Manuale codice "VATT DMST"

Questo documento illustrerà brevemente il funzionamento e la struttura del codice MATLAB per la valutazione di performance di una turbina marina ad asse verticale (VATT) mediante teoria DMST con aggiunta di sottomodelli per tener conto di fenomeni idrodinamici ulteriori.

Per la teoria del modello, si rimanda al lavoro di tesi di Stefano Deluca scaricabile sul portale ETD dell'Università di Pisa (Link).

Tutte le unità adottate in questo lavoro sono sempre SI, *m* per le lunghezze, *s* per il tempo, *kg* per la massa.

Si consiglia di leggere il manuale nella sua interezza prima di effettuare simulazioni, possibilmente visionando il codice sorgente allo stesso tempo.

Prerequisiti

Il programma è stato sviluppato e testato su sistema operativo **Windows 10 (64 bit)**. L'uso su altre piattaforme potrebbe richiedere modifiche al codice.

È necessario installare MATLAB R2018b con pacchetti:

- MATLAB Coder 4.1
- Parallel Computing Toolbox 6.13
- Image Processing Toolbox 10.3 (solo per simulazioni MIT)

Successivamente bisogna installare:

- MATLAB Support for MinGW-w64 C/C++ Compiler (Link)

L'uso di versioni diverse non è stato testato.

Come lanciare una simulazione normale

Questo paragrafo è volutamente discorsivo. Si rimanda all'analisi dei singoli file per informazioni sulla sintassi degli script.

1. Aprire il file *init_input.m* e definire le opzioni della simulazione.
2. Lanciare la funzione *dmst_vatt.m* specificando il profilo di velocità da usare, per simulazioni 3D, oppure semplicemente il valore di velocità del flusso, per simulazioni 2D. Facoltativamente, definire il nome del file di output.
3. Realizzare i plot desiderati usando i dati di output.

Come lanciare una simulazione con i dati MIT

Al momento sono disponibili due set di dati posizionati in due cartelle.

- Hz600mN010mw (griglia con risoluzione 600 m)
- Hz200mN010mw (griglia con risoluzione 200 m)

Il file con i dati è *out.nc*.

1. Per preparare i dati alla simulazione DMST, avviare lo script *MIT_read_nc_data.m* modificando la variabile `pe_file` al suo interno con il percorso al file .nc desiderato. Verrà prodotto, insieme ad altri file, il file *%nome_file%_processed.mat.*
2. Per avviare la simulazione MIT, aprire lo script MIT_run_simulation.m e modificare la variabile `nome_file` con il file *%nome_file%_processed.mat* creato precedentemente (senza estensione).

3. Modificare la variabile `sim_step` per impostare la risoluzione desiderata.
4. Avviare lo script.
5. I risultati complessivi sono salvati in *%nome_file%_sub_finished.mat.* Sono anche presenti i file delle singole simulazioni in una sottocartella.

Struttura dei file

In questo paragrafo verranno investigati i singoli script, di cui verranno spiegati i rispettivi input e output. Non si entrerà nei meriti del codice in maniera troppo tecnica in quanto gli script sono sufficientemente commentati. Per informazioni sugli script nella cartella *func*, data la loro natura molto specifica, si consiglia di leggere direttamente il codice sorgente.

vatt_dmst.m

Questo file è la function da cui lanciare le simulazioni.

Sintassi

```
[data_post, data_geom, data_vel, data_out_geom, data_out, data_dyn, sim_input, sim_settings] = vatt_dmst(vel_input, [output_file], [tsr_override]);
```

Output		
Nome	*Tipo*	*Descrizione*
`data_post`	Struttura	Contiene vettori utili per l'output
`data_geom`	Struttura	Contiene grandezze informazioni sulla geometria calcolata del rotore
`data_vel`	Struttura	Contiene grandezze relative al flusso in ingresso
`data_out_geom`	Matrice	Contiene grandezze relative alla geometria del rotore e del flusso calcolate su ogni cella del rotore. Dimensioni: nz*n_ring*16
`data_out`	Matrice	Contiene grandezze di output calcolate su ogni cella del rotore. Dimensioni: nz*n_ring*8
`data_dyn`	Matrice	Contiene grandezze relative alla routine di stallo dinamico per ogni cella del rotore. È una matrice nz*n_ring*8
`sim_input`	Struttura	Contiene impostazioni della simulazione
`sim_settings`	Struttura	Contiene informazioni sui sottomodelli attivi

Input		
Nome	*Tipo*	*Descrizione*
`vel_input`	Matrice/ Scalare	Contiene informazioni sul flusso indisturbato. • *Simulazioni 2D* Scalare della velocità. • *Simulazioni 3D* Matrice di 2 colonne. La prima indica le posizioni z a cui la velocità viene misurata da 0 (pelo libero del mare) a $-\infty$ (fondale). La seconda, il valore di velocità.
`output_file`	Stringa	Opzionale. Indica il file di output dove salvare le simulazioni. Se non fornito, viene generato un nome dal programma.
`tsr_override`	Scalare	Opzionale. Imposta un TSR diverso da quello specificato in `init_input.m`

Per un esempio su come avviare una simulazione, fare riferimento allo script *ESEMPIO_lanciasim_singola.m* e *ESEMPIO_dmst_optimizer.m*

I nomi delle variabili all'interno delle strutture sono piuttosto intuitivi. In caso di dubbio, risalire dal codice sorgente alla grandezza calcolata.

Ora verranno illustrate le grandezze calcolate nelle matrici, riferite alla cella in posizione azimutale *i*, variabile fra 1 e n_ring, e posizione verticale *k*, variabile fra 1 e nz.

data_out(k,i,x)			
1	Fattore di induzione assiale non corretto	5	Fattore di perdita alle pute
2	Fattore di induzione assiale a monte, se la cella è in downstream	6	CL senza perdite alle punte
3	CL con perdita alle punte	7	CD senza perdite alle punte
4	CD con perdita alle punte	8	Fattore di induzione assiale dopo correzione espansione streamtubes

data_out_geom(k,i,x)					
1	R	Raggio	9	$\cos(\alpha)$	Coseno dell'angolo di attacco
2	$\cos(\theta)$	Coseno di pos azimutale	10	α [deg]	Angolo di attacco
3	$\sin(\theta)$	Seno di pos azimutale	11	α_{virt}	Angolo di attacco virtuale
4	W_0	Componente orizzontale di vel relativa	12	Δz	Altezza della cella
5	W_1	Componente verticale di vel relativa	13	θ	Posizione azimutale
6	$ModW$	Modulo della vel relativa	14	TSR_{loc}	TSR locale nella cella
7	Re_c	Numero di Reynolds	15	U	Componente orizzontale di vel assoluta
8	$\sin(\alpha)$	Seno dell'angolo di attacco	16	V	Componente verticale di vel assoluta

La definizione della matrice `data_dyn(k,i,x)` dipende dal tipo di modello di stallo dinamico adottato. Si invita ad ispezionare *dmst_calc.m* per maggiori informazioni. Queste informazioni non dovrebbero essere particolarmente rilevanti ai fini dell'output del codice DMST.

init_input.m

Questo file è lo script che definisce le impostazioni della simulazione a livello di geometria di turbina, tipo di flusso, discretizzazione del rotore, ecc. Seguire i commenti nel file per configurare la simulazione come si desidera.

Sintassi

```
[sim_settings, sim_input] = init_input
```

Per gli output si rimanda al paragrafo *vatt_dmst.m*.

init_geom.m

Questo file si occupa di realizzare la discretizzazione spaziale del rotore in nz*n_ring celle.

Sintassi

```
[data_geom] = init_geom(sim_input)
```

Per gli input e output si rimanda al paragrafo *vatt_dmst.m*.

init_vel.m

Questo file si occupa di applicare il flusso indisturbato ad ogni cella del rotore.

Sintassi

```
[data_vel] = init_vel(sim_input, data_geom, vel_input)
```

Per gli input e output si rimanda al paragrafo *vatt_dmst.m*.

dmst_update.m

Questo file gestisce la singola iterazione del solutore DMST. Nello specifico, separa le informazioni disponibili per ogni piano di turbina *k* e successivamente risolve in parallelo i singoli piani di turbina (sezione `parfor`).

Sintassi

```
[data_out_geom, data_out, data_dyn] = dmst_update(sim_settings, sim_input, data_geom, data_vel)
```

Per gli input e output si rimanda al paragrafo *vatt_dmst.m*.

Si sottolinea che la variabile interna `dyn_input` è dichiarata come variabile di tipo `persistent` e pertanto continua ad esistere fra le varie chiamate di *dmst_update.m.*

dmst_par_loop.m

Questo file gestisce la soluzione del singolo piano di turbina. Viene risolta prima la parte upstream, trovando il fattore di induzione che fa tornare i bilanci per ogni cella azimutale della prima metà di turbina, poi c'è l'eventuale correzione di espansione dei tubi di flusso e successivamente viene risolto il downstream.

Sintassi

```
[geom_out_data, out_data, dyn_data] = dmst_par_loop(sim_settings, sim_input, dmst_input)
```

Input		
Nome	*Tipo*	*Descrizione*
`dmst_input`	Struttura	Struttura che contiene informazioni sul singolo piano da risolvere

Per i restanti input e output si rimanda al paragrafo *vatt_dmst.m*.

dmst_calc.m

Questo file risolve il singolo streamtube per il valore del fattore di induzione fornito.

Sintassi

```
[f_eff, data_geom, data_out, data_dyn] = dmst_calc(sim_settings, sim_input, dmst_input, pos_theta, a, a_upstream)
```

Output		
Nome	*Tipo*	*Descrizione*
`f_eff`	Scalare	Differenza fra forza di spinta calcolata con teoria Actuator Disk e Blade Element

Input		
Nome	*Tipo*	*Descrizione*
`pos_theta`	Scalare	Indice della posizione azimutale
`a`	Scalare	Fattore di induzione da adottare
`a_upstream`	Scalare	Fattore di induzione della cella a monte, se applicabile

Per i restanti input e output si rimanda al paragrafo *vatt_dmst.m*.

dmst_post.m

Questo file crea vettori utili per l'output

Sintassi

```
post_data = dmst_post(sim_input, out_geom_data, geom_data, out_data, data_vel)
```

Per gli input e output si rimanda al paragrafo *vatt_dmst.m*.

dmst_plot_update.m

Questo file crea e aggiorna i plot di interesse ad ogni iterazione.

Sintassi

```
dmst_plot_update(fig_out, data_post)
```

Input		
Nome	*Tipo*	*Descrizione*
`fig_out`	Figura	Figura dove creare/aggiornare plots

Per gli input e output si rimanda al paragrafo *vatt_dmst.m*.

Future work

Segue una lista di possibili spunti per il miglioramento del codice, in ordine di crescente difficoltà di implementazione (circa), con relativi commenti.

1. **Cambiare il database di dati di riferimento per profili in accordo con (Link).**
   Bisogna valutare se effettivamente ne valga la pena, ma è molto semplice da fare, bisogna solo modificare *static_data.m* e il sorgente della routine di stallo dinamico con i nuovi valori.
2. **Raffinare modello di curvatura di streamlines**
   Si consiglia di effettuare nuove simulazioni CFD 2D con vario numero di pale, TSR, ecc. In ognuna bisognerà impostare un report per ogni timestep della quantità *Velocity Angle* su una circonferenza centrata nell'origine (dov'è il centro della turbina) di raggio pari al raggio nominale della turbina. Ogni report dovrà essere mediato nel tempo (per usare valori del velocity angle privi di fluttuazioni). L'idea è ricavare un database di velocity angle al variare di posizione azimutale, TSR e numero di pale ed eventualmente altro. Successivamente il fattore $f_{sc}(\theta)$ dovrà essere ricavato da una interpolazione lineare in più dimensioni (ispirarsi al modello di perdite alle punte nuovo).
3. **Raffinare modello di espansione degli streamtube**
   Migliorare la giustificazione fisica del modello corrente o trovarne uno nuovo. Sicuramente migliorare il fitting effettuato sul modello attuale.
4. **Passare ad una discretizzazione verticale non più uniforme, con infittimento verso le punte.**
   Potrebbe essere piuttosto ostico, soprattutto perché bisognerebbe cambiare molte formule in giro per il codice.
5. **Implementare effetti di curvatura del flusso.**
   Bisogna prima capire perché il modello presente non funziona per codici DMST ma va bene per UDF su Fluent. Se non dovesse andar bene, va reimplementato in altro modo.
6. **Migliorare modello di perdite alle punte**
   Servono dati da simulazioni CFD 3D, possibilmente dipendenti anche dalla posizione azimutale. Bisogna capire quanto è importante. Non particolarmente difficile di per sé, ma richiede simulazioni CFD molto lunghe.