UNIVERSITÀ DEGLI STUDI DI TRIESTE

Dipartimento di Fisica

CORSO DI LAUREA TRIENNALE IN FISICA

TESI DI LAUREA

UTILIZZO DEI SENSORI DI SHACK-HARTMANN PER L'ANALISI DI FRONTI D'ONDA NEI SISTEMI OTTICI

Laureanda:
Alberta Candussi

Relatore:
Edoardo Milotti

ANNO ACCADEMICO 2021-2022

Ai miei genitori

Indice

1 Introduzione alla misura della fase di un fronte d'onda e ai sensori SHWF

In questa sezione introduco brevemente la teoria legata all'ottica geometrica prendendo spunto dal libro "Fundamentals of Photonics" di Saleh e Teich e analizzo in modo qualitativo il funzionamento e la struttura di un sensore di Shack-Hartmann.

1.1 Breve introduzione teorica all'ottica geometrica

I raggi luminosi, in relazione al mezzo in cui si propagano, possono seguire traiettorie diverse. In ogni caso è comunque possibile definire una funzione scalare $S(\vec{r})$ tale per cui la condizione $S(\vec{r})$ =costante rappresenta tutte le superfici perpendicolari alle traiettorie stesse. Tali superfici vengono chiamate fronti d'onda geometrici e sono rappresentati schematicamente in figura 1.

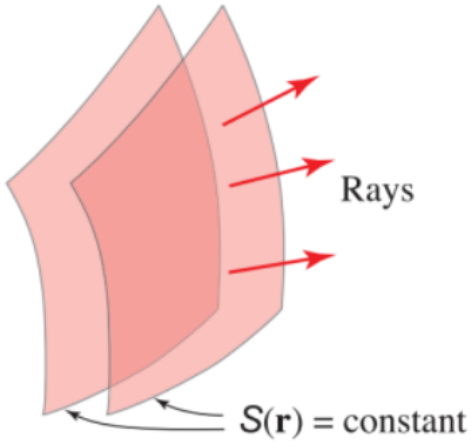


Figura 1: Fronti d'onda geometrici, figura tratta da [3].

La conoscenza della funzione $S(\vec{r})$ è fondamentale in quanto, facendone il gradiente $\nabla S(\vec{r})$, si ottengono direttamente le traiettorie dei raggi luminosi. Si può dunque fare un parallelismo con quanto avviene in elettromagnetismo dove vale la relazione $\vec{E} = -\nabla V$; in questo caso il ruolo delle linee di campo elettrico è ricoperto dai raggi luminosi mentre quello del potenziale dalla funzione $S(\vec{r})$ che dunque, data la sua importanza, prende il nome di funzione iconale.

Dovendo rispettare il principio di Fermat, che afferma che il cammino ottico dei raggi luminosi che si propagano da un punto A ad un punto B dello spazio è sempre quello che richiede il minor tempo di percorrenza, la funzione iconale deve rispettare la condizione:

$$|\nabla S(\vec{r})|^2 = n^2(\vec{r}) \tag{1}$$

che scritta in forma esplicita diventa:

$$\left|\frac{\partial S\left(\vec{r}\right)}{\partial x}+\frac{\partial S\left(\vec{r}\right)}{\partial y}+\frac{\partial S\left(\vec{r}\right)}{\partial z}\right|^{2}=n^{2}\left(\vec{r}\right) \tag{2}$$

dove $n\left(\vec{r}\right)$ è l'indice di rifrazione del mezzo in cui le onde luminose si propagano.

Si può dunque fare una differenziazione tra i mezzi di propagazione omogenei, caratterizzati dall'avere $n(\vec{r})$ =costante e per i quali il modulo di $|\nabla S\left(\vec{r}\right)|$ è costante, che avranno traiettorie dei raggi rettilinee e quelli disomogenei, dove $n(\vec{r})$ $\neq$costante, le cui traiettorie dei raggi possono essere calcolate risolvendo l'equazione iconale.

Analogamente a quanto si fa in elettromagnetismo per il potenziale scalare, è possibile integrare la funzione iconale tra due punti A e B. Di seguito si riporta il calcolo:

$$\int_{A}^{B}\left|\nabla S\left(\vec{r}\right)\right|ds=\int_{A}^{B}n\,ds=S\left(\vec{r}_{B}\right)-S\left(\vec{r}_{A}\right) \tag{3}$$

Quanto calcolato corrisponde alla lunghezza del cammino ottico tra i punti A e B dello spazio ed è l'analogo della differenza di potenziale elettromagnetico.

Il formalismo della funzione iconale fornisce dunque la base teorica che permette di trattare la propagazione dei fronti d'onda elettromagnetici mediante l'ottica geometrica, nel limite di lunghezza d'onda trascurabile.

1.2 Studio di un fronte d'onda tramite i sensori SHWF

Un metodo per la misura della distribuzione della fase di un fronte d'onda è rappresentato dall'utilizzo dei sensori di Shack-Hartmann anche detti in inglese Shack-Hartmann Wavefront Sensors (SHWF).

La figura 2 mostra lo schema concettuale di un sensore di Schack-Hartmann che riassume il loro funzionamento. Internamente è costituito da due piani, in quello superiore è disposto un array di $n \times n$ microlenti convergenti mentre in quello inferiore è posta una camera digitale caratterizzata da $m \times m$ pixel, con $m > n$. Questo dispositivo permette di registrare l'impulso luminoso come segnale elettronico tramite un sensore di immagine che produce una corrente o una tensione proporzionale all'irradianza incidente. I due piani si trovano ad una distanza d pari a quella focale dell'array di microlenti. Alle volte tale separazione può essere leggermente maggiore o minore a d in modo tale da ottenere una variazione delle dimensioni dello spot luminoso proiettato sulla camera digitale secondo le specifiche esigenze operative.

Il principio di funzionamento è molto semplice: un generico fronte d'onda elettromagnetico colpisce l'array di microlenti; se si considera la porzione di esso che incide sulla singola microlente il fronte d'onda può essere considerato come piano; la luce viene focalizzata dalle microlenti sul piano della camera digitale e l'immagine dei raggi focalizzati viene acquisita

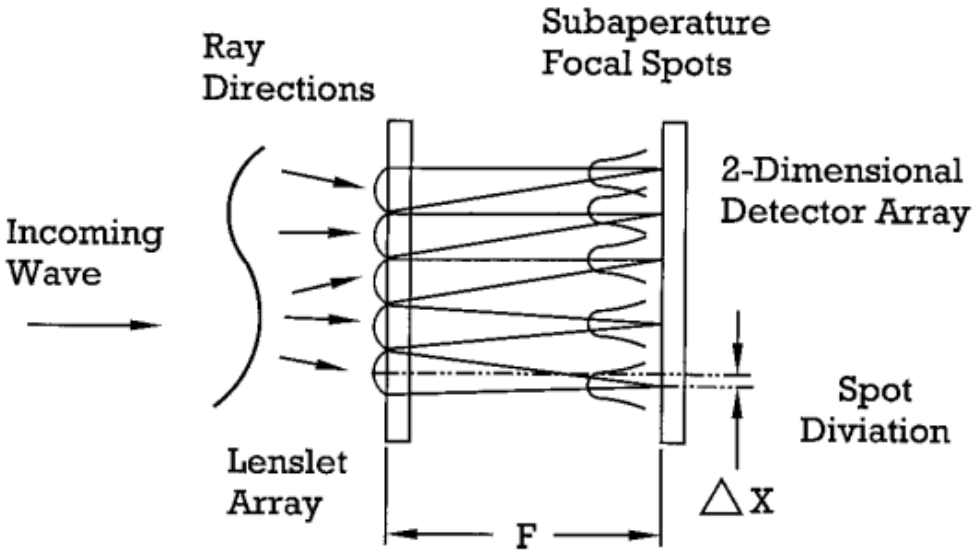


Figura 2: Schema concettuale di un sensore Shack-Hartmann, da [4].

dal sistema digitale. Se il fronte d'onda incidente è perfettamente piano e perpendicolare al sensore, ogni microlente focalizzerà la luce esattamente nella proiezione del suo centro sul piano della camera digitale. Questi punti di focalizzazione sono considerati come punti di focalizzazione di riferimento. Se il fronte d'onda non è perfettamente piano ma presenta delle aberrazioni e distorsioni di vario genere, allora il punto di focalizzazione non sarà più quello di riferimento ma si troverà più o meno scostato rispetto ad esso in base all'angolo con cui il fronte d'onda, localmente approssimato come piano, ha inciso sulla lente. Dallo studio di tale scostamento è possibile comprendere e ricostruire la distribuzione della fase del generico fronte d'onda incidente.

2 Determinazione della posizione dell'asse ottico di un fronte d'onda sferico tramite SHWF

Consideriamo un fronte d'onda sferico che incide sul sensore. Trascurando la polarizzazione, il campo elettrico dell'onda è

$$E(r,t) = \frac{E_0}{r} e^{i\vec{k}\cdot(\vec{r}-\vec{v}t)} \tag{4}$$

L'asse ottico è definito come il segmento perpendicolare al piano del sensore passante per la sorgente dell'onda sferica. I parametri che vogliamo individuare corrispondono alla posizione dell'asse ottico sul piano del rivelatore e alla distanza della sorgente dal rivelatore.

In questa sezione descrivo un metodo analitico per ricavare questi parametri partendo inizialmente da una descrizione bidimensionale (con un rivelatore ridotto ad un segmento) per poi estenderla al caso di un rivelatore reale.

2.1 Caso unidimensionale

La situazione è schematizzata in figura 3:

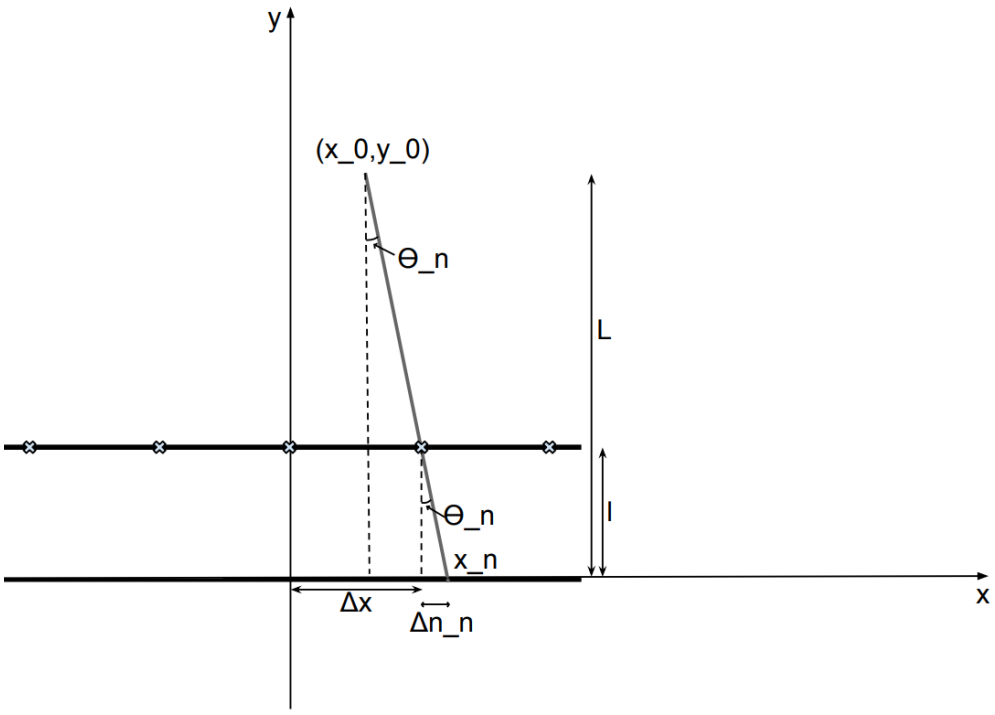


Figura 3: Schema del caso unidimensionale.

Posto un sistema di riferimento cartesiano in modo tale che l'asse delle ordinate coincida con l'asse di simmetria dell'array di microlenti, le incognite del problema risultano essere la

posizione del centroide, indicata con (x_0, y_0), e il valore del raggio di curvatura L del fronte d'onda sferico. Le quantità costanti Δx e l si riferiscono rispettivamente allo scostamento tra due punti di riferimento adiacenti e alla distanza focale delle n microlenti convergenti presenti nell'array, mentre x_n rappresenta l'ascissa dell'n-esimo spot luminoso che incide sulla camera digitale con un angolo θ_n rispetto alla perpendicolare al piano focale passante per il centroide. Si è indicato con Δn_n^{th} il valore teorico dello spostamento dell' n-simo spot luminoso dall'n-esimo punto di riferimento corrispondente mentre il valore sperimentale misurato è identificato con Δn_n^{exp} e corrisponde al valore teorico al quale si è sommata una componente Gaussiana di rumore.

Considerando la geometria del problema, mediante semplici relazioni trigonometriche, si dimostra che valgono le seguenti relazioni:

$$\Delta n_n^{\text{th}} = l \tan\theta_n \tag{5a}$$
$$x_n - x_0 = L \tan\theta_n \tag{5b}$$
$$x_n = \Delta n_n^{\text{th}} + n\Delta x \tag{5c}$$

Considerando $\theta \to 0$, ossia in approssimazione di piccoli angoli, è possibile espandere la tangente tramite uno sviluppo in serie di Taylor ottenendo al primo ordine:

$$\Delta n_n^{\text{th}} \simeq l\theta_n \tag{6a}$$
$$x_n - x_0 \simeq L\theta_n \tag{6b}$$

Si può dunque ricavare lo spostamento teorico tramite i seguenti passaggi:

$$\Delta n_n^{\text{th}} + n\Delta x - x_0 = L\theta_n \tag{7a}$$
$$\theta_n = \frac{\Delta n_n^{\text{th}} + n\Delta x - x_0}{L} \tag{7b}$$
$$\theta_n = \frac{l\theta_n + n\Delta x - x_0}{L} \tag{7c}$$
$$\left(1 - \frac{l}{L}\right)\theta_n = \frac{n\Delta x - x_0}{L} \tag{7d}$$
$$\theta_n = \frac{n\Delta x - x_0}{L - l} \tag{7e}$$
$$\Delta n_n^{\text{th}} = \frac{l}{L - l}\left(n\Delta x - x_0\right) \tag{7f}$$

Alla stessa espressione è possibile giungere anche senza considerare l'approssimazione di piccoli angoli in quanto entrambe le relazioni considerate hanno la medesima dipendenza dalla tangente dell'angolo di deflessione dalla perpendicolare alla superficie del sensore.

Definisco ora la funzione χ^2 come:

$$\chi^2 = \sum_n \frac{\left(\Delta n_n^{\text{th}} - \Delta n_n^{\text{exp}}\right)^2}{\sigma^2} \tag{8}$$

che minimizzo per ottenere una stima dei parametri incogniti x_0 e L. A tal scopo calcolo le derivate e risolvo il sistema di equazioni ottenuto rispetto a x_0 e L. Di seguito riporto in dettaglio il calcolo per la minimizzazione rispetto a x_0:

$$\frac{\partial \chi^2}{\partial x_0} = 0 \tag{9a}$$

$$\frac{\partial}{\partial x_0} \sum_n \frac{(\Delta n_n^{\text{th}} - \Delta n_n^{\text{exp}})^2}{\sigma^2} = 0 \tag{9b}$$

$$\frac{\partial}{\partial x_0} \sum_n \frac{1}{\sigma^2} \left[\frac{l}{L-l} (n\Delta x - x_0) - \Delta n_n^{\text{exp}} \right]^2 = 0 \tag{9c}$$

$$-2 \sum_n \frac{1}{\sigma^2} \left[\frac{l}{L-l} (n\Delta x - x_0) - \Delta n_n^{\text{exp}} \right] \left(\frac{l}{L-l} \right) = 0 \tag{9d}$$

$$-2 \left(\frac{l}{L-l} \right)^2 \sum_n \frac{1}{\sigma^2} \left[(n\Delta x - x_0) - \frac{L-l}{l} \Delta n_n^{\text{exp}} \right] = 0 \tag{9e}$$

$$\sum_n \frac{1}{\sigma^2} \left[(n\Delta x - x_0) - \frac{L-l}{l} \Delta n_n^{\text{exp}} \right] = 0 \tag{9f}$$

$$\sum_n \frac{1}{\sigma^2} (n\Delta x - x_0) - \frac{L-l}{l} \sum_n \frac{\Delta n_n^{\text{exp}}}{\sigma^2} = 0 \tag{9g}$$

$$\sum_n \frac{1}{\sigma^2} (n\Delta x - x_0) = \frac{L-l}{l} \sum_n \frac{\Delta n_n^{\text{exp}}}{\sigma^2} \tag{9h}$$

$$\sum_n \frac{1}{\sigma^2} (n\Delta x - x_0) = \frac{L}{l} \sum_n \frac{\Delta n_n^{\text{exp}}}{\sigma^2} - \sum_n \frac{\Delta n_n^{\text{exp}}}{\sigma^2} \tag{9i}$$

$$L = \frac{\sum_n \frac{n\Delta x}{\sigma^2} - \sum_n \frac{x_0}{\sigma^2} + \sum_n \frac{\Delta n_n^{\text{exp}}}{\sigma^2}}{\frac{1}{l} \sum_n \frac{\Delta n_n^{\text{exp}}}{\sigma^2}} \tag{9j}$$

La minimizzazione rispetto a L viene eseguita in maniera analoga:

$$\frac{\partial \chi^2}{\partial L} = 0 \tag{10a}$$

$$\frac{\partial}{\partial L} \sum_n \frac{\left(\Delta n_n^{\text{th}} - \Delta n_n^{\text{exp}}\right)^2}{\sigma^2} = 0 \tag{10b}$$

$$\frac{\partial}{\partial L} \sum_n \frac{1}{\sigma^2} \left[\frac{l}{L-l}\left(n\Delta x - x_0\right) - \Delta n_n^{\text{exp}}\right]^2 = 0 \tag{10c}$$

$$-2 \sum_n \frac{1}{\sigma^2} \left[\frac{l}{L-l}\left(n\Delta x - x_0\right) - \Delta n_n^{\text{exp}}\right] \frac{l\left(n\Delta x - x_0\right)}{\left(L-l\right)^2} = 0 \tag{10d}$$

$$-2 \frac{l^2}{\left(L-l\right)^3} \sum_n \frac{\left(n\Delta x - x_0\right)}{\sigma^2} \left[\left(n\Delta x - x_0\right) - \frac{L-l}{l} \Delta n_n^{\text{exp}}\right] = 0 \tag{10e}$$

$$\sum_n \frac{1}{\sigma^2} \left[\left(n\Delta x - x_0\right)^2 - \frac{L-l}{l} \Delta n_n^{\text{exp}} \left(n\Delta x - x_0\right)\right] = 0 \tag{10f}$$

$$\sum_n \frac{\left(n\Delta x - x_0\right)^2}{\sigma^2} - \frac{L-l}{l} \sum_n \frac{\Delta n_n^{\text{exp}}}{\sigma^2} \left(n\Delta x - x_0\right) = 0 \tag{10g}$$

$$\sum_n \frac{\left(n\Delta x - x_0\right)^2}{\sigma^2} = \frac{L-l}{l} \sum_n \frac{\Delta n_n^{\text{exp}}}{\sigma^2} \left(n\Delta x - x_0\right) \tag{10h}$$

$$\sum_n \frac{\left(n\Delta x - x_0\right)^2}{\sigma^2} = \frac{L}{l} \sum_n \frac{\Delta n_n^{\text{exp}}}{\sigma^2} \left(n\Delta x - x_0\right) - \sum_n \frac{\Delta n_n^{\text{exp}}}{\sigma^2} \left(n\Delta x - x_0\right) \tag{10i}$$

$$L = \frac{\sum_n \frac{\left(n\Delta x - x_0\right)^2}{\sigma^2} + \sum_n \frac{\Delta n_n^{\text{exp}}}{\sigma^2} \left(n\Delta x - x_0\right)}{\frac{1}{l} \sum_n \frac{\Delta n_n^{\text{exp}}}{\sigma^2} \left(n\Delta x - x_0\right)} \tag{10j}$$

Uguagliando le due espressioni ottenute per L si ottiene un'equazione nella sola incognita x_0

che permette di trovare la posizione ignota del centroide:

$$\frac{\sum_n \frac{n\Delta x}{\sigma^2} - \sum_n \frac{x_0}{\sigma^2} + \sum_n \frac{\Delta n_n^{\exp}}{\sigma^2}}{\frac{1}{l}\sum_n \frac{\Delta n_n^{\exp}}{\sigma^2}} = \frac{\sum_n \frac{(n\Delta x - x_0)^2}{\sigma^2} + \sum_n \frac{\Delta n_n^{\exp}}{\sigma^2}(n\Delta x - x_0)}{\frac{1}{l}\sum_n \frac{\Delta n_n^{\exp}}{\sigma^2}(n\Delta x - x_0)} \tag{11a}$$

$$\frac{\sum_n \frac{n\Delta x}{\sigma^2} - \sum_n \frac{x_0}{\sigma^2} + \sum_n \frac{\Delta n_n^{\exp}}{\sigma^2}}{\sum_n \frac{\Delta n_n^{\exp}}{\sigma^2}} = \frac{\sum_n \frac{1}{\sigma^2}(n\Delta x - x_0)^2 + \sum_n \frac{\Delta n_n^{\exp}}{\sigma^2}(n\Delta x - x_0)}{\sum_n \frac{\Delta n_n^{\exp}}{\sigma^2}(n\Delta x - x_0)} \tag{11b}$$

$$\frac{\sum_n \frac{n\Delta x}{\sigma^2} - \sum_n \frac{x_0}{\sigma^2} + \sum_n \frac{\Delta n_n^{\exp}}{\sigma^2}}{\sum_n \frac{\Delta n_n^{\exp}}{\sigma^2}} =$$

$$= \frac{\sum_n \frac{n^2\Delta x^2}{\sigma^2} + \sum_n \frac{x_0^2}{\sigma^2} + \sum_n \frac{2x_0 n\Delta x}{\sigma^2} + \sum_n \frac{\Delta n_n^{\exp} n\Delta x}{\sigma^2} - \sum_n \frac{\Delta n_n^{\exp} x_0}{\sigma^2}}{\sum_n \frac{\Delta n_n^{\exp} n\Delta x}{\sigma^2} - \sum_n \frac{\Delta n_n^{\exp} x_0}{\sigma^2}} \tag{11c}$$

Per semplificare la notazione si ridefiniscono le sommatorie

$$s_1 = \sum_n \frac{1}{\sigma^2} \quad s_2 = \sum_n \frac{\Delta n_n^{\exp}}{\sigma^2} \quad s_3 = \sum_n \frac{n\Delta x}{\sigma^2} \quad s_4 = \sum_n \frac{\Delta n_n^{\exp} n\Delta x}{\sigma^2} \quad s_5 = \sum_n \frac{n^2\Delta x^2}{\sigma^2} \tag{12}$$

Si ottiene dunque:

$$\frac{s_5 + x_0^2 s_1 + 2x_0 s_3 + s_4 - x_0 s_2}{s_4 - x_0 s_2} = \frac{s_3 - x_0 s_1 + s_2}{s_2} \tag{13a}$$

$$s_2 s_5 + x_0^2 s_1 s_2 + 2x_0 s_3 s_2 + s_4 s_2 - x_0 s_2^2 = s_3 s_4 - x_0 s_3 s_2 - x_0 s_1 s_4 + x_0^2 s_1 s_2 + s_2 s_4 - x_0 s_2^2 \tag{13b}$$

$$3x_0 s_3 s_2 + x_0 s_1 s_4 = s_3 s_4 - s_2 s_5 \tag{13c}$$

da cui si ricava la posizione del centroide:

$$x_0 = \frac{s_3 s_4 - s_2 s_5}{3s_3 s_2 + s_1 s_4} \tag{14}$$

e di conseguenza il valore del raggio di curvatura:

$$L = \frac{s_3 - x_0 s_1 + s_2}{s_2} l = \frac{s_3 + s_2 - s_1\left(\frac{s_3 s_4 - s_2 s_5}{3s_3 s_2 + s_1 s_4}\right)}{s_2} l \tag{15}$$

2.2 Caso bidimensionale

La situazione è schematizzata in figura 4. La trattazione è analoga a quella del caso unidimensionale ma ora è estesa ad un rivelatore planare. Il sistema di riferimento cartesiano è posizionato in modo tale che l'asse di simmetria del sensore e l'asse z della terna destrorsa coincidano. La posizione incognita del centroide è indicata con (x_0, y_0, z_0) mentre $L = z_0$ è il raggio ignoto della sfera. Le grandezze costanti risultano essere Δx e Δy, pari alle distanze tra i centri delle lenti adiacenti lungo i rispettivi assi, nonché l, la distanza focale dell'array di $n \times n$ lenti convergenti. $\Delta n_{xn}^{\text{th}}$ e $\Delta n_{yn}^{\text{th}}$ rappresentano lo scostamento teorico dell'n-esimo spot luminoso rispettivamente lungo l'asse x e y rispetto al punto focale di riferimento. I corrispettivi scostamenti sperimentali sono indicati con $\Delta n_{xn}^{\text{exp}}$ e $\Delta n_{yn}^{\text{exp}}$ e sono ottenuti dai valori teorici con l'aggiunta di una componente di rumore gaussiano. Infine n_x e n_y indicano il numero di lenti lungo i rispettivi assi.

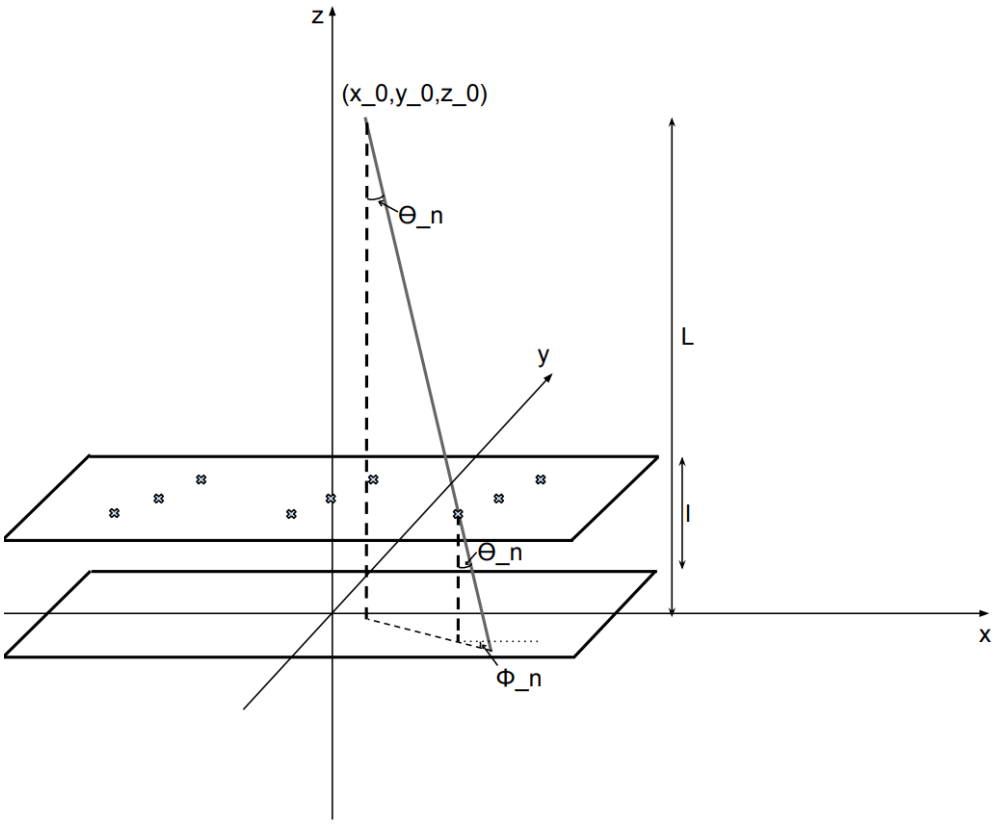


Figura 4: Schema del caso bidimensionale.

Dalla geometria del problema si evince che valgono le seguenti relazioni:

$$\Delta n_{xn}^{\text{th}} = l \tan\theta_n \cos\phi_n \tag{16a}$$
$$\Delta n_{yn}^{\text{th}} = l \tan\theta_n \sin\phi_n \tag{16b}$$
$$L \tan\theta_n \cos\phi_n = n_x \Delta x - x_0 + \Delta n_{xn}^{\text{th}} \tag{16c}$$
$$L \tan\theta_n \sin\phi_n = n_y \Delta y - y_0 + \Delta n_{yn}^{\text{th}} \tag{16d}$$

Assumendo $\theta_n \ll 1$ le formule si approssimano alle seguenti

$$\Delta n_{xn}^{\text{th}} = l\theta_n \cos\phi_n \tag{17a}$$
$$\Delta n_{yn}^{\text{th}} = l\theta_n \sin\phi_n \tag{17b}$$
$$L\theta_n \cos\phi_n = n_x \Delta x - x_0 + \Delta n_{xn}^{\text{th}} \tag{17c}$$
$$L\theta_n \sin\phi_n = n_y \Delta y - y_0 + \Delta n_{yn}^{\text{th}} \tag{17d}$$

Da cui:

$$\theta_n \cos\phi_n = \frac{n_x \Delta x - x_0 + \Delta n_{xn}^{\text{th}}}{L} \tag{18a}$$
$$\theta_n \sin\phi_n = \frac{n_y \Delta y - y_0 + \Delta n_{yn}^{\text{th}}}{L} \tag{18b}$$
$$\Delta n_{xn}^{\text{th}} = \frac{l}{L}\left(n_x \Delta x - x_0 + \Delta n_{xn}^{\text{th}}\right) \tag{18c}$$
$$\Delta n_{yn}^{\text{th}} = \frac{l}{L}\left(n_y \Delta y - y_0 + \Delta n_{yn}^{\text{th}}\right) \tag{18d}$$

e dunque:

$$\Delta n_{xn}^{\text{th}} = \frac{l}{L - l}\left(n_x \Delta x - x_0\right) \tag{19a}$$
$$\Delta n_{yn}^{\text{th}} = \frac{l}{L - l}\left(n_y \Delta y - y_0\right) \tag{19b}$$

Si definisce lo spostamento teorico totale dell'n-esimo spot luminoso come:

$$\Delta n_n^{\text{th}} = \sqrt{\left(\Delta n_{xn}^{\text{th}}\right)^2 + \left(\Delta n_{yn}^{\text{th}}\right)^2} \tag{20}$$

Analogamente al caso unidimensionale si utilizza il metodo dei minimi quadrati per stimare i parametri incogniti ossia la posizione del centroide e il raggio di curvatura. La funzione da

minimizzare è:

$$\chi^2 = \sum_n \frac{\left(\Delta n_n^{\text{th}} - \Delta n_n^{\text{exp}}\right)^2}{\sigma^2} = \sum_n \frac{\left(\sqrt{\left(\Delta n_{x,n}^{\text{th}}\right)^2 + \left(\Delta n_{y,n}^{\text{th}}\right)^2} - \Delta n_n^{\text{exp}}\right)^2}{\sigma^2}$$
$$= \sum_n \frac{\left\{\sqrt{\left[\frac{l}{L-l}\left(n_x\Delta x - x_0\right)\right]^2 + \left[\frac{l}{L-l}\left(n_y\Delta y - y_0\right)\right]^2} - \Delta n_n^{\text{exp}}\right\}^2}{\sigma^2} \tag{21}$$

Di seguito riporto la derivazione della funzione rispetto al parametro x_0:

$$\frac{\partial}{\partial x_0}\sum_n \frac{\left\{\left\{\left[\frac{l}{L-l}\left(n_x\Delta x - x_0\right)\right]^2 + \left[\frac{l}{L-l}\left(n_y\Delta y - y_0\right)\right]^2\right\}^{1/2} - \Delta n_n^{\text{exp}}\right\}^2}{\sigma^2} = 0 \tag{22a}$$

$$\frac{\partial}{\partial x_0}\frac{l}{L-l}\sum_n \frac{\left\{\left[\left(n_x\Delta x - x_0\right)^2 + \left(n_y\Delta y - y_0\right)^2\right]^{1/2} - \frac{L-l}{l}\Delta n_n^{\text{exp}}\right\}^2}{\sigma^2} = 0 \tag{22b}$$

$$-2\frac{l}{L-l}\sum_n \frac{1}{\sigma^2}\left\{\left[\left(n_x\Delta x - x_0\right)^2 + \left(n_y\Delta y - y_0\right)^2\right]^{1/2} - \frac{L-l}{l}\Delta n_n^{\text{exp}}\right\}\cdot$$
$$\cdot \frac{1}{2}\frac{2\left(n_x\Delta x - x_0\right)}{\left[\left(n_x\Delta x - x_0\right)^2 + \left(n_y\Delta y - y_0\right)^2\right]^{1/2}} = 0 \tag{22c}$$

$$\sum_n \frac{1}{\sigma^2}\left\{\left[\left(n_x\Delta x - x_0\right)^2 + \left(n_y\Delta y - y_0\right)^2\right]^{1/2} - \frac{L-l}{l}\Delta n_n^{\text{exp}}\right\}\cdot$$
$$\cdot \frac{n_x\Delta x - x_0}{\left[\left(n_x\Delta x - x_0\right)^2 + \left(n_y\Delta y - y_0\right)^2\right]^{1/2}} = 0 \tag{22d}$$

$$\sum_n \frac{1}{\sigma^2}\left(n_x\Delta x - x_0\right) = \sum_n \frac{\Delta n_n^{\text{exp}}}{\sigma^2}\frac{L-l}{l}\frac{n_x\Delta x - x_0}{\left[\left(n_x\Delta x - x_0\right)^2 + \left(n_y\Delta y - y_0\right)^2\right]^{1/2}} \tag{22e}$$

La derivazione di χ^2 rispetto al parametro y_0 risulta essere invece:

$$\frac{\partial}{\partial y_0}\sum_n \frac{\left\{\left\{\left[\frac{l}{L-l}\left(n_x\Delta x - x_0\right)\right]^2 + \left[\frac{l}{L-l}\left(n_y\Delta y - y_0\right)\right]^2\right\}^{1/2} - \Delta n_n^{\exp}\right\}^2}{\sigma^2} = 0 \tag{23a}$$

$$\frac{\partial}{\partial y_0}\frac{l}{L-l}\sum_n \frac{\left\{\left[\left(n_x\Delta x - x_0\right)^2 + \left(n_y\Delta y - y_0\right)^2\right]^{1/2} - \frac{L-l}{l}\Delta n_n^{\exp}\right\}^2}{\sigma^2} = 0 \tag{23b}$$

$$-2\frac{l}{L-l}\sum_n \frac{1}{\sigma^2}\left\{\left[\left(n_x\Delta x - x_0\right)^2 + \left(n_y\Delta y - y_0\right)^2\right]^{1/2} - \frac{L-l}{l}\Delta n_n^{\exp}\right\}\cdot \cdot\frac{1}{2}\frac{2\left(n_y\Delta y - y_0\right)}{\left[\left(n_x\Delta x - x_0\right)^2 + \left(n_y\Delta y - y_0\right)^2\right]^{1/2}} = 0 \tag{23c}$$

$$\sum_n \frac{1}{\sigma^2}\left\{\left[\left(n_x\Delta x - x_0\right)^2 + \left(n_y\Delta y - y_0\right)^2\right]^{1/2} - \frac{L-l}{l}\Delta n_n^{\exp}\right\}\cdot \cdot\frac{\left(n_y\Delta y - y_0\right)}{\left[\left(n_x\Delta x - x_0\right)^2 + \left(n_y\Delta y - y_0\right)^2\right]^{1/2}} = 0 \tag{23d}$$

$$\sum_n \frac{1}{\sigma^2}\left(n_y\Delta y - y_0\right) = \sum_n \frac{\Delta n_n^{\exp}}{\sigma^2}\frac{L-l}{l}\frac{\left(n_y\Delta y - y_0\right)}{\left[\left(n_x\Delta x - x_0\right)^2 + \left(n_x\Delta x - x_0\right)^2\right]^{1/2}} \tag{23e}$$

Infine la derivazione dell'espressione rispetto al parametro L è:

$$\frac{\partial}{\partial L}\sum_n \frac{\left\{\left\{\left[\frac{l}{L-l}\left(n_x\Delta x - x_0\right)\right]^2 + \left[\frac{l}{L-l}\left(n_y\Delta y - y_0\right)\right]^2\right\}^{1/2} - \Delta n_n^{\exp}\right\}^2}{\sigma^2} = 0 \tag{24a}$$

$$-2\sum_n \frac{1}{\sigma^2}\left\{\frac{l}{L-l}\left[\left(n_x\Delta x - x_0\right)^2 + \left(n_y\Delta y - y_0\right)^2\right]^{1/2} - \Delta n_n^{\exp}\right\}\frac{l}{(L-l)^2}\cdot \cdot\left[\left(n_x\Delta x - x_0\right)^2 + \left(n_y\Delta y - y_0\right)^2\right]^{1/2} = 0 \tag{24b}$$

$$\sum_n \frac{1}{\sigma^2}\frac{l}{L-l}\left[\left(n_x\Delta x - x_0\right)^2 + \left(n_y\Delta y - y_0\right)^2\right] = \sum_n \frac{\Delta n_n^{\exp}}{\sigma^2}\left[\left(n_x\Delta x - x_0\right)^2 + \left(n_y\Delta y - y_0\right)^2\right]^{1/2} \tag{24c}$$

Nel caso bidimensionale non è possibile trovare una soluzione analitica al sistema di parametri in quanto non si riesce a separare e isolare singolarmente i termini riguardanti l'ascissa e quelli riguardanti l'ordinata, per tal motivo si deve ricorrere ad una soluzione numerica.

3 Descrizione dell'algoritmo numerico

In questa sezione descrivo le stime della posizione del centroide e del raggio di curvatura del fronte d'onda sferico incidente sul sensore ottenute in diverse situazioni. A tale scopo, ho implementato il metodo numerico sia per il caso unidimensionale che per quello bidimensionale. I programmi in questione sono stati scritti in Python, e l'appendice riporta i codici sorgenti.

3.1 Caso unidimensionale

Ho considerato un array di $n = 9$ microlenti disposte lungo un segmento. La distanza focale di ciascuna microlente è $l = 5$ cm e la separazione delle microlenti adiacenti è $\Delta x = 2$ cm. Assumendo note le posizioni teoriche x_n degli n spot luminosi, ho simulato i valori sperimentali associati a ciascuna microlente sommando al valore teorico una fluttuazione estratta a caso da una distribuzione Gaussiana con media nulla e varianza pari a $\sigma^2 = 0.0004$ cm^2, costante per tutte le n estrazioni. I valori dei punti focali standard sono invece stati calcolati semplicemente moltiplicando la distanza tra i centri di microlenti adiacenti per il numero della microlente corrispondente. In particolar modo queste ultime sono state numerate in modo tale che la lente posizionata sull'asse y sia considerata come la "lente 0", quelle sul semiasse positivo delle ascisse siano numerate progressivamente, da sinistra verso destra, partendo da 1 mentre quelle nel semiasse negativo siano numerate in modo progressivo in valore assoluto da destra verso sinistra sempre partendo da 1. Note le posizioni teoriche e sperimentali dei raggi luminosi, nonché la posizione dei punti di riferimento standard, si è calcolato lo spostamento teorico e quello sperimentale dello spot luminoso rispetto al centro della lente come differenza in valore assoluto delle rispettive posizioni rispetto a quelle standard. La tabella 1 riporta i risultati ottenuti.

Numero lente [#]	-4	-3	-2	-1	0	1	2	3	4
Ascissa teorica [cm]	-8.52	-6.32	-4.17	-2.07	0.01	2.06	4.11	6.26	8.46
Ascissa sperimentale [cm]	-8.51	-6.34	-4.19	-2.10	0.03	2.07	4.12	6.25	8.49
Ascissa centri lenti [cm]	-8.00	-6.00	-4.00	-2.00	0.00	2.00	4.00	6.00	8.00
Δn_n^{th} [cm]	-0.52	-0.32	-0.17	-0.07	0.01	0.06	0.11	0.26	0.46
Δn_n^{exp} [cm]	-0.51	-0.34	-0.19	-0.10	0.03	0.07	0.12	0.25	0.49

Tabella 1: Risultati ottenuti nel caso unidimensionale.

Ci si aspetta che la differenza tra i valori di scostamento teorici Δn_n^{th} e quelli sperimentali Δn_n^{exp} abbia una distribuzione Gaussiana normale $N(0, \sigma^2)$ ossia che il valore di aspettazione sia $\mu = 0$ cm e che la varianza ad essa associata sia pari a σ^2. Per realizzare l'istogramma

della distribuzione corrispondente ho aumentato il numero di microlenti presenti nell'array da $n = 9$ a $n = 21$ e ripetuto la stima delle differenze per $m = 500$ volte in modo tale da aumentare il numero di valori da inserire nell'istogramma e per rendere dunque più evidente la tipologia di distribuzione delle differenze in esame. In figura 5 riporto l'istogramma in questione normalizzato e il suo fit. Dal grafico si osserva come effettivamente le differenze

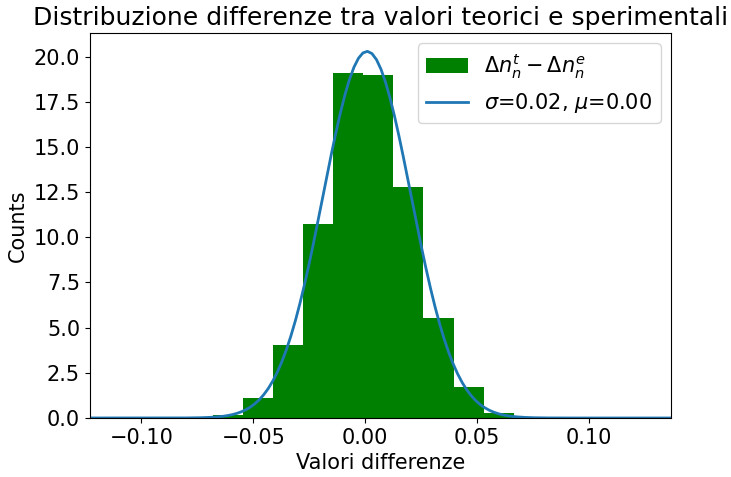


Figura 5: Distribuzione Gaussiana di $\Delta n_n^{\text{th}} - \Delta n_n^{\text{exp}}$.

abbiano l'andamento atteso. Il fit Gaussiano evidenzia che la deviazione standard è $\sigma = 0.02$ cm mentre il valore d'aspettazione risulta $\mu = 0.00$ cm. Simulando svariate volte i dati e realizzando di volta in volta l'istogramma si può incorrere in valori di deviazione standard e di aspettazione del fit leggermente diversi. Tali variazioni rispetto a quanto atteso sono sicuramente attribuibili al fatto che, nonostante abbia ampliato l'array di lenti e sia stato simulato m volte il calcolo delle differenze per riuscire ad ottenere delle evidenze migliori sull'andamento della distribuzione, la statistica rimane ancora troppo bassa. In tal caso, il valore di aspettazione μ nullo e una deviazione standard pari a 0.02 cm si possono dunque ottenere aumentando ulteriormente il numero di ripetizioni delle n differenze.

La posizione vera del centroide e il valore vero del raggio di curvatura si ottengono quando il rumore è nullo. In particolar modo, non essendo possibile richiedere una varianza nulla, li ho calcolati analiticamente con $\sigma^2 = 10^{-12}$ cm^2 e sono rispettivamente pari a $x_0 = 0.38$ cm e $L = 99.49$ cm. Dopo aver definito tutti i valori necessari per analizzare il problema ho inizialmente realizzato una stima analitica dei parametri incogniti utilizzando le espressioni ricavate con il metodo dei minimi quadrati nella sezione precedente. In particolar modo, la posizione del centroide risulta essere pari a 0.35 cm mentre il valore del raggio di curvatura è 96.34 cm. Si osserva dunque che il valore dell'ascissa del centroide è spostato di 0.03 cm rispetto a quello vero mentre il raggio di curvatura è sottostimato di 3.15 cm. Per la trattazione sull'errore associato a queste stime rimando alla prossima sezione.

Successivamente ho realizzato una stima dei medesimi parametri mediante il metodo di Newton-Raphson, anche noto come metodo delle tangenti. Questo metodo consiste nel trovare in modo approssimato lo zero di una funzione mediante l'iterazione di uno stesso blocco di istruzioni finché non viene soddisfatta una condizione di stop. In generale esso viene applicato in un intervallo chiuso $[a, b]$ ove è presente un'unica radice. Partendo da un valore iniziale casuale $x_i \in [a, b]$ e detta $f(x)$ la funzione di cui si vuole trovare la radice, si calcola inizialmente $f(x_i)$; se tale valore è nullo il problema è risolto e x_i rappresenta lo zero della funzione d'interesse, in caso contrario si calcola la derivata della funzione in x_i e se ne trova l'intersezione con l'asse delle ascisse individuando così x_1. Il valore di x_i viene aggiornato a x_1 e si ripete la procedura appena descritta fino a quando si trova un x_n in cui la funzione si annulla. Difficilmente però si riuscirà a trovare un valore di x_n tale per cui $f(x)$ sia esattamente 0 e dunque si è soliti imporre un valore di tolleranza arbitrario `tol` tale per cui, se $f(x_n) <$`tol`, x_n viene considerato lo zero della funzione terminando così il ciclo iterativo.

In formule quanto descritto si ottiene mediante lo sviluppo in serie di Taylor della funzione $f(x)$.

$$f(x) = f(x_i) + f'(x_i)(x - x_i) \tag{25a}$$
$$f(x) = 0 \tag{25b}$$
$$x = x_i - \frac{f(x_i)}{f'(x_i)} \tag{25c}$$

Nella stima numerica realizzata ho considerato separatamente i due parametri da stimare. In particolar modo dapprima ho fatto una stima di x_0 noto L e successivamente ho stimato L noto x_0. I valori presi per noti sono pari a quelli veri, ossia $x_0 = 0.38$ cm e $L = 99.49$ cm. Le due funzioni di cui si vuole trovare la radice sono dunque rispettivamente:

$$f_1(x_0) = \frac{\partial \chi^2}{\partial x_0} = -2 \sum_n \frac{1}{\sigma^2} \left[\frac{l}{L-l} (n\Delta x - x_0) - \Delta n_n^{\exp} \right] \left(\frac{l}{L-l} \right) \tag{26a}$$
$$f_2(L) = \frac{\partial \chi^2}{\partial L} = -2 \sum_n \frac{1}{\sigma^2} \left[\frac{l}{L-l} (n\Delta x - x_0) - \Delta n_n^{\exp} \right] \frac{l(n\Delta x - x_0)}{(L-l)^2} \tag{26b}$$

Le derivate corrispondenti le ho calcolate in modo numerico approssimato tramite rapporto incrementale, in particolare:

$$f_1'(x_0) = \frac{f(x_0 + h) - f(x_0)}{h} \qquad \text{con } h = 10^{-6} \tag{27a}$$
$$f_2'(L) = \frac{f(L + h) - f(L)}{h} \qquad \text{con } h = 10^{-6} \tag{27b}$$

I valori di partenza x_0 ed L li ho scelti casualmente in un intorno dei rispettivi valori attesi e i valori di tolleranza per la condizione di uscita da loop iterativo li ho scelti rispettivamente

come `tol`$= 10^{-6}$ per la determinazione della posizione del centroide e `tol`$= 10^{-6}$ per il raggio di curvatura.

I valori ottenuti sono pari a $x_0 = 0.36$ cm e $L = 96.38$ cm. Per la trattazione sugli errori rimando alla prossima sezione. Si osserva dunque una sottostima della posizione del centroide di 0.02 cm rispetto al valore vero mentre per il raggio di curvatura la sottostima è pari a 3.11 cm.

3.2 Caso bidimensionale

In questo caso ho considerato un'array di 25 microlenti identiche ma disposte ora su un piano quadrato 5×5. La distanza focale delle microlenti, analogamente al caso unidimensionale, è $l = 5$ cm. La distanza tra i centri delle lenti l'ho considerata, per semplicità di trattazione, uguale nelle due componenti e pari a $\Delta x = \Delta y = 2$ cm, supponendo così che tutte le microlenti convergenti abbiano o una forma circolare o quadrata. Questa ipotesi non è tuttavia riduttiva in quanto è possibile considerare anche altre forme geometriche, quali quella rettangolare, modificando opportunamente le quantità Δx e Δy.

La procedura per la generazione dei valori necessari per il calcolo della posizione del centroide ricalca quanto visto in precedenza ma ora è estesa anche alla componente y. In particolar modo sono patita da due matrici 5×5 costituite da valori arbitrari rappresentanti rispettivamente le ascisse e le ordinate teoriche degli spot luminosi. Successivamente ho realizzato le corrispettive matrici contenenti le ascisse e le ordinate degli spot sperimentali inserendo in ciascun valore una componente Gaussiana con varianza $\sigma^2 = 0.0004$ cm^2. Infine ho realizzato le matrici contenenti ascisse e le ordinate dei centri delle lenti moltiplicando la distanza Δx e Δy per il numero della lente che si sta considerando. Il metodo di numerazione delle lenti è il medesimo del caso unidimensionale solo che ora viene iterato su ciascuna riga ed esteso anche alle ordinate. A partire da ciò ho calcolato gli spostamenti teorici, $\Delta n_{xn}^{\text{th}}$ e $\Delta n_{yn}^{\text{th}}$, e quelli sperimentali, $\Delta n_{xn}^{\text{exp}}$ e $\Delta n_{yn}^{\text{exp}}$, come semplici differenze in valore assoluto rispettivamente delle ascisse e delle ordinate delle posizioni degli spot teoriche e sperimentali da quelle dei centri delle microlenti.

Riporto in figura 6 una rappresentazione schematica esemplificativa della situazione.
I 25 quadrati indicano le lenti dell'array, i cerchi al centro di ognuno sono i punti di riferimento mentre i cerchietti scostati sono i punti in cui i raggi luminosi vengono focalizzati da ciascuna microlente. Ogni spostamento è indicato dalle rispettive frecce.

Di seguito riporto un esempio dei valori numerici in questione:

- Numero lenti ascisse: $\begin{pmatrix} -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \end{pmatrix}$ Numero lenti ordinate: $\begin{pmatrix} 2 & 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 & -1 \\ -2 & -2 & -2 & -2 & -2 \end{pmatrix}$

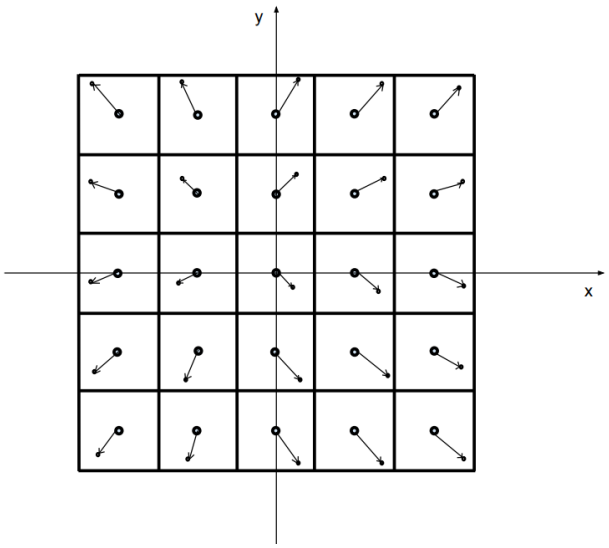


Figura 6: Schema array lenti.

- x^{th}: $\begin{pmatrix} -4.20 & -2.06 & 0.08 & 2.21 & 4.29 \\ -4.17 & -2.05 & 0.07 & 2.17 & 4.26 \\ -4.11 & -2.04 & 0.06 & 2.11 & 4.21 \\ -4.15 & -2.05 & 0.07 & 2.15 & 4.23 \\ -4.19 & -2.06 & 0.08 & 2.19 & 4.27 \end{pmatrix}$ y^{th}: $\begin{pmatrix} 4.29 & 4.25 & 4.19 & 4.21 & 4.26 \\ 2.21 & 2.16 & 2.11 & 2.13 & 2.18 \\ 0.08 & 0.07 & 0.06 & 0.07 & 0.08 \\ -2.16 & -2.13 & -2.08 & -2.09 & -2.11 \\ -4.27 & -4.21 & -4.16 & -4.19 & -4.24 \end{pmatrix}$
- x^{exp}: $\begin{pmatrix} -4.21 & -2.04 & 0.07 & 2.22 & 4.29 \\ -4.19 & -2.06 & 0.07 & 2.16 & 4.26 \\ -4.10 & -2.05 & 0.09 & 2.11 & 4.23 \\ -4.16 & -2.05 & 0.03 & 2.14 & 4.22 \\ -4.18 & -2.05 & 0.08 & 2.17 & 4.24 \end{pmatrix}$ y^{exp}: $\begin{pmatrix} 4.30 & 4.25 & 4.22 & 4.19 & 4.27 \\ 2.23 & 2.15 & 2.14 & 2.13 & 2.15 \\ 0.09 & 0.04 & 0.07 & 0.09 & 0.08 \\ -2.16 & -2.14 & -2.06 & -2.09 & -2.11 \\ -4.25 & -4.21 & -4.14 & -4.17 & -4.21 \end{pmatrix}$
- Ascisse centri lenti: $\begin{pmatrix} -4.00 & -2.00 & 0.00 & 2.00 & 4.00 \\ -4.00 & -2.00 & 0.00 & 2.00 & 4.00 \\ -4.00 & -2.00 & 0.00 & 2.00 & 4.00 \\ -4.00 & -2.00 & 0.00 & 2.00 & 4.00 \\ -4.00 & -2.00 & 0.00 & 2.00 & 4.00 \end{pmatrix}$ Ordinate centri lenti: $\begin{pmatrix} 4.00 & 4.00 & 4.00 & 4.00 & 4.00 \\ 2.00 & 2.00 & 2.00 & 2.00 & 2.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ -2.00 & -2.00 & -2.00 & -2.00 & -2.00 \\ -4.00 & -4.00 & -4.00 & -4.00 & -4.00 \end{pmatrix}$
- Δn_{xn}^{th}: $\begin{pmatrix} 0.20 & 0.06 & 0.08 & 0.21 & 0.29 \\ 0.17 & 0.05 & 0.07 & 0.17 & 0.26 \\ 0.11 & 0.04 & 0.06 & 0.11 & 0.21 \\ 0.15 & 0.05 & 0.07 & 0.15 & 0.23 \\ 0.19 & 0.06 & 0.08 & 0.19 & 0.27 \end{pmatrix}$ Δn_{yn}^{th}: $\begin{pmatrix} 0.29 & 0.25 & 0.19 & 0.21 & 0.26 \\ 0.21 & 0.16 & 0.11 & 0.13 & 0.18 \\ 0.08 & 0.07 & 0.06 & 0.07 & 0.08 \\ 0.16 & 0.13 & 0.08 & 0.09 & 0.11 \\ 0.27 & 0.21 & 0.16 & 0.19 & 0.24 \end{pmatrix}$

- Δn_{xn}^{exp}: $\begin{pmatrix} 0.21 & 0.04 & 0.07 & 0.22 & 0.29 \\ 0.19 & 0.06 & 0.07 & 0.16 & 0.26 \\ 0.10 & 0.05 & 0.09 & 0.11 & 0.23 \\ 0.16 & 0.05 & 0.03 & 0.14 & 0.22 \\ 0.18 & 0.05 & 0.08 & 0.17 & 0.24 \end{pmatrix}$ Δn_{yn}^{exp}: $\begin{pmatrix} 0.30 & 0.25 & 0.22 & 0.19 & 0.27 \\ 0.23 & 0.15 & 0.14 & 0.13 & 0.15 \\ 0.09 & 0.04 & 0.07 & 0.09 & 0.08 \\ 0.16 & 0.14 & 0.06 & 0.09 & 0.11 \\ 0.25 & 0.21 & 0.14 & 0.17 & 0.21 \end{pmatrix}$

Non essendo possibile una risoluzione analitica per la stima della posizione del centroide e del raggio di curvatura ho implementato ora solo il metodo numerico di Newton-Raphson. A differenza del caso unidimensionale le funzioni di cui bisogna trovare la radice sono tre e non più due e corrispondono alle derivate di χ^2 rispetto ai tre parametri ignoti. In particolar modo sono:

$$f_1(x_0, y_0, L) = \frac{\partial \chi^2}{\partial x_0} = \sum_n \frac{1}{\sigma_n^2}(n_x \Delta x - x_0) - \frac{\Delta n_n^{\exp}}{\sigma_n^2} \frac{L-l}{l} \frac{(n_x \Delta x - x_0)}{\left[(n_x \Delta x - x_0)^2 + (n_y \Delta y - y_0)^2\right]^{1/2}} \tag{28a}$$

$$f_2(x_0, y_0, L) = \frac{\partial \chi^2}{\partial y_0} = \sum_n \frac{1}{\sigma_n^2}(n_y \Delta y - y_0) - \frac{\Delta n_n^{\exp}}{\sigma_n^2} \frac{L-l}{l} \frac{(n_y \Delta y - y_0)}{\left[(n_x \Delta x - x_0)^2 + (n_x \Delta x - x_0)^2\right]^{1/2}} \tag{28b}$$

$$f_3(x_0, y_0, L) = \frac{\partial \chi^2}{\partial L} = \sum_n \frac{1}{\sigma_n^2} \frac{l}{L-l} \left[(n_x \Delta x - x_0)^2 + (n_y \Delta y - y_0)^2\right] + - \sum_n \frac{\Delta n_n^{\exp}}{\sigma_n^2} \left[(n_x \Delta x - x_0)^2 + (n_y \Delta y - y_0)^2\right]^{1/2} \tag{28c}$$

Le loro derivate le ho calcolate in modo numerico approssimato tramite rapporto incrementale con $h \to 0$, ovvero:

$$f_1'(x_0, y_0, L) = \frac{f(x_0 + h, y_0, L) - f(x_0, y_0, L)}{h}, \quad \text{con } h = 10^{-8} \tag{29a}$$

$$f_2'(x_0, y_0, L) = \frac{f(x_0, y_0 + h, L) - f(x_0, y_0, L)}{h}, \quad \text{con } h = 10^{-8} \tag{29b}$$

$$f_3'(x_0, y_0, L) = \frac{f(x_0, y_0, L + h) - f(x_0, y_0, L)}{h}, \quad \text{con } h = 10^{-8} \tag{29c}$$

Dapprima ho implementato un programma dove dati due parametri per noti ho stimato il terzo. In particolar modo, non potendo conoscere la posizione esatta del centroide e il valore del raggio di curvatura come nel caso unidimensionale mediante la trattazione analitica non è possibile controllare direttamente che i risultati ottenuti per via numerica coincidano con quanto atteso. Ho pertanto proceduto in questo modo: ho attribuito dei valori arbitrari a y_0 e L e ho trovato x_0 tramite l'algoritmo di Newton; successivamente ho implementato una seconda volta l'algoritmo di Newton per trovare y_0 noti ora L, che ha mantenuto lo

stesso valore della prima implementazione, e x_0, appena stimato. Infine ho fatto una terza implementazione per trovare L noti x_0 e y_0 dalle due implementazioni precedenti. Procedendo in questo modo ho potuto controllare l'accuratezza sulla stima dell'ordinata e del raggio di curvatura in quanto i rispettivi valori ottenuti devono essere pari ai valori arbitrari assegnati all'inizio delle tre implementazioni. Nel dettaglio i valori attribuiti inizialmente a y_0 e L sono: $y_0 = -0.2$ cm e $L = 100$ cm. I risultati ottenuti sono: $x_0 = -0.48$ cm, $y_0 = -0.23$ cm e $L = 97.05$ cm. Anche in questo caso, per la trattazione degli errori da cui sono affette queste stime rimando alla sezione successiva.

Si osserva dunque che la differenza tra il valore iniziale e quello stimato per le ordinate vale 0.03 cm mentre quella riguardante il raggio di curvatura è pari a 2.95 cm. Queste differenze sono inevitabilmente presenti in quanto, essendo il metodo di Newton un metodo approssimato e considerando anche l'approssimazione utilizzata per il calcolo delle derivate, l'implicazione inversa a quella iniziale, ossia trovare y_0 noti x_0 e L dopo che ho stimato x_0 a partire dagli altri due parametri, risente doppiamente del peso di queste approssimazioni.
Per la condizione di stop alle iterazioni all'interno dei metodi di Newton ho richiesto una tolleranza pari a `tol`$= 10^{-9}$ per tutti i parametri.

Ho successivamente implementato il metodo di Newton multidimensionale per riuscire a stimare contemporaneamente tutti e tre i parametri, senza dover supporre alcuni di essi noti. Anche in questo caso l'algoritmo iterativo viene derivato dall'espansione in serie di Taylor della funzione $\vec{f}(\vec{x})$. In generale si ha infatti che:

$$0 = \vec{f}\left(\vec{x}_{\text{sol}} =\right) = \vec{f}\left(\vec{x}_i + \delta\vec{x}\right) \approx \vec{f}\left(\vec{x}_i\right) + \mathbb{J}\left(\vec{x}_i\right)\delta\vec{x} \tag{30a}$$

$$\delta\vec{x} = -\mathbb{J}^{-1}\left(\vec{x}_i\right) f\left(\vec{x}_i\right) \tag{30b}$$

dove $\mathbb{J}\left(\vec{x}\right)$ è la matrice Jacobiana ossia quella contenente le derivate prime delle funzioni rispetto a tutti i parametri, vale a dire

$$\mathbb{J}(\vec{x}) = \begin{pmatrix} \frac{\partial f_1(\vec{x})}{\partial x_0} & \frac{\partial f_1(\vec{x})}{\partial y_0} & \frac{\partial f_1(\vec{x})}{\partial L} \\ \frac{\partial f_2(\vec{x})}{\partial x_0} & \frac{\partial f_2(\vec{x})}{\partial y_0} & \frac{\partial f_2(\vec{x})}{\partial L} \\ \frac{\partial f_3(\vec{x})}{\partial x_0} & \frac{\partial f_3(\vec{x})}{\partial y_0} & \frac{\partial f_3(\vec{x})}{\partial L} \end{pmatrix} \tag{31}$$

A questo punto, tramite l'algoritmo di eliminazione di Gauss, si calcola il vettore spostamento $\delta\vec{x}$ e si definisce un nuovo vettore delle stime dei parametri incogniti, $\vec{x}_{i+1} = \vec{x}_i + \delta\vec{x}$ e si ricomincia il ciclo calcolando funzioni e derivate con il nuovo array di parametri. La condizione di uscita da esso è data, al pari del caso precedente, da un valore di tolleranza `tol`.

Nel caso in questione, essendo tre i parametri da stimare, si ha un sistema di tre equazioni in tre incognite. Ho definito con $\vec{x}$ l'array di parametri:

$$\vec{x} = (x_0, y_0, L) \tag{32}$$

mentre con $\vec{f}(\vec{x})$ il vettore delle funzioni $\vec{f}: \mathbb{R}^3 \to \mathbb{R}^3$, di cui voglio trovare la radice, ossia le derivate di χ^2 rispetto ai parametri. Si ha quindi:

$$\vec{f}\,(\vec{x}) = (f_1\,(\vec{x})\,, f_2\,(\vec{x})\,, f_3\,(\vec{x})) = \left(\frac{\partial\chi^2}{\partial x_0}, \frac{\partial\chi^2}{\partial y_0}, \frac{\partial\chi^2}{\partial L}\right) \tag{33}$$

Nell'implementazione effettuata tutte le componenti della matrice Jacobiana le ho calcolate tramite rapporto incrementale per il quale ho scelto un valore dello scalare h pari a $h = 10^{-8}$.

Essendo difficile la convergenza al valore nullo di tutte e tre le funzioni contemporaneamente, al ciclo iterativo di Newton ho imposto un numero massimo di ripetizioni, fissato a 15000, oltre il quale il programma si ferma senza avere però trovato i valori da stimare. Nel caso in cui si verifichi la convergenza entro il numero di ripetizioni, il valore di tolleranza l'ho fissato a `tol`$= 10^{-7}$ e ho utilizzato un'unica condizione di stop. L'uscita dal ciclo non avviene infatti quando ogni componente del vettore delle funzioni è inferiore al valore di tolleranza, ma si verifica quando la radice quadrata del prodotto tra la trasposta della matrice delle funzioni e la matrice delle funzioni stessa è un valore inferiore a quello di tolleranza; in formule:

$$\sqrt{\left\|\vec{f}(\vec{x})\right\|^2} = \sqrt{\vec{f}\,(\vec{x})^T\,\vec{f}\,(\vec{x})} < \texttt{tol} \tag{34}$$

Il metodo per la generazione dei dati è il medesimo della prima implementazione mentre i valori inizialmente forniti li ho scelti in un intorno dei valori attesi per facilitare la convergenza del programma. In particolar modo sono partita da $L = 102.00$ cm, $x_0 = -0.45$ cm e $y_0 = -0.25$ cm.

Con i livelli di accuratezza imposti e il numero di ripetizioni richiesto il programma converge raramente tuttavia i valori ottenuti in uno specifico run convergente sono: $x_0 = -0.25$ cm, $y_0 = -0.31$ cm e $L = 97.76$ cm. Per il calcolo degli errori su queste stime rimando alla sezione successiva. Si osserva dunque come in questo specifico caso il valore delle ordinate si discosti da $y_0 = -0.2$ cm, richiesto nella prima implementazione, di 0.11 cm mentre i valori di L e x_0 siano minori rispetto a quanto trovato in precedenza.

4 Effetto del rumore elettronico e di quantizzazione nella determinazione della posizione del centroide e di conseguenza dell'asse ottico

Come visto in precedenza, le misurazioni effettuate con questi sensori, essendo essi degli strumenti digitali, sono soggette ad una componente di rumore gaussiano. Questa contiene al suo interno sia l'effetto del rumore dovuto all'elettronica dell'apparato sia quello legato alla quantizzazione. In particolar modo la prima componente vede la presenza di rumore termico dovuto alla camera digitale mentre la seconda è legata alla presenza di un ADC, ossia un analog to digital converter. Quest'ultimo ha una risoluzione finita, in genere pari a 10 bits, e può dunque registrare il segnale analogico come sequenza di tensioni alte e basse, quindi di 0 e 1, solo con un'accuratezza limitata che tuttavia risulta essere, in genere, sufficiente per l'utilizzo che si vuole fare del sensore. Se si considera, a differenza di quanto visto fin ora, anche la dinamica dell'arrivo di un fronte d'onda e come questo si evolve nel tempo, la presenza dell'ADC comporta una seconda quantizzazione, quella in tempo. Esso ha infatti un tempo di campionamento finito e in generale più questo è breve più la misura risulta accurata tuttavia non sarà mai perfettamente nullo.

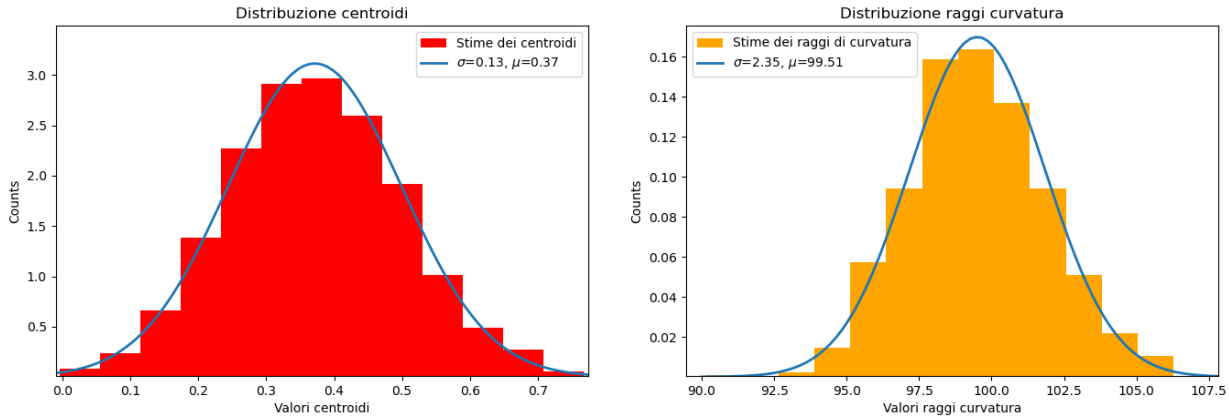
E' possibile inoltre citare una terza fonte di rumore che è quella legata al carattere quantistico della luce. Essendo essa quantizzata, i singoli fotoni che impattano sulla superficie del sensore determinano un flusso che non è continuo ma "granulare" andando così a definire quello che è lo shot noise. Tanto maggiore è il numero di fotoni per unità di superficie tanto minore sarà l'effetto di questo rumore che diventa dunque rilevante solo in caso di bassissime intensità luminose.

Lo scopo di questa sezione è trattare come ho determinato l'errore sulla stima dei parametri incogniti e fare delle considerazioni sull'influenza del rumore gaussiano nel determinare la posizione del centroide.

4.1 Caso unidimensionale

Per il caso in questione, essendo possibile una risoluzione analitica, ho implementato una simulazione Montecarlo iterando $m = 1000$ volte il processo di stima tramite il metodo dei minimi quadrati ottenendo così 1000 stime analitiche per la posizione del centroide e del raggio di curvatura. Ho istogrammato i risultati ottenuti per osservarne la distribuzione.

Nelle figure 7a e 7b riporto gli istogrammi delle distribuzioni normalizzate delle stime con i rispettivi fit gaussiani.



(a) Distribuzione stime delle posizioni dei centroidi.

(b) Distribuzione stime dei raggi di curvatura.

Figura 7: Distribuzioni parametri.

Si osserva come la distribuzione dei centroidi sia una Gaussiana come atteso. In particolar modo il valore di aspettazione è $\mu = 0.37$ cm mentre la deviazione standard associata è $\sigma = 0.13$ cm. Per quanto concerne la distribuzione dei raggi di curvatura si osserva invece che l'andamento non è perfettamente gaussiano ma presenta una leggera coda più allungata verso stime più elevate. Il valore di aspettazione è infatti $\mu = 99.51$ cm ma la deviazione standard, essendo pari a $\sigma = 2.35$ cm, è un valore piuttosto elevato ma comunque non oltre le 5 σ necessarie a rifiutare l'ipotesi di andamento gaussiano.

Ho parimenti realizzato, come riportato in figura 8, un grafico a dispersione che riporta su un unico piano entrambe le stime accoppiate.

Per l'effettiva stima dell'errore ho optato per utilizzare il metodo grafico andando a rappresentare la funzione iniziale χ^2 in funzione delle m stime realizzate. Ciò l'ho implementato separatamente per i due parametri; dapprima ho rappresentato χ^2 in funzione delle m stime di x_0 supponendo noto L, con $L = 103.89$ cm e successivamente ho rappresentato χ^2 in funzione delle m stime di L dato per noto $x_0 = 0.09$ cm. Per ciascuna delle due situazioni si ho realizzato un fit quadratico dei punti rappresentati ottenendo così i parametri a,b e c delle parabole. Successivamente ho calcolato il minimo di esse tramite la formula del vertice di parabole con asse parallelo all'asse y: $V = \left(-\frac{b}{2a}, -\frac{\Delta}{4a}\right)$, dove $\Delta = b^2 - 4ac$. Rappresentando l'asse di simmetria della parabola di equazione $ax : x = -\frac{b}{2a}$ e la retta r di equazione $y = -\frac{\Delta}{4a} + 1$ ottengo la stima grafica della deviazione standard σ relativa a ciascun parametro. In particolar modo, essa risulta pari alla distanza tra il punto di intersezione tra la retta r e il fit a parabola e quello dovuto all'intersezione della medesima retta con l'asse ax. In formule:

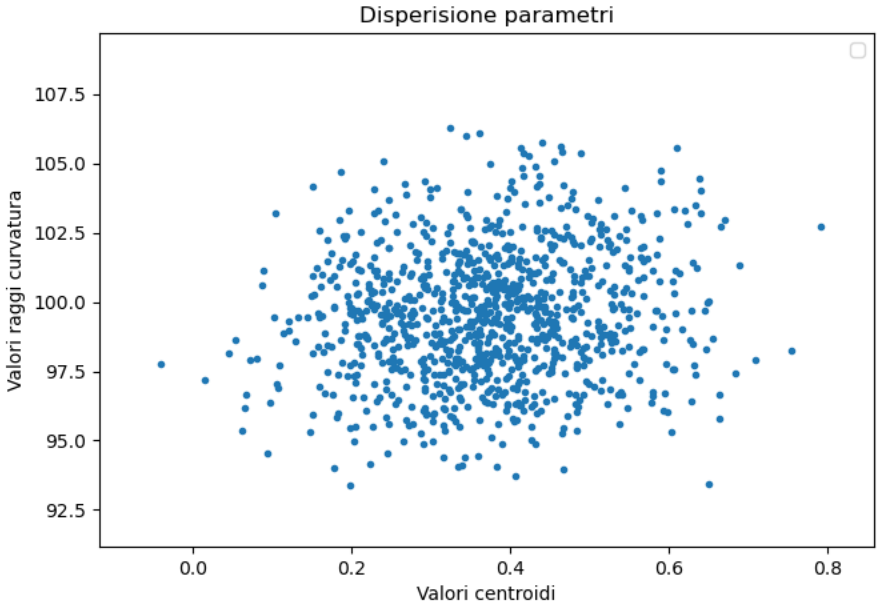


Figura 8: Scatter-plot dei parametri stimati.

$$\sigma = \left| -\frac{b}{2a} + \frac{4a^2b - \sqrt{16a^4b^2 - 4\left(16a^5c + 4a^3b - 16a^4c\right)}}{8a^3} \right| \tag{35}$$

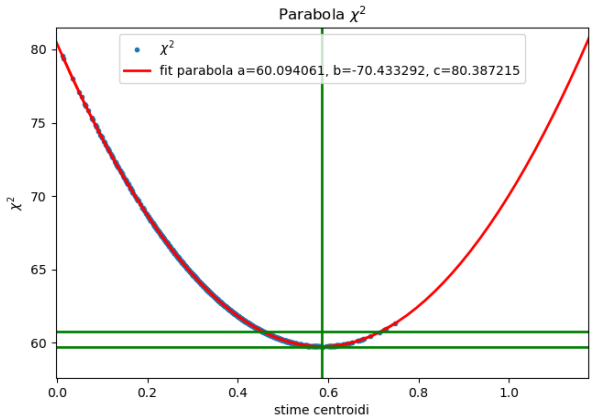
Numericamente la deviazione standard sulla stima del centroide risulta pari a $\sigma = 0.12$ cm mentre quella relativa alla stima del raggio di curvatura è $\sigma = 2.02$ cm.

Nelle figure 9a e 9b riporto i grafici di cui sopra.
Si può dunque affermare con certezza che i valori ottenuti mediante la stima numerica sono compatibili con quelli analitici.

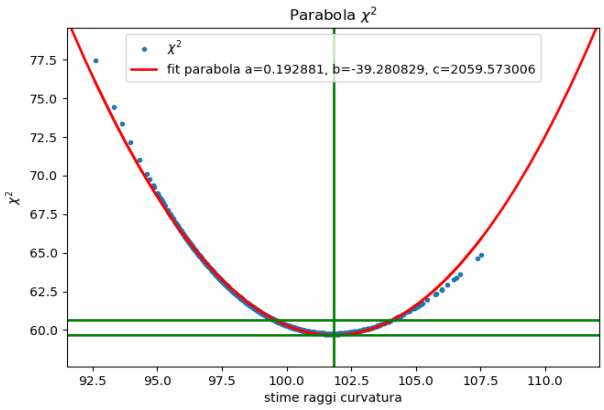
Per il caso unidimensionale ho inoltre realizzato una seconda coppia di grafici riportanti la dispersione delle stime analitiche in funzione della varianza utilizzata per generare i dati sperimentali.

Per realizzare ciò ho scelto sei valori arbitrari di varianza, riportati in tabella 2, e per ciascuno di essi ho stimato separatamente i valori dell'ascissa del centroide e del raggio di curvatura per dieci volte. Questi grafici permettono di osservare come le stime dei parametri variano all'aumentare o al diminuire del rumore da cui sono affette.

Nelle figure 10a e 10b riporto gli scatter-plot delle stime in funzione delle varianze con cui sono state ottenute.



(a) Stima grafica della deviazione standard per il centroide.

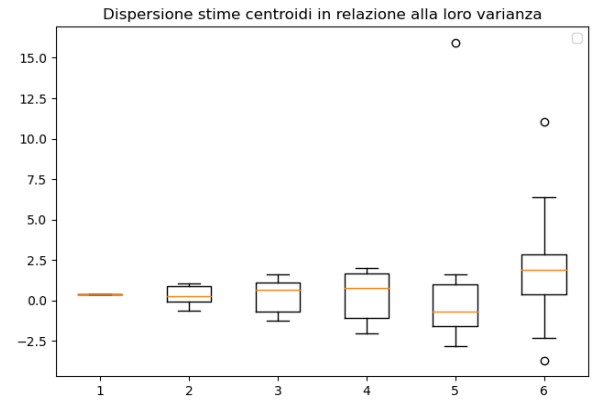


(b) Stima grafica della deviazione standard per il raggio di curvatura.

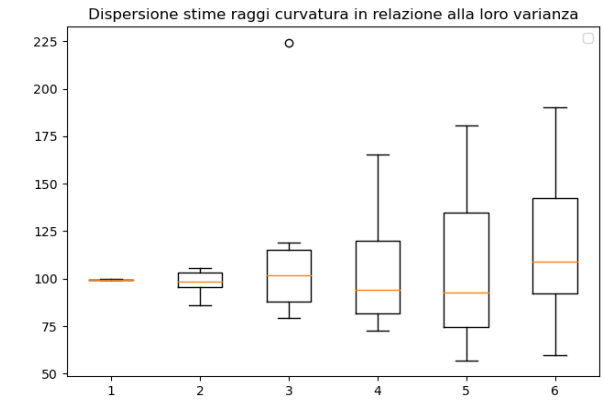
Figura 9: Stima grafica deviazioni standard.

σ_1^2	$1 \cdot 10^{-6}$
σ_2^2	$1 \cdot 10^{-2}$
σ_3^2	$2.5 \cdot 10^{-2}$
σ_4^2	$5 \cdot 10^{-2}$
σ_5^2	$1 \cdot 10^{-1}$
σ_6^2	$1.5 \cdot 10^{-1}$

Tabella 2: Valori varianza scelti



(a) Dispersione stime centroidi.



(b) Dispersione stime raggi di curvatura.

Figura 10: Dispersione parametri in funzione della varianza.

Si osserva che, esattamente come atteso, le stime ripetute ad una determinata varianza presentano una maggiore dispersione a mano a mano che il valore della varianza con cui sono state calcolate aumenta. In entrambi i casi si nota infatti come a varianza pressoché nulla, ossia tendendo al caso ideale in cui i valori teorici corrispondono a quelli sperimentali e non vi è alcun tipo di rumore, le stime sono concentrate in un intorno molto piccolo del valore che ci aspettiamo di trovare in assenza di rumore, mentre a mano a mano che la varianza aumenta si nota come le stime inizino ad allontanarsi sempre di più da tale valore. Ciò è evidente dal grafico confrontando la dispersione delle stime ottenute con $\sigma^2 = 10^{-6}$ cm^2 e quelle ottenute con $\sigma^2 = 0.15$ cm^2.

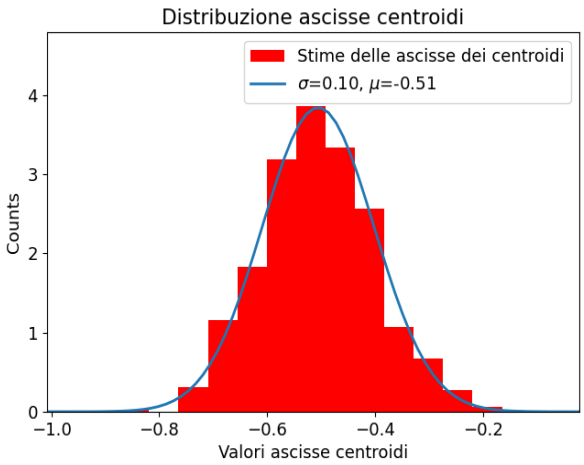
Lanciando svariate volte il programma si può tuttavia osservare come in alcuni casi quanto detto sembrerebbe essere non vero in quanto è possibile che, data la dispersione delle stime ad una determinata varianza, quelle con varianza minore abbiano una dispersione maggiore. Questo è legato all'utilizzo di una statistica piuttosto bassa nella ripetizione delle stime per ciascuna varianza e alla scelta dei valori sperimentali come numeri random di gaussiane centrate nei rispettivi valori teorici.

4.2 Caso bidimensionale

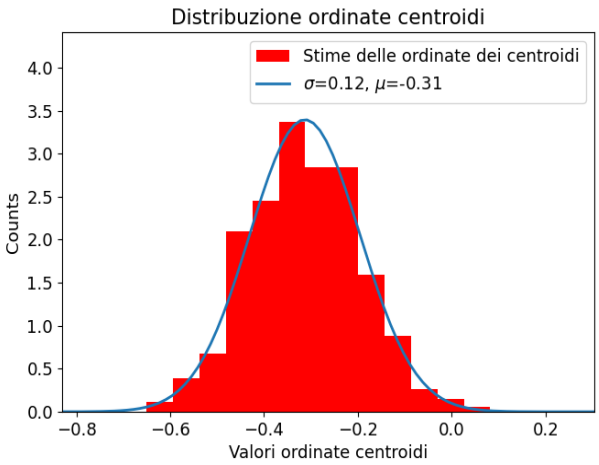
Anche per il caso bidimensionale ho implementato il metodo Montecarlo e ho studiato la distribuzione dei parametri stimati. In particolar modo, partendo dai tre metodi di Newton-Raphson concatenati descritti nella sezione precedente, ho realizzato tre cicli di ripetizione separati, uno per ciascun parametro stimato. Nello specifico, dapprima ho realizzato le ripetizioni delle stime delle ascisse inserendo in un ciclo iterativo l'algoritmo di Newton-Raphson per x_0, noti y_0 e L arbitrari, ottenendo così m stime di x_0. Ho poi istogrammato tali stime facendone un fit gaussiano e ho ottenuto un valore di aspettazione μ_{x_0}. Tale valore l'ho usato come valore noto insieme ad L per la stima di y_0. Ripetendo l'algoritmo di Newton per y_0 m volte ho ottenuto m stime delle ordinate che a loro volta ho istogrammato ricavando μ_{y_0}. Analogamente a quanto visto per y_0, questo valore l'ho utilizzato come valore noto insieme a μ_{x_0} per stimare L. Ho infine ripetuto anche quest'ultima stima m volte ottenendo così un terzo istogramma. In tutti i casi il numero di ripetizioni è pari a $m = 600$, un numero piuttosto basso ma che rappresenta il giusto compromesso tra risultato desiderato e velocità di esecuzione del programma.

Nelle figure 11a e 11b riporto le distribuzioni delle ascisse e delle ordinate del centroide con i rispettivi fit.

Nonostante la limitata statistica dal fit gaussiano si evidenzia che sono centrate in $\mu_x = -0.51$ cm e $\mu_y = -0.31$ cm con rispettive deviazioni standard $\sigma_x = 0.10$ cm e $\sigma_y = 0.12$ cm. Si osserva che il valore di aspettazione delle ordinate risulta spostato a sinistra rispetto quanto atteso di 0.11 cm ed inoltre i valori delle deviazioni standard sono ora molto più elevati rispetto a quelli trovati nel caso unidimensionale a causa di una maggiore dispersione delle stime. Ciò potrebbe essere imputabile sia all'utilizzo di una ridotta statistica sia alla chiamata dei numeri random gaussiani, centrati nei corrispondenti valori teorici, per determinare le coordinate sperimentali degli spot luminosi.



(a) Distribuzione stime ascisse centroidi.



(b) Distribuzione stime ordinate centroidi.

Figura 11: Distribuzione coordinate centroidi.

In figura 12 riporto inoltre la distribuzione delle stime dei raggi di curvatura e il corrispettivo fit.

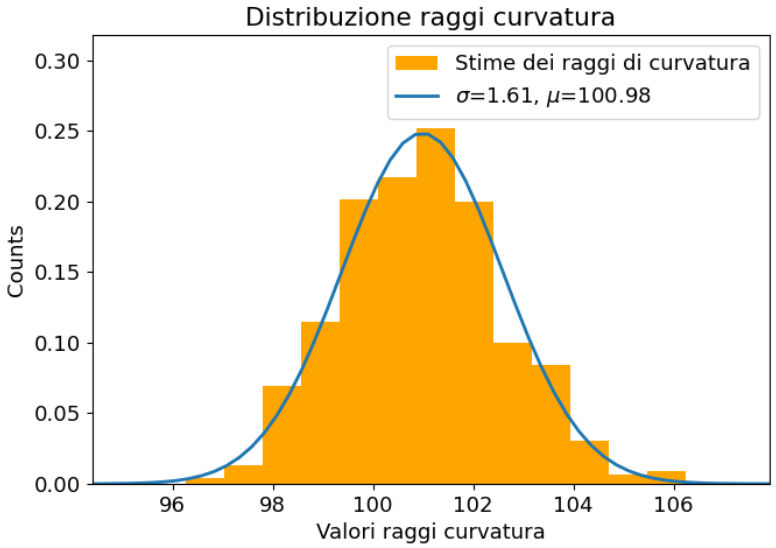


Figura 12: Distribuzione dei raggi di curvatura.

Anche in questo caso, analogamente a quanto visto per la situazione unidimensionale, la distribuzione non è perfettamente Gaussiana ma presenta una coda più allungata verso i valori più elevati. Ciò è evidenziato dal valore della deviazione standard che è molto elevato anche se non prossimo al valore utilizzato per rigettare l'ipotesi essendo pari a $\sigma = 1.61$ cm.

Si nota come il valore di aspettazione sia pari a $\mu = 100.98$ cm e dunque risulti spostato a sinistra di 0.98 cm rispetto a quanto atteso. Questo può essere spiegato grazie alle stesse motivazioni addotte per lo spostamento della stima dell'ordinata del centroide.

Ho realizzato inoltre un confronto tra la distanza media rispetto all'origine degli m centroidi ottenuti fissando y_0 e stimando m volte x_0 e quella ottenuta dalla posizione del centroide $(x_0 = \mu_{x_0}, y_0)$. Nel caso riportato nell'istogramma in figura 11a, la prima risulta essere $d = \sum_s \frac{1}{m}\sqrt{x_{0_s}^2 + y_0^2} = 0.55$ cm mentre la seconda è $d = \sqrt{\mu_{x_0}^2 + y_0^2} = 0.55$ cm. La stessa cosa l'ho fatta per gli m centroidi stimati fissando $x_0 = \mu_{x_0}$ cm e stimando y_0 e la posizione del centroide $(x_0 = \mu_{x_0}, y_0 = \mu_{y_0})$. Per il caso riportato in figura 11b, la prima risulta $d = \frac{1}{m}\sum_s \sqrt{\mu_{x_0}^2 + y_{0_s}^2} = 0.60$ cm mentre la seconda è $d = \sqrt{\mu_{x_0}^2 + \mu_{y_0}^2} = 0.60$ cm. I valori, arrotondati alla seconda cifra decimale risultano così in entrambi i casi uguali.

Riporto in figura 13 un grafico a dispersione che simula la posizione degli m centroidi stimati rispetto a piano costituito dall'array di microlenti.

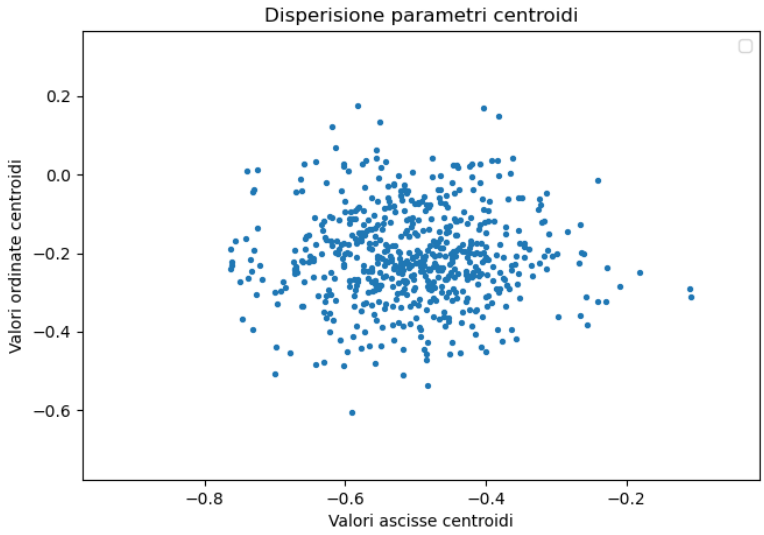


Figura 13: Scatter-plot ascisse e ordinate centroidi.

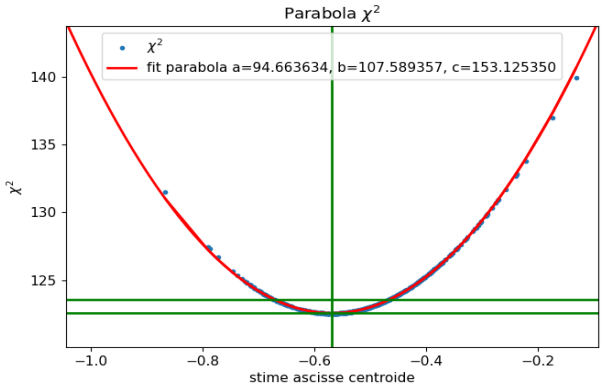
Per ottenere questo grafico ho implementato un metodo Montecarlo simile a quello realizzato per ottenere le distribuzioni dei parametri ma che al posto di utilizzare μ_{x_0} come unico valore noto per stimare m volte y_0, considera ogni valore stimato delle ascisse e ne stima uno per le ordinate riuscendo così a concatenare le due variabili. Anche in questo caso il numero di ripetizioni è stato fissato a $m = 600$.

Si osserva come la distribuzione sia centrata in $x_0 = -0.51$ cm e in $y_0 = -0.31$ cm come atteso. Lo scatter plot mette tuttavia in evidenza anche la posizione leggermente scostata

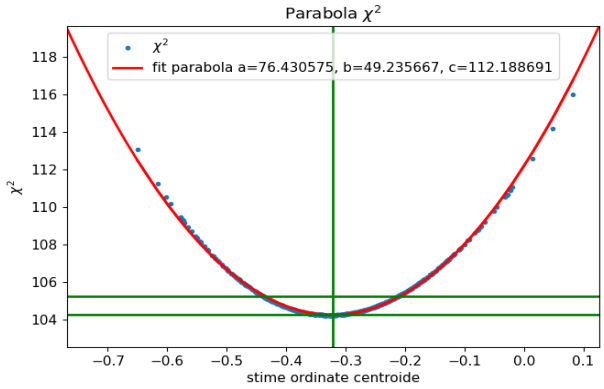
di alcune stime rispetto a tutte le altre che si dispongono su un'ellisse di eccentricità $e \to 0$. Questi punti spostati sono il motivo per cui le deviazioni standard, calcolate tramite fit gaussiano, nel caso bidimensionale siano maggiori rispetto a quelle unidimensionali.

Per la stima degli errori sui parametri, anche per la situazione bidimensionale, ho utilizzato il metodo grafico in quanto risulta piuttosto semplice e pratico nel caso in cui i parametri siano stimati uno alla volta come implementato inizialmente. In particolar modo, per la stima dell'errore su x_0 ho graficato i valori di χ^2 in funzione delle m stime di x_0 ottenendo una parabola. Dal fit quadratico ho ottenuti i parametri necessari a calcolarne il punto di minimo e considerando l'intersezione della retta passante per $y_{min} + 1$ con la parabola e il suo asse di simmetria ho ottenuta la stima numerica di σ_{x_0}. Il medesimo procedimento l'ho adottato anche per stimare le deviazioni standard per y_0 e L.

Nelle figure 14a e 14b riporto i grafici ottenuti con i rispettivi fit parabolici.



(a) Stima grafica della deviazione standard per l'ascissa del centroide.



(b) Stima grafica della deviazione standard per l'ordinata del centroide.

Figura 14: Stima grafica delle deviazioni standard coordinate centroide.

La deviazione standard per l'ascissa del centroide risulta pari a $\sigma_{x_0} = 0.10$ cm. Tale valore è giustificato dal fatto che il fit parabolico viene realizzato tenendo in considerazione tutte le ascisse stimate, incluse quelle lievemente spostate rispetto al valore atteso. Essendo dunque un'interpolazione, i valori scostati rispetto a quanto atteso tendono ad aumentare il valore del coefficiente a della parabola, allargandola. Per quanto riguarda le ordinate si osserva come la deviazione standard stimata graficamente nel caso riportato in figura 14b sia pari a $\sigma_{y_0} = 0.11$ cm. Anche in questo caso il motivo del valore piuttosto elevato è legato all'interpolazione fatta su tutti i valori stimati. In entrambi i casi l'utilizzo di una ridotta statistica influisce negativamente sul risultato.

Riporto infine, in figura 15, la stima grafica della deviazione standard per i raggi di curvatura.

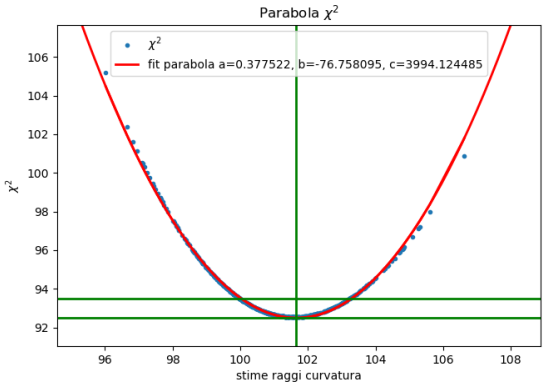


Figura 15: Stima grafica della deviazione standard per il raggio di curvatura.

Il fit parabolico e la conseguente stima della deviazione standard risentono della coda allungata verso valori più elevati dei raggi di curvatura. Ne consegue una leggera sovrastima dell'errore che risulta essere pari a $\sigma = 1.5$ cm.

Anche in questo caso si può affermare che le stime effettuate per le ordinate e i raggi di curvatura sono sensate in quanto considerando le rispettive deviazioni standard il valore atteso è contenuto all'interno dell'intervallo di incertezza calcolato.

5 Conclusioni

Lo scopo di questa tesi è fornire una breve analisi sul funzionamento dei sensori di Shack e Hartmann sia dal punto di vista qualitativo che quantitativo. In particolare ne ho descritto la struttura menzionando i vari componenti da cui sono costituiti e mettendoli in relazione alla funzione che ciascuno di essi svolge all'interno del sensore. Successivamente ho analizzato ciò che accade quando un generico fronte d'onda colpisce il sensore e infine ho simulato la stima dell'asse ottico con metodi diversi sia per la situazione semplificata unidimensionale che per quella bidimensionale. Ho osservato che in una dimensione i risultati ottenuti tramite il metodo analitico e quello numerico per la posizione del centroide e per il raggio di curvatura sono i medesimi mentre per il caso bidimensionale le stime ottenute in modo numerico sono compatibili entro una deviazione standard con i valori attesi.

Un possibile approfondimento della trattazione potrebbe essere legato allo studio dei polinomi di Zernike, una sequenza polinomiale molto utilizzata in ottica e alla base degli algoritmi per la ricostruzione del fronte d'onda tramite SHWF. In secondo luogo, potrebbe essere affrontato dal punto di vista tecnico il funzionamento e le caratteristiche delle principali tipologie di camere digitali utilizzate all'interno dei sensori, camere CCD o CMOS, mettendone in evidenza i pro e i contro di ciascuna.

I sensori di Shack e Hartmann, progettati e realizzati per la prima volta negli anni '60 in risposta all'esigenza pratica di ottenere immagini spaziali di migliore qualità, hanno avuto nel corso degli anni una rapida evoluzione non tanto dal punto di vista strutturale quanto per il loro impiego. Attualmente sono infatti ampiamente utilizzati in svariate realtà, da quella industriale alla ricerca applicata e di base, passando per l'ambito militare e quello medico. Questo è dovuto alla semplicità e versatilità dello strumento oltre che al ridotto costo e alla facilità con cui sono reperibili sul mercato. In futuro il loro utilizzo sarà sempre maggiore e potrebbero giocare un ruolo importante per il miglioramento di tecnologie esistenti e nella realizzazione di nuove tipologie di misurazioni fino ad ora mai effettuate.

6 Ringraziamenti

Vorrei in primis ringraziare il professore Edoardo Milotti per la disponibilità, la pazienza e la gentilezza che ha sempre dimostrato, non solo durante la stesura della tesi ma anche durante i corsi da lui tenuti. Lo ringrazio inoltre per l'enorme competenza e passione contagiosa di cui è portatore e per il grande numero di spunti offerti in questo percorso.

Ringrazio poi i miei genitori, Claudio e Gabriella, per aver reso possibile tutto ciò e in generale per tutto quello che fanno sempre per me.

Infine ringrazio gli amici, quelli di vecchia e nuova data, che mi sono stati accanto in questo percorso. Tra questi un grazie particolarmente sentito va a Iacopo, Elena, Gabriele e Chiara che hanno reso questi anni indimenticabili.

Riferimenti bibliografici

[1] M. Mansuripur, The Shack-Hartmann wavefront sensor, in Optics & Photonics News, April 1999.

[2] B. C. Platt, R. Shack, History and Principles of Shack-Hartmann Wavefront Sensing, in Journal of Refractive Surgery, Vol. 17, September/October 2001.

[3] B. E. A. Saleh, M. C. Teich, Ray Optics, in Fundamentals of Photonics, pp. 26-27, 3rd ed. 2019.

[4] da Thorlabs.com, Newton, New Jersey, Shack-Hartmann Wavefront Sensors.

Appendice: codici Python

Di seguito si riportano i codici Python realizzati per le simulazioni riportate alle sezioni 3 e 4.
Codice per la stima analitica e numerica dei parametri x_0 e L nel caso 1D.

```
'''PROGRAMMA MINIMIZZAZIONE CON METODO NEWTON-RAHPSON'''
'''PROGRAMMA DI MINIMIZZAZIONE ANALITICO'''

import matplotlib.pyplot as plt
import numpy as np
from math import sqrt

#STIMA DEL CENTROIDE E RAGGIO DI CURVATURA IN MODO ANALITICO

n=9
var=0.0004
dx=2
df=5
s1=0
s2=0
s3=0
s4=0
s5=0

ll=[]
lxt=np.array([-8.52,-6.32,-4.17,-2.07,0.01,2.06,4.11,6.26,8.46])
lxs=np.empty(n)
lxl=np.empty(n)
dnt=np.empty(n)
dne=np.empty(n)
dnl=np.empty(n)

lxs=np.random.normal(lxt[:],sqrt(var),size=None)

for j in range(-int(n/2),int(n/2)+1):
 di=j*dx
 ll.append(di)
lxl=np.asarray(ll)

dnt=(lxt-lxl)
dne=(lxs-lxl)
dnl=(lxl)

print(f'spostamento teorico spot luminoso centro lente {dnt}')
print(f'spostamento sperimentale spot luminoso centro lente {dne}')

for j in range(-int(n/2),int(n/2)+1):
 s1+=1/var
for j in range(0,n):
 s2+=dne[j]/var
 s3+=dnl[j]/var
```

```
 s4+=dnl[j]*dne[j]/var
 s5+=dnl[j]**2/var

xc=(s3*s4-s2*s5)/(3*s3*s2+s1*s4)
L=df*(s3+s2-s1*((s3*s4-s2*s5)/(3*s3*s2+s1*s4)))/s2

print(f'centroide analitico {xc}')
print(f'raggio di curvatura analitico {L}')

L=99.49
xc=0.38

def funzionex(x0,L):
    fx=0
    for i in range(0,n):
     fx+=(-2*(df/(L-df))**2*((dnl[i]-x0)-(dne[i]*(L-df)/df)))/var
    return fx

def funzionel(l,xc):
    fl=0
    for i in range(0,n):
        fl+=(-2*((df/(l-df))*(dnl[i]-xc)-dne[i])*(df*(dnl[i]-xc)/(l-df)**2))/var
    return fl

def derivatax(x0,fx,L):
    h=1e-6
    derivx=(fx(x0+h,L)-fx(x0,L))/h
    return derivx

def derivatal(l,fl,xc):
    h=1e-6
    derivl=(fl(l+h,xc)-fl(l,xc))/h
    return derivl

def newtonx(x0,fx,derivx):
    tol=1e-6
    xn=x0
    funzrefx=fx(xn,L)
    while abs(funzrefx)>tol:
        xn1=xn-(fx(xn,L)/derivx(xn,fx,L))
        xn=xn1
        funzrefx=fx(xn1,L)
    return xn1

def newtonl(l,fl,derivl):
    tol=1e-6
    ln=l
    funzrefl=fl(ln,xc)
    while abs(funzrefl)>tol:
        ln1=ln-(fl(ln,xc)/derivl(ln,fl,xc))
        ln=ln1
        funzrefl=fl(ln1,xc)
    return ln1

```

```
#STIMA DEL CENTROIDE E DEL RAGGIO DI CURVATURA CON METODO NEWTON

x0=-0.1
l=90

ascissa=newtonx(x0,funzionex,derivatax)
raggio=newtonl(l,funzionel,derivatal)

print(f'centroide Newton {ascissa}')
print(f'raggio Newton {raggio}')

dif=abs(xc-ascissa)
print(f'differenza centroide {dif}')

dif=abs(L-raggio)
print(f'differenza raggio {dif}')
```

Codice per la realizzazione della distribuzione Gaussiana delle differenze $\Delta n^{\text{th}} - \Delta n^{\text{exp}}$ nel caso 1D.

```
'''PROGRAMMA PER L'ISTOGRAMMA DELLE DIFFERENZE'''

import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import norm
from math import sqrt

n=21
var=0.0004
dx=2
df=5
k=500

ll=[]
lista=[]
lxt=np.array([-20.90,-18.72,-16.56,-14.42,-12.30,-10.20,-8.12,-6.08,
-4.04,-2.02,0.01,2.02,4.05,6.10,8.17,10.26,12.37,14.50,16.65,18.82,21.00])
lxs=np.empty(n)
lxl=np.empty(n)
dnt=np.empty(n)
dne=np.empty(n)
dnl=np.empty(n)
diff=np.empty((k,n))

for j in range(-int(n/2),int(n/2)+1):
 di=j*dx
 ll.append(di)
lxl=np.asarray(ll)

dnt=abs(lxt-lxl)
dnl=abs(lxl)

for i in range(0,k):
```

```
 lxs=np.random.normal(lxt[:],sqrt(var),size=None)
 dne=abs(lxs-lxl)
 diff[i,:]=dne-dnt

print(np.shape(diff))

for i in range(diff.shape[0]):
 for j in range(diff.shape[1]):
  lista.append(diff[i,j])

#PLOT DISTRIBUZIONE GAUSSIANA DIFFERENZE VALORI TEORICI VALORI SPERIMENTALI

plt.rcParams['figure.figsize']=[7.5,5]
plt.rcParams['font.size']=15
plt.rcParams['lines.linewidth']=2
plt.hist(lista,histtype='bar',bins=13,color='green',label=f'$\Delta n_n^t-\Delta
    n_n^e$',density=True)
(mu,sigma)=norm.fit(diff)
x=np.linspace(-0.15,0.15,150)
y=norm.pdf(x,mu,sigma)
plt.plot(x,y,label=f'$\sigma$={sigma:.2f}, $\mu$={mu:.2f}')
plt.xlabel('Valori differenze')
plt.ylabel('Counts')
plt.xlim(-0.15,0.15)
plt.title('Distribuzione differenze tra valori teorici e sperimentali')
plt.legend()
plt.show()
plt.savefig('figura.png')
```

Codice per il calcolo dei valori veri del centroide e del raggio di curvatura nel caso 1D.

```
'''PROGRAMMA CALCOLO VALORI VERI UNIDIMENSIONALE'''

import matplotlib.pyplot as plt
import numpy as np
from math import sqrt

n=9
var=0.000000000001
dx=2
df=5
s1=0
s2=0
s3=0
s4=0
s5=0

ll=[]
lxt=np.array([-8.52,-6.32,-4.17,-2.07,0.01,2.06,4.11,6.26,8.46])
lxs=np.empty(n)
lxl=np.empty(n)
dnt=np.empty(n)
dne=np.empty(n)
```

```
dnl=np.empty(n)

lxs=np.random.normal(lxt[:],sqrt(var),size=None)

for j in range(-int(n/2),int(n/2)+1):
 di=j*dx
 ll.append(di)
lxl=np.asarray(ll)

dnt=(lxt-lxl)
dne=(lxs-lxl)
dnl=(lxl)

for j in range(-int(n/2),int(n/2)+1):
 s1+=1/var
for j in range(0,n):
 s2+=dne[j]/var
 s3+=dnl[j]/var
 s4+=dnl[j]*dne[j]/var
 s5+=dnl[j]**2/var

xc=(s3*s4-s2*s5)/(3*s3*s2+s1*s4)
L=df*(s3+s2-s1*((s3*s4-s2*s5)/(3*s3*s2+s1*s4)))/s2

print(f'centroide vero {xc}')
print(f'raggio di curvatura vero {L}')
```

Codice per la realizzazione del metodo Montecarlo, distribuzione dei parametri e stima delle loro deviazioni standard nel caso 1D.

```
'''CODICE PER METODO MONTECARLO STIME ANALITICHE CON VARIANZA COSTANTE'''

import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import norm
from scipy.optimize import curve_fit
from math import sqrt

def fun(x,a,b,c):
 return x**2*a+x*b+c

m=1000
n=9
var=0.0004
dx=2
df=5

xc=0.38
L=99.49

af=[]
ac=[]
ll=[]
```

```
lxt=np.array([-8.52,-6.32,-4.17,-2.07,0.01,2.06,4.11,6.26,8.46])
lxs=np.empty(n)
lxl=np.empty(n)
dnt=np.empty(n)
dne=np.empty(n)
dnl=np.empty(n)
axc=np.empty(m)
al=np.empty(m)
chi1=np.empty(m)
chi2=np.empty(m)

for j in range(-int(n/2),int(n/2)+1):
 di=j*dx
 ll.append(di)
lxl=np.asarray(ll)

dnt=(lxt-lxl)
dnl=(lxl)

for i in range(0,m):
    lxs=np.random.normal(lxt[:],sqrt(var),size=None)
    dne=(lxs-lxl)
    s1=0
    s2=0
    s3=0
    s4=0
    s5=0
    for j in range(0,n):
     s1+=1/var
     s2+=dne[j]/var
     s3+=dnl[j]/var
     s4+=dnl[j]*dne[j]/var
     s5+=dnl[j]**2/var

    xc=(s3*s4-s2*s5)/(3*s3*s2+s1*s4)
    L=df*(s3+s2-s1*((s3*s4-s2*s5)/(3*s3*s2+s1*s4)))/s2
    axc[i]=xc
    al[i]=L

for i in range(0,m):
    f1=0
    f2=0
    for j in range(0,n):
        f1+=(((df/(L-df))*(dnl[j]-axc[i]))-dne[j])**2/var
        f2+=(((df/(al[i]-df))*(dnl[j]-xc))-dne[j])**2/var
    af.append(f1)
    ac.append(f2)
chi1=np.asarray(af)
chi2=np.asarray(ac)

som1=0
som2=0
som3=0
alm=np.mean(al)
```

```
axcm=np.mean(axc)
for i in range(0,m):
 som1+=(axc[i]-axcm)**2
 som2+=(al[i]-alm)**2
 som3+=(al[i]-alm)*(axc[i]-axcm)
r=som3/sqrt(som1*som2)
print(r)


#PLOT DISTRIBUZIONE CENTROIDI ANALITICI CON FIT

plt.rcParams['figure.figsize']=[7.5,5]
plt.rcParams['font.size']=10
plt.rcParams['lines.linewidth']=2
plt.hist(axc,bins=13,histtype='bar',color='red',label=f'Stime dei centroidi',
    density=True)
(mu,sigma)=norm.fit(axc)
x=np.linspace(-0.2,0.9,150)
plt.ylim(0,10)
y=norm.pdf(x,mu,sigma)
plt.plot(x,y,label=f'$\sigma$={sigma:.2f}, $\mu$={mu:.2f}')
plt.xlabel('Valori centroidi')
plt.ylabel('Counts')
plt.xlim(-0.0,0.8)
plt.title('Distribuzione centroidi')
plt.legend()
plt.show()
plt.savefig('figura.png')

#PLOT DISTRIBUZIONE RAGGI CURVATURA ANALITICI CON FIT

plt.rcParams['figure.figsize']=[7.5,5]
plt.rcParams['font.size']=10
plt.rcParams['lines.linewidth']=2
plt.hist(al,bins=13,histtype='bar',color='orange',label=f'Stime dei raggi di
    curvatura',density=True)
(mu,sigma)=norm.fit(al)
x=np.linspace(90,120,150)
y=norm.pdf(x,mu,sigma)
plt.plot(x,y,label=f'$\sigma$={sigma:.2f}, $\mu$={mu:.2f}')
plt.xlabel('Valori raggi curvatura')
plt.ylabel('Counts')
plt.xlim(90,120)
plt.title('Distribuzione raggi curvatura')
plt.legend()
plt.show()
plt.savefig('figura.png')

#SCATTER PLOT PARAMETRI STIMATI

plt.rcParams['figure.figsize']=[7.5,5]
plt.rcParams['font.size']=10
plt.rcParams['lines.linewidth']=2
plt.scatter(axc,al,marker='.')
```

```
plt.xlim(-0.3,0.9)
plt.ylim(90,120)
plt.xlabel('Valori centroidi')
plt.ylabel('Valori raggi curvatura')
plt.title('Disperisione parametri')
plt.legend()
plt.show()
plt.savefig('figura.png')

#PLOT ERRORE STIMA CENTROIDE

order=np.argsort(axc)
xs=np.array(axc)[order]
ys=np.array(chi1)[order]
popt,pcov=curve_fit(fun,xs,ys)
plt.rcParams['figure.figsize']=[7.5,5]
plt.rcParams['font.size']=10
plt.rcParams['lines.linewidth']=2
plt.scatter(xs,ys,marker='.',label='$\chi^2$')
plt.plot(xs,fun(xs,*popt),color='red',label='fit parabola a=%1f, b=%1f, c=%1f' %
    tuple(popt))
x=np.linspace(-1.5,1.5,150)
plt.ylim(0,65)
plt.plot(x,fun(x,*popt),color='red')
vx=-popt[1]/(2*popt[0])
vy=-(popt[1]**2-4*popt[0]*popt[2])/(4*popt[0])
plt.axhline(y=vy,color='green',linestyle='-')
plt.axhline(y=vy+1,color='green',linestyle='-')
plt.axvline(x=vx,color='green',linestyle='-')
plt.xlabel('stime centroidi')
plt.ylabel('$\chi^2$')
plt.title('Parabola $\chi^2$')
plt.legend()
plt.show()
plt.savefig('figura.png')

#PLOT ERRORE STIMA RAGGIO CURVATURA

order=np.argsort(al)
xs=np.array(al)[order]
ys=np.array(chi2)[order]
popt,pcov=curve_fit(fun,xs,ys)
plt.rcParams['figure.figsize']=[7.5,5]
plt.rcParams['font.size']=10
plt.rcParams['lines.linewidth']=2
plt.scatter(xs,ys,marker='.',label='$\chi^2$')
plt.plot(xs,fun(xs,*popt),color='red',label='fit parabola a=%1f, b=%1f, c=%1f' %
    tuple(popt))
x=np.linspace(60,140,150)
plt.ylim(0,65)
plt.plot(x,fun(x,*popt),color='red')
vx=-popt[1]/(2*popt[0])
vy=-(popt[1]**2-4*popt[0]*popt[2])/(4*popt[0])
plt.axhline(y=vy,color='green',linestyle='-')
```

```
plt.axhline(y=vy+1,color='green',linestyle='-')
plt.axvline(x=vx,color='green',linestyle='-')
plt.xlabel('stime raggi curvatura')
plt.ylabel('$\chi^2$')
plt.title('Parabola $\chi^2$')
plt.legend()
plt.show()
plt.savefig('figura.png')
```

Codice per la stima dei parametri con varianza variabile tra una stima e l'altra 1D.

```
'''PROGRAMMA DIPENDENZA STIME VARIANZA'''

import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import norm
from math import sqrt

p=10
n=9
m=6
l=p*n
var=np.empty(m)
var=([1e-6,1e-2,2.5e-2,5e-2,1e-1,1.5e-1])
dx=2
df=5

ll=[]
lxt=np.array([-8.52,-6.32,-4.17,-2.07,0.01,2.06,4.11,6.26,8.46])
lxs=np.empty(n)
lxl=np.empty(n)
dnt=np.empty(n)
dne=np.empty(n)
dnl=np.empty(n)
axc1=np.empty(p)
al1=np.empty(p)
axc2=np.empty(p)
al2=np.empty(p)
axc3=np.empty(p)
al3=np.empty(p)
axc4=np.empty(p)
al4=np.empty(p)
axc5=np.empty(p)
al5=np.empty(p)
axc6=np.empty(p)
al6=np.empty(p)

for j in range(-int(n/2),int(n/2)+1):
 di=j*dx
 ll.append(di)
lxl=np.asarray(ll)

dnt=abs(lxt-lxl)
```

```
dnl=abs(lxl)


for i in range(0,p):
 lxs=np.random.normal(lxt[:],sqrt(var[0]),size=None)
 dne=abs(lxs-lxl)
 s1=0
 s2=0
 s3=0
 s4=0
 s5=0
 for j in range(0,n):
  s1+=1/var[0]
  s2+=dne[j]/var[0]
  s3+=dnl[j]/var[0]
  s4+=dnl[j]*dne[j]/var[0]
  s5+=dnl[j]**2/var[0]
 xc=(s3*s4-s2*s5)/(3*s3*s2+s1*s4)
 L=df*(s3+s2-s1*((s3*s4-s2*s5)/(3*s3*s2+s1*s4)))/s2
 axc1[i]=xc
 al1[i]=L

for i in range(0,p):
 lxs=np.random.normal(lxt[:],sqrt(var[1]),size=None)
 dne=abs(lxs-lxl)
 s1=0
 s2=0
 s3=0
 s4=0
 s5=0
 for j in range(0,n):
  s1+=1/var[1]
  s2+=dne[j]/var[1]
  s3+=dnl[j]/var[1]
  s4+=dnl[j]*dne[j]/var[1]
  s5+=dnl[j]**2/var[1]
 xc=(s3*s4-s2*s5)/(3*s3*s2+s1*s4)
 L=df*(s3+s2-s1*((s3*s4-s2*s5)/(3*s3*s2+s1*s4)))/s2
 axc2[i]=xc
 al2[i]=L

for i in range(0,p):
 lxs=np.random.normal(lxt[:],sqrt(var[2]),size=None)
 dne=abs(lxs-lxl)
 s1=0
 s2=0
 s3=0
 s4=0
 s5=0
 for j in range(0,n):
  s1+=1/var[2]
  s2+=dne[j]/var[2]
  s3+=dnl[j]/var[2]
  s4+=dnl[j]*dne[j]/var[2]
```

```
  s5+=dnl[j]**2/var[2]
 xc=(s3*s4-s2*s5)/(3*s3*s2+s1*s4)
 L=df*(s3+s2-s1*((s3*s4-s2*s5)/(3*s3*s2+s1*s4)))/s2
 axc3[i]=xc
 al3[i]=L

for i in range(0,p):
 lxs=np.random.normal(lxt[:],sqrt(var[3]),size=None)
 dne=abs(lxs-lxl)
 s1=0
 s2=0
 s3=0
 s4=0
 s5=0
 for j in range(0,n):
  s1+=1/var[3]
  s2+=dne[j]/var[3]
  s3+=dnl[j]/var[3]
  s4+=dnl[j]*dne[j]/var[3]
  s5+=dnl[j]**2/var[3]
 xc=(s3*s4-s2*s5)/(3*s3*s2+s1*s4)
 L=df*(s3+s2-s1*((s3*s4-s2*s5)/(3*s3*s2+s1*s4)))/s2
 axc4[i]=xc
 al4[i]=L

for i in range(0,p):
 lxs=np.random.normal(lxt[:],sqrt(var[4]),size=None)
 dne=abs(lxs-lxl)
 s1=0
 s2=0
 s3=0
 s4=0
 s5=0
 for j in range(0,n):
  s1+=1/var[4]
  s2+=dne[j]/var[4]
  s3+=dnl[j]/var[4]
  s4+=dnl[j]*dne[j]/var[4]
  s5+=dnl[j]**2/var[4]
 xc=(s3*s4-s2*s5)/(3*s3*s2+s1*s4)
 L=df*(s3+s2-s1*((s3*s4-s2*s5)/(3*s3*s2+s1*s4)))/s2
 axc5[i]=xc
 al5[i]=L

for i in range(0,p):
 lxs=np.random.normal(lxt[:],sqrt(var[5]),size=None)
 dne=abs(lxs-lxl)
 s1=0
 s2=0
 s3=0
 s4=0
 s5=0
 for j in range(0,n):
  s1+=1/var[5]
```

```
  s2+=dne[j]/var[5]
  s3+=dnl[j]/var[5]
  s4+=dnl[j]*dne[j]/var[5]
  s5+=dnl[j]**2/var[5]
 xc=(s3*s4-s2*s5)/(3*s3*s2+s1*s4)
 L=df*(s3+s2-s1*((s3*s4-s2*s5)/(3*s3*s2+s1*s4)))/s2
 axc6[i]=xc
 al6[i]=L

axcf=[axc1,axc2,axc3,axc4,axc5,axc6]
alf=[al1,al2,al3,al4,al5,al6]

#PLOT STIMA POSIZIONE CENTROIDI IN FUNZIONE VARIANZA CORRELATA AD OGNI STIMA

plt.rcParams['figure.figsize']=[7.5,5]
plt.rcParams['font.size']=10
plt.rcParams['lines.linewidth']=2
plt.boxplot(axcf)
plt.title('Dispersione stime centroidi in relazione alla loro varianza')
plt.legend()
plt.show()
plt.savefig('figura.png')

#PLOT STIMA RAGGI CURVATURA IN FUNZIONE VARIANZA CORRELATA

plt.rcParams['figure.figsize']=[7.5,5]
plt.rcParams['font.size']=10
plt.rcParams['lines.linewidth']=2
plt.boxplot(alf)
plt.title('Dispersione stime raggi curvatura in relazione alla loro varianza')
plt.legend()
plt.show()
plt.savefig('figura.png')
```

Codice per la stima numerica dei parametri x_0, y_0 e L nel caso 2D.

```
'''PROGRAMMA PER IMPLEMENTAZIONE NUMERICA BIDIMENSIONALE METODO NEWTON-RAHPSON'''

import matplotlib.pyplot as plt
import numpy as np
from math import sqrt

def funzionex(x0,y0,l):
    fx=0
    for j in range(0,m):
        for i in range(0,n):
         fx+=((dnlx[j,i]-x0)-dne[j,i]*((l-df)/df)*((dnlx[j,i]-x0)/sqrt((dnlx[j,i]-
    x0)**2+(dnly[j,i]-y0)**2)))/var
    return fx

def derivatax(x0,y0,l,fx):
    h=1e-8
    derivx=(fx(x0+h,y0,l)-fx(x0,y0,l))/h
```

```
    return derivx

def newtonx(x0,fx,derivx):
 tol=1e-9
 xn=x0
 funzrefx=fx(xn,y0,l)
 while abs(funzrefx)>tol:
  xn1=xn-(fx(xn,y0,l)/derivx(xn,y0,l,fx))
  xn=xn1
  funzrefx=fx(xn1,y0,l)
 return xn1

#STIMA DI X0
x0=-0.2
y0=-0.2
l=100

n=5
m=5
var=0.0004
dx=2
dy=2
df=5

aly=np.empty(m)
lx=[]
ly=[]
lxt=np.array([[-4.20,-2.06,0.08,2.21,4.29],[-4.17,-2.05,0.07,2.17,4.26],
[-4.11,-2.04,0.06,2.11,4.21],[-4.15,-2.05,0.07,2.15,4.23],
[-4.19,-2.06,0.08,2.19,4.27]])
lyt=np.array([[4.29,4.25,4.19,4.21,4.26],[2.21,2.16,2.11,2.13,2.18],
[0.08,0.07,0.06,0.07,0.08],[-2.16,-2.13,-2.08,-2.09,-2.11],
[-4.27,-4.21,-4.16,-4.19,-4.24]])
lxs=np.empty((m,n))
lys=np.empty((m,n))
lxl=np.empty((m,n))
lyl=np.empty((m,n))
dntx=np.empty((m,n))
dnty=np.empty((m,n))
dnex=np.empty((m,n))
dney=np.empty((m,n))
dnlx=np.empty((m,n))
dnly=np.empty((m,n))
dne=np.empty((m,n))
dnt=np.empty((m,n))

lxs=np.random.normal(lxt[:],sqrt(var),size=None)
lys=np.random.normal(lyt[:],sqrt(var),size=None)


for i in range(-int(n/2),int(n/2)+1):
 di=i*dx
 lx.append(di)
for j in range(0,m):
```

```
 lxl[j]=np.asarray(lx)

for j in range(-int(m/2),int(m/2)+1):
 dj=j*dy
 ly.append(dj)
ly=sorted(ly,reverse=True)
aly=np.asarray(ly)
aly=aly.reshape(-1,1)
for i in range(0,n):
 lyl[None,:]=aly

dntx=abs(lxt-lxl)
dnty=abs(lyt-lyl)
dnex=abs(lxs-lxl)
dney=abs(lys-lyl)
dnlx=abs(lxl)
dnly=abs(lyl)

dne=dnex**2+dney**2
for i in range(0,n):
 for j in range(0,m):
  dne[j,i]=sqrt(dne[j,i])
dnt=dntx**2+dnty**2
for i in range(0,n):
 for j in range(0,m):
  dnt[j,i]=sqrt(dnt[j,i])

print(lxt)
print(lyt)
print(lxs)
print(lys)
print(lxl)
print(lyl)
print(dntx)
print(dnty)
print(dnex)
print(dney)
print(dnlx)
print(dnly)

ascissa=newtonx(x0,funzionex,derivatax)
print(f'ascissa centroide {ascissa}')

def funzioney(xc,yc,lc):
    fy=0
    for j in range(0,m):
        for i in range(0,n):
         fy+=((dnly[j,i]-yc)-dne[j,i]*((lc-df)/df)*((dnly[j,i]-yc)/sqrt((dnlx[j,i
    ]-xc)**2+(dnly[j,i]-yc)**2)))/var
    return fy

def derivatay(xc,yc,lc,fy):
 h=1e-8
 derivy=(fy(xc,yc+h,lc)-fy(xc,yc,lc))/h
```

```
 return derivy

def newtony(yc,fy,derivy):
    tol=1e-9
    yn=yc
    funzrefy=fy(xc,yn,lc)
    while abs(funzrefy)>tol:
     yn1=yn-(fy(xc,yn,lc)/derivy(xc,yn,lc,fy))
     yn=yn1
     funzrefy=fy(xc,yn1,lc)
    return yn1

#STIMA DI Y0
xc=ascissa
yc=-0.25
lc=100

n=5
m=5
var=0.0004
dx=2
dy=2
df=5

aly=np.empty(m)
lx=[]
ly=[]
lxt=np.array([[-4.20,-2.06,0.08,2.21,4.29],[-4.17,-2.05,0.07,2.17,4.26],
[-4.11,-2.04,0.06,2.11,4.21],[-4.15,-2.05,0.07,2.15,4.23],
[-4.19,-2.06,0.08,2.19,4.27]])
lyt=np.array([[4.29,4.25,4.19,4.21,4.26],[2.21,2.16,2.11,2.13,2.18],
[0.08,0.07,0.06,0.07,0.08],[-2.16,-2.13,-2.08,-2.09,-2.11],
[-4.27,-4.21,-4.16,-4.19,-4.24]])
lxs=np.empty((m,n))
lys=np.empty((m,n))
lxl=np.empty((m,n))
lyl=np.empty((m,n))
dntx=np.empty((m,n))
dnty=np.empty((m,n))
dnex=np.empty((m,n))
dney=np.empty((m,n))
dnlx=np.empty((m,n))
dnly=np.empty((m,n))
dne=np.empty((m,n))
dnt=np.empty((m,n))

lxs=np.random.normal(lxt[:],sqrt(var),size=None)
lys=np.random.normal(lyt[:],sqrt(var),size=None)


for i in range(-int(n/2),int(n/2)+1):
 di=i*dx
 lx.append(di)
for j in range(0,m):
```

```
 lxl[j]=np.asarray(lx)

for j in range(-int(m/2),int(m/2)+1):
 dj=j*dy
 ly.append(dj)
ly=sorted(ly,reverse=True)
aly=np.asarray(ly)
aly=aly.reshape(-1,1)
for i in range(0,n):
 lyl[None,:]=aly

dntx=abs(lxt-lxl)
dnty=abs(lyt-lyl)
dnex=abs(lxs-lxl)
dney=abs(lys-lyl)
dnlx=abs(lxl)
dnly=abs(lyl)

dne=dnex**2+dney**2
for i in range(0,n):
 for j in range(0,m):
  dne[j,i]=sqrt(dne[j,i])
dnt=dntx**2+dnty**2
for i in range(0,n):
 for j in range(0,m):
  dnt[j,i]=sqrt(dnt[j,i])

ordinata=newtony(yc,funzioney,derivatay)
print(f'ordinata centroide {ordinata}')

def funzionel(xcc,ycc,lcc):
 fl=0
 for j in range(0,m):
  for i in range(0,n):
   fl+=((df/(lcc-df)**3)*((dnlx[j,i]-xcc)**2+(dnly[j,i]-ycc)**2)-(dne[j,i]/(lcc-df
   )**2)*sqrt((dnlx[j,i]-xcc)**2+(dnly[j,i]-ycc)**2))/var
 return fl

def derivatal(xcc,ycc,lcc,fl):
    h=1e-8
    derivl=(fl(xcc,ycc,lcc+h)-fl(xcc,ycc,lcc))/h
    return derivl

def newtonl(lcc,fl,derivl):
    tol=1e-9
    ln=lcc
    funzrefl=fl(xcc,ycc,ln)
    while abs(funzrefl)>tol:
     ln1=ln-(fl(xcc,ycc,ln)/derivl(xcc,ycc,ln,fl))
     ln=ln1
     funzrefl=fl(xcc,ycc,ln1)
    return ln1

#STIMA DI L
```

```
lcc=120
xcc=ascissa
ycc=ordinata

n=5
m=5
var=0.0004
dx=2
dy=2
df=5

aly=np.empty(m)
lx=[]
ly=[]
lxt=np.array([[-4.20,-2.06,0.08,2.21,4.29],[-4.17,-2.05,0.07,2.17,4.26],
[-4.11,-2.04,0.06,2.11,4.21],[-4.15,-2.05,0.07,2.15,4.23],
[-4.19,-2.06,0.08,2.19,4.27]])
lyt=np.array([[4.29,4.25,4.19,4.21,4.26],[2.21,2.16,2.11,2.13,2.18],
[0.08,0.07,0.06,0.07,0.08],[-2.16,-2.13,-2.08,-2.09,-2.11],
[-4.27,-4.21,-4.16,-4.19,-4.24]])
lxs=np.empty((m,n))
lys=np.empty((m,n))
lxl=np.empty((m,n))
lyl=np.empty((m,n))
dntx=np.empty((m,n))
dnty=np.empty((m,n))
dnex=np.empty((m,n))
dney=np.empty((m,n))
dnlx=np.empty((m,n))
dnly=np.empty((m,n))
dne=np.empty((m,n))
dnt=np.empty((m,n))

lxs=np.random.normal(lxt[:],sqrt(var),size=None)
lys=np.random.normal(lyt[:],sqrt(var),size=None)

for i in range(-int(n/2),int(n/2)+1):
 di=i*dx
 lx.append(di)
for j in range(0,m):
 lxl[j]=np.asarray(lx)

for j in range(-int(m/2),int(m/2)+1):
 dj=j*dy
 ly.append(dj)
ly=sorted(ly,reverse=True)
aly=np.asarray(ly)
aly=aly.reshape(-1,1)
for i in range(0,n):
 lyl[None,:]=aly

dntx=abs(lxt-lxl)
dnty=abs(lyt-lyl)
dnex=abs(lxs-lxl)
```

```
dney=abs(lys-lyl)
dnlx=abs(lxl)
dnly=abs(lyl)

dne=dnex**2+dney**2
for i in range(0,n):
 for j in range(0,m):
  dne[j,i]=sqrt(dne[j,i])
dnt=dntx**2+dnty**2
for i in range(0,n):
 for j in range(0,m):
  dnt[j,i]=sqrt(dnt[j,i])

raggio=newtonl(lcc,funzionel,derivatal)
print(f'raggio curvatura {raggio}')

print(f'differenza valore ottenuto e atteso ordinate {abs(ordinata+0.2)}')
print(f'differenza valore ottenuto e atteso raggio {abs(raggio-100)}')
```

Codice per il metodo di Newton multidimensionale nel caso 2D.
Codice per la realizzazione dello scatter plot di ascisse e ordinate del centroide nel caso 2D.

```
'''PROGRAMMA SCATTER PLOT ASCISSE ORDINATE'''

import matplotlib.pyplot as plt
import numpy as np
from math import sqrt
from scipy.stats import norm

def funzionex(x0,y0,l):
    fx=0
    for j in range(0,m):
        for i in range(0,n):
         fx+=((dnlx[j,i]-x0)-dne[j,i]*((l-df)/df)*((dnlx[j,i]-x0)/sqrt((dnlx[j,i]-x0)**2+(dnly[j,i]-y0)**2)))/var
    return fx

def derivatax(x0,y0,l,fx):
    h=1e-8
    derivx=(fx(x0+h,y0,l)-fx(x0,y0,l))/h
    return derivx

def newtonx(x0,fx,derivx):
 tol=1e-9
 xn=x0
 funzrefx=fx(xn,y0,l)
 while abs(funzrefx)>tol:
  xn1=xn-(fx(xn,y0,l)/derivx(xn,y0,l,fx))
  xn=xn1
  funzrefx=fx(xn1,y0,l)
 return xn1

#STIMA DI X0
```

```
x0=-0.2
y0=-0.2
l=100

mc=600
n=5
m=5
var=0.0004
dx=2
dy=2
df=5

aly=np.empty(m)
lx=[]
ly=[]
lxt=np.array([[-4.20,-2.06,0.08,2.21,4.29],[-4.17,-2.05,0.07,2.17,4.26],
[-4.11,-2.04,0.06,2.11,4.21],[-4.15,-2.05,0.07,2.15,4.23],
[-4.19,-2.06,0.08,2.19,4.27]])
lyt=np.array([[4.29,4.25,4.19,4.21,4.26],[2.21,2.16,2.11,2.13,2.18],
[0.08,0.07,0.06,0.07,0.08],[-2.16,-2.13,-2.08,-2.09,-2.11],
[-4.27,-4.21,-4.16,-4.19,-4.24]])
lxs=np.empty((m,n))
lys=np.empty((m,n))
lxl=np.empty((m,n))
lyl=np.empty((m,n))
dntx=np.empty((m,n))
dnty=np.empty((m,n))
dnex=np.empty((m,n))
dney=np.empty((m,n))
dnlx=np.empty((m,n))
dnly=np.empty((m,n))
dne=np.empty((m,n))
dnt=np.empty((m,n))
axc=np.empty(mc)


for i in range(-int(n/2),int(n/2)+1):
 di=i*dx
 lx.append(di)
for j in range(0,m):
 lxl[j]=np.asarray(lx)

for j in range(-int(m/2),int(m/2)+1):
 dj=j*dy
 ly.append(dj)
ly=sorted(ly,reverse=True)
aly=np.asarray(ly)
aly=aly.reshape(-1,1)
for i in range(0,n):
 lyl[None,:]=aly

dntx=abs(lxt-lxl)
dnty=abs(lyt-lyl)
dnlx=abs(lxl)
```

```
dnly=abs(lyl)

for k in range(0,mc):
    lxs=np.random.normal(lxt[:],sqrt(var),size=None)
    lys=np.random.normal(lyt[:],sqrt(var),size=None)
    dnex=abs(lxs-lxl)
    dney=abs(lys-lyl)
    dne=dnex**2+dney**2
    for i in range(0,n):
     for j in range(0,m):
      dne[j,i]=sqrt(dne[j,i])
    ascissa=newtonx(x0,funzionex,derivatax)
    axc[k]=ascissa

def funzioney(axc,yc,lc):
 for r in range(0,mc):
  fy=0
  for j in range(0,m):
   for i in range(0,n):
    fy+=((dnly[j,i]-yc)-dne[j,i]*((lc-df)/df)*((dnly[j,i]-yc)/sqrt((dnlx[j,i]-axc[
    r])**2+(dnly[j,i]-yc)**2)))/var
  return fy

def derivatay(axc,yc,lc,fy):
 h=1e-8
 derivy=(fy(axc,yc+h,lc)-fy(axc,yc,lc))/h
 return derivy

def newtony(yc,fy,derivy):
    tol=1e-9
    yn=yc
    funzrefy=fy(axc,yn,lc)
    while abs(funzrefy)>tol:
     yn1=yn-(fy(axc,yn,lc)/derivy(axc,yn,lc,fy))
     yn=yn1
     funzrefy=fy(axc,yn1,lc)
    return yn1

#STIMA DI Y0
yc=-0.25
lc=100

mc=600
n=5
m=5
var=0.0004
dx=2
dy=2
df=5

aly=np.empty(m)
lx=[]
ly=[]
lxt=np.array([[-4.20,-2.06,0.08,2.21,4.29],[-4.17,-2.05,0.07,2.17,4.26],
```

```
[-4.11,-2.04,0.06,2.11,4.21],[-4.15,-2.05,0.07,2.15,4.23],
[-4.19,-2.06,0.08,2.19,4.27]])
lyt=np.array([[4.29,4.25,4.19,4.21,4.26],[2.21,2.16,2.11,2.13,2.18],
[0.08,0.07,0.06,0.07,0.08],[-2.16,-2.13,-2.08,-2.09,-2.11],
[-4.27,-4.21,-4.16,-4.19,-4.24]])
lxs=np.empty((m,n))
lys=np.empty((m,n))
lxl=np.empty((m,n))
lyl=np.empty((m,n))
dntx=np.empty((m,n))
dnty=np.empty((m,n))
dnex=np.empty((m,n))
dney=np.empty((m,n))
dnlx=np.empty((m,n))
dnly=np.empty((m,n))
dne=np.empty((m,n))
dnt=np.empty((m,n))
ayc=np.empty(mc)

for i in range(-int(n/2),int(n/2)+1):
 di=i*dx
 lx.append(di)
for j in range(0,m):
 lxl[j]=np.asarray(lx)

for j in range(-int(m/2),int(m/2)+1):
 dj=j*dy
 ly.append(dj)
ly=sorted(ly,reverse=True)
aly=np.asarray(ly)
aly=aly.reshape(-1,1)
for i in range(0,n):
 lyl[None,:]=aly

dntx=abs(lxt-lxl)
dnty=abs(lyt-lyl)
dnlx=abs(lxl)
dnly=abs(lyl)

for k in range(0,mc):
 lxs=np.random.normal(lxt[:],sqrt(var),size=None)
 lys=np.random.normal(lyt[:],sqrt(var),size=None)
 dnex=abs(lxs-lxl)
 dney=abs(lys-lyl)
 dne=dnex**2+dney**2
 for i in range(0,n):
  for j in range(0,m):
   dne[j,i]=sqrt(dne[j,i])
 ordinata=newtony(yc,funzioney,derivatay)
 ayc[k]=ordinata

#PLOT DISPERSIONE STIME X0 E Y0

plt.rcParams['figure.figsize']=[7.5,5]
```

```
plt.rcParams['font.size']=10
plt.rcParams['lines.linewidth']=2
plt.scatter(axc,ayc,marker='.',linewidths=0.8)
plt.xlim(-1,1)
plt.ylim(-1,1)
plt.xlabel('Valori ascisse centroidi')
plt.ylabel('Valori ordinate centroidi')
plt.title('Disperisione parametri centroidi')
plt.legend()
plt.show()
plt.savefig('figura.png')
```

Codice per la realizzazione del metodo Montecarlo e per la stima delle deviazioni standard dei parametri nel caso 2D.

```
'''PROGRAMMA METODO MONTECARLO BIDIMENSIONALE'''

import matplotlib.pyplot as plt
import numpy as np
from math import sqrt
from scipy.stats import norm
from scipy.optimize import curve_fit

def fun(x,a,b,c):
 return x**2*a+x*b+c

def funzionex(x0,y0,l):
    fx=0
    for j in range(0,m):
        for i in range(0,n):
         fx+=((dnlx[j,i]-x0)-dne[j,i]*((l-df)/df)*((dnlx[j,i]-x0)/sqrt((dnlx[j,i]-x0)**2+(dnly[j,i]-y0)**2)))/var
    return fx

def derivatax(x0,y0,l,fx):
    h=1e-8
    derivx=(fx(x0+h,y0,l)-fx(x0,y0,l))/h
    return derivx

def newtonx(x0,fx,derivx):
 tol=1e-9
 xn=x0
 funzrefx=fx(xn,y0,l)
 while abs(funzrefx)>tol:
  xn1=xn-(fx(xn,y0,l)/derivx(xn,y0,l,fx))
  xn=xn1
  funzrefx=fx(xn1,y0,l)
 return xn1

#STIMA DI X0
x0=-0.2
y0=-0.2
l=100
```

```
38
39 mc=600
40 n=5
41 m=5
42 var=0.0004
43 dx=2
44 dy=2
45 df=5
46
47 aly=np.empty(m)
48 lx=[]
49 ly=[]
50 lxt=np.array([[-4.20,-2.06,0.08,2.21,4.29],[-4.17,-2.05,0.07,2.17,4.26],
51 [-4.11,-2.04,0.06,2.11,4.21],[-4.15,-2.05,0.07,2.15,4.23],
52 [-4.19,-2.06,0.08,2.19,4.27]])
53 lyt=np.array([[4.29,4.25,4.19,4.21,4.26],[2.21,2.16,2.11,2.13,2.18],
54 [0.08,0.07,0.06,0.07,0.08],[-2.16,-2.13,-2.08,-2.09,-2.11],
55 [-4.27,-4.21,-4.16,-4.19,-4.24]])
56 lxs=np.empty((m,n))
57 lys=np.empty((m,n))
58 lxl=np.empty((m,n))
59 lyl=np.empty((m,n))
60 dntx=np.empty((m,n))
61 dnty=np.empty((m,n))
62 dnex=np.empty((m,n))
63 dney=np.empty((m,n))
64 dnlx=np.empty((m,n))
65 dnly=np.empty((m,n))
66 dne=np.empty((m,n))
67 dnt=np.empty((m,n))
68 axc=np.empty(mc)
69 dist1=np.empty(mc)
70 achi1=np.empty(mc)
71
72 for i in range(-int(n/2),int(n/2)+1):
73  di=i*dx
74  lx.append(di)
75 for j in range(0,m):
76  lxl[j]=np.asarray(lx)
77
78 for j in range(-int(m/2),int(m/2)+1):
79  dj=j*dy
80  ly.append(dj)
81 ly=sorted(ly,reverse=True)
82 aly=np.asarray(ly)
83 aly=aly.reshape(-1,1)
84 for i in range(0,n):
85  lyl[None,:]=aly
86
87 dntx=abs(lxt-lxl)
88 dnty=abs(lyt-lyl)
89 dnlx=abs(lxl)
90 dnly=abs(lyl)
91
```

```
for k in range(0,mc):
    lxs=np.random.normal(lxt[:],sqrt(var),size=None)
    lys=np.random.normal(lyt[:],sqrt(var),size=None)
    dnex=abs(lxs-lxl)
    dney=abs(lys-lyl)
    dne=dnex**2+dney**2
    for i in range(0,n):
     for j in range(0,m):
      dne[j,i]=sqrt(dne[j,i])
    ascissa=newtonx(x0,funzionex,derivatax)
    axc[k]=ascissa
    distanza1=sqrt(ascissa**2+(0.2)**2)
    dist1[k]=distanza1
distmedasc=np.mean(dist1)
print(distmedasc)

#PLOT DISTRIBUZIONE ASCISSE CENTROIDI

plt.rcParams['figure.figsize']=[7.5,5]
plt.rcParams['font.size']=13
plt.rcParams['lines.linewidth']=2
plt.hist(axc,bins=13,histtype='bar',color='red',label=f'Stime delle ascisse dei
    centroidi',density=True)
(mu1,sigma)=norm.fit(axc)
x=np.linspace(-1.5,1.5,220)
y=norm.pdf(x,mu1,sigma)
plt.plot(x,y,label=f'$\sigma$={sigma:.2f}, $\mu$={mu1:.2f}')
plt.xlabel('Valori ascisse centroidi')
plt.ylabel('Counts')
plt.xlim(-1.5,1.5)
plt.ylim(0,5)
plt.title('Distribuzione ascisse centroidi')
plt.legend()
plt.show()
plt.savefig('figura.png')

def funzioney(xc,yc,lc):
    fy=0
    for j in range(0,m):
        for i in range(0,n):
         fy+=((dnly[j,i]-yc)-dne[j,i]*((lc-df)/df)*((dnly[j,i]-yc)/sqrt((dnlx[j,i
    ]-xc)**2+(dnly[j,i]-yc)**2)))/var
    return fy

def derivatay(xc,yc,lc,fy):
 h=1e-8
 derivy=(fy(xc,yc+h,lc)-fy(xc,yc,lc))/h
 return derivy

def newtony(yc,fy,derivy):
    tol=1e-9
    yn=yc
    funzrefy=fy(xc,yn,lc)
    while abs(funzrefy)>tol:
```

```
     yn1=yn-(fy(xc,yn,lc)/derivy(xc,yn,lc,fy))
     yn=yn1
     funzrefy=fy(xc,yn1,lc)
    return yn1

#STIMA DI Y0
xc=mu1
yc=-0.25
lc=100

mc=600
n=5
m=5
var=0.0004
dx=2
dy=2
df=5

aly=np.empty(m)
lx=[]
ly=[]
lxt=np.array([[-4.20,-2.06,0.08,2.21,4.29],[-4.17,-2.05,0.07,2.17,4.26],
[-4.11,-2.04,0.06,2.11,4.21],[-4.15,-2.05,0.07,2.15,4.23],
[-4.19,-2.06,0.08,2.19,4.27]])
lyt=np.array([[4.29,4.25,4.19,4.21,4.26],[2.21,2.16,2.11,2.13,2.18],
[0.08,0.07,0.06,0.07,0.08],[-2.16,-2.13,-2.08,-2.09,-2.11],
[-4.27,-4.21,-4.16,-4.19,-4.24]])
lxs=np.empty((m,n))
lys=np.empty((m,n))
lxl=np.empty((m,n))
lyl=np.empty((m,n))
dntx=np.empty((m,n))
dnty=np.empty((m,n))
dnex=np.empty((m,n))
dney=np.empty((m,n))
dnlx=np.empty((m,n))
dnly=np.empty((m,n))
dne=np.empty((m,n))
dnt=np.empty((m,n))
ayc=np.empty(mc)
dist2=np.empty(mc)
achi2=np.empty(mc)

for i in range(-int(n/2),int(n/2)+1):
 di=i*dx
 lx.append(di)
for j in range(0,m):
 lxl[j]=np.asarray(lx)

for j in range(-int(m/2),int(m/2)+1):
 dj=j*dy
 ly.append(dj)
ly=sorted(ly,reverse=True)
aly=np.asarray(ly)
```

```
aly=aly.reshape(-1,1)
for i in range(0,n):
 lyl[None,:]=aly

dntx=abs(lxt-lxl)
dnty=abs(lyt-lyl)
dnlx=abs(lxl)
dnly=abs(lyl)

for k in range(0,mc):
 lxs=np.random.normal(lxt[:],sqrt(var),size=None)
 lys=np.random.normal(lyt[:],sqrt(var),size=None)
 dnex=abs(lxs-lxl)
 dney=abs(lys-lyl)
 dne=dnex**2+dney**2
 for i in range(0,n):
  for j in range(0,m):
   dne[j,i]=sqrt(dne[j,i])
 ordinata=newtony(yc,funzioney,derivatay)
 ayc[k]=ordinata
 distanza2=sqrt(mu1**2+ordinata**2)
 dist2[k]=distanza2
distmedord=np.mean(dist2)
print(distmedord)

#PLOT DISTRIBUZIONE ORDINATE CENTROIDI

plt.rcParams['figure.figsize']=[7.5,5]
plt.rcParams['font.size']=13
plt.rcParams['lines.linewidth']=2
plt.hist(ayc,bins=13,histtype='bar',color='red',label=f'Stime delle ordinate dei
    centroidi',density=True)
(mu2,sigma)=norm.fit(ayc)
x=np.linspace(-1.5,1.5,220)
y=norm.pdf(x,mu2,sigma)
plt.plot(x,y,label=f'$\sigma$={sigma:.2f}, $\mu$={mu2:.2f}')
plt.xlabel('Valori ordinate centroidi')
plt.ylabel('Counts')
plt.xlim(-1.5,1.5)
plt.ylim(0,5)
plt.title('Distribuzione ordinate centroidi')
plt.legend()
plt.show()
plt.savefig('figura.png')

def funzionel(xcc,ycc,lcc):
 fl=0
 for j in range(0,m):
  for i in range(0,n):
   fl+=((df/(lcc-df)**3)*((dnlx[j,i]-xcc)**2+(dnly[j,i]-ycc)**2)-(dne[j,i]/(lcc-df
    )**2)*sqrt((dnlx[j,i]-xcc)**2+(dnly[j,i]-ycc)**2))/var
 return fl

def derivatal(xcc,ycc,lcc,fl):
```

```
    h=1e-8
    derivl=(fl(xcc,ycc,lcc+h)-fl(xcc,ycc,lcc))/h
    return derivl

def newtonl(lcc,fl,derivl):
    tol=1e-9
    ln=lcc
    funzrefl=fl(xcc,ycc,ln)
    while abs(funzrefl)>tol:
     ln1=ln-(fl(xcc,ycc,ln)/derivl(xcc,ycc,ln,fl))
     ln=ln1
     funzrefl=fl(xcc,ycc,ln1)
    return ln1

#STIMA DI L
lcc=120
xcc=mu1
ycc=mu2

mc=600
n=5
m=5
var=0.0004
dx=2
dy=2
df=5

aly=np.empty(m)
lx=[]
ly=[]
lxt=np.array([[-4.20,-2.06,0.08,2.21,4.29],[-4.17,-2.05,0.07,2.17,4.26],
[-4.11,-2.04,0.06,2.11,4.21],[-4.15,-2.05,0.07,2.15,4.23],
[-4.19,-2.06,0.08,2.19,4.27]])
lyt=np.array([[4.29,4.25,4.19,4.21,4.26],[2.21,2.16,2.11,2.13,2.18],
[0.08,0.07,0.06,0.07,0.08],[-2.16,-2.13,-2.08,-2.09,-2.11],
[-4.27,-4.21,-4.16,-4.19,-4.24]])
lxs=np.empty((m,n))
lys=np.empty((m,n))
lxl=np.empty((m,n))
lyl=np.empty((m,n))
dntx=np.empty((m,n))
dnty=np.empty((m,n))
dnex=np.empty((m,n))
dney=np.empty((m,n))
dnlx=np.empty((m,n))
dnly=np.empty((m,n))
dne=np.empty((m,n))
dnt=np.empty((m,n))
lc=np.empty(mc)
achi3=np.empty(mc)

for i in range(-int(n/2),int(n/2)+1):
 di=i*dx
 lx.append(di)
```

```
for j in range(0,m):
 lxl[j]=np.asarray(lx)

for j in range(-int(m/2),int(m/2)+1):
 dj=j*dy
 ly.append(dj)
ly=sorted(ly,reverse=True)
aly=np.asarray(ly)
aly=aly.reshape(-1,1)
for i in range(0,n):
 lyl[None,:]=aly

dntx=abs(lxt-lxl)
dnty=abs(lyt-lyl)
dnlx=abs(lxl)
dnly=abs(lyl)

for k in range(0,mc):
 lxs=np.random.normal(lxt[:],sqrt(var),size=None)
 lys=np.random.normal(lyt[:],sqrt(var),size=None)
 dnex=abs(lxs-lxl)
 dney=abs(lys-lyl)
 dne=dnex**2+dney**2
 for i in range(0,n):
  for j in range(0,m):
   dne[j,i]=sqrt(dne[j,i])
 raggio=newtonl(lcc,funzionel,derivatal)
 lc[k]=raggio

#PLOT DISTRIBUZIONE RAGGI CURVATURA

plt.rcParams['figure.figsize']=[7.5,5]
plt.rcParams['font.size']=13
plt.rcParams['lines.linewidth']=2
plt.hist(lc,bins=13,histtype='bar',color='orange',label=f'Stime dei raggi di
    curvatura',density=True)
(mu3,sigma)=norm.fit(lc)
x=np.linspace(80,135,220)
y=norm.pdf(x,mu3,sigma)
plt.plot(x,y,label=f'$\sigma$={sigma:.2f}, $\mu$={mu3:.2f}')
plt.xlabel('Valori raggi curvatura')
plt.ylabel('Counts')
plt.xlim(80,135)
plt.ylim(0,5)
plt.title('Distribuzione raggi curvatura')
plt.legend()
plt.show()
plt.savefig('figura.png')

for k in range(0,mc):
    chi1=0
    chi2=0
    chi3=0
    for j in range(0,m):
```

```
        for i in range(0,n):
          chi1+=((df/(l-df))*sqrt((dnlx[j,i]-axc[k])**2+(dnly[j,i]-y0)**2)-dne[j,i])**2/var
          chi2+=((df/(l-df))*sqrt((dnlx[j,i]-mu1)**2+(dnly[j,i]-ayc[k])**2)-dne[j,i])**2/var
          chi3+=((df/(lc[k]-df))*sqrt((dnlx[j,i]-mu1)**2+(dnly[j,i]-mu2)**2)-dne[j,i])**2/var
        achi1[k]=chi1
        achi2[k]=chi2
        achi3[k]=chi3

print(achi1)
print(achi2)
print(achi3)

#PLOT STIMA ERRORE X0

order=np.argsort(axc)
xs=np.array(axc)[order]
ys=np.array(achi1)[order]
popt,pcov=curve_fit(fun,xs,ys)
plt.rcParams['figure.figsize']=[7.5,5]
plt.rcParams['font.size']=11
plt.rcParams['lines.linewidth']=2
plt.scatter(xs,ys,marker='.',label='$\chi^2$')
plt.plot(xs,fun(xs,*popt),color='red',label='fit parabola a=%1f, b=%1f, c=%1f' %tuple(popt))
x=np.linspace(-1.5,1.5,220)
plt.ylim(10,70)
plt.plot(x,fun(x,*popt),color='red')
vx=-popt[1]/(2*popt[0])
vy=-(popt[1]**2-4*popt[0]*popt[2])/(4*popt[0])
plt.axhline(y=vy,color='green',linestyle='-')
plt.axhline(y=vy+1,color='green',linestyle='-')
plt.axvline(x=vx,color='green',linestyle='-')
plt.xlabel('stime ascisse centroide')
plt.ylabel('$\chi^2$')
plt.title('Parabola $\chi^2$')
plt.legend()
plt.show()
plt.savefig('figura.png')

#PLOT STIMA ERRORE Y0

order=np.argsort(ayc)
xs=np.array(ayc)[order]
ys=np.array(achi2)[order]
popt,pcov=curve_fit(fun,xs,ys)
plt.rcParams['figure.figsize']=[7.5,5]
plt.rcParams['font.size']=11
plt.rcParams['lines.linewidth']=2
plt.scatter(xs,ys,marker='.',label='$\chi^2$')
plt.plot(xs,fun(xs,*popt),color='red',label='fit parabola a=%1f, b=%1f, c=%1f' %tuple(popt))
```

```
x=np.linspace(-1.5,1.5,220)
plt.ylim(10,60)
plt.plot(x,fun(x,*popt),color='red')
vx=-popt[1]/(2*popt[0])
vy=-(popt[1]**2-4*popt[0]*popt[2])/(4*popt[0])
plt.axhline(y=vy,color='green',linestyle='-')
plt.axhline(y=vy+1,color='green',linestyle='-')
plt.axvline(x=vx,color='green',linestyle='-')
plt.xlabel('stime ordinate centroide')
plt.ylabel('$\chi^2$')
plt.title('Parabola $\chi^2$')
plt.legend()
plt.show()
plt.savefig('figura.png')

#PLOT STIMA ERRORE L

order=np.argsort(lc)
xs=np.array(lc)[order]
ys=np.array(achi3)[order]
popt,pcov=curve_fit(fun,xs,ys)
plt.rcParams['figure.figsize']=[7.5,5]
plt.rcParams['font.size']=10
plt.rcParams['lines.linewidth']=2
plt.scatter(xs,ys,marker='.',label='$\chi^2$')
plt.plot(xs,fun(xs,*popt),color='red',label='fit parabola a=%1f, b=%1f, c=%1f' %
    tuple(popt))
x=np.linspace(85,130,220)
plt.ylim(10,60)
plt.plot(x,fun(x,*popt),color='red')
vx=-popt[1]/(2*popt[0])
vy=-(popt[1]**2-4*popt[0]*popt[2])/(4*popt[0])
plt.axhline(y=vy,color='green',linestyle='-')
plt.axhline(y=vy+1,color='green',linestyle='-')
plt.axvline(x=vx,color='green',linestyle='-')
plt.xlabel('stime raggi curvatura')
plt.ylabel('$\chi^2$')
plt.title('Parabola $\chi^2$')
plt.legend()
plt.show()
plt.savefig('figura.png')
```