Relazione di Laboratorio

Corso: Sperimentazioni di Fisica 1 – LT
Anno accademico: 2013 – 2014
Docente: dott. C. Sada

Gruppo di Lavoro: Riccardo Milocco – 1070552 – riccardo.milocco@studenti.unipd.it
Andrea de Marco – 1069510 – andrea.demarco.2@studenti.unipd.it

Data di consegna della relazione: martedì, 10 giugno 2014

Pendolo a Torsione

Obbiettivi delle esperienze

Stimare ω_f, $\omega_R, \omega_s, \gamma, \omega_0$ e costruire la curva di risonanza.

Apparato strumentale

Cilindro esterno in materiale plastico contenente dell'acqua e all'interno del quale avviene l'esperimento. Sulla sommità è dotato di una serie di dispositivi elettronici e meccanici che permettono il funzionamento dello strumento.

Cilindro in acciaio Peso di massa pari a $115.5 \pm 0.5\ g$, diametro pari a $22.7 \pm 0.1\ mm$ e altezza pari a $34.0 \pm 0.1\ mm$, il quale è saldato a un filo in acciaio armonico di lunghezza di circa $75\ cm$ e diametro di circa $0.4\ mm$.

Piattaforma rotante di diametro pari a $8\ cm$ la cui rotazione è impartita da un servomotore grazie ad un alimentatore che fornisce tensione costante. Tale piattaforma trasmette la rotazione al filo alla quale è collegata e di conseguenza al cilindro in acciaio.

Sensore di movimento che rileva la posizione angolare della piattaforma rotante, ovvero l'ampiezza dell'oscillazione.

Programma di acquisizione Permette di controllare i dispositivi precedentemente descritti e quindi le condizioni iniziali dell'esperimento, e di registrare i dati raccolti. Grazie ad esso è possibile calibrare il sistema, condizione iniziale importante per evitare di incominciare l'esperimento con errori sistematici di cui non si conosce entità; impostare la frequenza e l'ampiezza della forzante (la forza impressa dalla piattaforma rotante). Inoltre costruisce un grafico ampiezza-tempo simultaneamente all'acquisizione dei dati.

Metodologia di misura

In primo luogo, abbiamo scelto di fissare un'ampiezza di oscillazione del motore a torsione e poi variare la frequenza di oscillazione.
In particolare, l'ampiezza arbitrariamente impostata nel programma è pari a 10; mentre per la frequenza, siamo partiti dal minimo valore consentito di 0.900 Hz, per arrivare ad un massimo di 1.100 Hz, variando la frequenza di 20 Hz ad ogni campione: abbiamo quindi acquisito un totale di 11 campioni.
Fatto partire il motore a torsione e avviata la registrazione dei dati, abbiamo lasciato che il moto oscillatorio si stabilizzasse, in modo da essere sicuri che il transiente non influenzasse più le nostre misurazioni. Abbiamo osservato che 60s erano sufficienti per raggiungere questa fase delle oscillazioni; poi, raggiunti i 100 secondi abbiamo tolto il momento della forzante e abbiamo aspettato che l'ampiezza decrescesse sensibilmente, per terminare l'acquisizione dei dati.

Abbiamo ripetuto la stessa metodologia di misura per ogni frequenza scelta.

Metodologia di rielaborazione dati

Prima di esporre la nostra metodologia di rielaborazione dei dati acquisiti, forniamo l'equazione differenziale che descrive il moto del cilindro immerso nell'acqua:

$$I\ddot{\theta} = -k\theta - \mathbb{C}\dot{\theta} + M_{0,forz}\cos\left(\omega_f t\right)$$

dove
$-\ I:\ inerzia\ del\ pendolo\ a\ torsione$;
$-\ \ddot{\theta}:\ accelerazione\ angolare\ del\ pendolo$;
$-\ k$: $costante\ di\ richiamo$;
$-\ \theta$: $angolo\ di\ oscillazione$;
$-\ \mathbb{C}:\ funzione\ rispetto\ al\ momento\ d'attrito$;
$-\ M_{0,forz}:\ momento\ iniziale\ della\ forzante$;
$-\ \omega_f$: $pulsazione\ quando\ é\ in\ atto\ la\ forzante$;
$-\ t$: $periodo\ di\ oscillazione$.

La rielaborazione dei dati è stata divisa in due parti, ognuna relativa ad una parte della curva d'oscillazione presa in considerazione :

- determinare la curva di risonanza e ω_R, attraverso la stima di ω_f per ogni frequenza;
- utilizzare il decadimento dell'ampiezza per computare γ, ω_s e infine ω_0.

Fase con la forzante in atto

La curva di risonanza è una particolare funzione che si ottiene mettendo in grafico $\theta_{0,part}$ in funzione della pulsazione ω_f.
Per quanto riguarda $\theta_{0,part}$, questo valore può essere ottenuto mediante la media pesata dei massimi relativi di ogni oscillazione, ottenuti con l'interpolazione quadratica delle creste positive della funzione.
Per fare in modo che le stime avessero dei valori non troppo alti è stata applicata una traslazione dello zero fino ad un punto arbitrario che distasse dal primo massimo circa 0.1 s.

Pertanto, interpolando 9 concavità con la parabola:

$$y = a + bx + cx^2$$

i valori sopra l'asse delle x, abbiamo trovato l'ascissa e l'ordinata di 9 massimi. Gli errori connessi a questa grandezze sono stati calcolati con la formula di propagazione degli errori.
Siccome T_f poteva essere considerata come la minima distanza che separava i due massimi, ne abbiamo calcolati tutti i diversi valori per ogni coppia di massimi ottenuti. In seguito, attraverso la media pesata è stata ottenuta la miglior stima della $\bar{T}_f \pm \sigma_{\bar{T}_f}$.
Direttamente dalla stima di $\bar{T}_f$, abbiamo calcolato il valore della pulsazione media $\omega_f \pm \sigma_{\omega_f}$, ripetendo il procedimento per ogni frequenza ottenendo 11 stime $\omega_f \pm \sigma_{\omega_f}$; quindi abbiamo elaborato la curva di risonanza per stimare la pulsazione di risonanza $\omega_R \pm \sigma_{\omega_R}$. In particolare, ponendo in grafico la dipendenza $\theta_{0,part.}$ in funzione di tutte le ω_f si ottiene approssimativamente una curva a campana con l'ascissa del massimo pari a ω_R.
La stima di ω_R è data dall'interpolazione con una parabola dei punti più prossimi al massimo e, ottenuti i valori di a, b, c, si calcola ω_R come si è già fatto per i massimi dell'interpolazione precedente. Determinata la ω_R abbiamo potuto passare alla fase di smorzamento.

Fase di smorzamento

Dopo aver calcolato la ω_R siamo passati ad analizzare la fase di smorzamento corrispondente.
Per avere una buona stima del periodo di smorzamento, $T_s \pm \sigma_{T_s}$, abbiamo deciso di considerare 11 creste consecutive sopra l'asse delle ascisse. Quindi abbiamo interpolato la prima e l'ultima cresta ciascuna con una parabola, per calcolare in seguito l'ascissa del massimo e l'errore associato per propagazione. Quindi abbiamo calcolato la distanza tra i due massimi l'abbiamo divisa per il numero dei massimi diminuito di uno, per ottenere la stima di un periodo di smorzamento T_s, associandogli l'errore per propagazione, σ_{T_s}.
Quindi abbiamo calcolato ω_s a partire da T_s, e σ_{ω_s} per propagazione.

In seguito, abbiamo interpolato le 11 creste con delle parabole per ottenere le coordinate dei massimi di ciascuna parabola. Queste infatti ci hanno permesso di porre in grafico la dipendenza $(t^*; \ln(y_{max}))$, dove t^* è l'ascissa del massimo della parabola e y_{max} la sua ordinata.
Posto in grafico tale dipendenza ci siamo accorti che i punti 2, 3, 5, 7, 8 sono molto distanti dalla retta, pertanto li abbiamo scartati e ricalcolato la retta di interpolazione, il cui coefficiente coincide con il valore di $\gamma \pm \sigma_\gamma$, uno degli obbiettivi che ci eravamo posti.

Infine risolvendo il sistema

$$\begin{cases} \omega_s = \sqrt{\omega_0^2 - \gamma^2} \\ \omega_R = \sqrt{\omega_0^2 - 2\gamma^2} \end{cases}$$

si è riusciti a stimare il valore di ω_0 e l'errore, σ_{ω_0}, per propagazione.

Presentazione dei dati sperimentali – Fase forzante

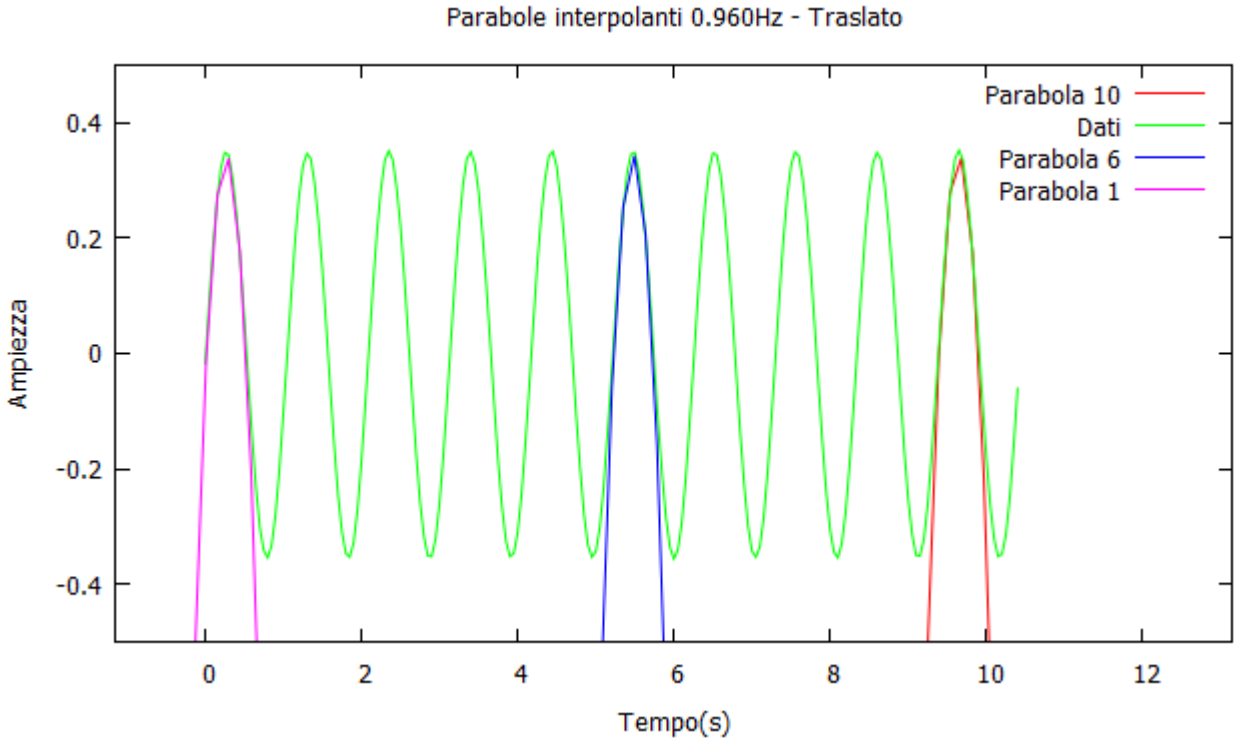
Legenda:

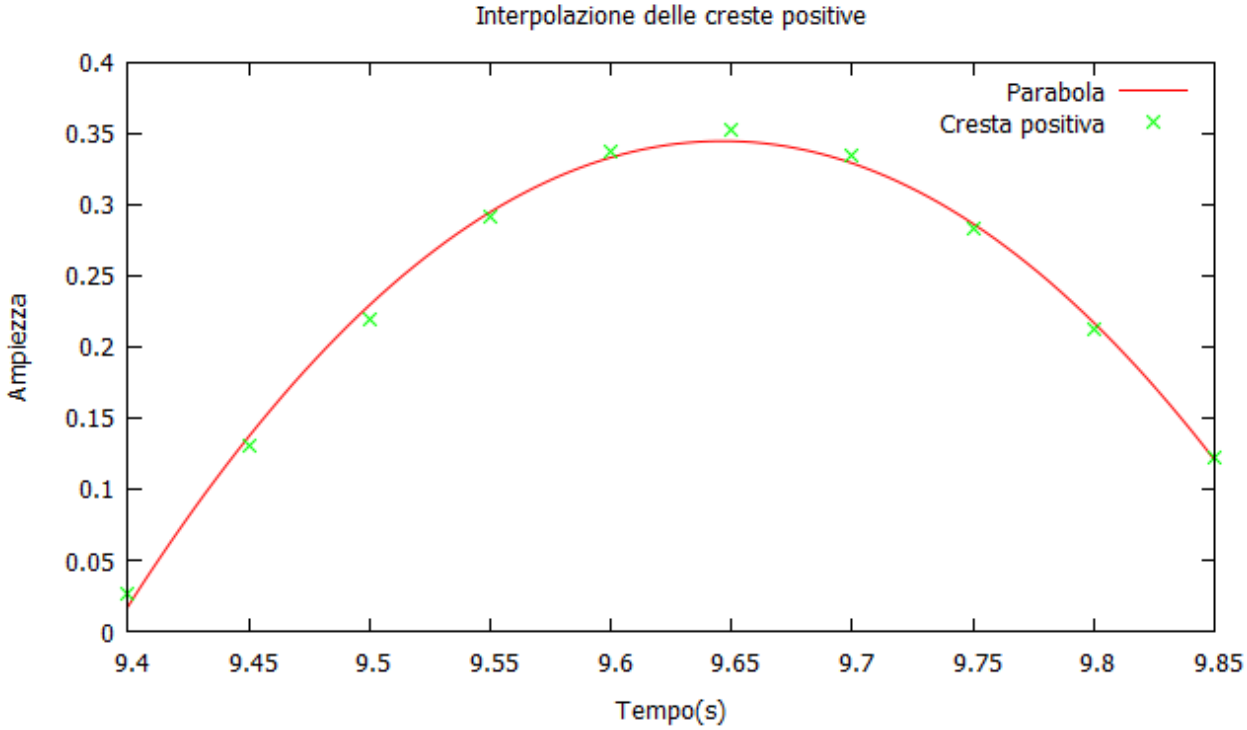
$< \boldsymbol{y_{max}} >$: *media pesata delle ordinate dei massimi delle parabole interpolanti*;
$< \boldsymbol{\sigma_{y_m}} >$: *errore dell'ordinata*;
$< \boldsymbol{T_f} >$; *media pesata dei periodi di oscillazione*;
$< \boldsymbol{\sigma_{T_f}} >$: *errore della media pesata dei periodi*;
$\boldsymbol{\omega_f}$: *pulsazione della forzante*;
$\boldsymbol{\sigma_{\omega_f}}$: *errore della pulsazione forzante*;
(a): *adimensionale*;

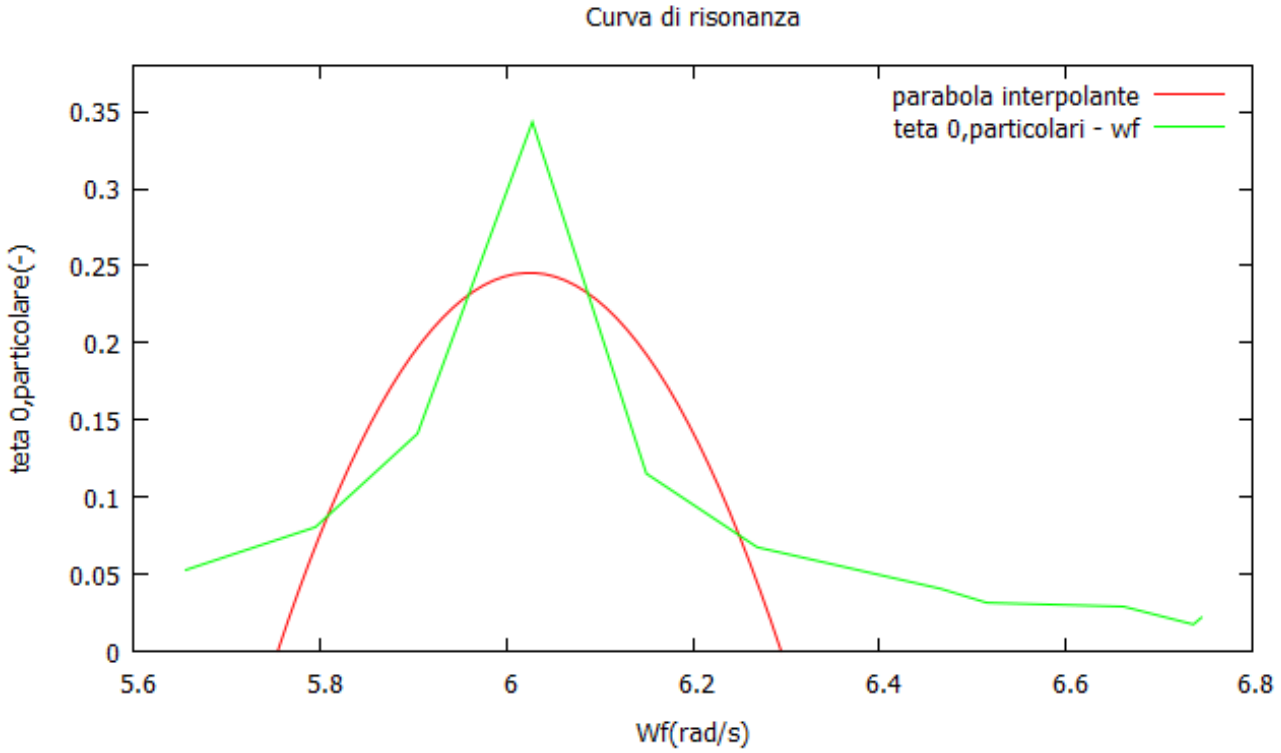
$\boldsymbol{Freq.(Hz)}$	$< \boldsymbol{y_{max}} > (a)$	$< \boldsymbol{\sigma_{y_m}} > (a)$	$< \boldsymbol{T_f} > (s)$	$< \boldsymbol{\sigma_{T_f}} > (s)$	$\boldsymbol{\omega_f}\ (rad/s)$	$\boldsymbol{\sigma_{\omega_f}}(rad/s)$
0.9	0.05	0.13	1.1	0.08	5.7	0.4
0.92	0.08	0.1	1.08	0.05	5.8	0.3
0.94	0.1	0.009	1.06	0.04	5.9	0.2
0.96	0.3	0.02	1.04	0.04	6.03	0.2
0.98	0.1	0.01	1.02	0.04	6.1	0.2
1.00	0.07	0.004	1.002	0.03	6.3	0.2
1.02	0.04	0.03	1.0	0.08	6.5	0.6
1.04	0.03	0.006	1.0	0.07	6.5	0.5
1.06	0.03	0.002	0.9	0.05	6.7	0.3
1.08	0.02	0.01	0.9	0.1	6.7	0.8
1.10	0.02	0.1	0.9	0.2	6.7	1.2

Grafici – Fase forzante

Per brevità abbiamo mostrato in grafico solo alcune interpolazioni quadratiche.






Curva di risonanza
parabola interpolante
teta 0,particolari - wf
teta 0,particolare(-)
Wf(rad/s)
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
5.6
5.8
6
6.2
6.4
6.6
6.8

Presentazione dei dati sperimentali – Fase di smorzamento

Le seguenti tabelle riportano i valori utilizzati per calcolare i massimi delle parabole interpolanti, e l'ascissa di massimo che ci è servita per determinare il periodo di smorzamento T_s.

$Prima\ cresta\ traslata\ O(\ 111.3\ s; 0\ )$	
$Tempo\ (s) - Ascissa$	$Ampiezza\ (a) - Ordinata$
0.10	0.08
0.15	0.2
0.20	0.2
0.25	0.3
0.30	0.3
0.35	0.3
0.40	0.3
0.45	0.2
0.50	0.2
0.55	0.07
$Ascissa\ di\ massimo \pm errore\ (s)$	$0.3 \pm\ 0.0001$

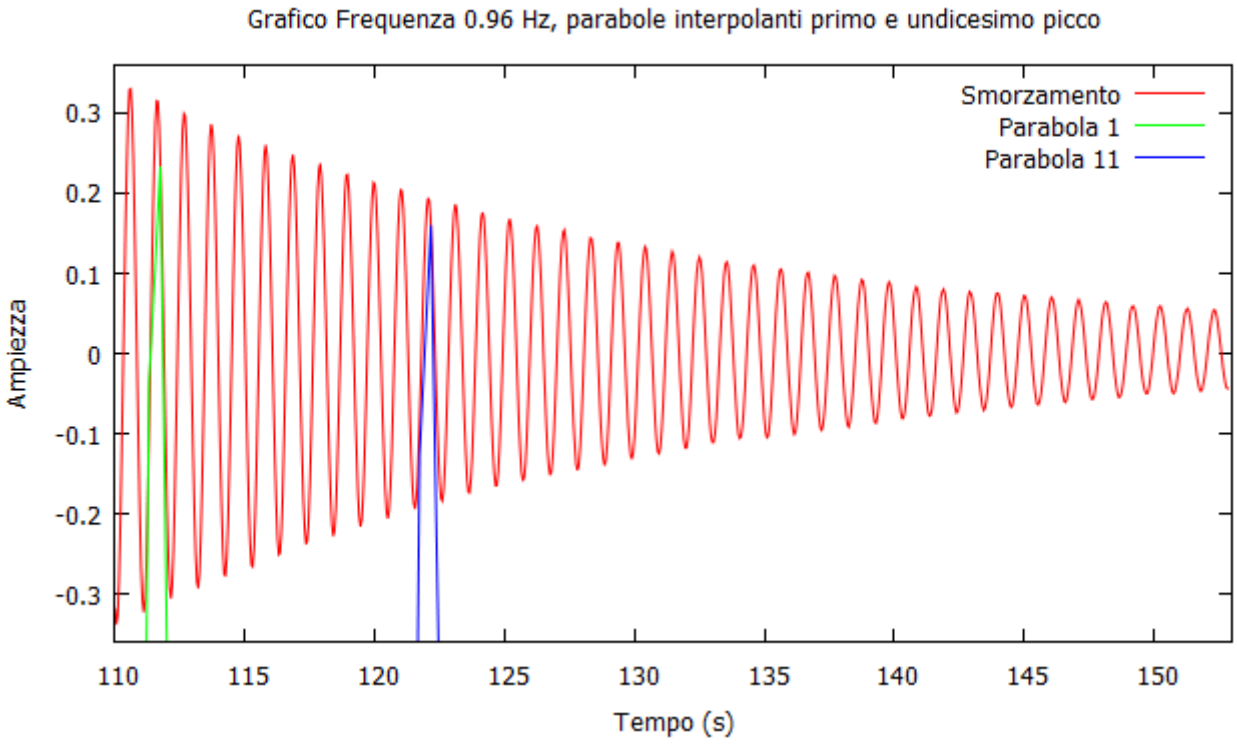
$Undicesima\ cresta\ traslata\ O(\ 111.3\ s; 0\ )$	
$Tempo\ (s) - Ascissa$	$Ampiezza\ (a) - Ordinata$
10.50	0.003
10.55	0.06
10.60	0.1
10.65	0.2
10.70	0.2
10.75	0.2
10.80	0.2
10.85	0.2
$Ascissa\ di\ massimo \pm errore\ (s)$	10.8 ± 0.1

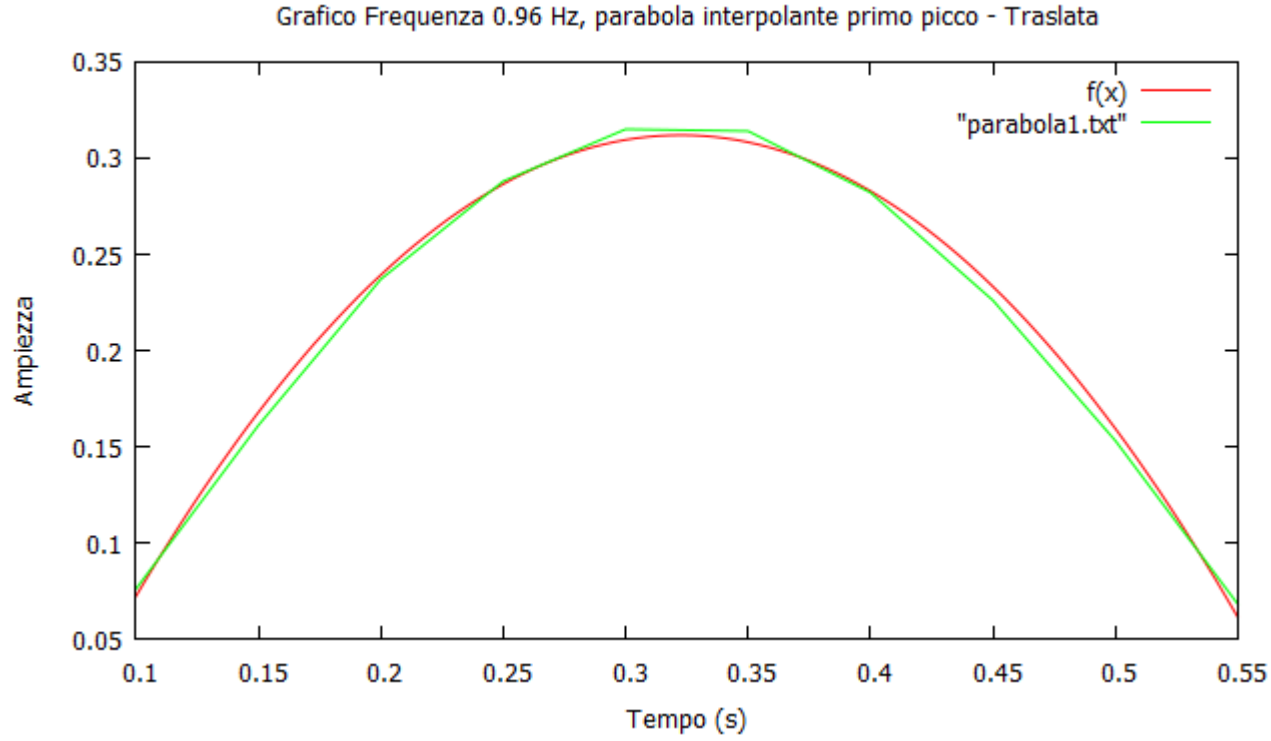
La seguente tabella riassume i valori ottenuti per i periodi quando è in atto la forzante e quando si è nella fase di smorzamento (rispettivamente T_f e T_s), e le relative pulsazioni, facenti riferimento ai dati della frequenza di risonanza (0.96 Hz); la pulsazione di risonanza calcolata come descritto in precedenza, ω_R, la pulsazione propria ω_0 e il parametro γ.

$\boldsymbol{T_f \pm \sigma_{T_f}\,(s)}$	1.04 ± 0.04
$\boldsymbol{\omega_f \pm \sigma_{\omega_f}(rad/s)}$	6.03 ± 0.2
$\boldsymbol{T_s \pm \sigma_{T_s}\,(s)}$	1.04 ± 0.01
$\boldsymbol{\omega_s \pm \sigma_{\omega_s}(rad/s)}$	6.02 ± 0.08
$\boldsymbol{\omega_R \pm \sigma_{\omega_R}(rad/s)}$	6.02 ± 4.8
$\boldsymbol{\omega_0 \pm \sigma_{\omega_0}(rad/s)^{*}}$	6.02 ± 4.8
$\boldsymbol{\omega_0 \pm \sigma_{\omega_0}(rad/s)^{**}}$	6.02 ± 0.006
$\boldsymbol{\gamma \pm \sigma_{\gamma}}$	-0.04 ± 0.006

* calcolato a partire da ω_R e da ω_s
** calcolato a partire da ω_s e da γ

Grafici – Fase di smorzamento





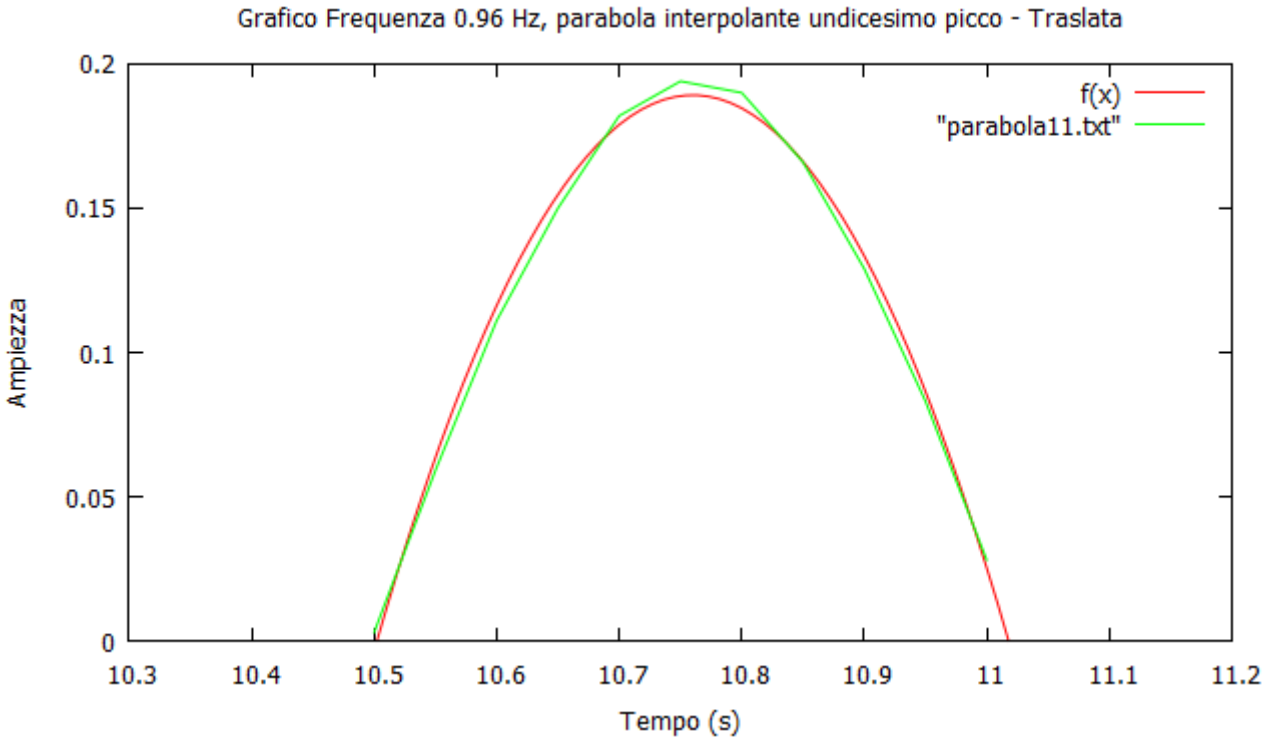


Grafico Frequenza 0.96 Hz, parabola interpolante undicesimo picco - Traslata
f(x)
"parabola11.txt"
Ampiezza
Tempo (s)
0.2
0.15
0.1
0.05
0
10.3
10.4
10.5
10.6
10.7
10.8
10.9
11
11.1
11.2

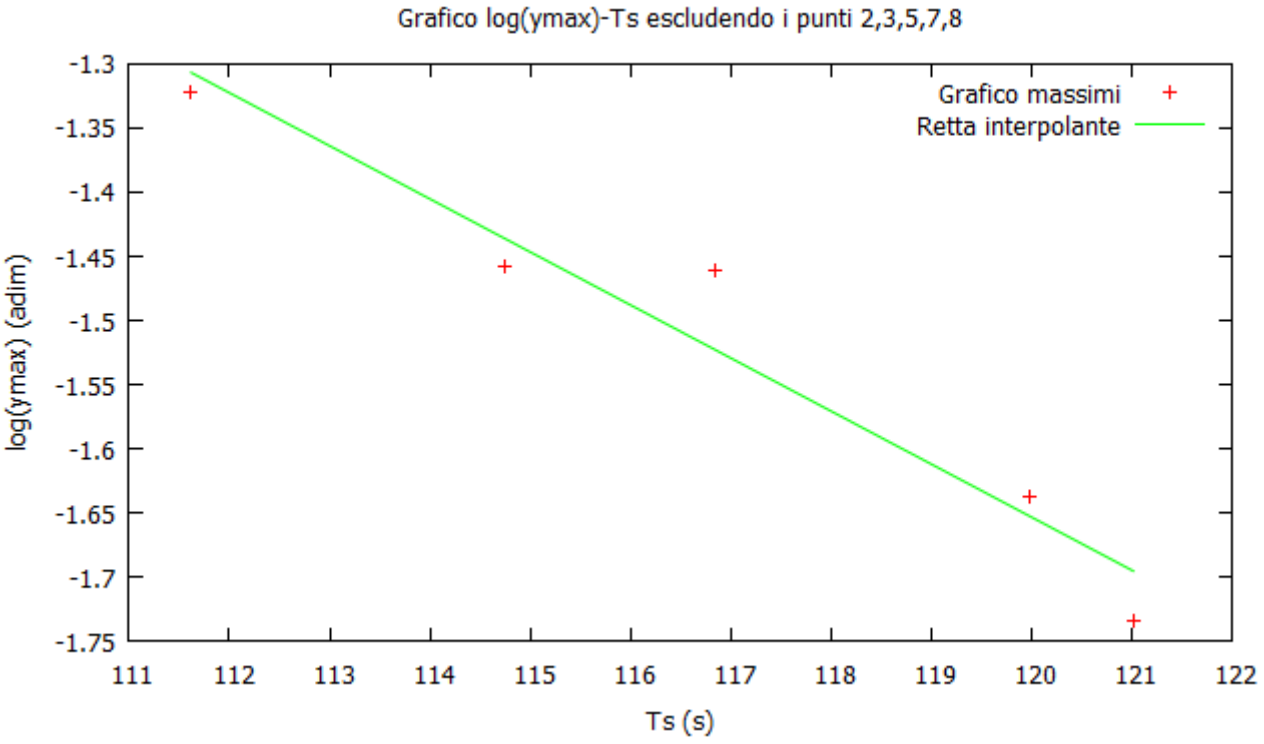


Grafico log(ymax)-Ts escludendo i punti 2,3,5,7,8
Grafico massimi
Retta interpolante
log(ymax) (adim)
Ts (s)
-1.3
-1.35
-1.4
-1.45
-1.5
-1.55
-1.6
-1.65
-1.7
-1.75
111
112
113
114
115
116
117
118
119
120
121
122

Analisi statistica dei dati presentati

La traslazione degli assi è stato un artificio molto utile per attribuire l'errore di T_f per propagazione, in quanto abbiamo proceduto stimando il valore di 9 intervalli facendo la differenza tra i massimi delle parabole (10) e in seguito facendo una media pesata degli intervalli. Infatti, essendo il massimo di una parabola dato dalla relazione:

$$x = -\frac{b}{2c}$$

dove b e c sono i parametri della parabola ($y = a + bx + cx^2$), e dipendendo quindi l'errore σ_x dai parametri e dagli errori ad essi associati, se non avessimo operato in questo modo saremmo stati costretti ad attribuire alla x un errore troppo elevato (≈ 800%).
In questo modo abbiamo ottenuto un valore dell'errore di T_f molto soddisfacente, circa il 3,8%.

Per quanto riguarda il periodo T_s (calcolato come già detto nella metodologia di rielaborazione dei dati), il suo errore per propagazione è dato da:

$$\sigma_{Ts} = \frac{\sqrt{\sigma_{max1}^2 + \sigma_{max11}^2}}{10}$$

dove σ_{max1} e σ_{max11} sono gli errori associati al primo e all'undicesimo massimo. Anche in questo caso il risultato è stato soddisfacente ≈ 1%.

Le due pulsazioni calcolate a partire dalle stime dei periodi, sono risultate essere prossime alla pulsazione di risonanza e con un errore relativamente basso, non superiore al 3%.

Per quanto riguarda invece ω_R non essendo riusciti a infittire le misure attorno alla frequenza di risonanza, l'interpolazione parabolica (che è stata eseguita sui 5 dati più prossimi al picco massimo della curva di risonanza) ha prodotto degli errori estremamente elevati, circa dell'80%, e per tanto la stima ottenuta non è soddisfacente.

Come anticipato in precedenza, il computo di ω_0 è stato possibile conoscendo la relazione che lega ω_s e ω_R al parametro γ, e risolvendo il sistema presentato nella metodologia di rielaborazione dati, ottenendo:

$$\omega_0^2 = 2\omega_s^2 - \omega_R^2$$

e il suo errore per propagazione:

$$\sigma_{\omega_0} = \sqrt{\frac{4\,\omega_s^2 \cdot \sigma_{\omega_s}^2 + \omega_R^2 \cdot \sigma_{\omega_R}^2}{2\omega_s^2 - \omega_R^2}}$$

Tuttavia, dipendendo quest'ultimo dall'errore di ω_R, è risultato essere molto elevato (≈ 79%). Un tale peso d'errore non ci permette di stabilire la precisione del valore di ω_0, il cui valore fluttua in un intervallo estremamente grande.

Il parametro γ è stato calcolato mediante interpolazione lineare dei punti aventi come ascissa il punto di massimo delle parabole interpolanti le creste (>0) della funzione in smorzamento; e come ordinata il logaritmo dei massimi delle stesse parabole. In particolare sono stati scartati quei valori che più si discostano da una prima retta interpolante, e successivamente è stata calcolata una seconda retta attorno alla quale i punti rimanenti si distribuiscono in modo pressoché lineare.

Il parametro γ che vogliamo stimare coincide proprio con il coefficiente angolare della seconda retta interpolante, e l'errore associato coincide con l'errore di tale coefficiente.

Riconsiderando ω_0 e calcolandolo invece a partire dalla relazione:

$$\omega_0^2 = \omega_s^2 + \gamma^2$$

l'errore associato per propagazione è dato da:

$$\sigma_{\omega_0} = \sqrt{\frac{\omega_s^2 \cdot \sigma_{\omega_s}^2 + \gamma^2 \cdot \sigma_\gamma^2}{\omega_s^2 + \gamma^2}}$$

il quale ha prodotto una ben più soddisfacente stima dell'errore, pari allo 0.1%.

Conclusioni

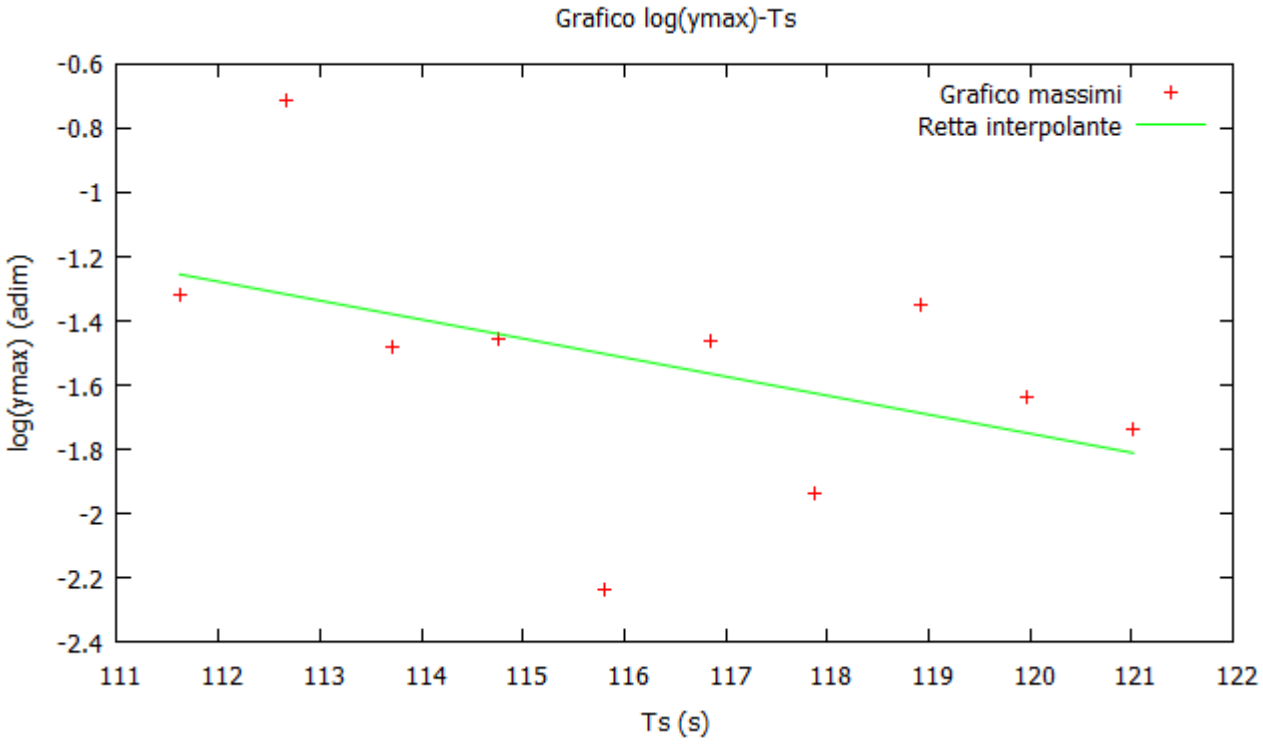
Dalle analisi effettuate sui dati acquisiti, siamo riusciti a stimare i valori delle $\omega_f, \omega_s, \omega_0$ e γ (e i relativi errori) per ogni frequenza.
Inoltre dal grafico della "Curva di risonanza" si può dedurre che la frequenza di risonanza sia la frequenza associata al massimo dei $\theta_{0,part}$ ($0{,}960 Hz$), e dunque siamo riusciti a computare la miglior stima della pulsazione di risonanza $\omega_R \pm \sigma_{\omega_R}$.
Per quanto riguarda invece la stima di ω_0, riteniamo valida quella ottenuta a partire dalla relazione:

$$\omega_0^2 = \omega_s^2 + \gamma^2$$

in quanto ha prodotto il risultato con errore minore.

Appendici

Di seguito viene riportato il grafico completo costruito per stimare il parametro γ.



Programmi:

- "tfwf.cxx";
- "rimodellatore.cxx";
- "wr.cxx";
- "ts.cxx";
- "traslazione.cxx";
- "grafico.cxx".

"tfwf.cxx":

```
#include <iostream>

#include <fstream>

#include <cmath>


int main()

{

        using namespace std;
```

```
	int dim = 10, k = 0, nmis = 60, tot = nmis / 6;

	double b[tot], errb[tot], c[tot], errc[tot], a[tot], erra[tot], smistatore[nmis/2],  errsmistatore[nmis/2], valore;

	char p, sl, m, ch, car, par1, par2, perc;


	ifstream fin( "fit1100.txt" );


	for( int i = 0; i < nmis/2; i++ )

	{

		fin >> car >> ch >> smistatore[i] >> p >> sl >> m >> errsmistatore[i] >> par1 >> valore >> perc >> par2;


		if ( k != 2 )

		{

			//cout << "Smistatore[ " << i << "]: " << smistatore[i] << "   errore" << errsmistatore[i] << endl;

			k++;

		}

		else if ( k == 2 )

		{

			//cout << "Smistatore[ " << i << "]: " << smistatore[i] << "   errore" << errsmistatore[i] << endl;

			//cout << endl;

			k = 0;

		}

	}


	fin.close();


	for( int i = 0; i < 10; i++ )
```

```
{
        a[i] = smistatore[3*i];
        b[i] = smistatore[3*i+1];
        c[i] = smistatore[3*i+2];
        /*cout << "a[ " << i << "]: " << a[i] << endl
                << "b[ " << i << "]: " << b[i] << endl
                << "c[ " << i << "]: " << c[i] << endl << endl;*/
}

for( int i = 0; i < 10; i++ )
{
        erra[i] = errsmistatore[3*i];
        errb[i] = errsmistatore[3*i+1];
        errc[i] = errsmistatore[3*i+2];
        /*cout << "erra[ " << i << "]: " << erra[i] << endl
                << "errb[ " << i << "]: " << errb[i] << endl
                << "errc[ " << i << "]: " << errc[i] << endl << endl;*/
}

double x[10];

for ( int i = 0; i < 10; i++ )
{
        x[i] =  -b[i]/( 2 * c[i]);
}

cout << endl;

double errx[dim];
```

```
for( int i = 0; i < dim; i++ )
{
        errx[i] = sqrt ( ( pow( b[i] * errc[i], 2 ) + pow( c[i] * errb[i], 2 ) ) / ( 4 * pow( c[i], 4 ) ) );
        cout << "x[" << i << "]: " << x[i] << "  errx: " << errx[i] << endl;
}

//calcolo tf

cout << endl;

double tf[9];
double errtf[9];

for ( int i = 0; i < 9; i++ )
{
        tf[i] =  fabs( x[i+1]-x[i] );
        errtf[i] = sqrt(( pow ( errx[i], 2 ) + pow( errx[i+1], 2 ) ));
        cout << "tf[" << i << "]: " << tf[i] << "      errtf: " << errtf[i] << endl;
}

//media pesata tf

double tmedia, sigmatmedia;

double peso[9];

for ( int i = 0; i < 9; i++ )
{
```

```
        peso[i] = 1.0 / pow ( errtf[i], 2 );
}

double num = 0;
for ( int i = 0; i < 9; i++ )
{
        num = ( tf[i] / pow ( errtf[i], 2 )) + num;
}

double den = 0;
for ( int i = 0; i < 9; i++ )
{
        den = peso[i] + den;
}

tmedia = num / den;

double errtmedia = sqrt ( 1.0 / den );

cout << "\nMedia pesata delle Tf = " << tmedia
        << "      Errore della media pesata = " << errtmedia << endl << endl;

//esporto tmedia e errore
ofstream fout;
fout.open("tfmedia.txt");
        fout << tmedia << "    " << errtmedia << endl;
fout.close();

// wf
```

```
double wf = 2.0*M_PI / tmedia;
double errwf = 2.0 * M_PI * ( errtmedia            / pow ( tmedia, 2 ) );



//ordinata da associare a wf ( media pesata ordinata dei massimi delle parabole )

double max[dim], errmax[dim];
double add1, add2, add3;
for ( int i = 0; i < 10; i++ )
{
        max[i] = - ( pow ( b[i], 2 ) - 4 * c[i] * a[i] ) / ( 4 * c[i] );



        add1 = pow (  -b[i] / ( 2 * c[i] ) * errb[i] , 2 );
        add2 = pow ( erra[i] , 2 );
        add3 = b[i]*b[i] / ( 4 * pow ( c[i], 2 ) ) * errc[i];
        errmax[i] =  add1 + add2 + pow ( add3, 2 ) ;
        errmax[i] = sqrt ( errmax[i] );
        //cout << "add1 " << add1 << "   add2 " << add2 << "   add3 " << add3 << endl;
        cout << "max[ " << i << "]:" << max[i] << "    errmaxf: " <<  errmax[i] << endl;

}



//media pesata max ( uso le stesse strutture )

for ( int i = 0; i < 10; i++ )
```

```
{
        peso[i] = 1.0 / pow ( errmax[i], 2 );
}

num = 0;
for ( int i = 0; i < 10; i++ )
{
        num = ( max[i] / pow ( errmax[i], 2 )) + num;
}

den = 0;
for ( int i = 0; i < 10; i++ )
{
        den = peso[i] + den;
}

tmedia = num / den;

errtmedia = sqrt ( 1.0 / den );

cout << "\nMedia pesata delle max = " << tmedia
        << "      Errore della media pesata = " << errtmedia << endl << endl;

cout << " wf: " << wf << "    errwf: " <<  errwf << endl << endl;

//esporto in diversi files


fout.open("tf.txt");
```

```
	for ( int i = 0; i < 9; i++ )
	{
		fout << tf[i] << "   " << errtf[i] << endl;
	}
	fout.close();

	//esporto maxmedia e errore
	fout.open("maxmedia.txt");
		fout << tmedia << "   " << errtmedia << endl;
	fout.close();

	fout.open("maxi.txt");
	for ( int i = 0; i < 9; i++ )
	{
		fout << max[i] << "   " << errmax[i] << endl;
	}
	fout.close();

	fout.open("wf.txt");
		fout << wf << "   " << errwf << endl;
	fout.close();

	return 0;
}
```

“rimodellatore.cxx”:

```
#include <iostream>
#include <fstream>
```

```
#include <cmath>
#include <string>
#include <iomanip>

int main()
{
	using namespace std;
	char c;
	char d;
	int righe = 3058;
	double tempo[righe];
	double ampiezza[righe];
	double pendolo[righe];

	ifstream fin;
	fin.open("frequenza0960ampiezza10.txt");

	for ( int i = 0; i < righe; i++ )
	{
		fin >> tempo[i] >> ampiezza[i] >> pendolo[i];
	}
	fin.close();

	ofstream fout;
	fout.open("risonanza.txt");

	for ( int i = 0; i < righe; i++ )
	{
		fout << ampiezza[i] << "	" << pendolo[i] << endl;
```

```
	}


	return 0;
}
```

“wr.cxx”:

```
#include <iostream>
#include <fstream>
#include <cmath>

int main()
{
	using namespace std;

	int dim = 1, k = 0, nmis = 6, tot = 1;
	double b[tot], errb[tot], c[tot], errc[tot], a[tot], erra[tot], smistatore[nmis/2],  errsmistatore[nmis/2], valore;
	char p, sl, m, ch, car, par1, par2, perc;

	ifstream fin( "abcwr.txt" );

	cout << endl;

	for( int i = 0; i < nmis/2; i++ )
	{
		fin >> car >> ch >> smistatore[i] >> p >> sl >> m >> errsmistatore[i] >> par1 >> valore >> perc >> par2;

		if ( k != 2 )
```

```
		{
			cout << "Smistatore[ " << i << "]: " << smistatore[i] << "   errore" << errsmistatore[i] << endl;
			k++;
		}
		else if ( k == 2 )
		{
			cout << "Smistatore[ " << i << "]: " << smistatore[i] << "   errore" << errsmistatore[i] << endl;
			cout << endl;
			k = 0;
		}
	}

	fin.close();

	for( int i = 0; i < dim; i++ )
	{
		a[i] = smistatore[3*i];
		b[i] = smistatore[3*i+1];
		c[i] = smistatore[3*i+2];
		cout << "a[ " << i << "]: " << a[i] << endl
			 << "b[ " << i << "]: " << b[i] << endl
			 << "c[ " << i << "]: " << c[i] << endl << endl;
	}

	for( int i = 0; i < dim; i++ )
	{
		erra[i] = errsmistatore[3*i];
		errb[i] = errsmistatore[3*i+1];
```

```
            errc[i] = errsmistatore[3*i+2];
            cout << "erra[ " << i << "]: " << erra[i] << endl
                    << "errb[ " << i << "]: " << errb[i] << endl
                    << "errc[ " << i << "]: " << errc[i] << endl << endl;
        }

        double x[10];

        for ( int i = 0; i < dim; i++ )
        {
            x[i] =  -b[i]/( 2 * c[i]);
        }

        cout << endl;

        double errx[dim];

        for( int i = 0; i < dim; i++ )
        {
            errx[i] = sqrt ( ( pow( b[i] * errc[i], 2 ) + pow( c[i] * errb[i], 2 ) ) / ( 4 * pow( c[i], 4 ) ) );
            cout << "x[" << i << "]: " << x[i] << "  errx: " << errx[i] << endl;
        }

        return 0;

}
```

"ts.cxx":

```
#include <iostream>
#include <fstream>
#include <cmath>

int main()
{
	using namespace std;

	int dim = 2;
	double b[dim], errb[dim], c[dim], errc[dim];

	ifstream fin( "stimats.txt" );

	for( int i = 0; i < dim; i++ )
		fin >> b[i] >> errb[i] >> c[i] >> errc[i];

	fin.close();

	double x[] = { -b[0]/( 2 * c[0]), -b[1]/( 2 * c[1])};

	double errx[dim]/*, corrbc[] = { -0.982, -1 }, covbc[dim]*/;

	/*for( int i = 0; i < dim; i++ )
	{
```

```
		covbc[i] = corrbc[i] * sqrt( errb[i] + errc[i] );
		cout << covbc[i] << endl;
	}
	*/
	for( int i = 0; i < dim; i++ )
	{
		errx[i] = ( pow( b[i] * errc[i], 2 ) + pow( c[i] * errb[i], 2 ) /*- 2 * b[i] * c[i] * covbc[i] */) / ( 4 * pow( c[i], 4 ) );
		cout << x[i] << "   " << errx[i] << endl;
	}

	double stimats = ( x[1] - x[0] ) / 10;
	double errts = sqrt( pow( errx[1], 2 ) + pow( errx[0], 2 ) ) / 10;

	cout << endl << stimats << "   " << errts << endl;

	double ws = 2 * M_PI / stimats,
		errws = 2 * M_PI * errts / pow( stimats, 2 ) ;

	cout << endl << ws << "    " << errws << endl;

	return 0;
}
```

“traslazione.cxx”:

```
#include <iostream>
#include <fstream>

int main()
{
	using namespace std;

	int dim = 22;

	double tempi[dim], unuseful[dim], traslazione[( dim - 1 )];

	ifstream fin( "frequenza0960ampiezza10.txt" );

	for( int i = 0; i < dim; i++ )
		fin >> tempi[i] >> unuseful[i];

	fin.close();

	cout << "\nTempi dopo traslazione" << endl << endl;

	for( int i = 0; i < dim - 1; i++ )
		traslazione[i] = tempi[( i + 1 )] - tempi[0];

	ofstream fout( "traslazione096hz.txt" );
```

```
	for( int i = 0; i < dim - 1; i++ )
	fout << traslazione[i] << "\t\t" << unuseful[( i + 1 )] << endl;

	fout.close();

	return 0;
}
```

“grafico.cxx”:

```
#include <iostream>
#include <fstream>
#include <cmath>

int main()
{
	using namespace std;

	ifstream fin( "abc.txt" );
	int dim = 10;
	double a[dim], erra[dim], b[dim], errb[dim], c[dim], errc[dim];
	char temp[3];

	for( int i = 0; i < dim; i++ )
		fin >> temp[i] >> a[i] >> erra[i] >> temp[i] >> b[i] >> errb[i] >> temp[i] >> c[i] >> errc[i];

	fin.close();
```

```
	double xmax[dim], ymax[dim], logymax[dim];

	for( int i = 0; i < dim; i++ )
	{
		xmax[i] = -b[i] / ( 2 * c[i] );
		ymax[i] = -( pow( b[i], 2 ) - 4 * a[i] * c[i] ) / ( 4 * c[i] );
		logymax[i] = log( ymax[i] );

		cout << i + 1 << ")  " << xmax[i] << endl << ymax[i] << endl << logymax[i] << endl <<
endl;
	}

	ofstream fout( "graficomax.txt" );

	for( int i = 0; i < dim; i++ )
		if ( i != 1 && i !=2 && i != 4 && i != 6 && i != 7 )
			fout << xmax[i] << "   " << logymax[i] << endl;

	fout.close();

	return 0;
}
```