

UNIVERSITÀ DEGLI STUDI DI MILANO
Facoltà di Scienze e Tecnologie
Corso di Laurea in Informatica

UNO STUDIO SUL PROBLEMA DI
IDENTIFICAZIONE E
INTERPOLAZIONE DI FUNZIONI
SINUSOIDALI SOVRAPPOSTE

Relatore: Prof. Giovanni RIGHINI

Tesi di:
Jonathan Junior AGYEKUM
Matricola: 935132

Anno Accademico 2022-2023

dedicato a mio padre, Daniel...

Ringraziamenti

Un sentito grazie a tutte le persone che mi hanno permesso di arrivare fin qui e di portare a termine questo lavoro di tesi.

Grazie al mio relatore G. Righini, sempre presente, puntuale e disponibile. Grazie al percorso intrapreso insieme ho sviluppato maggiormente la mia capacità di analisi e di problem solving.

Non posso non menzionare i miei genitori che da sempre mi sostengono nella realizzazione dei miei progetti. Non finirò mai di ringraziarvi per avermi permesso di arrivare fin qui.

Grazie ai miei amici Vincenzo, Francesco, Saverio, Andrea e Simone per essere stati sempre presenti durante le fasi del mio percorso di studi. Grazie per aver ascoltato i miei sfoghi, grazie per tutti i momenti di spensieratezza.

Indice

	ii
Ringraziamenti	iii
1 Introduzione	1
1.1 Organizzazione dei capitoli	3
1.2 Relazione con problemi simili	4
1.3 Strumenti utilizzati	4
1.4 Risultati sperimentali e finali	4
2 Descrizione del problema	6
3 Interpolazione di una sinusoide	10
3.1 Modello	10
3.1.1 Calcolo della regione pareto-ottima	10
3.2 Algoritmo	12
3.2.1 Criterio arresto	12
3.2.2 Punto di inizializzazione	13
3.2.3 Controllo del periodo	13
3.2.4 Controllo dell'ampiezza	14
3.2.5 Controllo della fase	14
3.2.6 Modifiche al modello	14
3.3 Risultati	16
3.3.1 Scelta del solutore	16
3.3.2 Selezione della soluzione	19
3.3.3 Sperimentazioni con osservazioni affette da errori	32
4 Identificazione di sinusoidi sovrapposte	40
4.1 Metodo	40
4.1.1 Creazione dei sottoproblemi	41
4.1.2 Gestione dei nodi	42

4.1.3	Gestione delle foglie	45
4.2	Valutazione delle prestazioni	48
4.3	Miglioramenti	53
5	Conclusioni	54

Capitolo 1

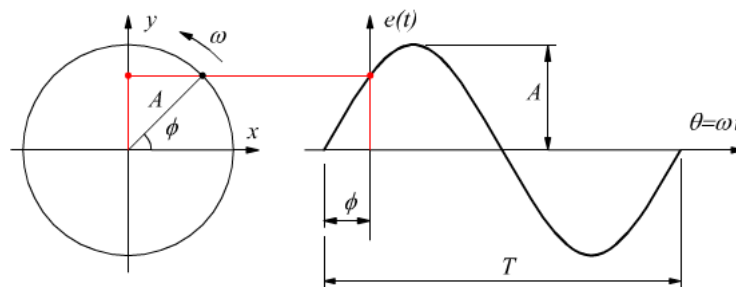
Introduzione

In questo lavoro di tesi affronto e discuto metodologie per l'individuazione ed distinzione di sinusoidi sovrapposte, osservate in funzione del tempo. L'obiettivo consiste nel riuscire a definire un programma, che dati m punti in un piano, generati da k sinusoidi, individui le formule analitiche delle k sinusoidi.

La tematica nasce dal caso di studio sull'osservazione di satelliti indistinguibili, della quale sia possibile osservarne la posizione o l'angolo rispetto al pianeta di riferimento. L'osservazione in modo continuo dell'angolo descritto da k satelliti intorno al loro pianeta, genera delle sinusoidi, questo apre alla possibilità di una identificazione dei satelliti attraverso l'analisi del loro moto orbitale.

Il legame tra l'orbite dei satelliti e le sinusoidi può essere spiegata considerando le orbite dei satelliti circolari e quindi attraverso le leggi del moto circolare.

Figura 1: Moto circolare uniforme



T : Periodo ϕ : Fase
 A : Ampiezza ω : Velocità angolare

Dove il periodo corrisponde al tempo di rivoluzione del satellite, l'ampiezza alla

massima distanza che intercorre nel periodo tra il satellite e il corrispettivo pianeta, $e(t)$ lo spostamento e la distanza del satellite lungo l'orbita. È quindi possibile descrivere la sinusoide risultante tramite l'espressione analitica: $e(t) = A \sin(\omega t + \phi)$

L'obiettivo è quello di riuscire ad attribuire ai k satelliti i k valori di ogni osservazione e di trovare periodo, ampiezza e fase delle k sinusoidi, supponendo di poter osservare i k valori simultaneamente, in n momenti successivi. Si suppone di non conoscere nulla dei k satelliti e soprattutto di non poterli distinguere in fase di rilevazione dei valori.

Questo si trasforma in un problema di identificazione di sinusoidi sovrapposte, a partire da una serie di osservazioni finita in funzione del tempo. Il compito consiste nel raggruppare le osservazioni in maniera tale da poter interpolare e ottenere le sinusoidi d'origine.

Figura 2: Esempio di osservazioni raccolte nel tempo

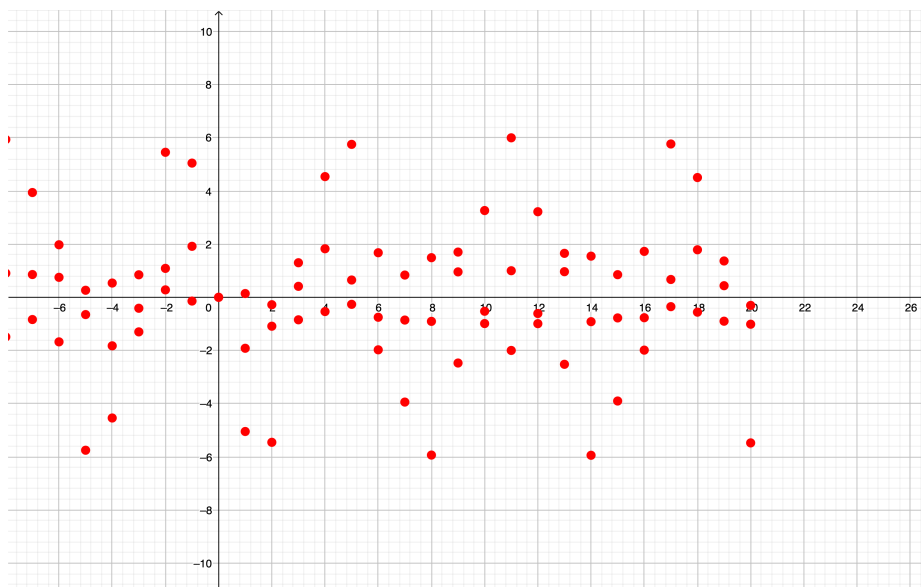
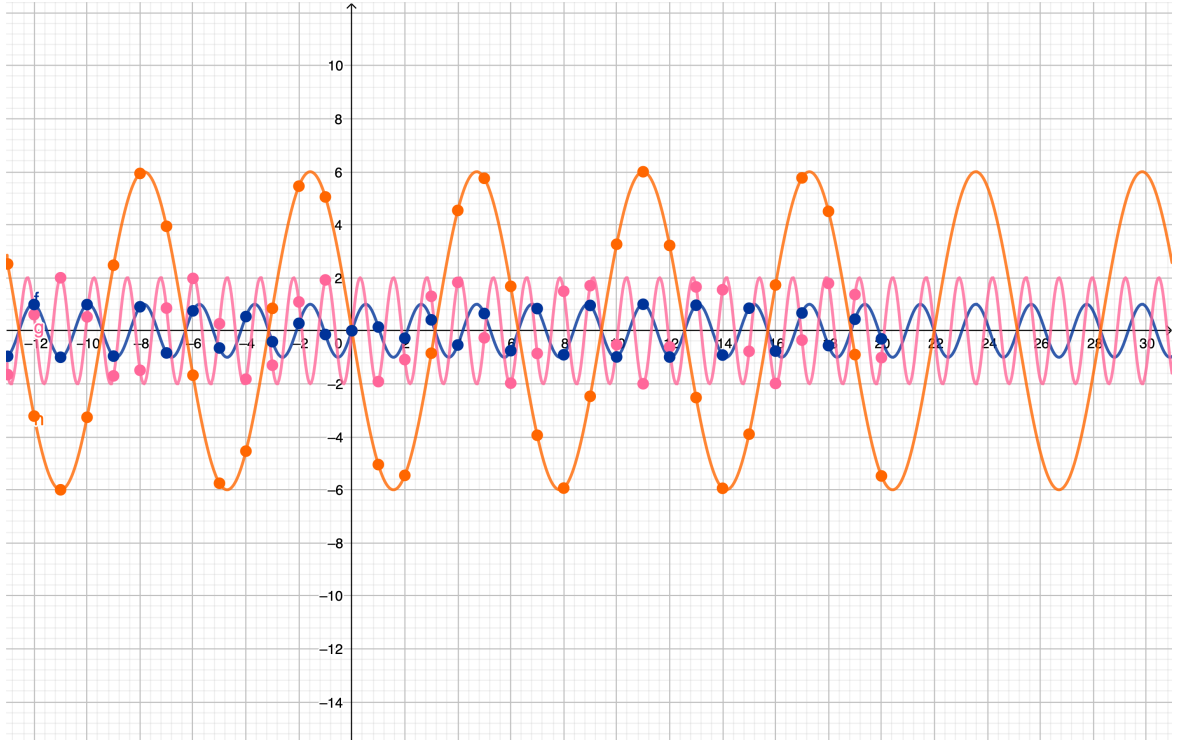


Figura 3: Esempio di individuazione di sinusoidi sovrapposte per fig. 2



1.1 Organizzazione dei capitoli

Il primo capitolo definisce un'introduzione alla tesi, specificando l'origine e la natura del problema, strumenti utilizzati, i risultati ottenuti e la struttura della tesi.

Il secondo capitolo astrae il problema, individuando moduli, parametri e variabili. Evidenziando le criticità e modalità per la risoluzione di esso.

Il terzo capitolo specifica il modello e l'algoritmo del modulo di interpolazione. In questo capitolo si analizzano inoltre le diverse metodologie di individuazione di soluzioni e di selezione di una soluzione considerata migliore.

Il quarto capitolo illustra e spiega l'algoritmo di enumerazione implementato per la risoluzione del problema di identificazione di sinusoidi sovrapposte. Si specificano strutture dati utilizzate, sottomoduli e algoritmi definiti.

1.2 Relazione con problemi simili

Il problema di interpolazione può essere individuato in diversi campi. Ad esempio può essere individuato nel problema di rilevamento di oggetti in movimento attraverso metodi basati sull'optical flow [1], dove vengono applicati metodi di interpolazione matematici come l'interpolazione lineare, polinomiale o spline. Nel campo della computer vision è utilizzata per individuare ed analizzare la direzione, la velocità e la distanza. La relazione con l'interpolazione nel problema di identificazione di sinusoidi sovrapposte, consiste nell'osservazione di un elemento in movimento in funzione del tempo, estrapolandone posizioni ed informazioni dai valori, che caratterizzano l'elemento in un determinato momento. Quindi si osserva l'evoluzione dei valori nel tempo, cercando di correlare i valori osservati attraverso funzioni matematiche di variabile reale.

1.3 Strumenti utilizzati

Per l'interpolazione di una sinusoide singola si utilizza un solutore PNL [12], attraverso la definizione di parametri, variabili, vincoli e funzione obiettivo. Un solutore PNL è un solutore di programmazione non lineare, utilizzato per l'individuazione di soluzioni a problemi con funzione obiettivo o vincoli non lineari. I solutori utilizzati tramite il software AMPL [4].

Per l'identificazione delle sinusoidi sovrapposte ho definito un algoritmo di enumerazione implicita, che ha il compito di enumerare e valutare tutte le sinusoidi restituite dal modulo di interpolazione in corrispondenza di un assegnamento di osservazioni. L'algoritmo è stato implementato tramite il linguaggio Python versione 3.9.13 [10].

I test sono stati effettuati su processore Intel Core i7 quad-core 2,3 GHz con una memoria di 8GB 1600 MHz DDR3.

1.4 Risultati sperimentali e finali

Dai test sperimentali si è conseguito che il modulo di interpolazione implementato riesce ad individuare sinusoidi che interpolano i punti, con uno scostamento medio di 84.17s su periodi appartenenti all'intervallo (0, 5000]s in un tempo medio di 1.1s. Il modulo presenta difficoltà ad individuare soluzioni dal periodo accettabile nei casi nel quale le osservazioni siano affette da errore e poco distribuite lungo il periodo della sinusoide relativa ai punti.

Il modulo di identificazione di sinusoidi sovrapposte ha una correttezza di assegnamento delle osservazioni del 100% per casi di test da 10 e 20 osservazioni in tempi di rispettivamente 394.3s e 786.2s, mentre il caso di test da 30 osservazioni si individua

una correttezza degli assegnamenti del 88.8% con un tempo medio di 525.2s. I casi di test sono stati eseguiti per 10 iterazioni con punti generati da sinusoidi con periodo e ampiezza definiti randomicamente all'interno dell'insieme $(0, 5000]$, e $[1, 10]$

Capitolo 2

Descrizione del problema

Il problema è scomponibile in due sottoproblemi: uno di interpolazione e uno di assegnamento. La componente di interpolazione è identificata nell'operazione di individuazione della sinusoide che si scosta il meno possibile dai valori, interpolazione di una singola sinusoide. Questo definisce un problema di interpolazione non-lineare e continuo: non-lineare poiché una sinusoide è una funzione periodica non-lineare, definita da una trasformazione elementare della funzione seno, continuo poiché il dominio di una sinusoide e delle sue componenti è definito in \mathbb{R} o in un suo sottoinsieme.

La sinusoide che si vuole ottenere dall'interpolazione dei punti deve avere determinate caratteristiche. Tra tutte le sinusoidi possibili che interpolano gli n valori, si vuole quella di massimo periodo e sicché le misurazioni possono essere affette da errore si deve combinare la ricerca del massimo periodo con quella di minimo errore, definendo un problema a due obiettivi.

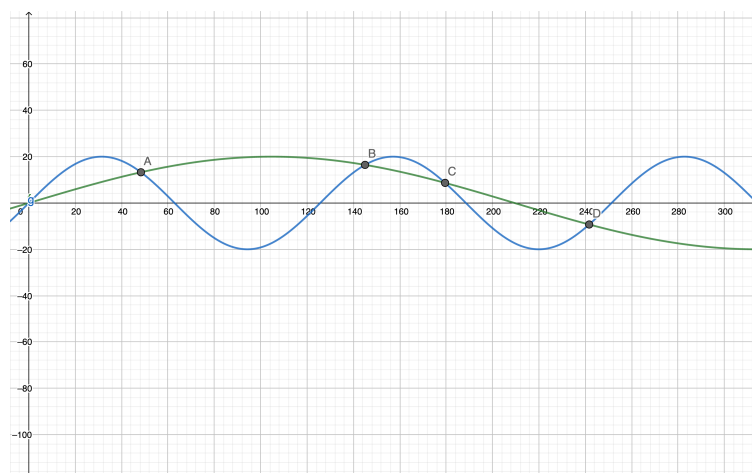
Il processo risolutivo per un problema a molti obiettivi in conflitto tra loro, cioè che il miglioramento di una comporti un peggioramento di un'altra, prevede:

1. Il calcolo della regione pareto-ottima, cioè l'insieme di soluzioni ammissibili non dominate, denominate anche soluzioni paretiane.
2. La scelta di una soluzione tra quelle individuate.

La determinazione della regione pareto-ottima può essere effettuata in diversi modi.

In questo processo di interpolazione è necessario fare attenzione alle sinusoidi con basso periodo rispetto alla sinusoide da individuare: si può osservare dalle figure 4, 5 e 6 che aumentando la frequenza, quindi al diminuire del periodo, esiste sempre un modo di interpolare n punti con una sinusoide minimizzando a piacimento l'errore di interpolazione, poiché i punti possono corrispondere o essere vicini alle intersezioni tra le sinusoidi.

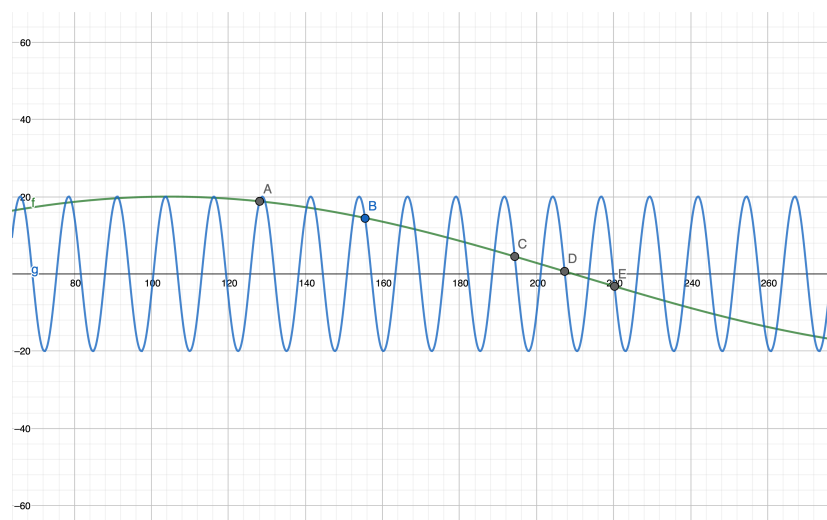
Figura 4: Problema sinuoidi da basso periodo rispetto alla sinusoide da individuare



Ascisse: Periodo (giorni)

Ordinate: Valore della sinusoide (")

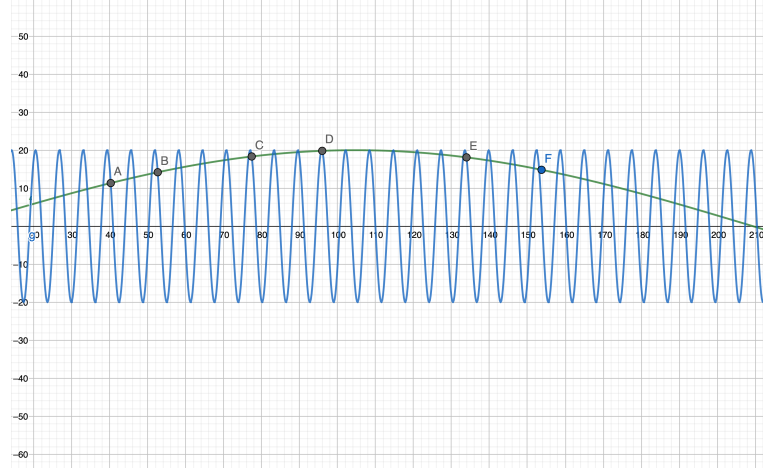
Figura 5: Problema sinuoidi da basso periodo rispetto alla sinusoide da individuare



Ascisse: Periodo (giorni)

Ordinate: Valore della sinusoide (")

Figura 6: Problema sinuoidi da basso periodo rispetto alla senoide da individuare



Ascisse: Periodo (giorni)

Ordinate: Valore della senoide (")

Parametri I dati a disposizione sono gli n punti assegnati da interpolare, il quale numero è determinato dal modulo di assegnamento. I punti sono individuati in un piano cartesiano dove l'asse delle ascisse rappresenta il tempo e l'asse delle ordinate l'angolo. I parametri del modello sono quindi le n ascisse t_i e le n ordinate e_i degli n punti, con i da 1 a n .

Variabili Le variabili sono individuate nelle componenti che determinano una senoide, data l'espressione analitica generale di una funzione sinusoidale:

$$e_i = A \sin(\omega t_i + \phi) \quad (1)$$

Dall'espressione ho definito le variabili che rappresentano fase ϕ , ampiezza A e la pulsazione ω .

Inoltre c'è la necessità di rappresentare l'errore che può essere generato dalle misurazioni o da un assegnamento di punti errato. L'espressione analitica diventa:

$$e_i = A \sin(\omega t_i + \phi) + \varepsilon_i \quad (2)$$

Perciò ho definito n variabili aggiuntive che rappresentano l'errore per ogni punto.

Nel definire le variabili si suppone che il valore medio v_{medio} (3) delle sinusoidi sia sempre zero, come nel caso delle sinusoidi generate dai satelliti. È da sottolineare

che v_{medio} potrebbe essere diverso da 0 in applicazioni differenti, determinando la tipologia di ricerca e gestione dell'ampiezza.

$$e_i = v_{medio} + A \sin(\omega t_i + \phi) + \varepsilon_i \quad (3)$$

Nonostante il possesso di tre variabili da calibrare, ampiezza, pulsazione e fase, il numero di sinusoidi che passano da tre punti è infinito, questo a causa della periodicità delle sinusoidi. Al contrario nel dominio delle rette, che con due variabili, rispettivamente coefficiente angolare e ordinata all'origine, il numero di rette che passano per due punti individuabili è solo uno. Perciò è necessario tramite una definizione accurata di un modello cercare di ridurre il numero di sinusoidi adatti al contesto.

La componente di assegnamento è individuata nel processo di selezione dei punti tra le diverse osservazioni che rappresentano una sinusoide, cioè l'identificazione di sinusoidi sovrapposte. La funzione di questa componente è quella di individuare k sinusoidi, generate dall'interpolazione di una assegnazione di valori.

Capitolo 3

Interpolazione di una sinusoide

Ancor prima di considerare il problema delle sinusoidi sovrapposte, è necessario studiare e analizzare il problema di identificazione di una singola sinusoide e quindi di ampiezza, pulsazione e fase. Per questo compito ho definito un modello e algoritmo di gestione e di selezione tra soluzioni generate, sperimentando il comportamento con osservazioni affette e non affette da errori.

3.1 Modello

I parametri e le variabili utilizzate nella specificazione dei seguenti modelli corrispondono a quelli definiti in nel capitolo 2.

Funzione obiettivo Le due funzioni obiettivo da ottimizzare sono:

$$\min f_1 = \omega \quad (4)$$

Minimizzazione della pulsazione

$$\min f_2 = \frac{\sum_{i=1}^n \varepsilon_i^2}{n} \quad (5)$$

Per la minimizzazione dell'errore complessivo, calcolo l'errore quadratico complessivo per l'eliminazione dei possibili valori negativi che si possono generare.

3.1.1 Calcolo della regione pareto-ottima

Metodo dei pesi

Il metodo dei pesi consiste nel dare un peso a ciascuna delle funzioni obiettivo, così da ridurre il problema a molti obiettivi in uno di programmazione matematica

parametrica.

Vincoli Ho definito come unico vincolo l'equazione (2)

I pesi definiti per le funzioni f_1 e f_2 sono rispettivamente 1 e -1, quindi ottenendo la funzione obiettivo:

$$\min f = f_1 + f_2 \quad (6)$$

Questa configurazione dei pesi di partenza permette di penalizzare tutte quelle soluzioni ammissibili che massimizzano il periodo ma hanno un errore elevato.

Il processo di individuazione delle soluzioni paretiane, con questo metodo e numero di funzioni, consiste nel eseguire un'analisi parametrica sui pesi, cioè analizzare e individuare soluzioni non dominate al variare dei pesi entro un certo range. In questo caso può essere sufficiente solo effettuare la variazione del peso dell'errore.

Un modello che dà priorità al periodo ottenendo un alto errore è per certo non utilizzabile, perché l'alto errore denota una completa mancanza di relazione tra i punti e la sinusoide individuata, questo spiega l'analisi parametrica solo sul peso dell'errore. È anche possibile non effettuare alcuna variazione dei pesi, ed accontentarsi della prima soluzione individuata tramite la configurazione dei pesi di partenza.

Metodo dei vincoli

Il metodo dei vincoli consiste nell'ottimizzare una delle funzioni obiettivo, trasformando le rimanenti in vincoli, utilizzando un termine noto parametrico.

Vincoli Si ha l'equazione (2) e la trasformazione in vincolo della funzione obiettivo legato al periodo (4)

$$\frac{2\pi}{\omega} \geq \beta \quad (7)$$

Il termine noto β sta ad indicare il minimo periodo ammissibile per una soluzione, questo permette di escludere le soluzioni a basso periodo rendendo inammissibili soluzioni con periodo più piccolo del termine noto.

Oppure se si applica il metodo dei vincoli sulla funzione obiettivo sull'errore (5) si ottiene:

$$\frac{\sum_{i=1}^n \varepsilon_i^2}{n} \leq \alpha \quad (8)$$

Il termine noto α rappresenta il massimo errore ammissibile per una soluzione.

Funzione obiettivo La funzione obiettivo definita è la funzione (5).

L'individuazione delle soluzioni paretiane è effettuata variando il termine noto parametrico nel vincolo (7). Ho scelto di definire il vincolo (7) perché permette di avere un controllo migliore sull'esplorazione dei periodi possibili della sinusoide da individuare. Un vincolo sull'errore sarebbe di difficile gestione dato dal fatto che esistono infinite sinusoidi che possono interpolare perfettamente, quindi riducendo l'errore, i punti relativi alle osservazioni.

3.2 Algoritmo

Il modello che ho scelto di implementare è il modello che utilizza il metodo dei vincoli per i vantaggi citati nella sua specificazione. Il metodo dei pesi non consente di individuare tutte le soluzioni paretiane per problemi discreti, possono esistere soluzioni paretiane che non sono ottime per alcuna scelta dei pesi. Una volta definito l'implementazione del modello è necessario specificare un algoritmo di controllo e di esecuzione di esso. Il modello applica il metodo dei vincoli nel seguente modo.

Si vincola il periodo ad appartenere ad un intervallo prefissato, l'intervallo nella quale si presume sia presente la sinusoide ricercata. Iterativamente si va ad effettuare una ricerca della sinusoide con il minimo errore e successivamente con il massimo errore. Lo scopo della ricerca del massimo errore è quella di modificare l'intervallo e delimitare quindi le regioni dei minimi locali all'interno dell'intervallo.

Mostrando un esempio di applicazione, se l'intervallo prefissato corrisponde a $[0, 10]s$ e la ricerca della sinusoide dal massimo periodo e massimo errore da' una sinusoide dal periodo pari a $9s$, il nuovo intervallo potrebbe corrispondere a $[0, 8]s$. Si fissa a $8s$ cosicché il periodo individuato diventi inammissibile per favorire una nuova ricerca.

Perciò i passi che caratterizzano l'algoritmo di controllo sono:

1. Ricerca della sinusoide dal massimo periodo e minimo errore.
2. Ricerca della sinusoide dal massimo periodo e massimo errore.
3. Aggiornamento del limite superiore o del limite inferiore dell'intervallo prefissato.

3.2.1 Criterio arresto

È necessario specificare un criterio di arresto per l'algoritmo di controllo. L'idea applicata è quella di determinare l'arresto fissando un periodo limite. Effettuando una ricerca sui satelliti del sistema solare, ho individuato che il massimo periodo di un satellite nel sistema solare è pari a 9374.0 giorni [2].

Quindi è possibile fissare un periodo limite, effettuando un mappaggio da giorni in secondi, pari a 5000s, quindi definendo un intervallo di $[0, 5000]$ s. Per maggiore accuratezza è possibile eseguire l'algoritmo di controllo all'interno di una serie di intervalli più piccoli all'interno dell'intervallo $[0, 5000]$ s.

Si continua ad eseguire l'algoritmo di controllo iterativamente finché l'intervallo non è sufficientemente piccolo, almeno un decimo o un centesimo dell'intervallo iniziale oppure se si individua nella ricerca del massimo errore una senoide dal periodo pari a uno dei due limiti dell'intervallo.

3.2.2 Punto di inizializzazione

Il processo di inizializzazione di un modello, consiste nel fissare le variabili del problema a dei valori di partenza. Generalmente definire una soluzione iniziale per il solutore nei problemi di programmazione non lineare è importante, poiché ne determina l'abilità di individuare un minimo locale diverso da un altro, e poter individuare un minimo locale migliore sotto una determinata caratteristica.

Ho deciso di fissare come punto di inizializzazione la metà dell'intervallo corrente nelle interazioni dell'algoritmo di controllo.

Questo permette di bilanciare la ricerca della senoide all'interno dell'intervallo, lasciando il compito di delimitare il minimo locale corrispondente alla senoide ricercata alla definizione iterativa degli intervalli.

3.2.3 Controllo del periodo

Una volta effettuata la ricerca del massimo periodo è necessario determinare il nuovo intervallo stabilendo la selezione dell'intervallo inferiore o superiore rispetto al periodo individuato.

Ad esempio se si effettua una ricerca del massimo periodo e massimo errore in un intervallo $[0, 10]$ s individuando una senoide dal periodo pari a 7s, è necessario determinare se il nuovo intervallo di ricerca della senoide dal minimo errore dev'essere interno a $[7, 10]$ s oppure $[0, 7]$ s.

La selezione può essere determinata andando ad effettuare una ricerca della senoide dal massimo periodo e minimo errore all'interno degli intervalli individuati, confrontando gli errori delle sinusoidi individuate.

Per rendere inammissibile la senoide dal massimo errore si fissa il nuovo limite sommando o sottraendo la metà della dimensione dell'intervallo al limite inferiore o superiore.

3.2.4 Controllo dell'ampiezza

In base ai valori delle osservazioni è possibile determinare un intervallo nella quale si presume che il valore dell'ampiezza sia situato, grazie alla periodicità delle funzioni sinusoidali.

Il valore minimo delle osservazioni determina il limite inferiore, il limite superiore è invece determinato effettuando uno studio dei valori dei punti, evidenziando due casi:

- Un andamento non monotono, in questo caso il valore massimo delle osservazioni corrisponde all'ampiezza della sinusoide, è possibile vincolare l'ampiezza ad essere inferiore al doppio di questo valore per un grado di incertezza.
- Un andamento monotono, in questo caso il valore massimo delle osservazioni potrebbe non corrispondere all'ampiezza perciò per precauzione è possibile moltiplicare il valore massimo per una costante sufficientemente grande.

Dalle sperimentazioni ho individuato 100 come un valore sufficiente per l'intervallo di periodi prefissato.

L'applicazione di questo vincolo permette di facilitare l'individuazione della sinusoide che interpola i punti, restringendo l'insieme delle possibili soluzioni.

3.2.5 Controllo della fase

È possibile vincolare la fase ad appartenere ad un intervallo $[\pi, -\pi]$, poiché all'esterno di tale intervallo gli effetti delle sinusoidi si ripetono ciclicamente.

3.2.6 Modifiche al modello

Per permettere il controllo del modello nelle modalità precedentemente spiegate è necessario apportare alcune modifiche, per poter avere il controllo sul periodo, sull'ampiezza e sulla fase.

Variabili

Le variabili rimangono invariate, si ha:

- ω rappresenta la pulsazione.
- A rappresenta l'ampiezza.
- θ rappresenta la fase.
- ϵ_i rappresenta l'errore per ogni punto.

Vincoli

I vincoli risultano essere i seguenti, per il periodo:

- $\omega \leq \max W$
- $\omega \geq \min W$

Dove $\max W$ e $\min W$ sono costanti che corrispondono rispettivamente ai valori di pulsazione del periodo limite inferiore e periodo limite superiore citati in 3.2 ed in 3.2.3.

Per l'ampiezza:

- $A > \min A$
- $A < \max A$

Dove $\min A$ e $\max A$ sono costanti corrispondenti all'analisi dei valori dei punti spiegato in 3.2.4.

Per la fase:

- $\theta > -\pi$
- $\theta < \pi$

Funzione obiettivo

Si aggiunge anche una funzione obiettivo di massimizzazione dell'errore attivato in maniera esclusiva con la funzione obiettivo di minimizzazione (5).

$$\max f_2 = \frac{\sum_{i=1}^n \varepsilon_i^2}{n} \quad (9)$$

Algoritmo 1: Algoritmo di controllo del modello

input : data Collezione di punti da interpolare
output: soluzioni insieme delle soluzioni individuate

```

1 maxPeriodo  $\leftarrow$  val1
2 minPeriodo  $\leftarrow$  val2
3 config  $\leftarrow$  valori di maxA, minA, e limiti per la fase
4 while (maxPeriodo – minPeriodo)  $\leq$  valore d'arresto do
5   | soluzioni  $\leftarrow$  minInterpolazione(data, config, maxPeriodo, minPeriodo)
6   | maxSoluzione  $\leftarrow$  maxInterpolazione(data, config, maxPeriodo,
7   |   minPeriodo)
8   | tempSol1  $\leftarrow$  minInterpolazione(data, config, maxPeriodo,
9   |   maxSoluzione[Periodo])
10  | tempSol2  $\leftarrow$  minInterpolazione(data, config, maxSoluzione[Periodo],
11  |   minPeriodo)
12  | if tempSol1[errore] < tempSol2[errore] then
13  |   | minPeriodo = maxSoluzione[Periodo]
14  | else
15  |   | maxPeriodo = maxSoluzione[Periodo]
16  | end
17  | soluzioni  $\leftarrow$  tempSol1
18  | soluzioni  $\leftarrow$  tempSol2
19 end
```

3.3 Risultati

3.3.1 Scelta del solutore

Per valutare i solutori, eseguo l'algoritmo di controllo per l'individuazione delle soluzioni di osservazioni con numero di punti pari a 10 e periodi interni all'intervallo fissato in 3.2.1, pari a $[0, 5000]$ s.

L'algoritmo di controllo viene eseguito iterativamente in intervalli più piccoli suddividendo in 5 intervalli il range $[0, 5000]$ s.

I risultati delle sperimentazioni sono illustrati nelle tabelle da 1 a 4. I parametri dell'algoritmo 1 per le seguenti sperimentazioni sono:

- valore d'arresto = 10

La valutazione dei solutori si basa sul tempo di computazione, il numero di soluzioni individuate nell'intorno di grandezza 100s, del periodo della soluzione ideale. I solutori utilizzati corrispondono a SNOPT [6], KNITRO [5], LGO [7], MINOS [8], LOQO [9], tutti solutori presenti in Ampl, linguaggio di modellazione [4].

Tabella 3: Prestazioni dei solutori: Sinusoide con $\omega = 0.005 \text{ rad/s}$

Solutore	n. soluzioni individuate	n. soluzioni nell'intorno	Tempo di calcolo (s)
SNOPT	20	4	0.8
KNITRO	19	4	0.8
LGO	14	0	6.3
MINOS	23	3	0.8
LOQO	23	7	0.8

Tabella 1: Prestazioni dei solutori: Sinusoide con $\omega = 0.8 \text{ rad/s}$

Solutore	n. soluzioni individuate	n. soluzioni nell'intorno	Tempo di calcolo (s)
SNOPT	17	2	0.74
KNITRO	20	2	1.2
LGO	17	0	6.7
MINOS	17	2	0.7
LOQO	23	4	0.8

Tabella 2: Prestazioni dei solutori: Sinusoide con $\omega = 0.0125 \text{ rad/s}$

Solutore	n. soluzioni individuate	n. soluzioni nell'intorno	Tempo di calcolo (s)
SNOPT	17	1	0.7
KNITRO	29	7	1.3
LGO	14	0	6.7
MINOS	26	6	0.9
LOQO	20	2	0.6

Tabella 4: Prestazioni dei solutori: Sinusoide con $\omega = 0.0013 \text{ rad/s}$

Solutore	n. soluzioni individuate	n. soluzioni nell'intorno	Tempo di calcolo (s)
SNOPT	23	1	0.9
KNITRO	23	1	1.1
LGO	17	0	8.2
MINOS	20	1	0.7
LOQO	23	0	0.8

I solutori che individuano mediamente più soluzioni nell'intorno definito sono LOQO e KNITRO con rispettivamente 3.25 e 3.5 soluzioni individuate. Questi due solutori si equivalgono anche per numero medio di soluzioni individuate con circa 22 soluzioni individuate, con KNITRO un po' più lento, 1.1s e LOQO in 0.8s in media.

Data la condizione di parità tra i due solutori ho deciso di confrontarli andando a confrontare le soluzioni più vicine alla soluzione ideale individuate dai solutori per ogni problema.

Le righe delle tabelle seguenti indicano rispettivamente le soluzioni individuate dai solutori per i problemi generati dalle sinusoidi con $A = 1$, $\theta = 0$ e

- $\omega = 0.8 \frac{\text{rad}}{\text{s}}$
- $\omega = 0.0125 \frac{\text{rad}}{\text{s}}$
- $\omega = 0.0050 \frac{\text{rad}}{\text{s}}$
- $\omega = 0.0013 \frac{\text{rad}}{\text{s}}$

Tabella 5: Prestazioni del solutore KNITRO

Pulsazione ($\frac{\text{rad}}{\text{s}}$)	Ampiezza	Fase (rad)	errore quadratico medio (m^2)
0.79	1.0	0	1.52e-7
0.0139	0.8	0	6.46e-11
0.0046	1.0	0	7.41e-14
0.0014	0.9	0	2.58e-16

Tabella 6: Prestazioni del solutore LOQO

Pulsazione ($\frac{rad}{s}$)	Ampiezza	Fase (rad)	errore quadratico medio (m^2)
0.79	1.0	0	7.00e−11
0.0125	0.9	0	2.20e−14
0.0039	1.2	0	6.91e−13
0.0018	0.7	0	2.4e−14

Al confronto il solutore KNITRO risulta ottenere risultati meno distanti in termini di pulsazione, ampiezza e fase.

3.3.2 Selezione della soluzione

La selezione di una soluzione tra le tante individuate determina un passo fondamentale per tutto il modulo del sottoproblema di interpolazione, poiché si potrebbe incorrere nel rischio di scartare la sinusoide che rappresenta esattamente uno dei k satelliti. I due metodi identificati per questo compito si basano su diverse ipotesi sulla sinusoide ideale e sulle caratteristiche di tutte le sinusoidi individuate, questi sono il criterio del punto di utopia, e criterio degli standard solo sull'errore.

Criterio del punto di utopia

Il punto di utopia è la soluzione, che nello spazio degli obiettivi, ha come coordinate i valori ottimi di ciascuno. Tipicamente questa soluzione non è ammissibile, quindi il criterio del punto di utopia si basa sul concetto di selezionare la soluzione individuata più vicina al punto di utopia nello spazio degli obiettivi. La distanza viene calcolata tramite la misura della lunghezza del segmento avente per estremi i due punti, e quindi tramite la distanza euclidea.




Nel problema di interpolazione, il punto di utopia corrisponde al punto avente per coordinate (5000, 0), con 5000s il valore massimo dei periodi delle sinusoidi fissato nella sezione 3.2.1, e 0 il minimo valore d'errore possibile.

Già a partire dalle sperimentazioni per 10 osservazioni, si può notare che questo criterio non sia adatto per il contesto. Nonostante siano individuate soluzioni meno distanti e più corrette rispetto alla sinusoide che ha generato i punti, questi non vengono selezionati.

La criticità principale è individuata nella mancata normalizzazione dei domini dei periodi e dell'errore. Questo causato da una disparità tra le distanze dei periodi e dell'errore, avvantaggiando soluzioni con lo stesso periodo del punto di utopia piuttosto che soluzioni con errore minore.

Un'altra criticità è sulla effettiva posizione del punto di utopia, cioè nel fatto che non esista, per questo problema, un effettivo valore del periodo migliore a priori a differenza dell'errore. Dato queste problematiche, ho optato per l'utilizzo di un altro criterio di selezione.

Legenda per i grafici da figura 7 a figura 10:

-  : soluzione individuata.
-  : soluzione individuata scelta dal criterio.
-  : punto di utopia.

La seguente tabella contiene i risultati dei test effettuati per 10 osservazioni con i punti generati con t che definito con gli interi dell'insieme $[1, 10]$ s, con sinusoida: $e(t) = \sin(\omega t)$

Tabella 7: Risultati delle sperimentazioni sul criterio del punto di utopia

Periodo del problema (s)	Periodo della soluzione selezionata (s)	Ampiezza della soluzione selezionata	errore quadratico medio (m^2)
7.8	4999	1	0.452
502.4	4132.9	8	1.0e−9
1256.0	4117.0	3	3.6e−12
4830.7	4674.6	1	3.7e−15

Tabella 8: Soluzioni individuate migliori delle soluzioni scelte dal criterio

Periodo del problema (s)	Periodo della soluzione selezionata (s)	Ampiezza della soluzione selezionata	errore quadratico medio (m^2)
7.8	11.4	1	0.261
502.4	502.5	0.9	7.0e−16
1256.0	1251.1	1	2.4e−15

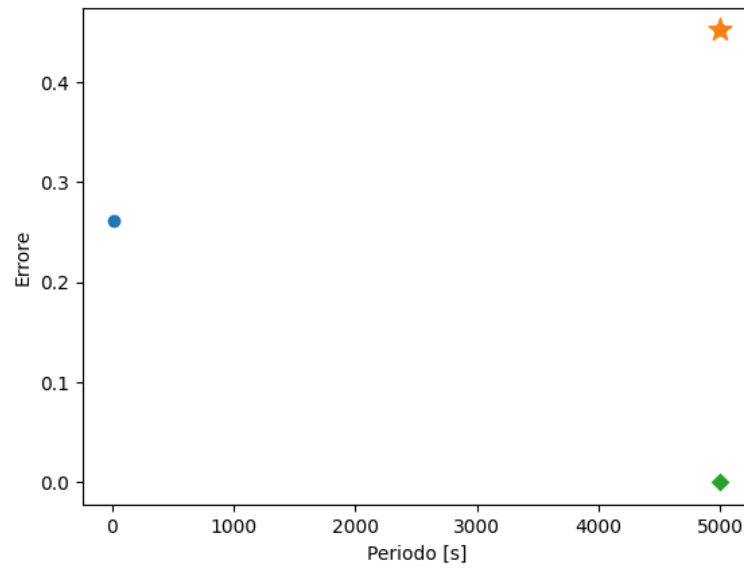


Figura 7: Regione paretiana per problema con periodo pari a 7.8s

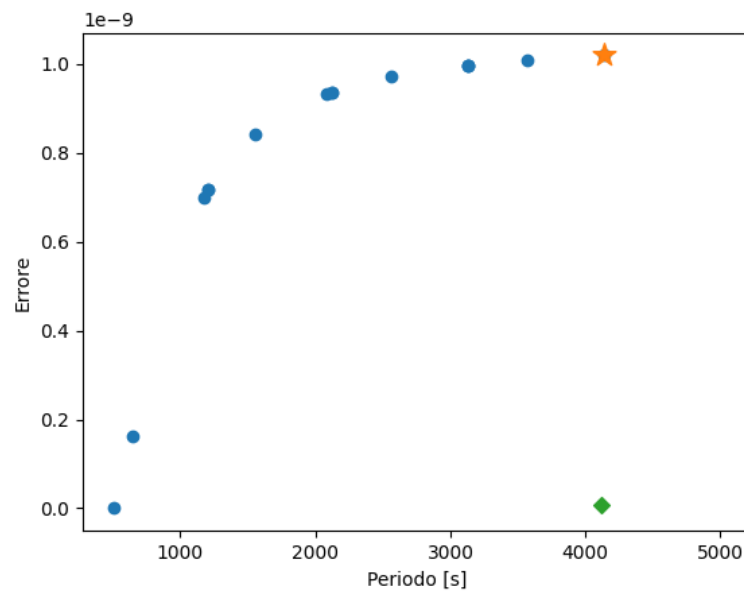


Figura 8: Regione paretiana per problema con periodo pari a 502.4s

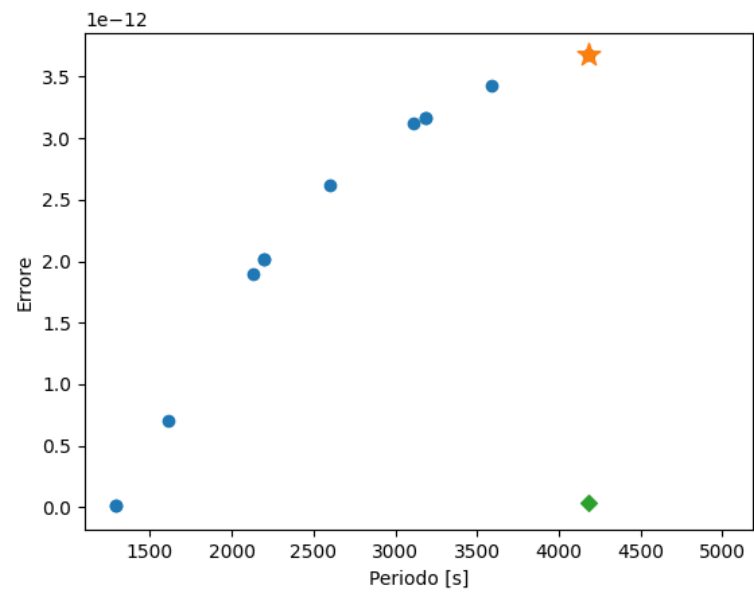


Figura 9: Regione paretiana per problema con periodo pari a 1256s

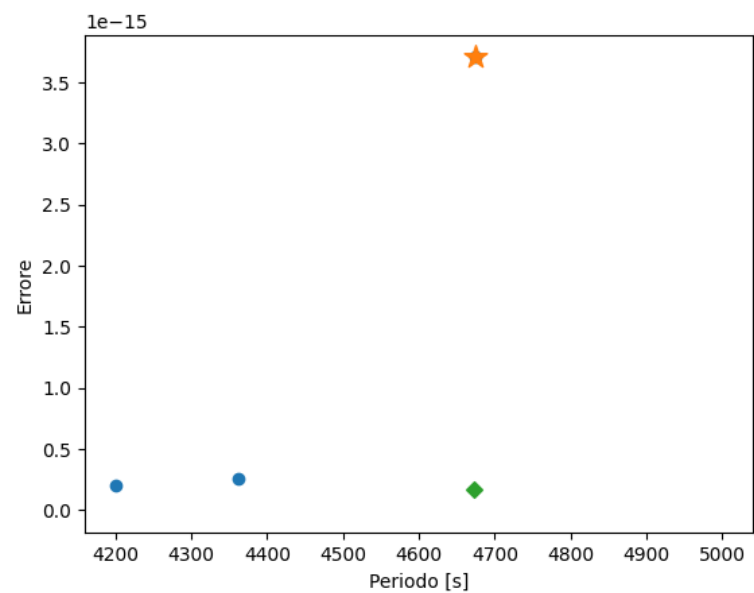


Figura 10: Regione paretiana per problema con periodo pari a 4830.7s

Criterio degli standard sull'errore

Gli standard sono valori-soglia al di sotto dei quali non si vuole che gli obiettivi possano peggiorare. Data la non predicibilità del periodo ottimale in questo problema, è possibile definire gli standard solo per l'errore. La modalità con il quale ho definito gli standard per la scelta della soluzione prevede la suddivisione in range del dominio dell'errore, ipotizzando che ogni soluzione in uno stesso range sia equivalente, se si considera solo l'errore. Ogni range è identificato con un numero ordinale specificandoli come livelli: primo livello, secondo livello, ... , n-esimo livello, dove per ordine crescente dei livelli si ha l'aumento del range d'errore. I range devono essere definiti a priori, in maniera attenta rispetto all'errore intrinseco dello strumento di rilevazione dei punti. È necessario inoltre effettuare ipotesi sul errore di una soluzione: errore di una soluzione ottimale, errore di una soluzione non correlata ai punti, errore di una soluzione legata ad un sottoproblema con un assegnamento di punti errato.

Il seguente pseudocodice illustra una procedura di una struttura dati che ho definito, per contenere la soluzione con livello più basso possibile e con più alto periodo tra tutte le soluzioni individuate all'interno del corrispettivo livello, perciò gli standard dell'errore sono fissati dalla soluzione con il livello minore.

Algoritmo 2: Criterio degli standard

```

input : solCorrente tupla contenente (errore, periodo, ampiezza, fase)
1 if LivelloDi(solCorrente) < LivelloDi(solOttimale) then
2   | solOttimale ← solCorrente
3 else
4   | if LivelloDi(solCorrente) = LivelloDi(solOttimale) then
5     |   if LivelloDi(solCorrente) = ultimoLv then
6       |     if solCorrente[errore] < solOttimale[errore] then
7         |       | solOttimale ← solCorrente
8       |     end
9     |   else
10      |     if solCorrente[periodo] > solOttimale[periodo] then
11        |       | solOttimale ← solCorrente
12      |     end
13    |   end
14  end
15 end

```

Tutte le soluzioni con un livello maggiore della soluzione ottimale vengono scartate (riga 1, algoritmo 2), invece se la soluzione corrente ha un livello inferiore, essa viene memorizzata come soluzione ottimale. Se i livelli della soluzione ottimale e della

soluzione corrente si equivalgono, si memorizza la soluzione corrente solo se essa ha un periodo maggiore della soluzione ottimale (riga 10, algoritmo 2).

Solo l'ultimo livello ha un criterio di selezione della soluzione ottimale differente, poiché si seleziona la soluzione con il minimo errore (riga 6, algoritmo 2). L'ultimo livello, rappresenta l'insieme dei valori d'errore più alti, che vanno da un valore fissato a $+\infty$, perciò all'interno di questo insieme illimitato, cerco di selezionare la soluzione che più si correla all'assegnamento dei punti.

Illustro i risultati dell'implementazione ed esecuzione del criterio degli standard.

I livelli fissati sono

- Livello 1: errore $< 1e-13$
- Livello 2: errore $< 1e-12$
- Livello 3: errore $< 1e-11$
- Livello 4: errore $< 1e-10$
- Livello 5: errore $< 1e-9$
- Livello 6: errore $< 1e-7$
- Livello 7: errore $< 1e-5$
- Livello 8: errore $< 1e-3$
- Livello 9: errore $< 1e-1$
- Livello 10: errore < 1
- Livello 11: errore ≥ 1

Ho fissato i range in maniera da ottenere maggiore precisione e diversificazione delle soluzioni con errore tra 0 e 1. Nella definizione dei livelli, non ho tenuto in considerazione l'errore intrinseco dello strumento di rilevazione, per ottenere delle prime informazioni sulla prestazione dell'algoritmo in una situazione ideale. A seguito si elencano problemi con punti generati da una sinusoide $e(t)$ e nelle tabelle le soluzioni individuate dall'algoritmo. Per ogni problema allego la regione paretiana corrispondente al livello della soluzione selezionata, cioè il livello migliore individuato, la regione paretiana non è presente solamente nei casi dove è stata individuata una singola soluzione non dominata.

Sperimentazioni sull'individuazione del periodo corretto

- $e(t) = \sin(0.0013t)$ sinusoide con periodo pari a: $T = 4830.7s$

Tabella 9: periodo da individuare uguale a 4830.7s

Numero di punti	Periodo della soluzione (s)	Tempo di calcolo (s)	Errore quadratico medio (m^2)
10	4361.3	1.1	$2.5e-16$
20	4624.3	1.5	$8.4e-16$
30	4731.7	1.0	$2.1e-15$

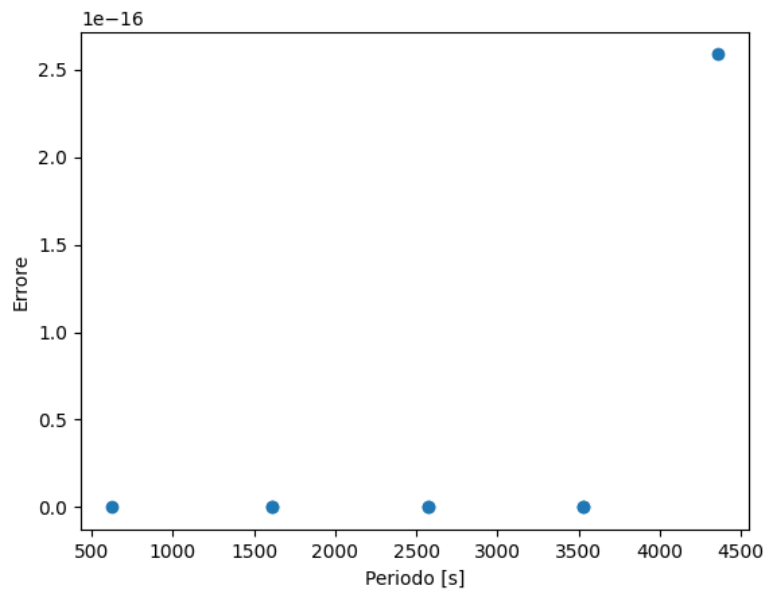


Figura 11: Regione paretiana per problema da 10 punti di tabella 9

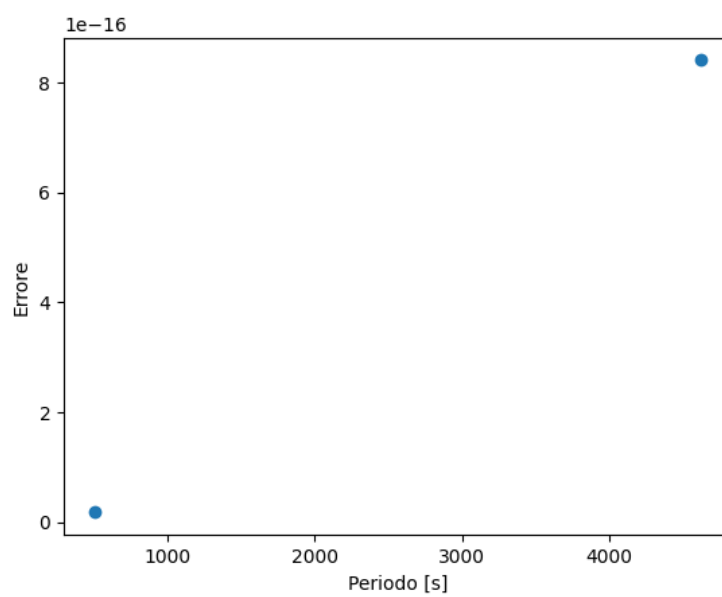


Figura 12: Regione paretiana per problema da 20 punti di tabella 9

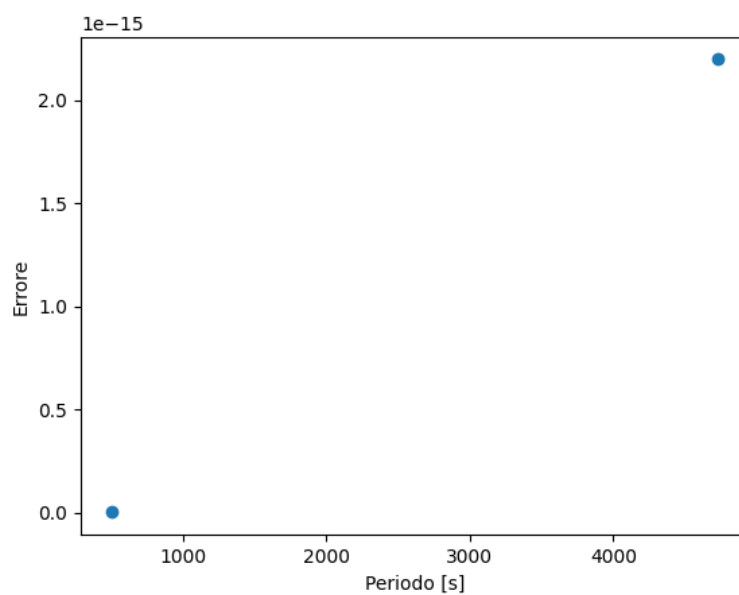


Figura 13: Regione paretiana per problema da 30 punti di tabella 9

- $e(t) = \sin(0.005t)$ sinusoide con periodo pari a: $T = 1256s$

Tabella 10: periodo da individuare uguale a 1256s

Numero di punti	Periodo della soluzione (s)	Tempo di calcolo (s)	Errore quadratico medio (m^2)
10	1426.7	0.7	$2.3e-13$
20	1349.7	1.4	$6.5e-12$
30	1251.1	1.3	$2.4e-15$

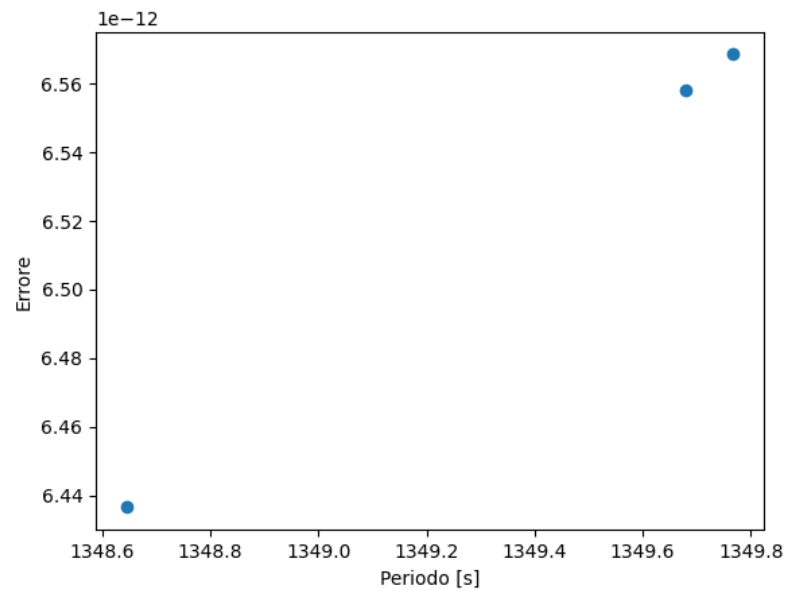


Figura 14: Regione paretiana per problema da 20 punti di tabella 10

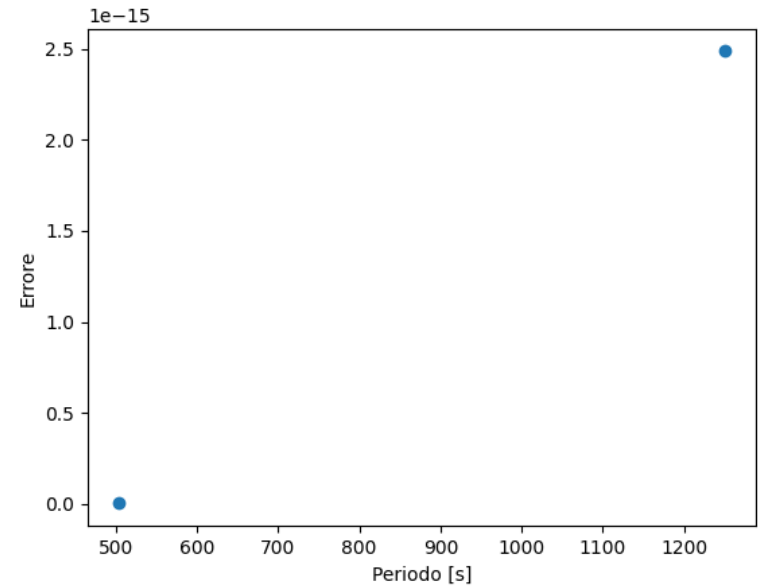


Figura 15: Regione paretiana per problema da 10 punti di tabella 10

- $e(t) = \sin(0.0125t)$ sinusoide con periodo pari a: $T = 502.4s$

Tabella 11: periodo da individuare uguale a 502.4s

Numero di punti	Periodo della soluzione (s)	Tempo di calcolo (s)	Errore quadratico medio (m^2)
10	449.9	1.3	6.4e-11
20	502.6	1.3	1.06e-18
30	502.6	1.3	3.0e-18

- $e(t) = \sin(0.8t)$ sinusoide con periodo pari a: $T = 7.85s$

Tabella 12: periodo da individuare uguale a 7.85s

Numero di punti	Periodo della soluzione (s)	Tempo di calcolo (s)	Errore quadratico medio (m^2)
10	7.8	1.1	$1.5e-7$
20	8.0	1.0	$1.1e-9$
30	7.8	1.3	$2.5e-7$

L'esecuzione dell'algoritmo è caratterizzato da uno scostamento medio dal periodo da individuare di 84.17s e una mediana di 28.8s ed infine un tempo medio di esecuzione di 1.1s e una mediana di 1.3s. Questo criterio di selezione della soluzione denota un netto miglioramento rispetto al criterio del punto di utopia.

Oltre alla valutazione rispetto allo scostamento del periodo delle sinusoidi sarebbe interessante valutare lo scostamento tra il numero di cicli della sinusode individuata rispetto a quella che ha generato i punti. Per far questo definisco una nuova serie di sinusoidi caratterizzati dall'avere periodi all'interno del numero delle osservazioni nel tempo e che quindi possa essere osservata almeno un periodo intero della sinusode ricercata.

I punti dei seguenti casi di test sono definiti con la seguente espressione:

$$e(t) = \sin(\omega t) \quad (10)$$

con t definito nell' sottoinsieme degli interi nell'insieme $(0, 30]s$, e quindi un numero di osservazioni pari a 30.

Periodo del problema (s)	Periodo della soluzione (s)	Numero di cicli completi del problema	Numero di cicli completi della soluzione
6.2	6.2	4	4
7.8	7.9	3	3
18.0	18.1	1	1
22.0	22.4	1	1

Da queste sperimentazioni si individua che lo scostamento tra il numero di cicli cioè il numero di periodi da $t = 0s$ a $t = 30s$ della sinusode del problema e la sinusode individuata è pari a 0.

Sperimentazioni sull'individuazione dell'angolo di fase corretto

Nelle successive sperimentazioni verifico il comportamento del solutore e criterio di selezione della soluzione al variare della fase. Verifico se il valore di angolo di fase individuato corrisponde a quella fissata in principalmente due casi, nel caso nel quale si ha almeno un periodo completo nelle osservazioni e nel caso contrario. Il numero delle osservazioni è pari a 30, mentre la fase varrà in ordine:

- $\phi = \frac{\pi}{3} \simeq 1.05$
- $\phi = \frac{\pi}{2} \simeq 1.57$
- $\phi = \frac{2\pi}{3} \simeq 2.09$
- $\phi = \frac{5\pi}{6} \simeq 2.61$
- $\phi = \pi \simeq 3.14$
- Sperimentazione con senoide corrispondente a: $e(t) = \sin(0.348t + \phi)$

Tabella 13: Verifica della fase con senoide con pulsazione di $0.348 \frac{rad}{s}$, cioè periodo di 18s

Pulsazione ($\frac{rad}{s}$)	Ampiezza	Fase (rad)	errore quadratico medio (m^2)
0.348	1.0	1.05	1.7e-16
0.348	1.0	1.57	3.8e-11
0.348	1.0	2.09	4.5e-11
0.348	1.0	2.61	4.9e-10
0.348	1.0	-3.14	1.6e-18

- Sperimentazione con senoide corrispondente a: $e(t) = \sin(0.0125t + \phi)$

Tabella 14: Verifica della fase con senoide con pulsazione di $0.0125 \frac{rad}{s}$, cioè periodo di 502.4s

Pulsazione ($\frac{rad}{s}$)	Ampiezza	Fase (rad)	errore quadratico medio (m^2)
0.0125	0.9	1.05	6.6e-14
0.0125	1.0	1.57	4.6e-12
0.0125	1.0	2.09	6.1e-18
0.0125	0.9	2.61	5.6e-16
0.0179	0.7	3.13	1.1e-06

Da come si può notare dai risultati delle sperimentazioni, viene individuato l'angolo di fase corretto, senza che essa vada ad influenzare l'individuazione dei valori di ampiezza e pulsazione corretti nella gran parte dei casi.

Sperimentazioni sull'individuazione dell'ampiezza corretta

In questi casi di test, verifico l'individuazione dell'ampiezza corretta per casi nel quale si ha un periodo completo e non, tra le osservazioni. Il numero delle osservazioni è pari a 30 e l'ampiezza varrà in ordine:

- $A = 2$
- $A = 4$
- $A = 8$
- $A = 10$
- Sperimentazione con senoide corrispondente a:

$$e(t) = A \sin(0.348t) \quad (11)$$

Tabella 15: Verifica dell'ampiezza con senoide con pulsazione di $0.348 \frac{rad}{s}$, cioè periodo di 18s

Pulsazione ($\frac{rad}{s}$)	Ampiezza	Fase (rad)	errore quadratico medio (m^2)
0.348	2.0	0	1.0e-17
0.348	4.0	0	4.3e-22
0.348	8.0	0	4.5e-14
0.348	10.0	0	1.5e-16

- Sperimentazione con senoide corrispondente a:

$$e(t) = A \sin(0.0125t) \quad (12)$$

Tabella 16: Verifica dell'ampiezza con senoide con pulsazione di $0.0125 \frac{rad}{s}$, cioè periodo di 502.4s

Pulsazione ($\frac{rad}{s}$)	Ampiezza	Fase (rad)	errore quadratico medio (m^2)
0.0125	1.9	0	2.2e−17
0.0125	4.0	0	3.4e−16
0.0125	7.9	0	1.4e−21
0.0125	9.9	0	7.16e−22

Dalle sperimentazioni effettuate si deduce che vengono individuati i valori di ampiezza corretti senza che essi vadano ad influenzare l'individuazione di periodo e fase corretti.

3.3.3 Sperimentazioni con osservazioni affette da errori

Le seguenti sperimentazioni sono state effettuate utilizzando il solutore KNITRO e il criterio degli standard, con il valore delle osservazioni $e(t)$ affette da un errore di al massimo del 10% tra le varie sperimentazioni e con un numero di punti pari a 10.

Lo scopo è verificare la robustezza del codice nell'individuare una soluzione accettabile, cioè con periodo, ampiezza e fase poco distanti da quella ideale nonostante il possibile errore sulle osservazioni.

L'algoritmo ha presentato diversi comportamenti in base alle caratteristiche del problema fornito.

Nei casi nel quale il numero di osservazioni copra gran parte del periodo delle sinusoidi che hanno generato i punti, nello specifico almeno la metà del periodo, viene individuata e selezionata una soluzione accettabile, nonostante la presenza di un errore di al massimo del 10% nelle osservazioni (tabelle 21, 22, 23).

Nei casi nel quale i periodi delle osservazioni siano molto maggiori del numero delle osservazioni raccolte nel tempo, nei casi d'errore del 5% e 10% risulta difficile al programma la selezione di una soluzione accettabile (tabelle 18, 19).

Questo accade poiché si combina la mancanza di informazioni a causa del numero esiguo di osservazioni rispetto ai periodi delle sinusoidi osservate, alla mancanza di precisione nei valori delle osservazioni generando soluzioni da un errore quadratico medio simile e difficilmente distinguibili dall'algoritmo di selezione di una soluzione.

Un miglioramento potrebbe essere quello di rendere dinamica la definizione dei livelli in base agli errori dell'insieme delle soluzioni individuate. Questo andrebbe ad adattare i livelli in maniera più granulare, nell'intervallo dei valori d'errore individuati, cosicché da suddividere le soluzioni individuate più efficacemente.

Un'altra soluzione è rilevare un maggior numero di osservazioni in base a ipotesi sul periodo totale degli elementi osservati, con almeno una copertura del 50% dei periodi ipotizzati, come nei casi di test di tabelle 20, 21, 22, 23.

- $e(t) = \sin(0.0013t)$ sinusoide con periodo pari a: $T = 4830.7s$

Tabella 17: periodo da individuare uguale a 4830.7s

Errore massimo nelle osservazioni	Periodo della soluzione (s)	Tempo di calcolo (s)	Errore quadratico medio (m^2)
1%	4674.6	1.09	$3.7e-15$
5%	4362.0	0.87	$1.1e-8$
10%	4646.8	0.9	$1.9e-7$

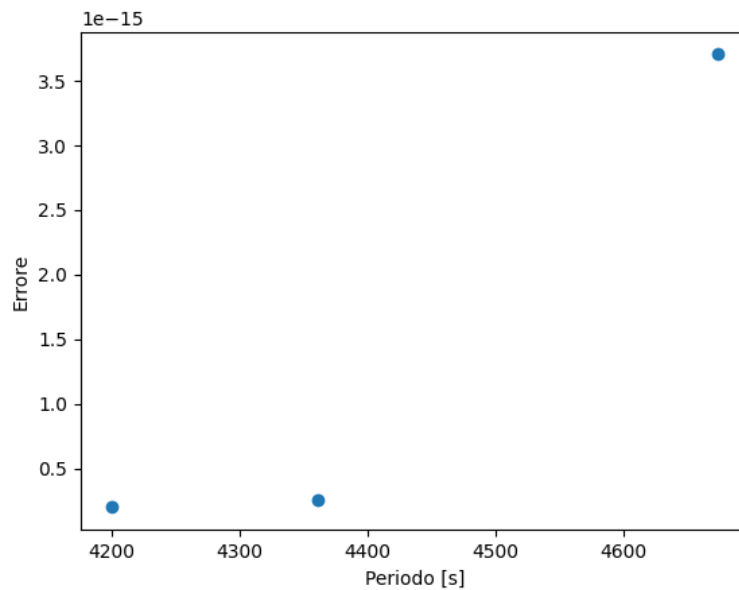


Figura 16: Regione paretiana per problema con errore del 1% punti di tabella 17

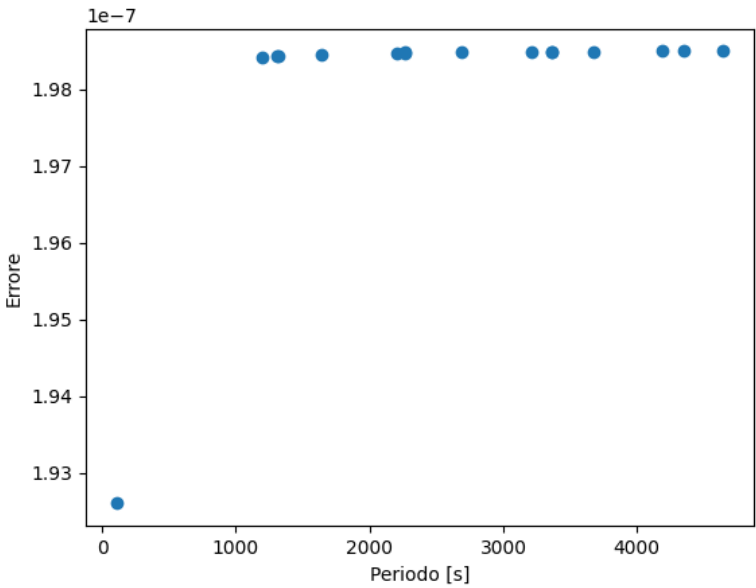


Figura 17: Regione paretiana per problema con errore del 10% punti di tabella 17

- $e(t) = \sin(0.005t)$ sinusoide con periodo pari a: $T = 1256s$

Tabella 18: periodo da individuare uguale a 1256s

Errore massimo nelle osservazioni	Periodo della soluzione (s)	Tempo di calcolo (s)	Errore quadratico medio (m^2)
1%	1286.6	0.83	1.3e-14
5%	4183.0	1.06	4.6e-7
10%	4174.9	1.07	1.2e-6

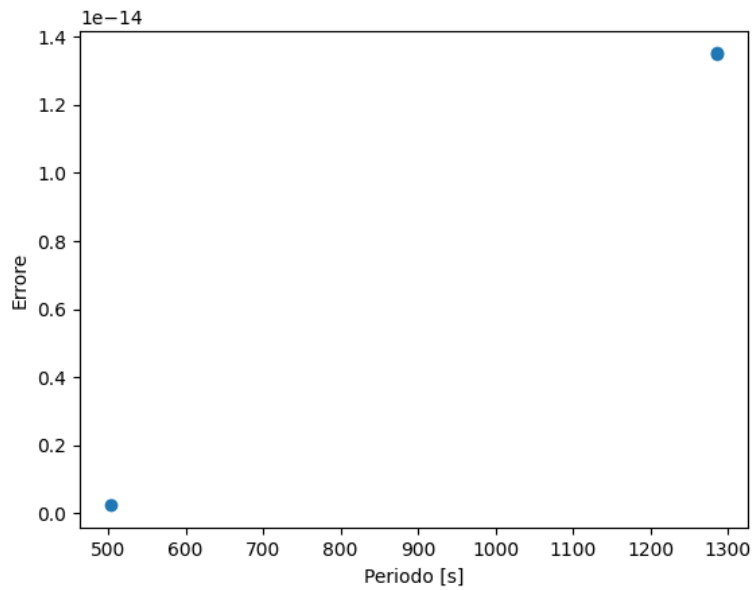


Figura 18: Regione paretiana per problema con errore del 1% punti di tabella 18

- $e(t) = \sin(0.0125t)$ sinusoide con periodo pari a: $T = 502.4s$

Tabella 19: periodo da individuare uguale a 502.4s

Errore massimo nelle osservazioni	Periodo della soluzione (s)	Tempo di calcolo (s)	Errore quadratico medio (m^2)
1%	502.5	1.09	2.4e-16
5%	4133.9	1.03	3.0e-6
10%	4230.0	0.9	1.0e-5

- $e(t) = \sin(0.8t)$ sinusoide con periodo pari a: $T = 7.85s$

Tabella 20: periodo da individuare uguale a 7.85s

Errore massimo nelle osservazioni	Periodo della soluzione (s)	Tempo di calcolo (s)	Errore quadratico medio (m^2)
1%	11.4	0.96	0.2
5%	7.8	0.95	1.9e-4
10%	0.8	0.89	3.1e-3

- $e(t) = \sin(t)$ sinusoide con periodo pari a: $T = 6.28s$

Tabella 21: periodo da individuare uguale a 6.28s

Errore massimo nelle osservazioni	Periodo della soluzione (s)	Tempo di calcolo (s)	Errore quadratico medio (m^2)
5%	6.27	0.5	3.7e-4
10%	6.3	0.3	4.4e-4

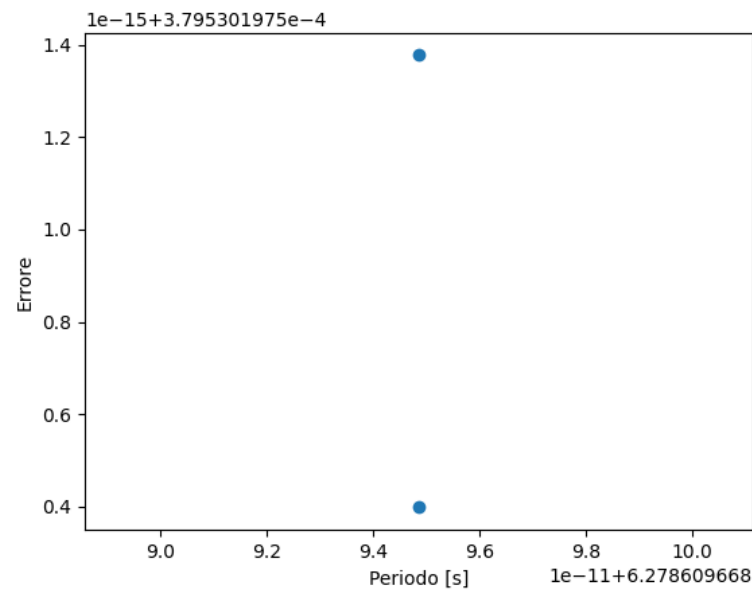


Figura 19: Regione paretiana per problema con errore del 5% punti di tabella 21

- $e(t) = \sin(0.348t)$ sinusoide con periodo pari a: $T = 18s$

Tabella 22: periodo da individuare uguale a 18s

Errore massimo nelle osservazioni	Periodo della soluzione (s)	Tempo di calcolo (s)	Errore quadratico medio (m^2)
5%	18.1	1.8	2.7e-4
10%	18.2	0.4	1.1e-3

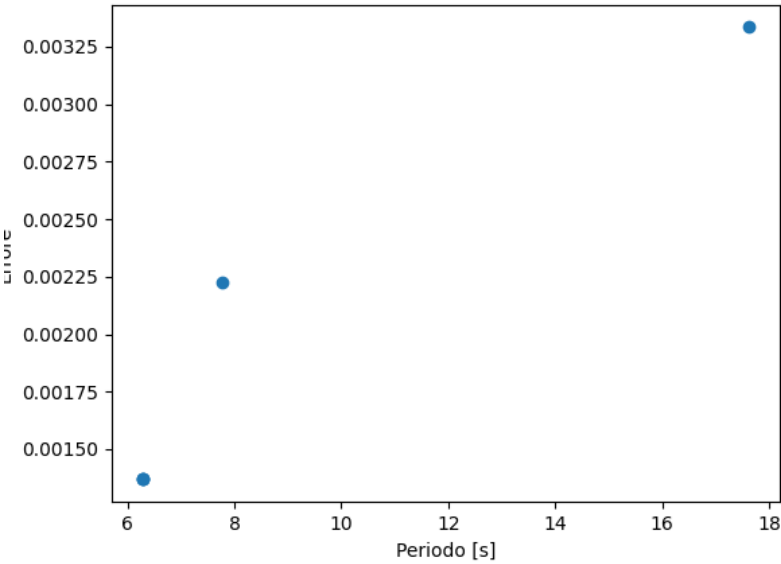


Figura 20: Regione paretiana per problema con errore del 10% punti di tabella 22

- $e(t) = \sin(0.028t)$ sinusoide con periodo pari a: $T = 22s$

Tabella 23: periodo da individuare uguale a 22s

Errore massimo nelle osservazioni	Periodo della soluzione (s)	Tempo di calcolo (s)	Errore quadratico medio (m^2)
5%	22.4	0.4	1.0e-4
10%	22.4	0.38	1.2e-3

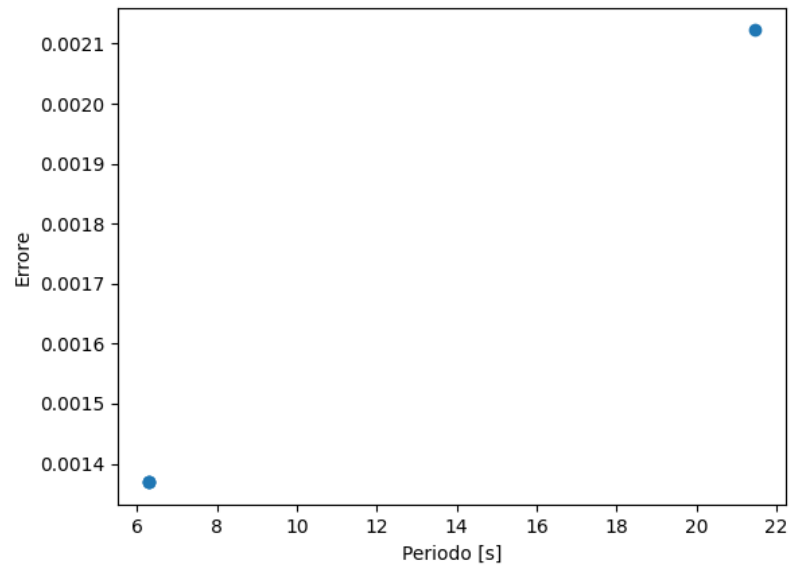


Figura 21: Regione paretiana per problema con errore del 10% punti di tabella 23

Capitolo 4

Identificazione di sinusoidi sovrapposte

Date n osservazioni composti da k valori, dove k è il numero di sinusoidi, l'algoritmo di assegnamento ha il compito di definire e valutare gli assegnamenti di n valori, presi da n osservazioni differenti. Lo scopo è di individuare gli assegnamenti che identificano le sinusoidi.

4.1 Metodo

Il concetto alla base è quello di generare tutte le permutazioni possibili delle n osservazioni, e di valutarle tramite i parametri restituiti dal modulo di interpolazione. Il numero delle permutazioni è pari a $(k!)^n$. Per generare le permutazioni viene utilizzato l'algoritmo di Heap [11].

La permutazione di una singola osservazione produce $k!$ elementi, ognuno di questi elementi deve essere assegnato a tutti i $k!$ elementi delle successive osservazioni fino alla n -esima osservazione:

$$\prod_{i=1}^n k! \quad (13)$$

Il numero delle permutazioni evidenzia il costo estensivo del processo e comporta l'applicazione di metodologie atti a ridurre il numero di permutazioni valutate per l'individuazione degli assegnamenti corretti in tempo utile. Non si effettua quindi una ricerca esaustiva, ma l'obiettivo è individuare e fermarsi alla migliore soluzione individuata, sotto un determinato criterio.

Per queste motivazioni, ho implementato un algoritmo divide et impera, che divide il problema di assegnamento in sottoproblemi dal numero di osservazioni minore e

unisce i risultati ottenuti nei vari nodi fino ad ottenere la soluzione del problema originario.

4.1.1 Creazione dei sottoproblemi

Per la generazione dei sottoproblemi, sfrutto la tecnica della ricorsione, generando un albero di chiamate dove ogni nodo corrisponde ad un sottoproblema. I nodi figli sono definiti in un nodo, suddividendo a metà il numero di osservazioni dati in input. Si effettua questo processo finché il numero di osservazioni in un nodo è minore o uguale ad un numero di osservazioni fissato, determinando le foglie.

Una volta che le foglie individuano una assegnazione dei punti ritenuta corretta, questa viene restituita al nodo padre che la unirà con la soluzione individuata dall'altro nodo figlio.

Algoritmo 3: Generatore Albero

```

input : valori Matrice contenente n osservazioni da k valori
output: Struttura dati contenente n osservazioni da k valori
1 nOss  $\leftarrow$  NumeroOsservazioni(valori)
2 if nOss  $\leq$  dimensioneFoglia then
3   | return OttimizzaFoglia(valori)
4 end
5 return OttimizzaNodo(GeneraAlbero(valori[0 : nOss/2]),
   GeneraAlbero(valori[nOss/2 : nOss]))

```

Per ogni osservazione si hanno k valori, i valori delle differenti osservazioni vengono assegnati tramite il modulo di interpolazione. Per il contenimento degli assegnamenti effettuati nei nodi viene utilizzata una matrice come illustrato in tabella 24, dove ogni colonna contiene gli assegnamenti, e le righe sono corrispettive alle osservazioni.

Tabella 24: Esempio di matrice utilizzata per contenere gli assegnamenti dei valori osservati dalle k sinusoidi

1° assegnamento	2° assegnamento	...	k-esimo assegnamento
val	val	...	val
val	val	...	val
val	val	...	val
val	val	...	val

Per "val" si intende un valore generico osservato dalle sinusoidi

4.1.2 Gestione dei nodi

Il compito di un nodo è quello di unire gli assegnamenti individuati nei nodi figli.

Gli assegnamenti vengono restituiti come matrici dove le colonne rappresentano gli assegnamenti dei valori individuati in una serie di osservazioni per una specifica sinusoide. Mentre le righe le osservazioni che sono definite da k valori con k il numero di sinusoidi come illustrato in tabella 24.

L'unione degli assegnamenti individuati nei due nodi figli avviene tramite la permutazione delle colonne di una delle due matrici, unendo per le colonne, la matrice non permutata con la matrice permutata.

Le colonne della matrice risultante sono valutate ad una ad una tramite il modulo di interpolazione, non appena si individua una colonna dall'errore con livello inferiore ad un livello soglia (riferimento ai livelli dell'algoritmo di selezione di una soluzione in 3.3.2), la si memorizza nella matrice da restituire, togliendo le colonne che compongono la colonna memorizzata dalla matrice da permutare e dalla matrice non permutata.

Nel caso non si trovi una colonna che rispetti il criterio di selezione, si seleziona la colonna con errore dal livello minore individuata nel corso di tutte le permutazioni.

Si effettuano queste operazioni finché non rimane una sola colonna nella matrice da permutare e nella matrice non permutata.

Il caso migliore è identificato nella individuazione della colonna ottimale nella prima iterazione delle permutazioni, dall'inizio fino ad ogni rimozione di colonna, dove l'algoritmo analizza $k - 1$ permutazioni, k rappresenta il numero di colonne. Il caso peggiore è identificato nell'individuazione della colonna ottimale nell'ultima iterazione delle permutazioni dall'inizio fino ad ogni rimozione di colonna, dove l'algoritmo analizza $\sum_{i=0}^k (k-i)!$ permutazioni. Viene effettuata una ricerca esaustiva unicamente nel caso peggiore. Se si valutassero le colonne di una permutazione complessivamente, quindi senza rimozioni di colonne, sarebbe necessario valutare tutte le permutazioni, l'algoritmo analizzerebbe $k!$ permutazioni.

Ho scelto di applicare l'operazione di rimozione per i vantaggi portati dal suo caso migliore e dal fatto che il caso peggiore non abbia un aumento così drastico delle permutazioni da analizzare.

Algoritmo 4: Gestione nodo

```

input : val1 Matrice contenente  $j$  osservazioni da  $k$  valori
        val2 Matrice contenente  $j$  osservazioni da  $k$  valori
output: Struttura dati contenente  $2j$  osservazioni da  $k$  valori
1 permutazioni  $\leftarrow$  GeneraPermutazioni(val2)
2 colonnaMigliore  $\leftarrow$  [ ]
3 while True do
4   for perm in permutazioni do
5     unione  $\leftarrow$  Unisci(val1, perm)
6     for col in unione do
7       sinusoide  $\leftarrow$  IndividuaSinusoide(col)
8       if LivelloDi(sinusoide)  $\leq$  livelloSoglia then
9         rimuovi le colonne di val1 e val2 che compongono col
10        rimuovi col da unione
11        permutazioni  $\leftarrow$  GeneraPermutazioni(val2)
12        aggiungi col ad assegnamentoOttimale
13        if NumeroColonne(unione) = 1 then
14          rimuovi le colonne di val1 e val2 che compongono la colonna
15          rimasta in unione
16          aggiungi la colonna rimasta in unione ad
17          assegnamentoOttimale
18          return assegnamentoOttimale
19        end
20        break
21      end
22      if LivelloDi(sin)  $\leq$  LivelloDi(colonnaMigliore) then
23        | colonnaMigliore  $\leftarrow$  col
24      end
25 end

```

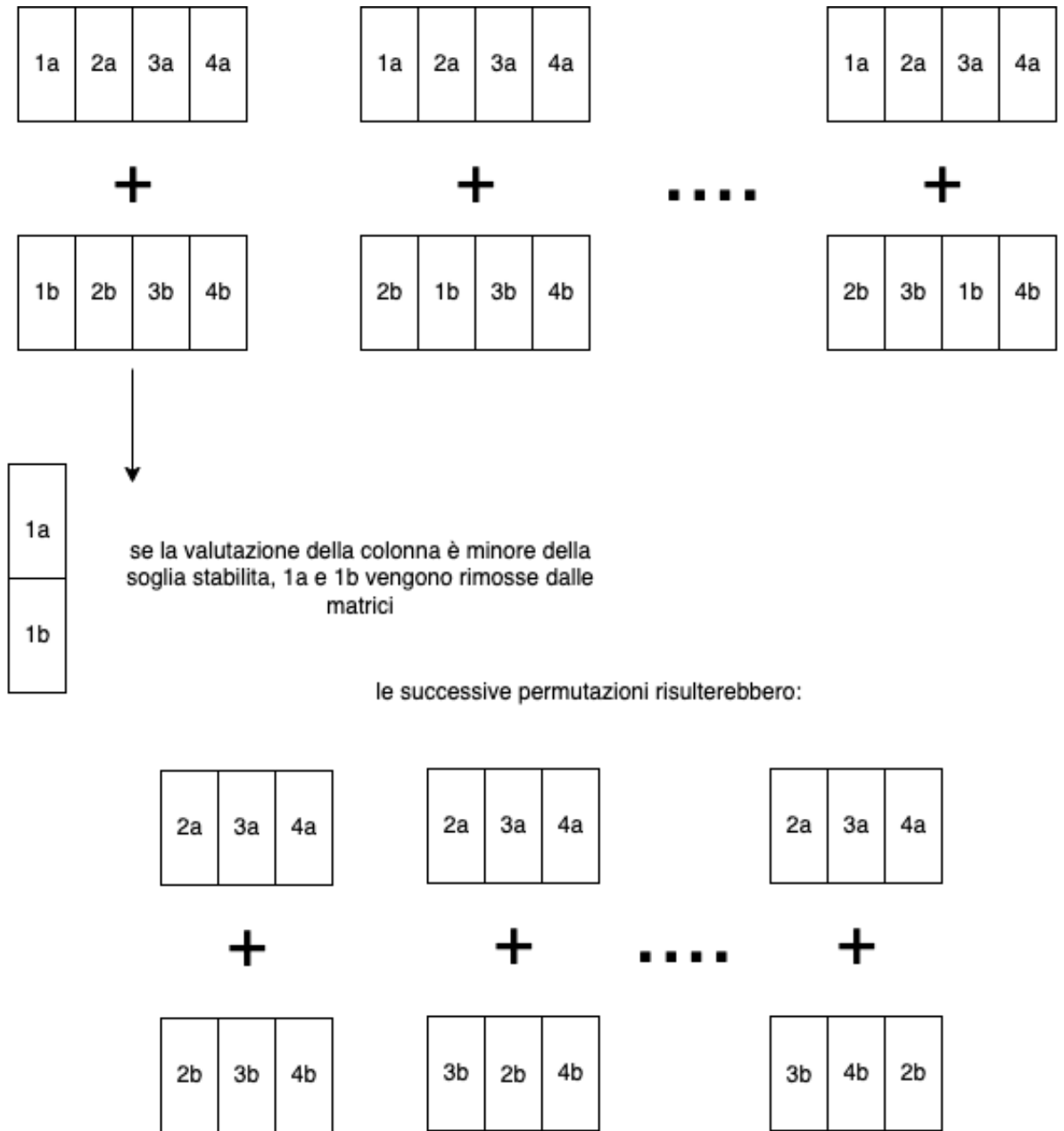


Figura 22: Esempio di valutazione delle osservazioni in un nodo

In figura 22, si illustra un esempio di valutazione delle osservazioni in un nodo dove ogni cella rappresenta la colonna rispettivamente da 1a ad 4a della prima matrice non permutata e 1b a 4b della seconda matrice permutata.

4.1.3 Gestione delle foglie

Le foglie sono caratterizzati da un numero di osservazioni minore o uguale ad un determinato valore (riga 2, algoritmo 3). Questo valore determina il numero di foglie che verranno generati, per esempio con foglie di dimensioni pari a 10 osservazioni su un totale di 20, si hanno $\frac{20}{10} = 2$ foglie, oppure con foglie di dimensione pari a 5, si hanno $\frac{20}{5} = 4$ foglie. Per ridurre il numero di permutazioni valutate, l'obiettivo è quello di generare quante più foglie possibili.

Nella determinazione del numero di osservazioni massimo per una foglia è necessario tenere a mente anche il modulo di interpolazione: non è fattibile o è di difficile esecuzione l'interpolazione di una sinusoidale a partire da pochi punti, come un solo punto o due.

Per la specificazione della dimensione dei nodi foglia, ho definito l'algoritmo 5, che fissa il parametro al maggior numero primo che divide il numero di osservazioni totale, scegliendo solo numeri primi maggiori di 2. Nel caso il numero di osservazioni sia un numero primo si valuta il numero precedente ad esso.

Il processo eseguito è quello descritto nella sezione 4.1, con la differenza che si ha una sola matrice, e che si effettua la permutazione dei valori per ogni riga. Si generano le permutazioni di ogni punto per ogni osservazione, effettuando le stesse operazioni di rimozione dei nodi. Non appena si individua una colonna dall'errore con livello inferiore ad un livello soglia (riferimento ai livelli dell'algoritmo di selezione di una soluzione in 3.3.2), la si memorizza nella matrice da restituire, togliendo la colonna dalla matrice da permutare, queste operazioni vengono effettuate finché non rimane una singola colonna nella matrice.

Il caso migliore corrisponde all'individuare una colonna ottimale alle prime permutazioni fino ad ogni rimozione, quindi il numero di permutazioni valutate corrisponde a $k - 1$, mentre il caso peggiore corrisponde all'individuazione della colonna ottimale nell'ultima iterazione delle permutazioni dall'inizio fino ad ogni rimozione di colonna, dove l'algoritmo analizza $\sum_{i=0}^{k-1} ((k - i)!)^n$ permutazioni. Nel caso non si effettuino operazioni di rimozione, il numero totale di permutazioni valutate corrisponde a $(k!)^n$. Come nella gestione dei nodi non foglia 4.1.2, per i vantaggi portati dal caso migliore, e un aumento non significativo delle permutazioni nel caso peggiore, ho scelto di applicare le operazioni di rimozione.

Algoritmo 5: Definizione della dimensione dei nodi foglia

input : nOsservazioni numero di osservazioni
output: dimFoglia numero di osservazioni massimo nei nodi foglia

```

1 dimFoglia  $\leftarrow$  0
2 if nOsservazioni è primo then
3   | nOsservazioni  $\leftarrow$  nOsservazioni  $-$  1
4 end
5 for  $i$  in [3, nOsservazioni] do
6   | if  $i$  è primo ed è un divisore di nOsservazioni then
7     |   dimFoglia  $\leftarrow$   $i$ 
8     | end
9   end
10 return dimFoglia

```

Algoritmo 6: Gestione nodo foglia

input : val Matrice contenente j osservazioni da k valori
output: assegnamentoOttimale Struttura dati contenente j osservazioni da k valori

```

1 permutazioni  $\leftarrow$  GeneraPermutazioni(val)
2 while True do
3   for  $perm$  in permutazioni do
4     for  $col$  in  $perm$  do
5       | sinusoide  $\leftarrow$  IndividuaSinusoide( $col$ )
6       | if LivelloDi(sinusoide)  $\leq$  livelloSoglia then
7         |   rimuovi  $col$  da  $perm$ 
8         |   permutazioni  $\leftarrow$  GeneraPermutazioni( $perm$ )
9         |   aggiungi  $col$  ad assegnamentoOttimale
10        |   if NumeroColonne( $perm$ ) = 1 then
11          |     aggiungi la colonna rimasta in  $perm$  ad
12            |     assegnamentoOttimale
13            |     return assegnamentoOttimale
14          |   end
15          |   break
16        |   end
17      end
18 end

```

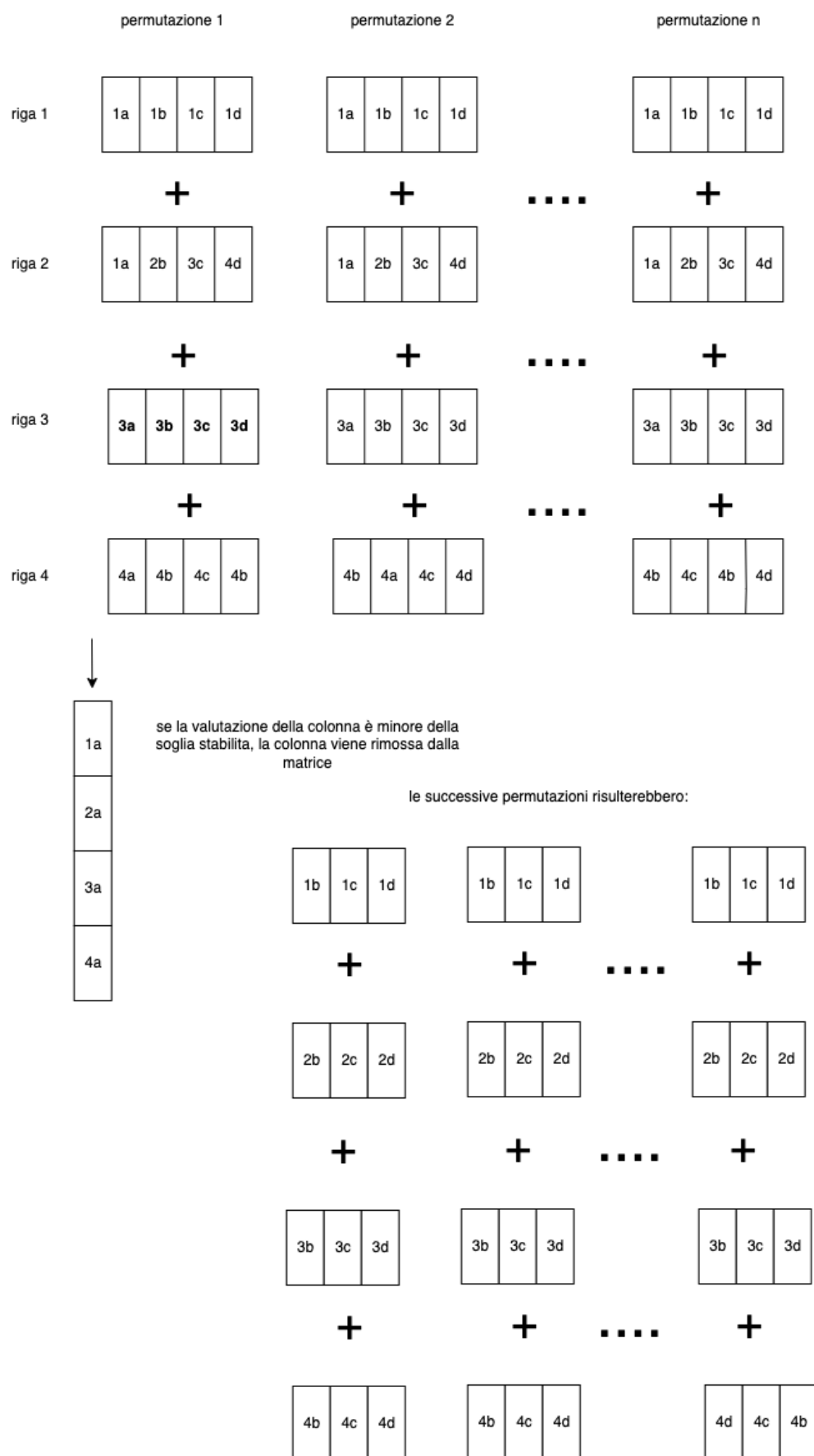


Figura 23: Esempio di valutazione delle osservazioni in una foglia

In figura 23 si illustra un esempio di valutazione in una foglia nel quale ogni cella rappresenta un singolo valore di una delle osservazioni.

4.2 Valutazione delle prestazioni

La valutazione delle prestazioni del modulo di assegnamento si concentra sull'analisi ed esame dei tempi di calcolo complessivi, tempo medio di calcolo per nodo, tempo medio di calcolo per foglia e percentuale di punti correttamente assegnati.

I punti in input sono riordinati in modo randomico utilizzando il modulo `random` di python perciò definito un caso di test, è necessario eseguire per un numero fissato di volte il modulo di assegnamento. In modo da recuperare ed analizzare il comportamento del modulo indipendentemente dalle caratteristiche dell'assegnamento iniziale in ingresso. Il numero di iterazioni fissato è pari a 10.

Le sinusodi sono definite con parametri stabiliti in maniera randomica utilizzando il modulo `random` di python ad ogni esecuzione. L'ampiezza viene definita da un insieme $[1, 10]$ mentre il periodo dall'insieme $[1, 5000]$ s. Questo permette di sperimentare e verificare il comportamento del modulo di assegnamento in ogni caso possibile e non basare considerazioni ed analisi su sinusodi specifiche, che potrebbero condizionare la bontà dei risultati.

I casi di test utilizzati si differenziano per numero di osservazioni. Il numero di osservazioni permette di verificare il comportamento dell'algoritmo di definizione del numero di osservazioni massimo nelle foglie, algoritmo 5 e la differenza in prestazioni dei nodi rispetto alle foglie e viceversa. I tre casi di test si differenziano in numero di osservazioni, che corrispondono rispettivamente a 10, 20, 30.

Tabella 26: caso di test 20 osservazioni

iterazione	tempo di esecuzione medio (s)	tempo medio nodi (s)	tempo medio foglie (s)	correttezza assegnamento (%)
1	607	5.1	148.0	100
2	537	5.5	130.2	100
3	745	14.1	175.8	100
4	444	3.9	108.0	100
5	837	11.9	200.3	100
6	435	6.0	104.4	100
7	1719	2.7	427.8	100
8	498	8.5	118.2	100
9	958	3.5	236.9	100
10	1082	6.2	265.9	100
media	786.2	6.7	191.5	100

Tabella 25: caso di test 10 osservazioni

iterazione	tempo di esecuzione (s)	tempo medio nodi (s)	tempo medio foglie (s)	correttezza assegnamento (%)
1	440	4.7	217	100
2	287	4.7	141	100
3	278	25.7	126	100
4	591	14.3	288	100
5	489	1.7	244	100
6	638	1.9	318	100
7	358	10.5	174	100
8	215	1.7	106	100
9	515	1.7	256	100
10	132	1.8	65	100
media	394.3	6.86	193.5	100

Il numero di permutazioni da valutare nelle foglie può essere molto maggiore rispetto al numero di permutazioni valutate nei nodi. I risultati delle sperimentazioni evidenziano ciò, la maggior parte del tempo di calcolo è impiegato nell'elaborazione delle foglie.

Tabella 27: caso di test 30 osservazioni

iterazione	tempo di esecuzione medio (s)	tempo medio nodi (s)	tempo medio foglie (s)	correttezza assegnamento (%)
1	585	14.6	60.3	91.1
2	493	9.1	53.6	95.5
3	703	15.2	74.5	92.2
4	798	16.7	85.1	96.6
5	810	6.4	95.5	95.5
6	576	16.2	57.8	65.5
7	364	7.0	39.4	97.7
8	369	4.6	41.9	95.5
9	259	12.1	21.6	65.5
10	295	11.5	26.8	93.3
media	525.2	11.3	55.6	88.8

Il tempo di esecuzione complessivo è fortemente influenzato dal numero di foglie definite e dal numero di osservazioni analizzate in esse. Ad esempio nel caso di test di tabella 25 e 26, la dimensione delle foglie può essere minore o uguale a 5 osservazioni e in quei casi di test corrispondo esattamente al valore fissato, poiché il quoziente della divisione tra numero di osservazioni totali e la dimensione massima delle foglie è un multiplo di 2, figura 24. Nel caso di test di tabella 25 si generano due foglie mentre in quello di tabella 26 si generano quattro foglie, perciò a parità di dimensione di foglia, al raddoppiare del numero di foglie, il tempo medio raddoppia.

La differenza tra il caso di test dei risultati di tabella 27 dai casi di test di tabella 25 e 26 è che seppur la dimensione massima delle foglie sia uguale, le foglie definite nel caso di test da 30 osservazioni sono pari a quattro o tre osservazioni, figura 25. Questo va a diminuire il numero di permutazioni valutate all'interno delle foglie, caratterizzando una diminuzione del tempo medio di esecuzione, ma comporta anche una diminuzione della correttezza dell'assegnamento.

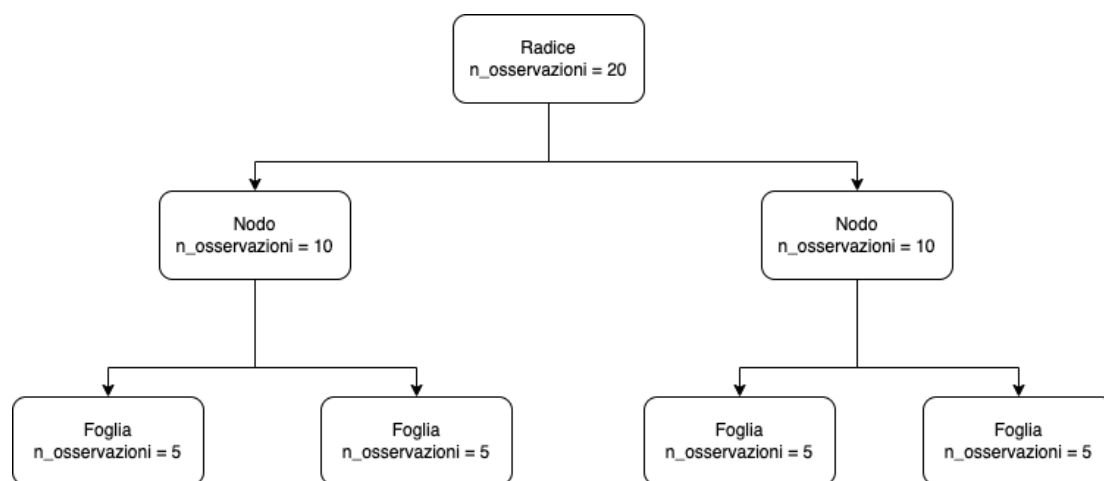


Figura 24: Albero dei sottoproblemi per il caso di test 26

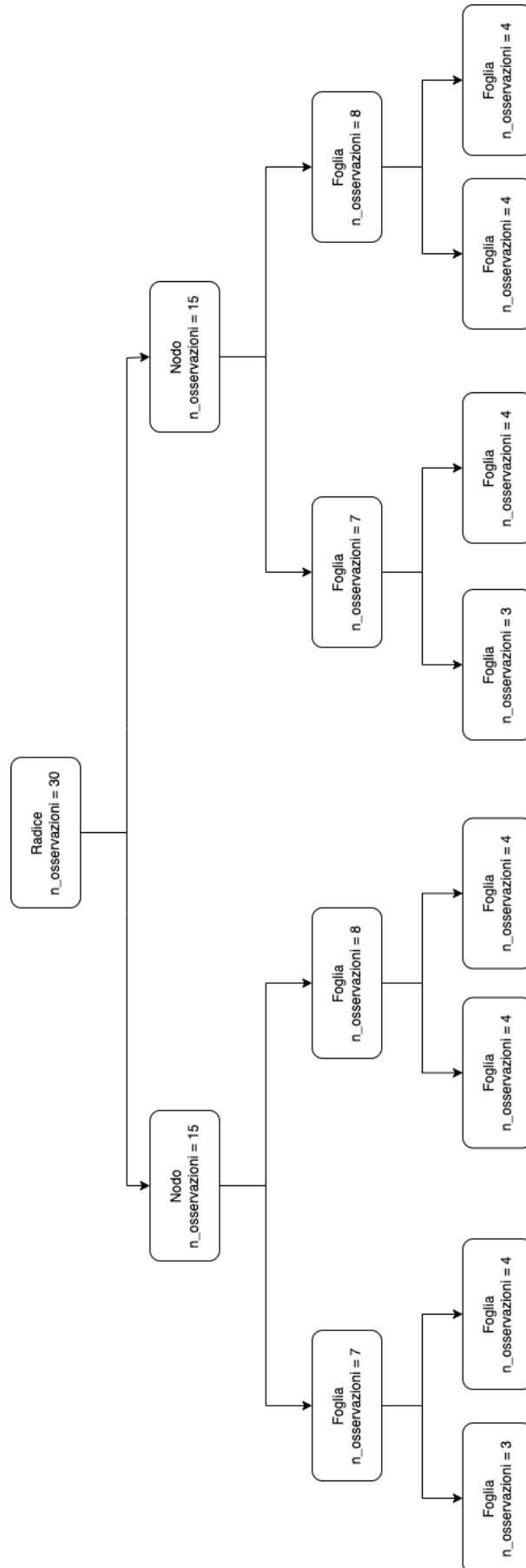


Figura 25: Albero dei sottoproblemi per il caso di test 27

4.3 Miglioramenti

Il threading è un miglioramento implementabile. Un thread o thread di esecuzione, è una suddivisione di un processo in due o più istanze o sottoprocessi che vengono eseguiti concorrentemente. L'esecuzione parallela nel caso del modulo di assegnamento non comporta alcuna problematica, siccome l'elaborazione dei nodi e delle foglie sono indipendenti l'uno dall'altra. È possibile quindi tramite l'utilizzo dei thread, ridurre drasticamente il tempo di esecuzione, tramite l'esecuzione parallela di interi rami o singoli nodi o foglie.

Capitolo 5

Conclusioni

Il lavoro presentato nella tesi è frutto delle conoscenze maturate durante i corsi universitari e l'esperienza ottenuta nel corso di Ricerca Operativa e tirocinio con il prof. Righini. Il sistema presentato tocca solo la superficie di quello che può essere l'analisi e lo studio del riconoscimento di elementi sul loro moto attraverso la ricerca operativa e modelli matematici.

Sicuramente possono essere presenti accortezze e miglioramenti applicabili alla soluzione presentata, ma a mio avviso la direzione espressa ed elaborata di astrazione e modellazione del problema sottolineano la duttilità e l'applicabilità della ricerca operativa a qualunque campo, come può essere quello di identificazione di satelliti.

I risultati ottenuti avvalgono la possibilità di identificare ed interpolare diverse sinusoidi sovrapposte con il minimo errore, a partire da una serie di valori ben distribuiti.

Sviluppi futuri possono essere la realizzazione di una interfaccia grafica che permetta una modalità di inserimento dei dati facile ed intuitiva, visualizzazione delle analisi effettuate e risultati individuati, oppure la definizione di diverse strategie utilizzabili per la risoluzione del problema di assegnamento. Inoltre sarebbe possibile estendere il processo di riconoscimento non solo a sinusoidi ma anche ad altre famiglie di funzioni matematiche.

Bibliografia

- [1] M. Sanna e G.Rodriguez, "Rilevamento di oggetti in movimento attraverso metodi basati sull'optical flow" 2016/2017
- [2] Dr. David R. Williams, Planetary Fact Sheets, NASA Goddard Space Flight Center 2016
- [3] Sartoretti P. and Schneide J. "On the detection of satellites of extrasolar planets with the method of transits" 1999, da pagina 553 a 560, SAO/NASA Astrophysics Data System.
- [4] Robert Fourer, David M. Gay, Brian W. Kernighan, AMPL (version 20220703), <https://www.ampl.com>
- [5] Artelys (2001), AMPL/Knitro (version 13.1.0)
- [6] Philip Gill, Walter Murray and Michael Saunders, SNOPT (version 7.5-1.2) (Jun 2015)
- [7] Pintér Consulting Services, Inc, LGO (2015-01-17)
- [8] Bruce Murtagh and Michael Saunders, AMPL/MINOS (version 5.51)
- [9] H. Y. Benson, Vanderbei, LOQO (Version 7.03 20150119)
- [10] The Python Software Foundation, Python (version 3.9.13), <https://python.org>
- [11] B. R. Heap, Permutations by interchanges, the Computer Journal, 6(3) (1963), pp. 293-298
- [12] Kuhn, H.W., Tucker, A.W. (2014). Nonlinear Programming. In: Giorgi, G., Kjeldsen, T. (eds) Traces and Emergence of Nonlinear Programming. Birkhäuser, Basel.