà quello di individuare degli errori di posizionamento realistici per gli elementi magnetici che, introdotti nel simulatore, determinino una matrice di risposta compatibile, entro le incertezze, con quella misurata sull'acceleratore reale.

Questa tecnica possiede dei limiti, principalmente dovuti alla natura e all'accuratezza dei sensori impiegati per le misure.

Nel caso si impieghino beam position monitor, per esempio, sarà possibile rilevare unicamente la posizione del fascio: si vedranno quindi solo gli effetti derivanti dai momenti di dipolo. La possibilità di rilevare unicamente gli effetti derivanti dalle sorgenti dipolari si tradurrà nell'impossibilità di vedere rotazioni lungo $\hat{\psi}$ per tutti quegli elementi che non possiedono momenti dipolari naturali. Parimenti non sarà possibile misurare il disallineamento lungo $\hat{x}$ e $\hat{y}$ per tutti quegli elementi che non possiedano momenti quadrupolari significativi.

Se la matrice di risposta viene individuata impiegando misuratori del profilo del fascio in grado di rilevare l'ampiezza delle oscillazioni di betatrone, la richiesta per determinare $\hat{\psi}$ sarà di possedere momenti quadrupolari, mentre per misurare $\hat{x}$ e $\hat{y}$ diventa quella di possedere momenti quadrupolari o sestupolari significativi. In generale, una matrice di risposta costruita impiegando le rivelazioni ottenute mediante una varietà di rivelatori consentirà deduzioni più approfondite.

In nessun caso sarà possibile determinare la rotazione lungo $\hat{\theta}$ e $\hat{\phi}$ a meno che non si tratti di una rotazione macroscopica, poiché tale rotazione non provoca effetti al primo ordine sulle funzioni magnetiche della componentistica affetta dal disallineamento. Questa è una limitazione solo apparente, dal momento che, se tali disallineamenti non producono effetti misurabili, la loro correzione non risulta prioritaria; viceversa se si dovesse trattare di una rotazione accentuata al punto di produrrre effetti misurabili allora tali effetti renderanno misurabile il disallineamento.

Infine la possibilità di rilevare disallineamenti consistenti in traslazioni lungo $\hat{s}$ dipende fortemente dalla configurazione dell'acceleratore, dalla natura del singolo pezzo e dai parametri utilizzati per manovrarlo. Non è, quindi, possibile determinare a priori la misurabilità delle traslazioni lungo $\hat{s}$.

## 3.5 Tecnica algoritmica

Come ho detto nel paragrafo (3.4), la tecnica di misura del *Beam Base Alignment* consiste nel cercare la corrispondenza tra una matrice di risposta misurata e una matrice di risposta simulata.

Una tecnica per eseguire tale ricerca potrebbe essere quella di adottare un approccio variazionale cercando di minimizzare la distanza tra la matrice misurata e quella simulata. Ho verificato che una tecnica efficace in tale senso consiste nel calcolare il gradiente della funzione distanza nello spazio dei disallineamenti e quindi nel procedere a minimizzare la distanza muovendosi nella direzione del gradiente. La minimizzazione deve essere eseguita muovendosi di passi di dimensioni opportune, verificando, ad ogni passo che la distanza tra le matrici continui a diminuire, e moltiplicando, ad ogni iterazione, il passo per un fattore numerico compreso tra 1 e 2. Nel momento in cui, a seguito dello spostamento, la distanza tra le matrici dovesse aumentare, è necessario moltiplicare il passo di scansione per $-\frac{1}{2}$. Quando il passo decresce al di sotto del valore inizialmente assegnatogli, si procede a ricalcolare il gradiente e si ripete l'intera procedura fintantoché il decremento ottenuto con ogni singola iterazione non diviene inferiore di una determinata soglia.

Questa tecnica, computazionalmente molto impegnativa, è altamente instabile numericamente. Consideriamo un disallineamento il quale induca una variazione nelle misure così piccola da non poter nemmeno essere rilevata: l'algoritmo iterativo produrrà un risultato altamente imprevedibile relativo a tale disallineamento. Questo avviene perché tale parametro verrà variato non con lo scopo di cercare di farlo corrispondere ad una reale risposta, bensì per farlo corrispondere al rumore presente nella misura della matrice di risposta campione che, relativamente a tale disallineamento, sarà prevalente. Per eseguire il *Beam Base Alignment* dovremo trovare un algoritmo più astuto.

### 3.5.1 Il tensore di risposta

L'unica speranza di riuscire a tenere sotto controllo i fenomeni descritti nel paragrafo precedente è quella di possedere un modello dell'effetto del disallineamento delle componenti sulle matrici di risposta. Per fare questo assumiamo che i disallineamenti siano abbastanza piccoli da generare una risposta lineare dei coefficienti della matrice di risposta. Pertanto andiamo ad approssimare la matrice di risposta modificata come:

$$\widetilde{K}_{i,j} \;\approx\; K_{i,j} \;+\; \sum_{k=0}^{n} T_{i,j,k} \cdot \Delta h_k \tag{3.17}$$

laddove $\Delta h_k$ è il disallineamento lungo il k-esimo grado di libertà, $K_{i,j}$ è la matrice di risposta del caso di disallineamento nullo e $T_{i,j,k}$ è un oggetto cui diamo il nome di **Tensore di Risposta**.

Ora, se applichiamo la definizione di matrice di risposta come miglior approssimazione lineare del comportamento della macchina, come dichiarato in (2.2), la definizione di tensore di risposta appena fornita permette di ricavare un'approssimazione della risposta della macchina anche in presenza di disallineamento delle componenti. Prima di tutto andiamo a scrivere la risposta della macchina secondo la definizione di matrice di risposta:

$$F(\vec{x}) \approx F(\vec{x}_0) + \hat{K} \cdot (\vec{x} - \vec{x}_0) \tag{3.18}$$

Ora, in presenza di disallineamenti possiamo eseguire uno sviluppo simile. Sia $\vec{h}$ il vettore dei disallineamenti. Assumiamo inoltre che lo sviluppo sia stato fatto intorno al punto in cui $\vec{h} = 0$.

$$F_h(\vec{x}_0) \approx F_0(\vec{x}_0) + \hat{H} \cdot \vec{h} \tag{3.19}$$

in cui $\hat{H}$ è la matrice di risposta avente come parametri in ingresso il disallineamento delle componenti di macchina. Mettendo insieme le due definizioni precedenti otteniamo:

$$F_h(\vec{x}) \approx F(\vec{x}_0) + \hat{H} \cdot \vec{h} + \hat{K}_h \cdot (\vec{x} - \vec{x}_0) \tag{3.20}$$

dove $\hat{K}_h$ è la parte della matrice di risposta $\hat{K}$ misurata con la componentistica di macchina disallineata del vettore $\vec{h}$. A questo punto utilizziamo la scrittura mostrata in (3.17) per sviluppare $\hat{K}_h$ :

$$F_h(\vec{x}) \approx F(\vec{x}_0) + \hat{K}_h \cdot \vec{h} + \widehat{T} \cdot \vec{h} \cdot (\vec{x} - \vec{x}_0) \tag{3.21}$$

Eseguito questo sviluppo definiamo il tensore di risposta $\widehat{T}$ come quel set di parametri che minimizza il valore di:

$$\sum_{\vec{x}} \left| \widehat{T} \cdot \vec{h} \cdot (\vec{x} - \vec{x}_0) + \hat{K}_h \cdot \vec{h} + F(\vec{x}_0) - F_h(\vec{x}) \right|^2 \tag{3.22}$$

### 3.5.2 Misura del tensore di risposta

La misura del tensore di risposta può avvenire nelle stesse modalità con le quali si misura una matrice di risposta, in particolare è possibile eseguirla mediante Monte Carlo con la definizione fornita nel punto (3.22), in alternativa mediante misura di rapporti incrementali, o ancora utilizzando entrambe le tecniche contemporaneamente introducendo per ogni sorgente di disallineamento una piccola variazione nel modello e misurando la matrice di risposta ottenuta a seguito di ogni variazione mediante Monte Carlo.

Ho trovato questa ultima tecnica di misura particolarmente fruttuosa in quanto nel caso del collisore DAΦNE la variazione dei coefficienti delle matrici di risposta è altamente lineare per piccoli disallineamenti, come mostrato nella figura (3.1)

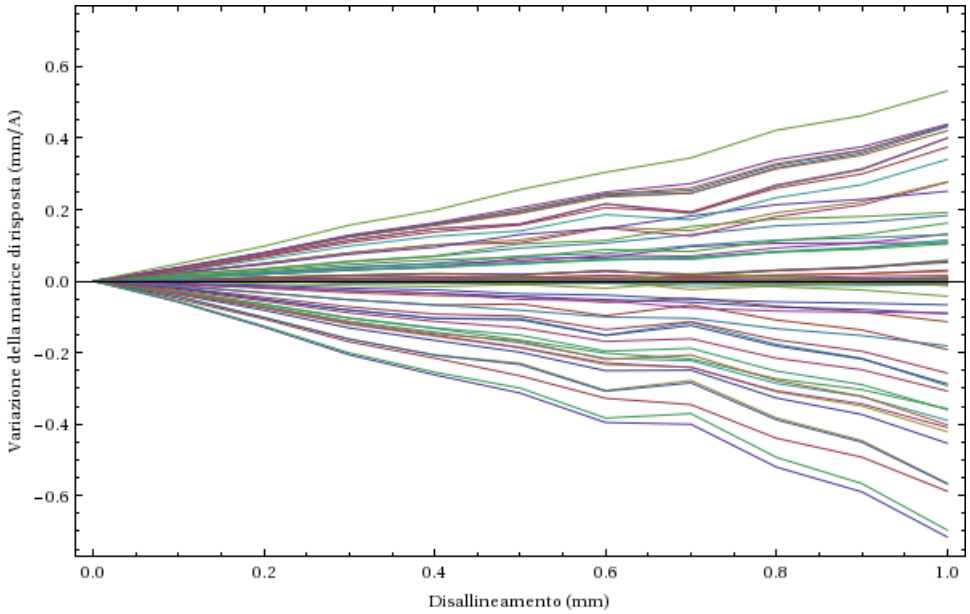



**Figura 3.1:** Linearità dei coefficienti della matrice di risposta in funzione del disallineamento di una componente di macchina. Il grafico è stato ottenuto mediante il simulatore *Mad-X* utilizzando il modello di macchina dell'acceleratore DAΦNE, variando la posizione orizzontale del quadrupolo QUAPL101, in ascissa viene riportato il disallinemento adoperato. In ordinata viene presentata una selezione di 94 elementi della matrice di risposta misurata mediante Monte Carlo, il comportamento lineare è evidente.

### 3.5.3 Beam Base Alignment mediante tensore di risposta

A questo punto siamo in grado di fornire una formula chiusa per ricostruire il disallineamento della componentistica di macchina partendo dal tensore di risposta. Sia $\hat{K}_{j,k}$ la matrice di risposta misurata sull'acceleratore reale, $\hat{K}_{0j,k}$ la matrice di risposta in assenza di disallineamnti prevista dal modello e $\widehat{T}_{i,j,k}$ il tensore di risposta ricavato dal modello. La ricostruzione dei disallineamenti si ottiene minimizzando in funzione del vettore dei disallineamenti $\vec{h}$ l'espressione:

$$\sum_{j,k} \Big| \sum_{i} T_{i,j,k} h_i - K_{j,k} - K_{0j,k} \Big|^2 \tag{3.23}$$

Per fare ciò è conveniente condensare gli indici $j$ e $k$ in un unico indice distruggendo la struttura matriciale di $K$. Ridotta la matrice di risposta ad un vettore che ne contiene le componenti ed il tensore di risposta ad una matrice, la formula (3.23) diviene:

$$\sum_{j} \Big| \sum_{i} T_{i,j} h_i - K_j - K_{0j} \Big|^2 \tag{3.24}$$

Per evitare confusione nell'impiego della notazione $\hat{K}$, d'ora in poi indicherò la forma contratta a due indici del tensore di risposta con il simbolo $\hat{A}$, riservando la notazione $\hat{T}$ alla forma a tre indici. La minimizzazione (3.24) può essere eseguita in formula chiusa, ottenendo:

$$\begin{aligned} \vec{h} &= \left({}^t\hat{A} \cdot \hat{A}\right)^{-1} \cdot {}^t\hat{A} \left(\vec{K} - \vec{K}_0\right) \\ &= \left({}^t\hat{A} \cdot \hat{A}\right)^{-1} \cdot {}^t\hat{A} \cdot \Delta\vec{K} \end{aligned} \tag{3.25}$$

Questo approccio contrasta le limitazioni dovute alla complessità computazionale generata dalla tecnica descritta nel paragrafo (3.5), ma continua ad essere affetto dagli stessi problemi di instabilità numerica.

### 3.5.4 La fattorizzazione SVD

Per risolvere i problemi di instabilità numerica illustrati nel paragrafo (3.5) è necessario agire sul tensore di risposta e modificarlo. Come vedremo nel momento in cui andremo ad analizzare la propagazione degli errori attraverso il *Beam Base Alignment* la natura stessa della minimizzazione proposta in (3.5) causa questa instabilità, per cui solo una modifica attiva del modello

di macchina (nel nostro caso nella sua versione approssimata dal tensore di risposta) può risolvere il problema.

Per eseguire la correzione del tensore di risposta è necessario prima di tutto riscriverlo in una forma che evidenzi la sorgente delle instabilità osservate. La scelta più comune è quella di fattorizzare la matrice mediante *Singular Value Decomposition*.

La *Singular Value Decomposition* (o fattorizzazione SVD) consiste nello scrivere una matrice reale o complessa $\hat{A}$ come prodotto di tre matrici $\hat{U}$, $\hat{B}$ e ${}^{*}\hat{V}$:

$$\hat{A} \;=\; \hat{U}\;\hat{B}\;{}^{*}\hat{V} \tag{3.26}$$

in cui $\hat{U}$ e $\hat{V}$ sono matrici ortogonali e $\hat{B}$ è una matrice diagonale. Dal momento che nel nostro caso la matrice $\hat{A}$ possiede coefficienti reali, le matrici $\hat{U}$ e $\hat{V}$ saranno matrici di rotazione e la coniugazione di $\hat{V}$ si ridurrà ad una semplice trasposizione, per cui nel nostro caso:

$$\hat{A} \;=\; \hat{U}\;\hat{B}\;{}^{t}\hat{V} \tag{3.27}$$

Inoltre, data la decomposizione SVD di una matrice, la decomposizione della trasposta discende immediatamente:

$${}^{t}\hat{A} \;=\; \hat{V}\;\hat{B}\;{}^{t}\hat{U} \tag{3.28}$$

Se ora andiamo a riscrivere la formula (3.25) impiegando la fattorizzazione SVD, otteniamo:

$$\begin{aligned}
\vec{h} \;&=\; \left(\hat{V}\;\hat{B}\;{}^{t}\hat{U}\,\hat{U}\;\hat{B}\;{}^{t}\hat{V}\right)^{-1}\hat{V}\;\hat{B}\;{}^{t}\hat{U}\cdot\Delta\vec{K} \\
&=\; \left(\hat{V}\;\hat{B}\;\hat{B}\;{}^{t}\hat{V}\right)^{-1}\hat{V}\;\hat{B}\;{}^{t}\hat{U}\cdot\Delta\vec{K} \\
&=\; \left(\hat{V}\;\hat{B}^{-2}\;{}^{t}\hat{V}\right)\hat{V}\;\hat{B}\;{}^{t}\hat{U}\cdot\Delta\vec{K} \\
&=\; \hat{V}\;\hat{B}^{-2}\;\hat{B}\;{}^{t}\hat{U}\cdot\Delta\vec{K} \\
&=\; \hat{V}\;\hat{B}^{-1}\;{}^{t}\hat{U}\cdot\Delta\vec{K}
\end{aligned} \tag{3.29}$$

Se si considera che il costo dell'inversione della matrice diagonale $B$ cresce linearmente con le dimensioni, si vede come il costo computazionale della regressione, una volta calcolato il tensore di risposta, si concentri interamente sulla decomposizione SVD del tensore di risposta.

Analizziamo, ora, come si propagano gli errori di misura attraverso l'operazione di regressione mostrata dall'equazione (3.29). Il nostro scopo non

sarà quello di eseguire una analisi completa delle incertezze relative ai valori restituiti dall'algoritmo (riserveremo in seguito a tale analisi una trattazione dettagliata), il nostro scopo, per il momento, sarà di tracciare la propagazione di una ben specifica sorgente di incertezza e cercarne di limitarne il contributo. Trascuriamo, quindi, l'incertezza con cui è conosciuto il tensore di risposta, così come gli errori dovuti all'approssimazione lineare della risposta della macchina e concentriamoci sugli effetti della imperfetta misura della variazione della matrice di risposta $\Delta\vec{K}$.

Sia $\vec{\delta}_K$ l'errore commesso nella misura di $\Delta\vec{K}$ e $\vec{\delta}_h$ l'errore propagato nella determinazione di $\vec{h}$. Per linearità varrà la relazione:

$$\begin{aligned} \delta_h &= \hat{V}\hat{B}^{-1}\,{}^t\hat{U}\,\delta_K & (3.30)\\ |\delta_h|^2 &= |\,\hat{V}\hat{B}^{-1}\,{}^t\hat{U}\delta_K\,|^2 & (3.31) \end{aligned}$$

Sfruttiamo l'ortogonalità di $\hat{V}$ per semplificarlo.

$$|\delta_h|^2 = |\,\hat{B}^{-1}\,{}^t\hat{U}\delta_K\,|^2 \qquad (3.32)$$

Definiamo, ora, per semplicità di notazione $\vec{\omega} \equiv {}^t\hat{U}\Delta\vec{K}$ e $\delta\vec{\omega} \equiv {}^t\hat{U}\vec{\delta}_K$.

$$\begin{aligned} |\delta_h|^2 &= |\,\hat{B}^{-1}\,\delta\vec{\omega}\,|^2 \\ &= \left|\sum_i B^{-1}_{i,i}\cdot\delta\vec{\omega}_i\right|^2 \\ &= \sum_i B^{-2}_{i,i}\cdot\delta\omega_i^2 & (3.33) \end{aligned}$$

Da questa scrittura risulta evidente l'origine dell'instabilità numerica finora discussa: i termini della $\hat{B}$ aventi modulo molto piccolo danno origine a valori di modulo elevato nel momento in cui sono elevati all'esponente $-2$, amplificando l'errore $\delta\omega_i$. Proviamo ad eliminare tale instabilità andando a modificare il tensore di risposta. Per farlo dividiamo ogni elemento diagonale della matrice $\hat{B}$ per un valore $s_i$ da noi deciso arbitrariamente. Ciò equivale a moltiplicare la matrice $\hat{B}^{-1}$ all'interno dello sviluppo per una matrice diagonale $\hat{S}$ costruita a partire dai valori $s_i$.

Andiamo a calcolare qual è l'errore complessivo $\vec{\delta}_h$ in queste condizioni tenendo contro sia dell'errore introdotto dall'errata misura della matrice di risposta, sia dell'errore introdotto dallla nostra modifica arbitraria del tensore di risposta:

$$
\begin{aligned}
\vec{h}_0 &= \hat{V}\hat{B}^{-1}\,{}^t\hat{U}\,\Delta\vec{K} && (3.34)\\
\vec{h} &= \hat{V}\hat{S}\hat{B}^{-1}\,{}^t\hat{U}\,(\Delta\vec{K}+\vec{\delta}_K) && (3.35)\\
\delta\vec{h} &= \vec{h}-\vec{h}_0 \\
&= \hat{V}\hat{S}\hat{B}^{-1}\,{}^t\hat{U}\,(\Delta\vec{K}+\vec{\delta}_K)-\hat{V}\hat{B}^{-1}\,{}^t\hat{U}\,\Delta\vec{K} && (3.36)\\
|\,\delta\vec{h}\,|^2 &= \Big|\ \hat{V}\Big(\hat{S}\hat{B}^{-1}\,{}^t\hat{U}\,(\Delta\vec{K}+\vec{\delta}_K)-\hat{B}^{-1}\,{}^t\hat{U}\,\Delta\vec{K}\Big)\ \Big|^2 \\
&= \Big|\ \hat{S}\hat{B}^{-1}\,{}^t\hat{U}\,\Delta\vec{K}+\hat{S}\hat{B}^{-1}\,{}^t\hat{U}\,\vec{\delta}_K-\hat{B}^{-1}\,{}^t\hat{U}\,\Delta\vec{K}\ \Big|^2 \\
&= \Big|\ \hat{S}\hat{B}^{-1}\vec{\omega}+\hat{S}\hat{B}^{-1}\vec{\delta\omega}-\hat{B}^{-1}\,\vec{\omega}\ \Big|^2 \\
&= \Big|\ (\hat{S}-\hat{1})\hat{B}^{-1}\,\vec{\omega}+\hat{S}\hat{B}^{-1}\vec{\delta\omega}\ \Big|^2 \\
&= \Big|\ \sum_i (s_i-1)B^{-1}_{i,i}\,\omega_i+s_iB^{-1}_{i,i}\delta\omega_i\ \Big|^2 \\
&= \sum_i \frac{1}{B^2_{i,i}}\ \Big|\ (s_i-1)\,\omega_i+s_i\,\delta\omega_i\ \Big|^2 && (3.37)
\end{aligned}
$$

Minimizzando per $s_i$, otteniamo $s_i = 1$ se $\delta\omega_i \leq \omega_i$ e $s_i = 0$ se $\delta\omega_i > \omega_i$. Ciò detto, valutando gli errori sulle componenti misurate di $\Delta\vec{K}$ e propagandole attraverso la moltiplicazione per la matrice ${}^tU$ è possibile determinare che valore dare ad ogni singola componente di $\hat{S}$. Si noti che, a differenza di quanto comunemente creduto a causa della somiglianza delle tecniche utilizzate per il *Beam Base Alignment* con le tecniche utilizzate per effettuare la correzione dell'orbita[11][12], il valore assoluto dell'autovalore $B_{i,i}$ non contribuisce a determinare se tale autovalore vada preservato o eliminato.

## 3.6 Verifica del funzionamento

Come ho illustrato nel paragrafo (1.4.4) una delle possibilità offerte dalla struttura di *Corolla* è la capacità di testare mediante simulatore strumenti disegnati per lavorare sulla macchina. Risulta, pertanto, naturale sfruttare questa opportunità di verificare le capacità dello strumento *Beam Base Alignment*.

Per eseguire il test ho utilizzato il simulatore Mad-X caricato con il modello di macchina dell'acceleratore DAΦNE aggiornato al luglio 2013. Questa scelta di acceleratore è assai impietosa nei confronti dell'algoritmo da testare, DAΦNE, infatti, è una macchina di notevole complessità ottica sulla quale LOCO[13], algoritmo allo stato dell'arte per il *Beam Base Alignment*, ha già fallito in passato. Tale inefficacia è principalmente riconducibile agli effetti

introdotti dai *fringe fields* sull'orbita. Ponendomi su di un piano slegato dall'ottica effettiva di macchina, il cui calcolo è affidato interamente al simulatore Mad-X, spero di superare questa difficoltà.

Per il primo dei test che illustrerò in questa trattazione ho misurato il tensore di risposta relativo al disallineamento lungo $\hat{x}$ e $\hat{y}$ dei quadrupoli non *skew* installati sull'arco *long* dell'anello positronico. Per ognuno dei 52 possibili disallineamenti è stata misurata la matrice di risposta ottenuta alterando la corrente di ciascun quadrupolo nell'intervallo $-0.5A \leq \Delta I \leq +0.5A$ ed eseguento la lettura su entrambi i canali di tutti i 47 *Beam Position Monitor* installati sulla macchina; per eseguire la misura di ciascuna matrice è stato utilizzato il metodo Monte Carlo con un campione di 300 punti.

Dopo aver misurato il tensore di risposta, una delle componenti dell'acceleratore di particelle virtuale è stata disallineata ed una ulteriore matrice di risposta è stata acquisita, questa volta utilizzando un campione di 3000 punti, è stato quindi utilizzato l'algoritmo di *Beam Base Alignment* descritto in questo capitolo per ricostruire tale disallineamento. L'errore sulla misura finale è stato ottenuto variando per 300 volte sia il tensore di risposta sia la matrice campione secondo le rispettive incertezze derivanti dal Monte Carlo e calcolando la varianza della distribuzione dei risultati così ottenuti.

Il risultato del test mostra una netta distinzione tra i casi nei quali il disallineamento è stato eseguito lungo l'asse $\hat{x}$ rispetto ai casi nei quali il disallineamento è stato eseguito lungo l'asse $\hat{y}$.

### 3.6.1 Disallineamento lungo l'asse $\hat{x}$

Nei casi in cui il disallineamento è stato introdotto in direzione dell'asse $\hat{x}$ il disallineamento viene parzialmente ricostruito nella maggior parte dei canali. Pochi canali (sempre gli stessi al variare del disallineamento inserito) presentano barre d'errore patologicamente ampie e, spesso, valori medi incompatibili. La conclusione è che la tecnica di *Beam Base Alignment* non possa ricostruire tutti i disallineamenti possibili della macchina, ma che i canali non ricostruibili siano individuabili mediante questo stesso semplice test.

Nelle figure (3.2), (3.3) e (3.4) si vedono tre esempi di disallineamenti correttamente ricostruiti, in figura (3.5), invece, si vede la ricostruzione del disallineamento di uno dei canali inaffidabili: lo scostamento ricostruito degli altri canali mette in evidenza gli altri canali non siano ricostruibili. In tutti i plot i canali non ricostruibili sono stati marcati in rosa.

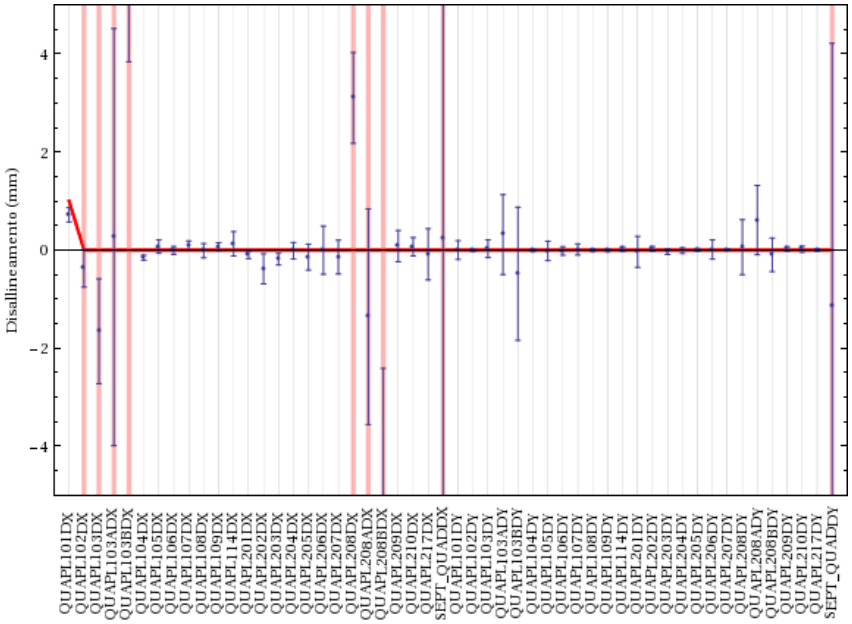



**Figura 3.2:** Ricostruzione mediante l'algoritmo di *Beam Base Alignment* di un disallineamento fittizio imposto al simulatore. In rosso il disallineamento introdotto, in blu la ricostruzione restituita dall'algoritmo, le bande rosa marcano i canali inaffidabili.

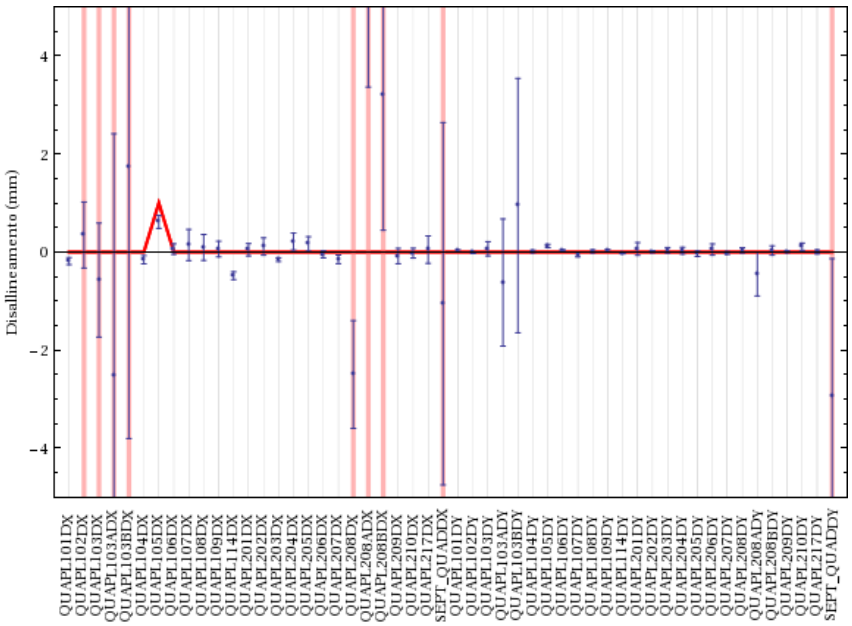



**Figura 3.3:** Vedasi fig. (3.2)

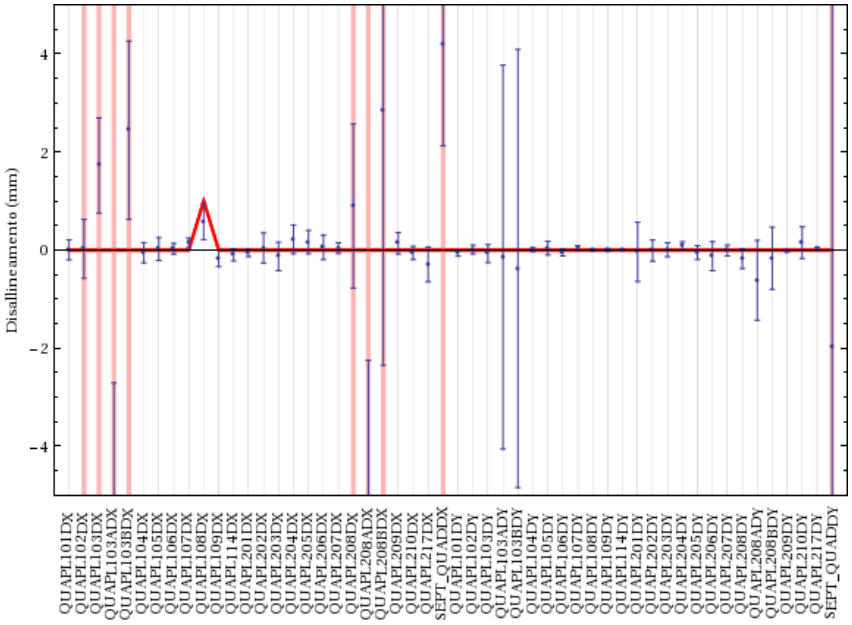



**Figura 3.4:** Vedasi fig. (3.2)

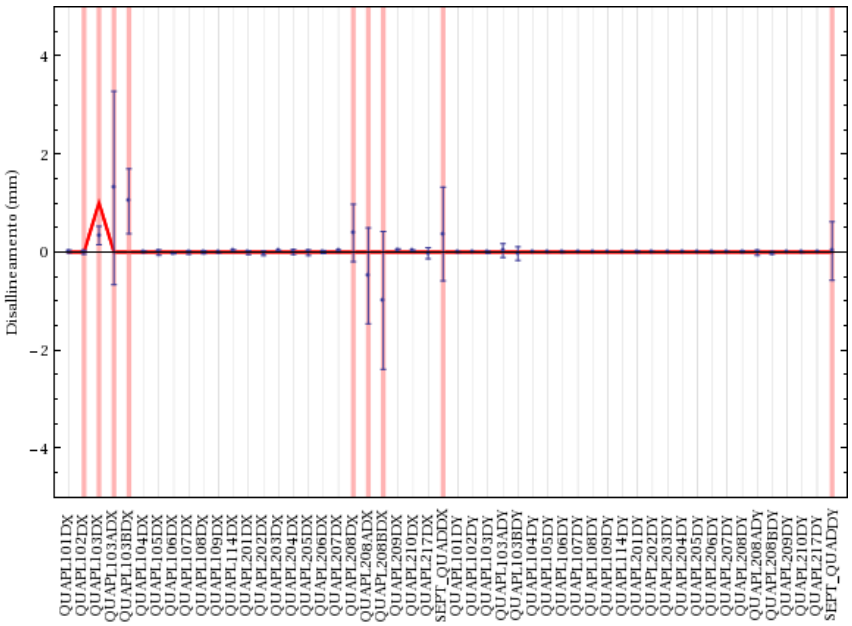



**Figura 3.5:** Vedasi fig. (3.2)

### 3.6.2 Scostamenti lungo l'asse $\hat{y}$

Con la profondità di campionamento adoperata, gli scostamenti lungo l'asse $\hat{y}$ si sono dimostrati impossibili da ricostruire; qualunque tentativo di ricostruzione è sovrastato dal rumore dovuto alla forte somiglianza negli effetti causati sulla matrice di risposta dallo scostamento lungo $\hat{y}$ di diversi quadrupoli.

In figura (3.6) vediamo la fallita ricostruzione di un disallineamento verticale nel quadrupolo *QUAPL106*. In figura (3.7) vediamo tre sezioni del tensore di risposta corrispondenti al disallineamento lungo l'asse $\hat{y}$ dei quadrupoli *QUAPL207*, *QUAPL208* e *QUAPL208B*; dal grafico è evidente come tali risposte siano poco dissimili le une dalle altre, questo comporta una grande difficoltà per l'algoritmo di *beam Base Alignment* nel discernere quale tra esse abbia causato l'effetto misurato e, in assenza di una precisione sufficiente, conduce all'impossibilità riscontrata.

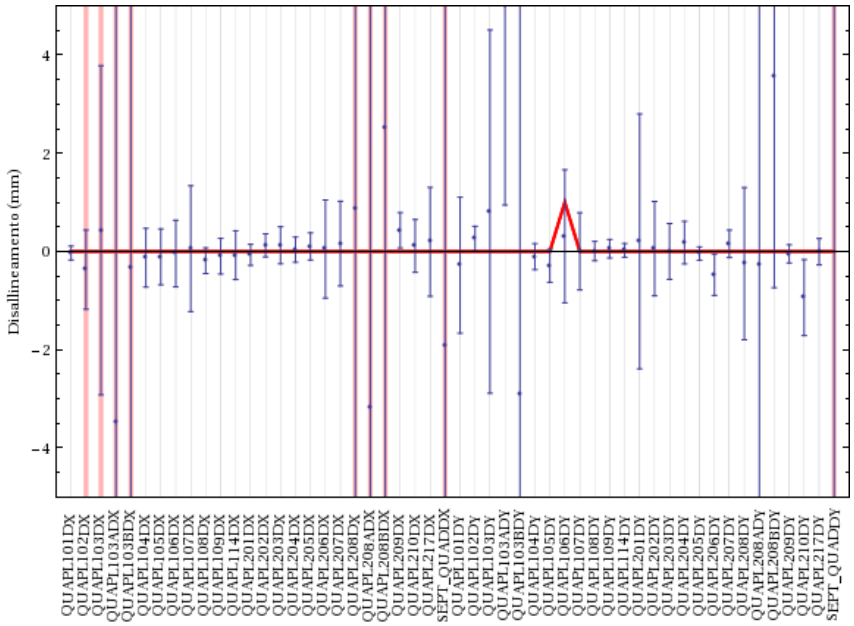



**Figura 3.6:** Vedasi fig. (3.2)

### 3.6.3 Accuratezza della stima degli errori

Per verificare l'accuratezza della stima degli errori adoperata, andiamo a studiare la distribuzione della differenza tra la posizione ricostruita degli elementi

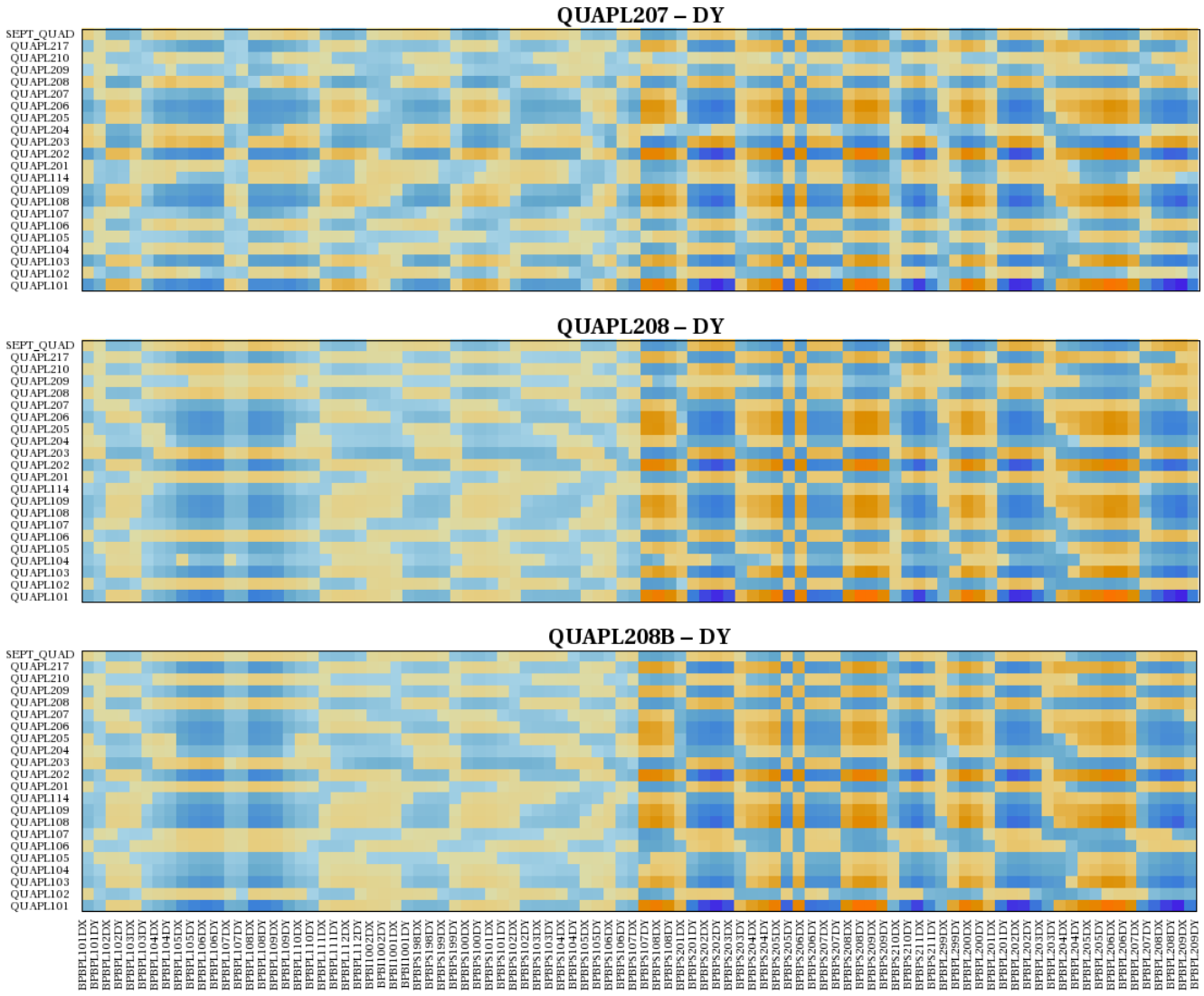



**Figura 3.7:** Sezioni del tensore di risposta corrispondenti alla variazione di matrice di risposta indotta da spostamento positivo lungo l'asse $\hat{y}$ dei quadrupoli *QUAPL207*, *QUAPL208* e *QUAPL208B*. Le tre sezioni mostrano come tutte e tre le variazioni modifichino la matrice di risposta in maniera simile rendendo difficile da distinguere la sorgente di tale alterazione per l'algoritmo di *Beam Base Alignment*

magnetici e quella imposta nel simulatore divise per l'errore sulla ricostruzione calcolato come decritto nel paragrafo (3.6).

La media di tale distribuzione $(-0.062 \pm 0.032)$ è compatibile con zero con una *Confidence Level* del 5%: troppo poco per affermare con certezza la compatibilità, troppo per affermare una sicura incompatibilità. La varianza della distribuzione $(1.040 \pm 0.041)$ è compatibile con 1, da cui deduco che la stima degli errori sia corretta. La forma della distribuzione non è gaussiana, ma, ancora una volta, è ben fittata da una distribuzione di Cauchy avente parametro di localizzazione pari alla media della distribuzione dei campioni e parametro di scala pari al doppio dello scarto quadratico medio.

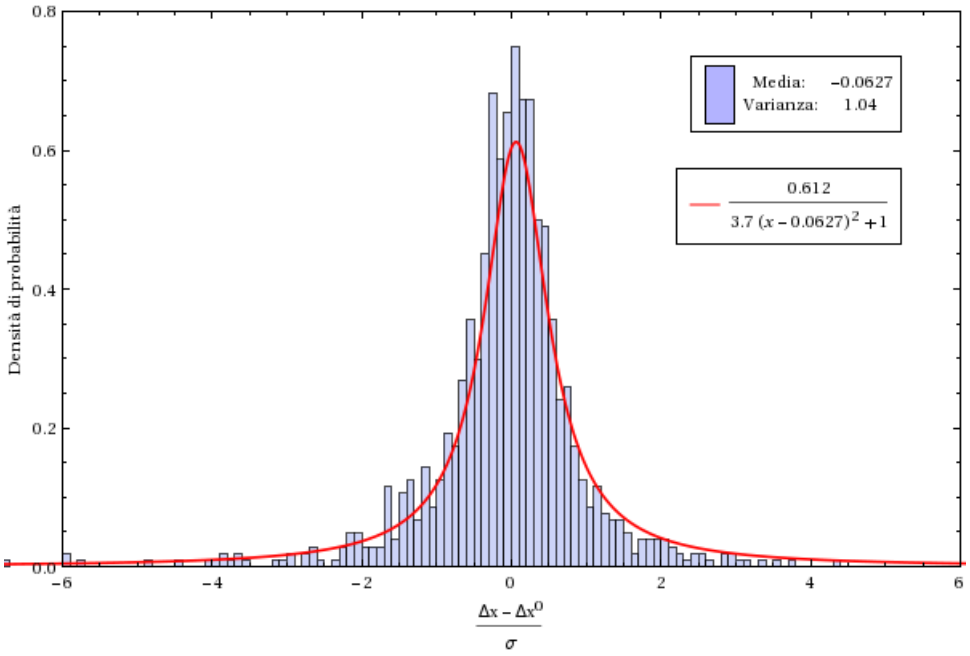



**Figura 3.8:** La differenza tra il disallineamento imposto al simulatore ed il disallineamento ricostruito normalizzata con la stima dell'errore fornita dall'algoritmo di *Beam Base Alignment* proposto. In rosso la distribuzione di Cauchy avente parametro di localizzazione pari alla media e parametro di scala pari al doppio scarto quadratico medio della distribuzione sperimentale.

## 3.7 Prospettive di miglioramento

Come ho detto nel paragrafo (2.4), l'errore sulla matrice di risposta è scomponibile in errore derivante dalla precisione strumentale ed errore dovuto al

Monte Carlo. Se la misurazione di differenti matrici di risposta viene eseguita scegliendo casualmente insiemi di punti ogni volta diversi, le incertezze sulle entrate di matrici di risposta differenti sono correlate solo nella componente dovuta all'errore sistematico della strumentazione di macchina.

La propagazione di tali errori può portare al completo deterioramento dell'informazione contenuta nei valori misurati. Si prenda ad esempio il caso della misurazione del tensore di risposta su di un'interfaccia di simulazione: ogni sezione di tale tensore è determinata dalla differenza di due matrici di risposta. Laddove questa differenza sia piccola rispetto all'errore sperimentale sulle due matrici di risposta l'informazione viene distrutta. Se alle misure eseguite su dataset casuali vengono sostituite misure eseguite su dataset fissi, invece, l'errore dovuto al Monte Carlo si ripercuoterà nel medesimo modo su ciascuna matrice misurata. La correlazione così generata permette, in tutti i casi in cui la misura non si basi sui valori effettivi delle entrate della matrice di risposta bensì sul confronto di matrici distinte, di limitare la propagazione dell'errore alla precisione strumentale.

Questa tecnica alternativa è implementata nativamente in *Corolla* mediante la variabile globale *FGMV::FixedRandom* che, se impostata al valore **true**, resetta il *seed* del generatore di numeri pseudocasuale ad un valore fissato prima di ogni Monte Carlo; se tale valore viene impostato a **false** il *seed* viene impostato ad un valore casuale generato mediante l'orologio di sistema prima di ogni Monte Carlo in modo da garantire l'assenza di correlazione.

Eseguire la misurazione delle matrici di risposta su dataset fissi si discosta dal concetto matematico di matrice di risposta in favore di una sua specifica approssimazione. Fatto questo passo potremmo andare fino in fondo decidendo di non eseguire il fit dei punti raccolti e di eseguire il *Beam Base Alignment* basandoci non sul confronto fra matrici di risposta bensì sul confronto di vettori di misurazioni non fittate. Questa possibilità incrementa certamente la precisione dal momento che evita la distruzione di informazione eseguita dal fit, tuttavia ha il difetto di non basarsi su di un oggetto matematico definito indipendentemente dallo specifico contesto in cui viene impiegato (le *n-uple* di punti possono essere scelte solo una volta fissati il numero di punti ed il numero di dimensioni). La mancanza di una definizione di respiro generale comporta l'impossibilità di adottare questa tecnica a misure eseguite mediante software differenti. Il *Beam Base Alignment* eseguito mediante matrici di risposta, invece, può essere applicato sia a matrici di risposta misurate mediante software differenti, sia ad ogni sottomatrice di una stessa matrice di risposta senza eseguire alcuna misura aggiuntiva.

# Bibliografia


[1] C. Milardi et al. *Present status of the DAΦNE upgrade and perspectives* Int.J.Mod.Phys.A24:360-368, 2009

[2] M. Zobov et al. *DAΦNE Developments for the KLOE-2 Experimental Run* Phys.Rev.Lett.104:174801, 2010

[3] A.Terebilo *Accelerator Toolbox for MATLAB* Stanford Linear Accelerator Center SLAC-PUB-8732, Maggio 2001

[4] Austin Joint Working Group *IEEE Standard for Information Technology - Portable Operating System Interface (POSIX(R))* IEEE Computer Society 1003.1, 2008

[5] Dave Shreiner, OpenGL Architecture Review Board. *OpenGL Reference Manual: The Official Reference Document to OpenGL, Version 1.4 (4th Edition)* Addison-Wesley Professional ISBN 0-321-17383-X, 2004

[6] J. Postel, J. Reynolds *File Transfer Protocol (FTP)* Information Sciences Institute RFC 959, Ottobre 1985

[7] MAD-X Home Page. http://mad.web.cern.ch/mad/

[8] M.D. Woodley et al. *Beam-Based Alignment at the KEK-ATF Damping Ring* LBNL-55728, Giugno 2004

[9] S. M. Liuzzo et al. *Tests for low vertical emittance at Diamond using LET algorithm* Proceedings of IPAC2011 WEPC013, 2011

[10] S. M. Liuzzo et al. *Tests of low emittance tuning techniques at SLS and DAΦNE* National Syncrotron Light Source of Brookhaven ISBN 978-3-95450-155-1 2012

[11] Eva Bozoki e Aharon Friedman *Optimization Method for Orbit Correction in Accelerators* National Syncrotron Light Source of Brookhaven PAC1993

[12] Aharon Friedman ed Eva Bozoki *Use of eigenvectors in understanding and correcting storage ring orbits* National Syncrotron Light Source of Brookhaven NIMPR A 344 (1994) 269-277, Settembre 1993

[13] J. Safrenek *Experimental determination of storage ring optics using orbit response measurements* National Syncrotron Light Source of Brookhaven NIMPR A 388 (1997) 27-36, Dicembre 1996