Descrizione della procedura "ottimizzata" di correzione degli accelerogrammi mediante correzione a tratti della linea base del segnale in velocità: implementazione del programma Matlab "basco.m" (POLI-MI)

L'ottimizzazione apportata al programma basco.m riguarda la ricerca automatizzata delle migliori condizioni che permettono la correzione a tratti della linea di base del segnale in velocità.

Il nucleo della procedura analitica del nuovo programma rispecchia essenzialmente quanto già descritto per il programma basco.m, a cui si rimanda per delucidazioni eventuali. Rispetto al programma precedente, il nuovo programma si differenzia per le caratteristiche descritte qui di seguito:

1. Caricamento di un accelerogramma da una directory "Input" mediante selezione tramite cursore e richiamando la funzione esterna leggi_acc.m;
2. Definizione interattiva dei parametri dell'accelerogramma (n. linee dell' header, n. colonne della registrazione, passo di campionamento, fattore di scala per passare ad accelerazione espressa in m/s^2);
3. Calcolo e visualizzazione dello "Husid Plot" relativo all'intensità di Arias della traccia analizzata, corretta per il valor medio ed integrata in velocità;
4. Possibilità di definire interattivamente i parametri della correzione da effettuare (parametri *lba*, *pre_ev*, *lb*v già introdotti nella versione originale di basco.m, tranne quelli relativi all'opzione di effettuare un filtraggio sulla traccia analizzata, opzione qui eliminata);
5. Ricerca su griglia delle migliori coppie di valori t_1 e t_2 che, utilizzati in modo analogo al programma originale basco.m, definiscono gli intervalli ottimali per la correzione a tratti della linea di base del segnale integrato numericamente in velocità;
6. Possibilità di accettare la soluzione ottimale individuata dal programma; un alternativa si può selezionare una delle soluzioni temporanee individuate dal programma oppure terminare il programma senza salvare alcuna soluzione.
7. Salvataggio in una directory "Output" di due files ASCII mono-colonna relativi allo spostamento corretto ed alla accelerazione ottenuta per doppia derivazione a partire dallo spostamento stesso;
8. Sostituzione della funzione "lsqr" (Matlab 6.5) con la funzione "lsqlin" (Matlab 7).

UTILIZZO E FUNZIONALITÀ DEL PROGRAMMA

basco_INGV_Roma1.m

Il programma consente di ottimizzare la determinazione dello spostamento ottenuto da una correzione di una registrazione accelerometrica, prevedendo di individuare i migliori parametri t_1 e t_2 attraverso una ricerca automatizzata su griglia.
Il programma è stato studiato al fine di ottenere uno spostamento corretto accettabile per lo meno per una durata corrispondente alla massima ampiezza del segnale.
E' degno di nota il fatto che la logica di ottimizzazione del programma è risultata "tarata" per eventi italiani in cui ragionevolmente non ci si aspetta uno spostamento residuo diverso da zero. E' possibile una sua implementazione per i casi in cui sussistano spostamenti residui.

Caricamento dati

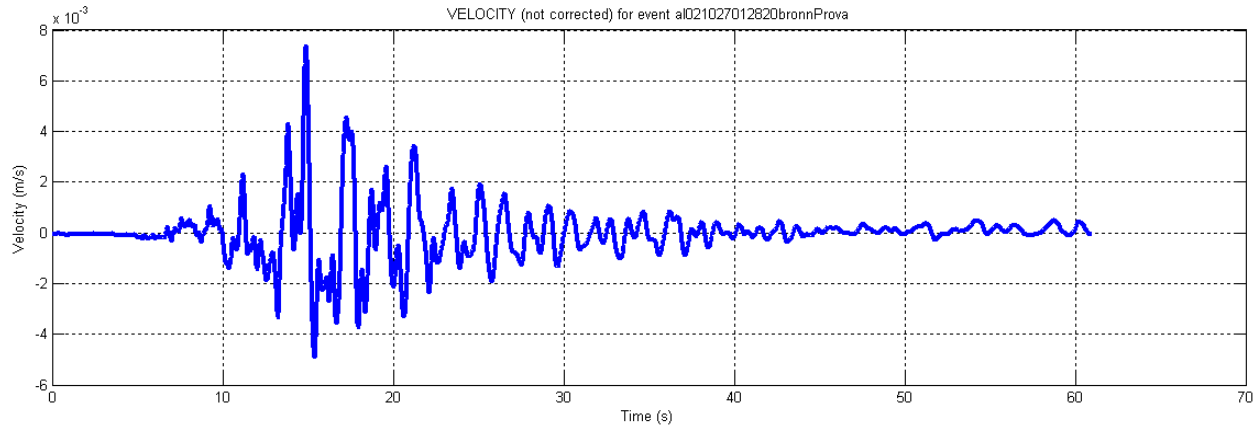
Il segnale accelerometrico in ingresso (con l'estensione .txt) viene selezionato tramite cursore, all'interno di una directory "Input" che deve esser creata precedentemente; il segnale viene letto dalla funzione esterna "*leggi_acc*", la quale prevede che vengano definiti interattivamente il numero di righe di header da saltare (*nlin*) ed il numero di colonne su cui sono disposti i dati (*ncol*). Inoltre, si definiscono interattivamente il passo di campionamento costante del segnale (*dt*) ed il fattore di scala che consente di passare dall' unità di grandezza del segnale a m/s^2.
In questa sezione si individua il nome e la directory di destinazione "Output" (da creare preventivamente) in cui verranno salvati due files in formato ASCII, relativi allo spostamento corretto ed all'accelerazione ottenuta da questo mediante doppia derivazione.

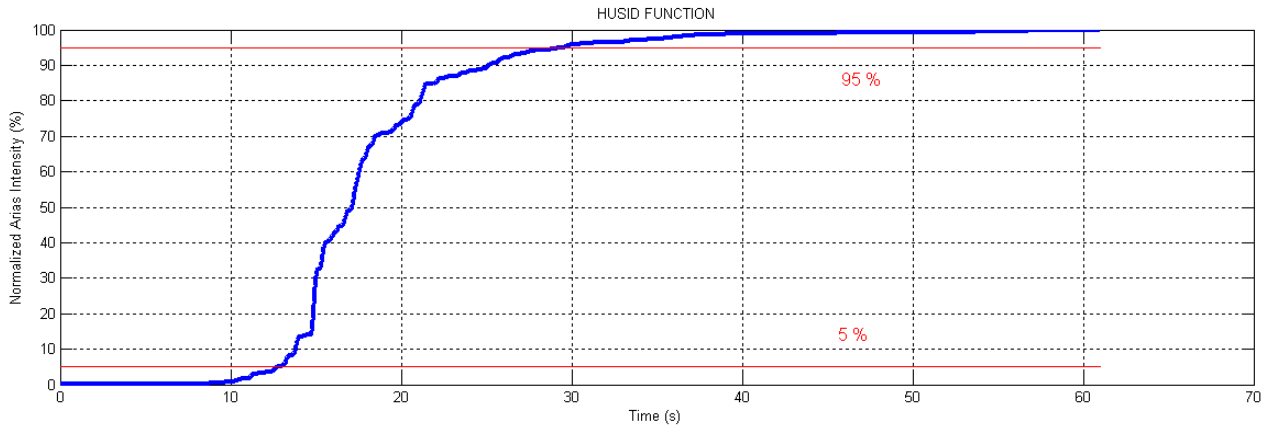
```
---------------------- CARICAMENTO DATI ------------------------

%Caricamento del file da analizzare%
[filename,pathname]=uigetfile('*.txt', 'Selezionare file di INPUT
(ascii con header e acc. su 1 o + colonne)')
fileid=filename(1:length(filename)-4);
%identificativo dei files in cui saranno salvati i dati corretti
nomefile2=[pathname(1:length(pathname)-6)    'Output\'    fileid
'_acc.cor'];  %accelerazione  dal  dato  corretto  con  Basco
ottimizzato
nomefile3=[pathname(1:length(pathname)-6)    'Output\'    fileid
'_disp.cor']; %spostamento tramite Basco ottimizzato
nlin=input('quante righe saltare? (righe di header): ');
ncol=input('quante colonne accelerogramma?: ');
dt=input('passo temporale accelerogramma?: ');
scala=input('fattore di scala per passare da unita grandezza del
segnale a m/s*s ?: ');
```

```
nomefile=[pathname,filename];
acc=leggi_acc(nomefile,nlin,ncol);  %funzione esterna per la lettura dell'accelerogramma%
durata_eff=(length(acc)-1)*dt; %durata effettiva del segnale *in tempo)%
npun_eff=length(acc);  %numero di punti di campionamento che costituiscono effettivamente il segnale%
t_eff=[0:dt:(npun_eff-1)*dt]; % definizione della variabile tempo effettivo %
```

Analisi sul dato originale

Il programma calcola il segnale in velocità ed in spostamento mediante integrazione numerica dell'accelerazione originale. Inoltre, calcola l'Intensità di Arias della traccia in velocità a partire dal segnale a cui è stata rimossa la media. L'intensità di Arias è visualizzata sotto forma di Husid Plot, normalizzato al valore massimo assunto dal parametro stesso.
Sovrapposti al grafico, sono evidenziati i livelli corrispondenti al 5% ed al 95% dell'intensità complessiva, i quali forniscono una indicazione di massima, rispettivamente, sull'inizio della fase delle onde S e sulla durata del segnale che riveste un ruolo significativo.





```
------------------- ANALISI SUL DATO ORIGINALE -----------------
……
%Calcolare e Plottare velocità non corretta e la relativa
l'intensità di Arias, evidenziando il 5% ed il 95 dell'intensità
risultante
h=cumtrapz(t_eff,(demean(vel_nc_husid(1:npun_eff))).^2);
h_norm=(h./max(h))*100; %intensità normalizzata
……
```

Definizione dei parametri per la correzione del segnale

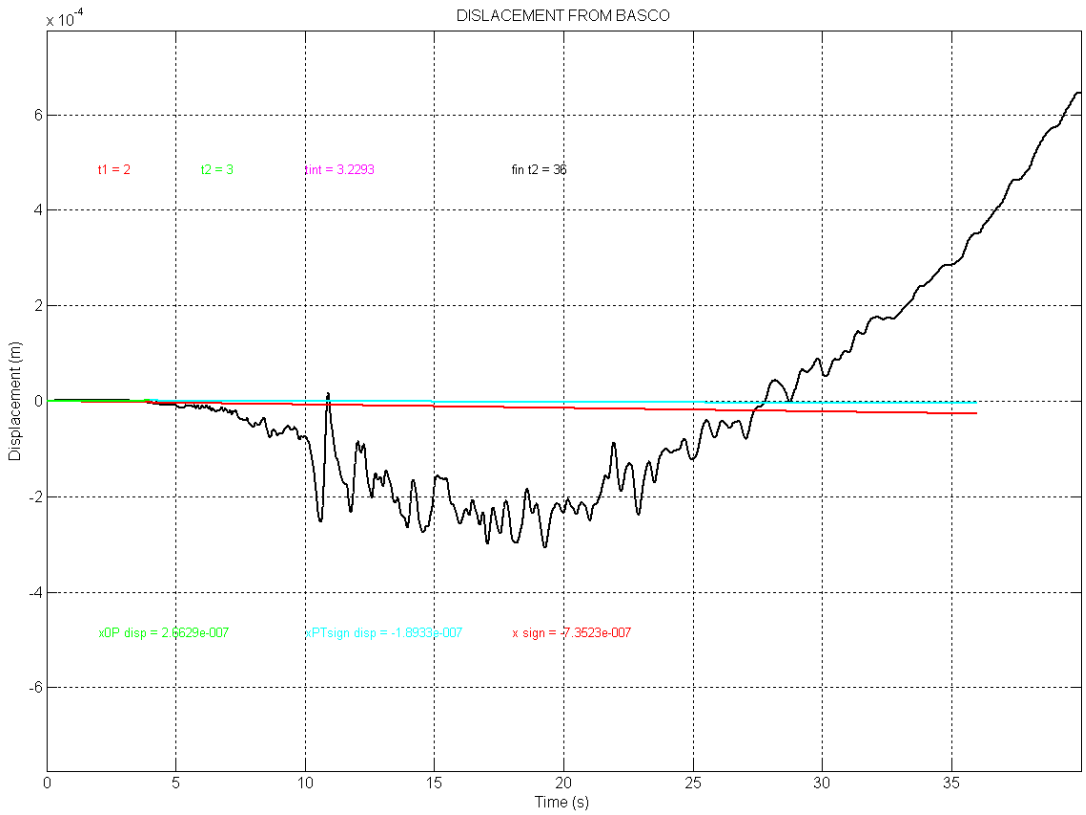
Il programma permette di selezionare interattivamente le opzioni di correzione già introdotte nella versione originale di basco.m (*lba*, *pre_ev*, *lbv*), tranne l'opzione di filtraggio che è stata eliminata nella versione ottimizzata.

In questa porzione del programma, vengono definiti i parametri utilizzati per la ricerca su griglia della coppia di valori di t_1 e t_2 che permettono la migliore correzione del segnale.
In particolare, vengono definiti i seguenti parametri:

- *in_t1* rappresenta il primo valore da assegnare al parametro t_1 all'interno della ricerca su griglia. Attualmente il parametro è stato posto pari a 2*dt1, ma può esser modificato manualmente all'interno del programma. In ogni caso, si consiglia di mantenere tale valore sufficientemente piccolo per consentire al programma di spaziare su un numero maggiore di possibilità.
- *dt1 e dt2* definiscono l'incremento con cui variano i valori di t_1 e t_2 durante i due cicli. Si consiglia di mantenere questi parametri sufficientemente piccoli, ad esempio tra 0.5 e 2 secondi, a seconda della lunghezza del segnale e della potenza di calcolo del PC.
- *dtTsign* definisce l'incremento con cui varia la lunghezza della porzione significativa del segnale, in base alla quale si calcolano alcuni parametri la cui minimizzazione consente di individuare la miglior coppia di valori t_1 e t_2 (si veda la sezione successiva per maggior dettagli sulla logica di ottimizzazione). Si consiglia di assegnare un valore compreso tra 1 e 4 secondi al fine di ottenere una buona convergenza alla soluzione ottimale. Si tenga presente che il parametro stesso condiziona fortemente il numero di cicli che si effettuano sul segnale, e di conseguenza la lunghezza dell'operazione di correzione.
- *arrivo_P* corrisponde approssimativamente al tempo di arrivo, in secondi, della fase P. In base all'esperienza acquisita, risulta preferibile fornire un valore leggermente inferiore al vero arrivo delle onde P, al fine di minimizzare quanto più possibile eventuali distorsioni nel segnale corretto in spostamento.
- *fin_t2_iniz* definisce la lunghezza della porzione significativa del segnale, in genere associata alla fine della fase di massima ampiezza che caratterizza le onde S. Questo parametro viene di volta in volta incrementato della quantità *dtTsign* all'interno dei cicli ideati.

Come esempio, nel segnale già mostrato precedentemente, tale parametro era stato fissato a 21 secondi.

- *durata_controllo* e *tempo_percontrollo* sono due parametri introdotti per controllare che le soluzioni individuate dal programma non portino ad una forma inaccettabile del segnale corretto in spostamento, come l'esempio mostrato nella figura successiva.
  Per garantire che si ottenga un segnale corretto in spostamento che includa almeno la porzione corrispondente alla massima ampiezza del segnale, i due parametri sono stati introdotti per controllare che il segnale non si assesti al di sotto od al di sopra della linea dello zero per un tempo troppo prolungato.
  Per garantire questa condizione, *durata_controllo* viene preso uguale all'intervallo corrispondente alla lunghezza in secondi del segnale contraddistinto da massima ampiezza (per default uguale alla lunghezza del segnale significativo *fin_t2_iniz*); *tempo_percontrollo* corrisponde all'intervallo temporale in secondi in cui il segnale deve poter cambiare segno: in generale, tale valore si assesta intorno ai 3 – 5 secondi ma può variare in funzione delle caratteristiche delle forme d'onda presenti nel segnale.




```
-----DEFINIZIONE DEI PARAMETRI PER LA CORREZIONE DEL SEGNALE-----
lba=input('vuoi eseguire rimozione costante della linea base? (1=Si 0=No): ');
```

```
pre_ev=input('durata temporale (in s) su cui calcola la linea
base (se=0 la calcola su tutto il segnale (non usato se lba=0)):
');
lbv=input('vuoi eseguire rimozione a tratti della linea base sul
segnale in velocità (1=Si 0=No): ');

%Definizione intervalli per correzione linea base a tratti in
velocità (non usato se lba=1 e pre_ev=0)%
in_t1=2*dt1; %primo valore da assegnare al t1 (all'occorrenza si
può modificare a mano sul programma, ma è sempre meglio
mantenerlo ad un numero sufficientemente basso
dt1=input('step di incremento per il t1(suggerito 0.5): ');
dt2=input('step di incremento per il t2(suggerito 0.5): ');
dtTsign=input('step di incremento per il dtTsign1(suggerito tra 1
e 5): ');
arrivo_P=input('tempo di arrivo delle onde P, in secondi
(suggerimento: mantenersi 1 o 2 secondi prima del vero arrivo
delle P): ');
ntarrivo_P=arrivo_P/dt;
fin_t2_iniz=input('lunghezza del segnale significativo, in
secondi (suggerimento: subito dopo la fase a max ampiezza delle
onde S): ');
ntfin_t2_iniz=fin_t2_iniz/dt;
fin_t1_iniz=fin_t2_iniz-dt1;
in_t2=in_t1+dt2;
t3=durata_eff; %durata effettiva del segnale, delimita la fine
del terzo intervallo%
tolleranza=(((max(abs(acc)))/1000)*scala)  %tolleranza per la
ricerca del punto di contatto tra r2 e r3 (1/1000 di acc max)%
tipo_reg_1=input('definizione tipologia di correzione sul primo
tratto (0 - t1); (1=retta per origine 2=nessuna correzione): ');

durata_controllo=fin_t2_iniz; %lunghezza del segnale su cui
effettuo il controllo sull'andamento della polarità dello
spostamento corretto: per default posta uguale alla lunghezza del
segnale significativo, ma può esser modificata variando la riga
del comando
tempo_percontrollo=input('per quanti secondi l andamento del
segnale in spostamento corretto non deve variare polarità: ');
```

Logica di ottimizzazione per la correzione del segnale

Il programma implementato mantiene sostanzialmente la modalità analitica del programma originale, in cui i valori di t_1, t_2 (e quindi t_{int}) e durata effettiva definiscono le spezzate di retta il cui valore viene sottratto al segnale integrato in velocità, prima di integrarlo nuovamente ed ottenere lo spostamento corretto. A differenza del programma originale, il programma implementato prevede che i valori di t_1 e t_2 varino secondo cicli nidificati all'interno di una griglia di dimensioni variabili.

Per ogni valore assunto da t_1 e t_2, il programma implementato prende in considerazione il segnale corretto in spostamento e calcola i valori dei coefficienti angolari nei seguenti tratti:

- "origine - arrivo_P" ➔ (coefficiente *x0P_disp*);
- "arrivo_P – fin_t2", dove fin_t2 è il valore assunto temporaneamente dalla lunghezza significativa del segnale ➔ (coefficiente *xPTsign_disp*);
- "origine - fin_t2" ➔ (coefficiente *x_sign*).

Per ogni soluzione che soddisfi i parametri di controllo sul segno della traccia in spostamento corretta, il programma implementato seleziona le soluzioni che rispettino una condizione di "forzatura" sul valore assunto dai coefficienti angolari *x0P_disp* e *xPTsign_disp*, di modo che risultino sufficientemente piccoli.
Testando il programma su alcune tracce disponibili, si è optato di definire la condizione di "forzatura" attraverso il comando logico

```
If
(((abs(x0P_disp))<=2*tolleranza)&((abs(xPTsign_disp))<=2.8*toller
anza))
```

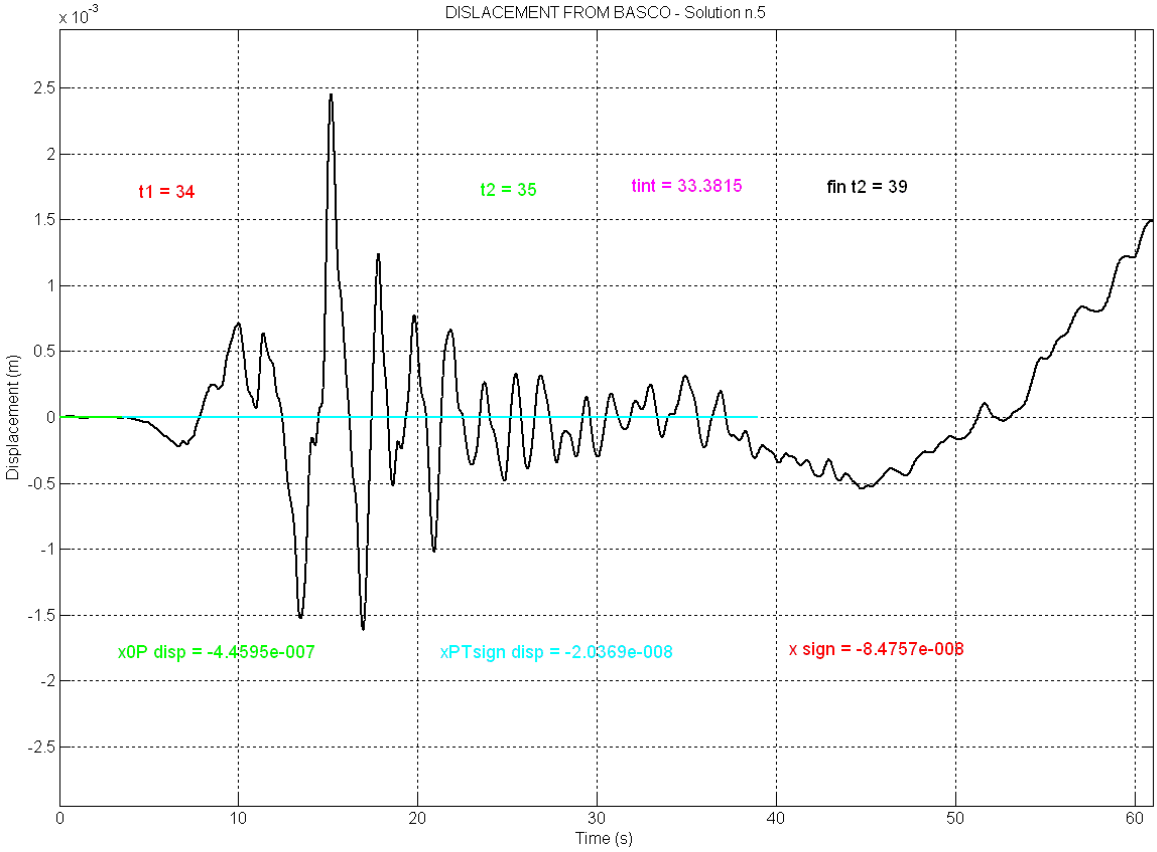
Ovviamente tale condizione può esser modificata agendo sulla riga di comando ed in funzione della "sensibilità" dell'operatore.

Purtroppo è difficile fornire un valore che si adatti ad ogni segnale analizzato, poiché la condizione di "forzatura" risente dell'influenza della portante del noise e dell'ampiezza stessa del segnale. In generale, si è notato che tale valore può esser messo in relazione al parametro *tolleranza* che era già stato definito nel programma originale.

Tra tutte le soluzioni temporanee che rispettino la condizione di "forzatura", il programma seleziona la soluzione ottimale, corrispondente a quella associata al minimo valore assunto dal coefficiente angolare *x_sign*.

Tutte le soluzioni temporanee individuate vengono graficate insieme ai valori temporanei assunti da t_1, t_2, t_{int}, *fin_t2, x0P_disp, xPTsign_disp* ed infine *x_sign*.
Al termine del processo di ottimizzazione, l'operatore può scegliere di accettare la soluzione ottimale individuata, oppure può selezionare la soluzione temporanea che preferisce. In alternativa, l'utente può scegliere di non accettare nessuna soluzione e quindi uscire dal programma.

Un esempio di soluzione finale individuata è descritto nella figura seguente.



Come emerge dalla figura, per quanto si riesca ad ottimizzare la correzione per un tratto sufficientemente lungo del segnale, ne permane una porzione che si discosta troppo dalla linea dello zero, per cui si deve prevedere l'eventualità di limitare la durata del segnale in spostamento (circa 35 secondi per il caso mostrato in figura).