POLITECNICO DI MILANO

V Facoltà di Ingegneria
Corso di Laurea Specialistica in Ingegneria Informatica

Sistemi per la costruzione automatica di simulatori di impianti industriali – Il caso di un impianto a ciclo combinato -

Relatore: Chiar.mo Prof. Sergio BITTANTI
Correlatori: Ing. Luigi BISONE
Ing. Antonio DE MARCO
Ing. Antonio GUAGLIARDI
Ing. Carlo SANDRONI

Tesi di Laurea di:
Arianna COMI
Matr. 720923

Anno accademico 2010-2011

Ringraziamenti

I primi ringraziamenti che voglio esprimere vanno all'Ing. Antonio De Marco che si è mostrato sempre molto disponibile e con il suo contributo ha permesso la realizzazione del lavoro.
Un ringraziamento all'Ing. Luigi Bisone che con la sua enorme esperienza informatica è stato di fondamentale aiuto e mi ha permesso di migliorare le mie conoscenze in questo campo.
Grazie all'Ing. Carlo Sandroni e all'Ing. Antonio Guagliardi che con il loro aiuto hanno contribuito al progetto.
Ringrazio anche il Professor Sergio Bittanti per avermi dato l'opportunità di svolgere questo lavoro, che reputo molto utile per la mia formazione professionale.
Desidero infine ringraziare i miei genitori che mi hanno sempre sostenuto in tutto il percorso di studi, e tutte le persone che mi sono state vicine in questi anni.

Abstract

Questa tesi è stata svolta in collaborazione con RSE – Ricerca sul Sistema Energetico S.p. A., sede di Milano.

L'obiettivo del lavoro è duplice: da un lato mettere a punto una procedura per la costruzione automatica di impianti complessi di tipo termo-idraulico a partire dai modelli dei singoli componenti, e dall'altro costruire – grazie al software realizzato – un simulatore di un impianto a ciclo combinato.

Il sistema è formato da:

- un software, detto **assiematore**, che produce il modello dell'impianto
- una libreria, che contiene i modelli dei componenti primitivi
- un solutore, che risolve il sistema di equazioni e permette di effettuare la simulazione

L'utente non dovrà costruire direttamente il sistema complessivo della rete: quando si desidera realizzare il modello di un impianto, per prima cosa occorre definire i modelli dei suoi singoli componenti. Solo se tali modelli non sono già presenti nella libreria, sarà necessario scrivere il codice C relativo, seguendo precise regole, descritte nella tesi.
Inoltre occorre definire le connessioni fra tutti gli elementi. Per far ciò bisogna realizzare un disegno dell'impianto, poiché la costruzione automatica è basata sulla topologia del modello.

Il disegno, definito nel linguaggio SVG, comprende la rappresentazione grafica dei vari elementi e i collegamenti tra di essi; l'assiematore, tramite un'analisi del disegno, esegue gli assegnamenti tra le variabili di elementi diversi e costruisce, grazie anche alle informazioni contenute nella libreria dei modelli, il codice C dell'impianto globale.
Il prodotto finale dell'assiematore viene poi integrato con il solutore, per risolvere il sistema di equazioni e testare il funzionamento dell'impianto.

Con il sistema proposto si è costruito il modello di un impianto a ciclo combinato, a partire dai modelli fisici, basati sui principi primi e sulle equazioni costitutive, dei singoli elementi che lo compongono.
In particolare, l'impianto studiato comprende circa trenta elementi, molti dei quali uguali tra loro, quindi la scelta di costruire i singoli moduli e poi assiemarli automaticamente ha reso meno oneroso il lavoro di modellizzazione.

Per simulare l'impianto è stato infine necessario effettuare un'identificazione dei parametri di funzionamento , sia statici che dinamici, a partire da dati di progetto e prove sperimentali.

INDICE

Indice delle figure

Introduzione

L'obiettivo del lavoro è duplice: da un lato mettere a punto una procedura per la costruzione automatica di impianti complessi di tipo termo-idraulico a partire dai modelli dei singoli componenti, e dall'altro costruire – grazie al software realizzato – un simulatore di un impianto a ciclo combinato.

Nella simulazione del comportamento delle reti elettriche, occorre ovviamente disporre dei simulatori dei vari impianti che forniscono potenza in rete. A questo fine occorre che i modelli dei singoli impianti di generazione siano abbastanza semplici e tarabili agevolmente a partire da registrazioni delle varie variabili nel normale funzionamento di impianto o nel corso di prove ad hoc richieste dal gestore della rete. Attualmente, nella rete elettrica italiana, gli impianti di generazione di maggior diffusione sono quelli a ciclo combinato a tre livelli di presssione (si tratta di impianti in cui vengono utilizzate sia turbine a gas sia turbine a vapore).

La tesi può essere divisa in tre parti:

1. Il Capitolo 1 descrive la metodologia per costruire il software utilizzabile per lo sviluppo dei simulatori di impianto. Vengono precisate le regole per la definizione dei singoli elementi costituenti gli impianti. Inoltre viene analizzato il problema della risoluzione del sistema di equazioni DAE del modello globale, presentando il metodo utilizzato.

2. Nei Capitoli 2-4 vengono descritti i singoli componenti di un ciclo combinato con i relativi modelli matematici, ottenuti grazie alle equazioni fisiche descrittive, e la loro implementazione in codice C, rispettando le regole descritte nel primo capitolo.
   La procedura e i modelli proposti vengono adoperati per costruire il modello del ciclo combinato

3. Nel Capitolo 5 si affronta il problema dell'identificazione dei parametri di funzionamento statici e dinamici.
   Per prima cosa si considera il funzionamento statico dell'impianto e si trovano i valori dei coefficienti di scambio e dei parametri di efflusso e di rendimento delle turbine.
   Noti questi valori è possibile eseguire una prima simulazione, assegnando ai parametri dinamici i valori ricavati dalla letteratura.
   Successivamente si propone una strategia per l'identificazione dei parametri dinamici (costanti di tempo) a partire da dati reali di funzionamento.

Capitolo 1. Sviluppo di un software per la costruzione automatica di un modello numerico di impianti termoidraulici complessi, nel caso cicli combinati

L'obiettivo del lavoro è realizzare un simulatore per verificare il funzionamento di impianti termoidraulici.

Il simulatore comprende:

- un software chiamato assiematore, che realizza automaticamente il modello di un impianto, a partire da quello dei singoli elementi che lo compongono e dalla topologia dell'impianto da realizzare
- una libreria di moduli, che contiene la descrizione dei singoli componenti, usati per costruire modelli più complessi. Per ogni elemento si realizza un codice C, con le definizioni delle strutture e delle funzioni del modello e un file grafico SVG che rappresenta l'elemento considerato
- un solutore, che risolve il sistema di equazioni algebrico-differenziale del modello complessivo

L'utente, quindi, non dovrà costruire direttamente il sistema complessivo della rete: quando si desidera realizzare il modello di un impianto, per prima cosa occorre definire i modelli dei suoi singoli componenti. Solo se tali modelli non sono già presenti nella libreria, sarà necessario scrivere il codice C relativo, seguendo precise regole, descritte nei capitoli successivi.
Inoltre occorre definire le connessioni fra tutti gli elementi. Per far ciò bisogna realizzare un disegno dell'impianto, poiché la costruzione automatica è basata sulla topologia del modello.

Il disegno, definito nel linguaggio SVG, comprende la rappresentazione grafica dei vari elementi e i collegamenti tra di essi; l'assiematore, tramite un'analisi del disegno, esegue gli assegnamenti tra le variabili di elementi diversi e costruisce, grazie anche alle informazioni contenute nella libreria dei modelli, il codice C dell'impianto globale.
Il prodotto finale dell'assiematore viene poi integrato con il solutore, per risolvere il sistema di equazioni e testare il funzionamento dell'impianto.

In figura 1.1 è riportato lo schema del lavoro.

Input

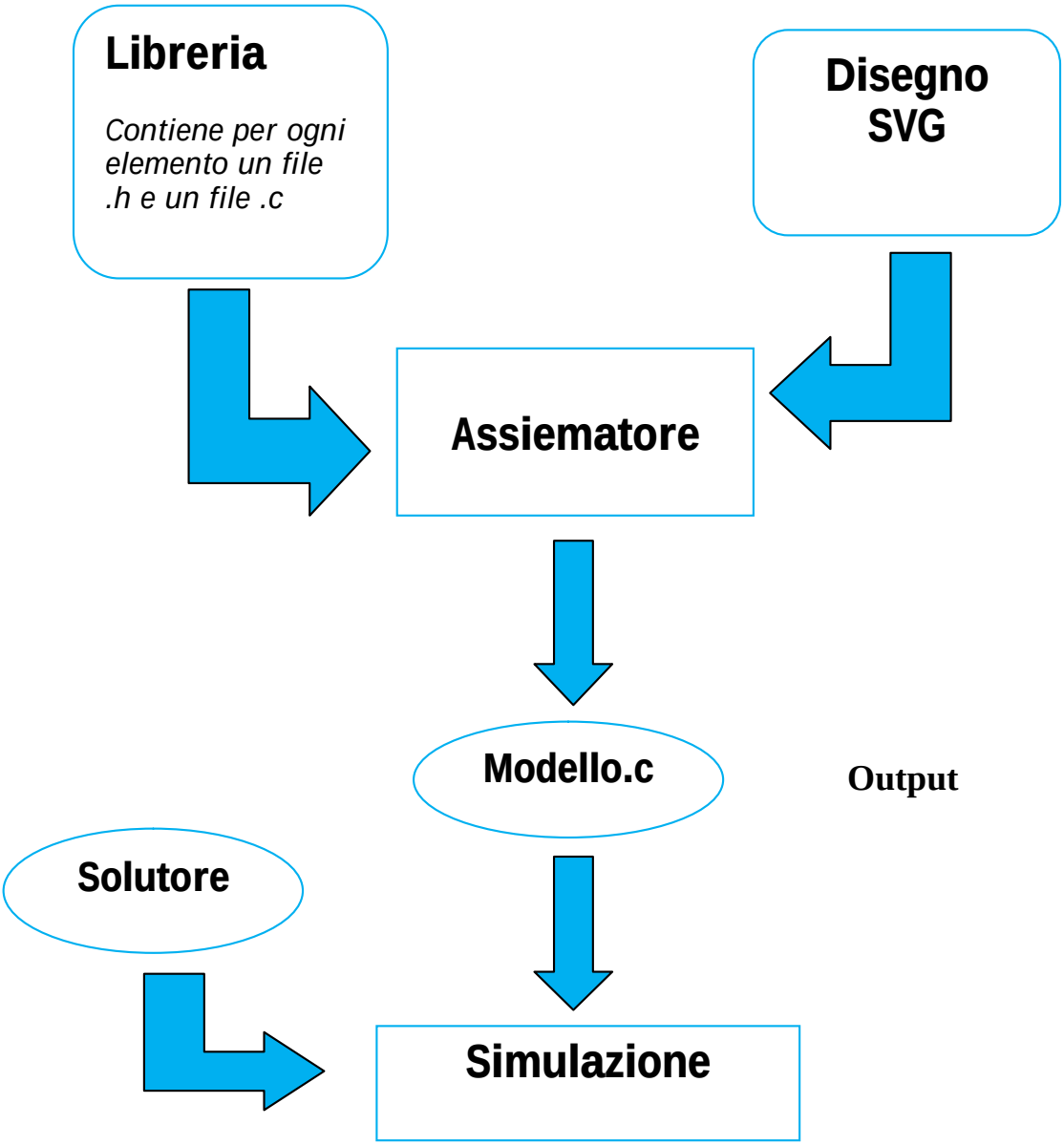

Figura 1.1 Schema del simulatore

1.1 Assemblaggio automatico di modelli

Dovendo adottare un linguaggio per descrivere la topologia di un modello costruito dall'assemblaggio di elementi selezionati tra quelli presenti nell'apposita libreria, l'adozione di SVG è apparsa immediatamente come la scelta più opportuna per almeno due buoni motivi:

- SVG è un linguaggio grafico vettoriale che si presta ad essere immediatamente tradotto in immagini praticamente da tutti i *browser*; questo comporta indubbi vantaggi per la documentazione del modello stesso così come per la sua costruzione, che può essere effettuata in maniera interattiva tramite un opportuno CAD;

- SVG deriva da XML di cui conserva la caratteristica di "estendibilità", cioè si presta ad essere integrato con parole chiave estranee al contesto grafico ma utili per la gestione di ulteriori contenuti necessari alla definizione completa dei modelli.

Quest'ultimo fatto ha consentito di specificare uno schema di descrizione grafica dell'impianto da realizzare, che potesse essere interpretato dal programma "*assiematore*" per produrre un modello in codice C, formato da diverse funzioni. Tale libreria risulta immediatamente integrabile in un apposito solutore che ne permette l'evoluzione nel tempo quindi, in definitiva, la simulazione dell'impianto.

1.2 Struttura del file SVG per la descrizione del modello

Il modello, come un qualunque documento XML, deve essere racchiuso in un elemento radice, nel nostro caso l'elemento **<svg>**, caratterizzato da una serie di attributi invarianti per la nostra applicazione.
Nella figura 1.2 è riportato lo scheletro del documento, svuotato cioè dei contenuti, per evidenziarne la struttura.

```
<svg>
 <script/>
 <defs id="LIBRERIA">
  . . . . . .          dichiarazioni elementi libreria
  <g id="INTERFACCE">
  . . . . . .          dichiarazioni tipi interfacce
  </g>
 </defs>
<g id="SCHEMA">
  <rect/>              sfondo (utile all'interazione)
  <g id="COLLEGAMENTI">
  . . . . . .          elenco collegamenti
  </g>
  <g id="ELEMENTI">
  . . . . . .          elenco elementi
  </g>
```

```
 </g>
</svg>
```

Figura 1.2 Scheletro del documento XML

I diversi blocchi sono raggruppati in un elemento **<g>**, tag che in SVG rappresenta un elemento di raggruppamento, a scopo prevalentemente semantico, che permette di definire attributi specifici per ogni blocco.

Il primo elemento figlio della radice è un elemento **<script>** che specifica il nome del file **libreria.js** contenente una libreria di funzioni Javascript che, oltre a consentire un minimo di interazione con l'immagine del modello, ne permettono una scrittura notevolmente semplificata, provvedendo automaticamente, nella fase di caricamento in memoria, al completamento di parti omesse in quanto ridondanti nella particolare specificazione ma necessarie al browser per il corretto rendering dello schema.

Segue un elemento **<defs>,** tag usato per racchiudere la parte del file SVG della libreria, cioè quella parte messa a fattor comune di tutte le applicazioni successive di questa metodologia. I primi elementi figli sono definizioni meramente grafiche finalizzate ad una maggior gradevolezza nell'estetica dello schema.
Segue la serie di interfacce ai componenti della libreria modellistica, ciascuna rappresentata da tre elementi di tipo **<symbol>**, immediatamente distinguibili per il loro attributo **id** che contiene il nome generico del componente, per esempio **evaporatore**, di volta in volta con il prefisso **B_**, **I_** e **D_**.

Ogni tag "symbol" racchiude la definizione di un componente della libreria.

```
<symbol id="B_evaporatore" width="85" height="150">
 <g>
  . . .  elementi grafici
 </g>
</symbol>
```

Figura 1.3 Tag symbol id= "B_evaporatore"

Il primo di questi tre elementi, nell'esempio con **id="B_evaporatore"**, rappresenta graficamente l'icona che distingue l'elemento nello schema. Non vi sono particolari vincoli a come l'immagine venga realizzata, naturalmente purché si rimanga aderenti alle caratteristiche di SVG. Si hanno comunque a disposizione un insieme esaustivo di primitive grafiche vettoriali oltre alla possibilità di incorporare immagini da file.

```
<symbol id="I_evaporatore" width="85" height="150"
  xmce:equazioni="5" xmce:stati="2" xmce:uscite="3"
  xmce:prefisso="evap">
  <circle cx="30" cy="5" r="5" xmce:tipo="acqvap"/>
  <circle cx="5" cy="35" r="5" xmce:tipo="acqvap">
   <xmce:passante xmce:simbolo="p" xmce:terminale="0"/>
  </circle>
  <circle cx="5" cy="120" r="5" xmce:tipo="arifum"/>
  <circle cx="80" cy="60" r="5" xmce:tipo="arifum">
   <xmce:passante xmce:simbolo="w" xmce:terminale="2"/>
   <xmce:passante xmce:simbolo="p" xmce:terminale="2"/>
  </circle>
 </symbol>
```

Figura 1.4 Tag symbol id= "I_evaporatore" e rappresentazione grafica dell'elemento

Il secondo elemento, sempre nell'esempio con **id="I_evaporatore"**, è finalizzato alla descrizione del modello matematico. In particolare quindi provvede a definire il numero di equazioni e di stati del modulo e le modalità di interfacciamento con gli altri componenti del modello. Le caratteristiche della descrizione matematica sono fissate da una serie di attributi che non hanno uno specifico significato grafico e sono distinguibili dagli altri elementi del linguaggio in quanto appartengono al namespace **xmce:** che li caratterizza.

Possono quindi essere presenti gli attributi

- **xmce:equazioni** definisce il numero di equazioni che descrivono il componente
- **xmce:stati** definisce il numero di stati interni del componente
- **xmce:prefisso** definisce il nome generico del componente che dovrà concorrere a formare i nomi di tutte le function e delle strutture informatiche riguardanti il componente in oggetto, così che possano essere opportunamente istanziate nel codice prodotto dall'assiematore.
- **xmce:ingressi** numero di variabili in ingresso all'elemento
- **xmce:uscite** numero di variabili in uscita dall'elemento

Il contenuto di **id="I_evaporatore"** è rappresentato da uno o più **<circle>**, tanti quanti sono i terminali del componente stesso. Questi cerchi hanno un significato topologico in quanto consentono all'assiematore di identificare la struttura dei collegamenti tra i vari componenti, quindi normalmente non vengono visualizzati. Se con il mouse ci si sovrappone all'elemento si ottiene, come evidente nell'icona riportata in figura 1.4, la visualizzazione della numerazione progressiva dei terminali stessi nel colore del tipo di collegamento specificato dall'attributo **xmce:tipo**. Nello stesso tempo viene visualizzato il valore dell'attributo **xmce:prefisso**, nel caso **evA**.
Questo elemento **<circle>**, può inoltre contenere uno o più elementi **<xmce:passante>** che specificano come una grandezza del collegamento riferito al terminale in questione ed identificata dalla propria sigla contenuta nell'attributo **xmce:simbolo** sia da considerarsi la stessa dell'omologa grandezza riferita al terminale il cui numero progressivo è specificato dall'attributo **xmce:terminale**. L'associazione tra le sigle delle grandezze ed il tipo di

collegamento, che sarà descritta in seguito, può essere controllata sia da un CAD che dall'assiematore in modo da prevenire dichiarazioni non corrette. Così come può essere controllata la correttezza dell'attestazione di un collegamento ad un terminale, verificando che questi siano compatibili.

```
<symbol id="D_evaporatore">
 <text xmce:gruppo="P">
  <tspan>Pressione vapore in ingresso terminale 0</tspan>
  . . .
  <tspan>Massa metallica dei riser</tspan>
  . . .
  <tspan>Calore specifico del metallo</tspan>
  . . .
  <tspan>Sezione del drum</tspan>
 </text>
 <text xmce:gruppo="U">
  <tspan>Pressione corpo cilindrico, Pc</tspan>
  . . .
  <tspan>Temperatura fumi in uscita terminale 3</tspan>
 </text>
</symbol>
```

Figura 1.5 Tag symbol id= "D_evaporatore"

Il terzo elemento con **id="D_evaporatore"**, completa la caratterizzazione di un componente dell'impianto. Ha infatti lo scopo di fornire al CAD un template per assistere l'operatore nella compilazione interattiva di un file, con lo stesso nome del modello ma estensione XML, contenente, per ogni istanza di un elemento, la lista dei parametri e dei dati necessari per l'inizializzazione di ciascun componente. In particolare l'elemento contiene degli elementi **<text>** distinti dall'attributo **xmce:gruppo**. Il gruppo identificato dalla lettera **"P"** racchiude i prompt per sollecitare l'introduzione di parametri e dati, il gruppo identificato dalla lettere **"U"** è relativo alle sigle delle grandezze in uscita. Infine il gruppo identificato dal valore **"I"** dell'attributo, nell'esempio non presente in quanto non pertinente, permette di definire gli ingressi.

L'ultimo elemento della libreria è identificato da **id="INTERFACCE"** che contiene le descrizioni di tutti i tipi di collegamento presenti nel modello.

```
<g xmce:tipo="acqvap" xmce:colore="darkblue">
 <g xmce:simbolo="w" xmce:grandezza="portata"/>
 <g xmce:simbolo="p" xmce:grandezza="pressione"/>
 <g xmce:simbolo="h" xmce:grandezza="entalpia"/>
</g>
```

Figura 1.6 Tag con descrizione delle interfacce

Ciascun tipo di collegamento è rappresentato da un elemento **<g>** caratterizzato dagli attributi

- **xmce:tipo** che assegna al collegamento un nome da cui venga identificato, per esempio nella definizione del terminale
- **xmce:colore** che assegna al colore con cui viene tracciato il collegamento l'insieme di grandezze tipiche del collegamento stesso

L'elemento contiene a sua volta l'elenco delle grandezze costituenti il collegamento stesso, ciascuna rappresentata da un elemento **<g>** caratterizzato dagli attributi

- **xmce:simbolo** che assegna alla grandezza il simbolo con cui questa viene identificata, per esempio nella definizione delle grandezze passanti
- **xmce:grandezza** che assegna al simbolo una descrizione minimamente più estesa

L'elemento **<g>** immediatamente al seguito della libreria, con **id="SCHEMA"**, contiene per l'appunto lo schema dell'impianto suddiviso in due elementi figli, sempre di tipo **<g>**, uno con **id="COLLEGAMENTI"** e l'altro con **id="ELEMENTI".**

```
<polyline id="L17" points="235,65 235,40 255,40" stroke="darkblue"/>
```

Figura 1.7 Descrizione dell'elemento polyline, per i collegamenti

I collegamenti tra componenti sono rappresentati da una serie di elementi di tipo **<polyline>** dove gli attributi

- **id** contiene un identificatore che viene visualizzato quando con il mouse ci si sovrappone al collegamento stesso
- **points** contiene le coordinate dei vertici della spezzata che costituisce il collegamento; la coppia iniziale e la coppia finale vengono utilizzate per identificare elemento e suo terminale di partenza ed elemento e suo terminale di arrivo del collegamento stesso
- **stroke** contiene il nome del colore che identifica il collegamento; il suo valore deve essere uguale al valore dell'attributo **xmce:colore** dell'interfaccia dei terminali che sono collegati da questa linea

Ciascun componente dell'impianto è rappresentato da un contenitore **<g>** caratterizzato dall'attributo **id** che identifica in maniera univoca la particolare istanza nel modello del componente specifico, nell'esempio **evA**, oltre che dalla coppia di coordinate **x** e **y** che posizionano il vertice alto a sinistra dell'elemento nello schema. Tale identificatore contribuirà a comporre i nomi delle variabili che l'assiematore definisce per la specifica istanza del componente.

```
<g id="ELEMENTI">
 . . .
 <g id="evA" x="641" y="85">
  <use xlink:href="#B_evaporatore"/>
  <use xlink:href="#I_evaporatore"/>
 </g>
```

```
 . . .
</g>
```

Figura 1.8 Tag id=”ELEMENTI”

Il contenuto dell’elemento è invece costituito da due elementi **<use>** che con i loro attributi **xlink:href** indicano rispettivamente l’elemento di libreria che definisce l’aspetto grafico del componente ed il suo interfacciamento.

1.3 L’assiematore

Questo programma elabora il file SVG contenente la libreria dei componenti e la topologia del modello e produce un file di codice C, genericamente **modello.c**, contenente i moduli necessari per inizializzare le strutture dati ed effettuare le chiamate di inizializzazione e di evoluzione previste da un generico ambiente per la simulazione di processi industriali. Viene prodotto anche un file di intestazione **elementi.h** indispensabile per definire correttamente l’interfacciamento tra i moduli e l’ambiente di evoluzione.
Oltre al file SVG precedentemente descritto l’assiematore prevede in ingresso, e se non lo trova ne crea uno precompilato, un file con lo stesso nome, tranne che per l’estensione **.xml**, contenente valori di dati e parametri necessari all’inizializzazione di tutti gli elementi. In questa struttura devono inoltre essere contenute le sigle delle variabili di ingresso ed uscita tramite cui interagire con gli elementi durante l’evoluzione temporale.
Altri prodotti dell’elaborazione sono un file con la stessa radice ma estensione **.txt** contenente la topologia del modello come desunta dallo schema grafico, utile a fini diagnostici.
Pure assai utile risulta un file grafico, anch’esso prodotto automaticamente con suffisso **_mappa** del nome del file contenente il modello, che riporta una completa descrizione grafica dello Jacobiano prodotto.
Questo file può essere visualizzato in un qualsiasi browser, meglio se con la contemporanea presenza nella directory della libreria Javascript **mappaJacobiano.js** che ne implementa le funzioni di pan e zoom, molto utili nel caso di mappe molto estese.
Nella figura 1.9 è esemplificata con grande evidenza l’utilità di tale funzione, in presenza di matrici di grandi dimensioni e quindi difficilmente rappresentabili nella loro globalità: il tassello con riquadro rosso rappresenta la visualizzazione ingrandita di un particolare della mappa.

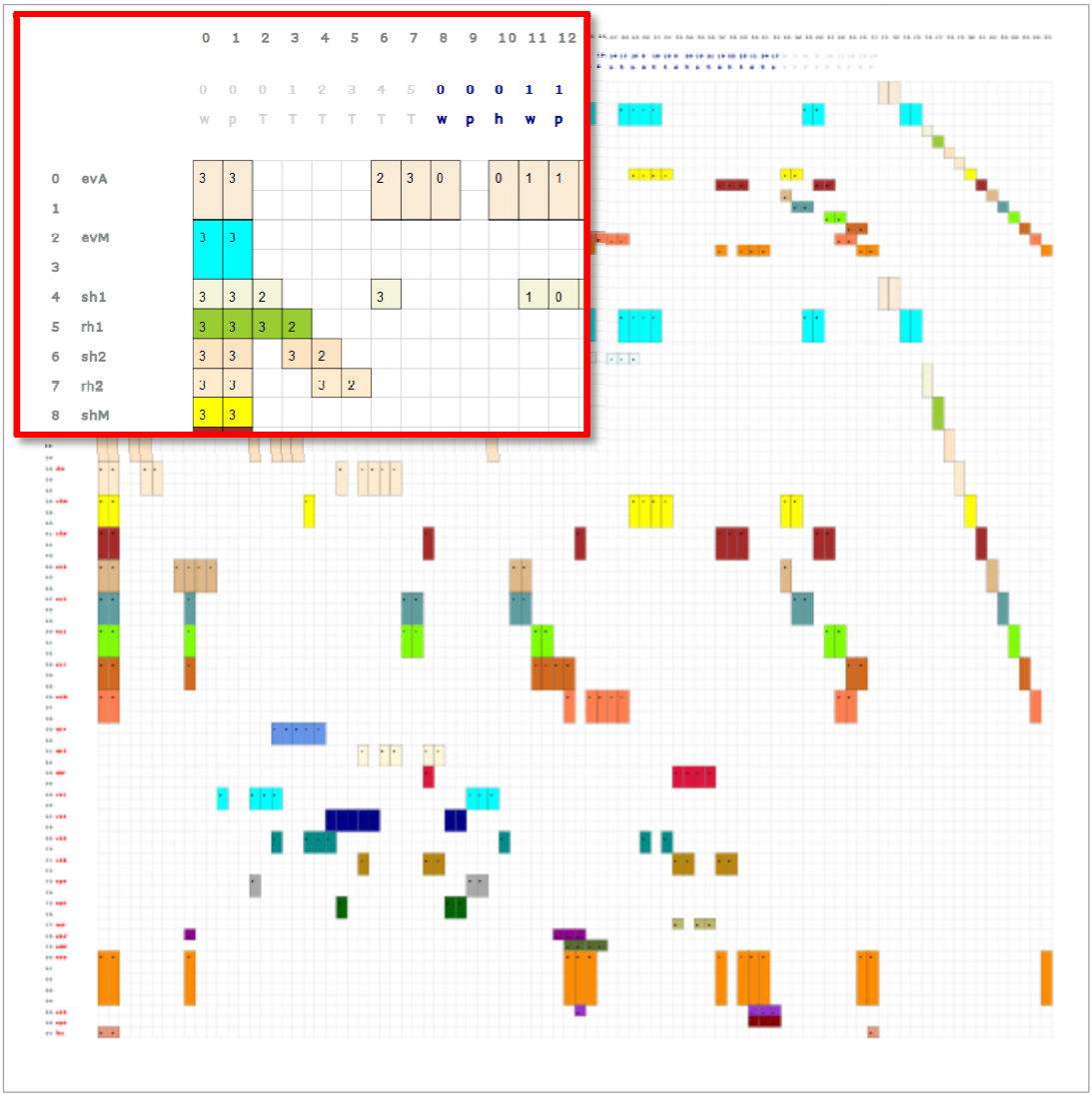

Figura 1.9 Rappresentazione grafica dello jacobiano del sistema, realizzata dall’assiematore

La prima riga dell’intestazione riporta la numerazione delle colonne della matrice dello jacobiano. Seguono, sulla seconda e sulla terza riga, rispettivamente il numero progressivo e l’identificatore della variabile terminale cui la colonna si riferisce, nel colore che individua la relativa tipologia di collegamento. Le ultime colonne sulla destra, dove l’elencazione delle variabili si interrompe, sono relative agli stati interni degli elementi.

Il corpo della matrice è numerato progressivamente sulla sinistra con a fianco l’indicazione dell’identificatore dell’elemento cui le equazioni, una per riga, appartengono. Sono riportate prima le equazioni differenziali, con il nome dell’elemento in grigio, poi le equazioni algebriche, con nomi in rosso. Le celle sottese da equazioni e variabili terminali per uno specifico elemento, rappresentate nello stesso colore, sono identificate dal numero progressivo del terminale all’interno dell’elemento stesso.
Volendo effettuare una sintesi delle operazioni svolte dal programma assiematore è importante ricordare innanzitutto la definizione della struttura globale che comprende le singole strutture, tipiche di ogni componente, una per ogni istanza del componente stesso.
Una seconda operazione non banale consiste nella definizione dell’insieme delle variabili di collegamento tra i componenti del modello con un’analisi topologica del disegno effettuata con procedure ricorsive attraverso gli elementi ove questi dichiarino grandezze “passanti”. Si evita così l’allocazione di ulteriori colonne dello Jacobiano quando una variabile di un

collegamento non venga modificata passando ad un collegamento adiacente attraverso un elemento che, per l’appunto, la dichiari “passante” cioè non modifichi la variabile stessa. Si riesce così a limitare notevolmente le dimensioni del problema a tutto vantaggio dei tempi di esecuzione.
Per concludere resta da citare la produzione del codice per l’allocazione di tutte le variabili e le funzioni, richiamate dal solutore, per l’inizializzazione e l’evoluzione del calcolo e l’interazione con ogni componente del modello rappresentata da una serie di istanze alle specifiche funzioni del relativo componente generico.

1.4 Descrizione del codice prodotto dall’assiematore

L’assiematore produce un file C, chiamato **modello.c**, che contiene il modello dell’impianto che si vuole simulare.
Il modello è definito da una struttura **Model**, M, che contiene lo jacobiano, le equazioni, le costanti di tempo, le variabili algebriche e gli stati.
Inoltre contiene una struttura **Elementi,** definita dall’assiematore in base alla topologia. Questa struttura contiene per ogni elemento presente nell’impianto un riferimento alla struttura del modello corrispondente, in modo da poter memorizzare tutte le variabili necessarie per definire l’elemento.

Il modello dell’impianto è definito dalle seguenti funzioni

```
void init(Model *M);
void setJac(Model *M, double t);
void setJacU(Model *M);
void setS(Model *M);
void setT(Model *M);
void setU(Model *M);
void setY(Model *M);
void terminate(Model *M);
```

La funzione **init** è la più importante, perché permette di allocare in modo corretto tutte le variabili necessarie.
Per prima cosa si allocano le variabili dello jacobiano, delle variabili algebriche e degli stati, e la matrice delle costanti di tempo, tutte riferite al sistema complessivo.
Per l’allocazione dinamica occorre conoscere la dimensione dei vettori da allocare.
Per le variabili algebriche e gli stati, l’assiematore ricava la dimensione dei vettori calcolando il numero di variabili algebriche e di stati di ciascun elemento e sommando i valori ottenuti. Si dichiara quindi un vettore Z, con tale dimensione; il vettore è diviso in due parti, la prima contiene le variabili algebriche, la seconda gli stati.
Si usa lo stesso procedimento per trovare la dimensione del vettore delle equazioni algebriche e differenziali e si dichiara il vettore S, diviso in due parti, la prima riservata alle equazioni differenziali, la seconda a quelle algebriche.
Per lo jacobiano e le costanti di tempo, il numero di righe è la somma delle equazioni dei singoli elementi, il numero delle colonne è dato dalla somma degli stati e delle variabili algebriche del modello globale. Mentre il numero degli stati è uguale alla somma degli stati di ciascun elemento, per le variabili algebriche occorre considerare solo le *vere* variabili algebriche, cioè quelle indipendenti. Lo jacobiano globale deve risultare quadrato, per consentire la risoluzione del sistema di equazioni del modello complessivo.
Quando si collegano due elementi, le loro variabili terminali coincidono, quindi nel calcolo delle colonne bisogna aggiungere per ogni variabile terminale collegata una sola colonna invece che due.

Dopo aver calcolato il numero delle colonne, sono note le dimensioni e si dichiarano le variabili J, per lo jacobiano globale, e T per le costanti di tempo.
Lo jacobiano è composto da 4 blocchi: due blocchi per le derivate delle equazioni differenziali rispetto alle variabili algebriche e agli stati, e due per le derivate delle equazioni algebriche rispetto alle variabili algebriche e agli stati, come riportato in figura 1.10.

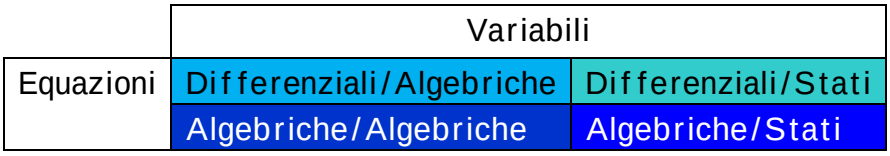

Figura 1.10 Divisione in blocchi dello jacobiano

Tutte le variabili definite rappresentano il modello completo; ogni singolo modulo occupa una propria posizione nel modello complessivo, quindi scrive o legge solo in determinate posizioni delle matrici delle variabili globali.
In base alla topologia dell'impianto, l'assiematore calcola la posizione occupata da ciascun elemento e passa al modulo che lo rappresenta le variabili globali con indice corretto, in modo che ciascun elemento modifichi solo le posizioni che gli corrispondono.
Questi riferimenti vengono passati alla funzione *init* del modulo corrispondente, per l'inizializzazione *locale*, come mostrato nella figura 1.11.

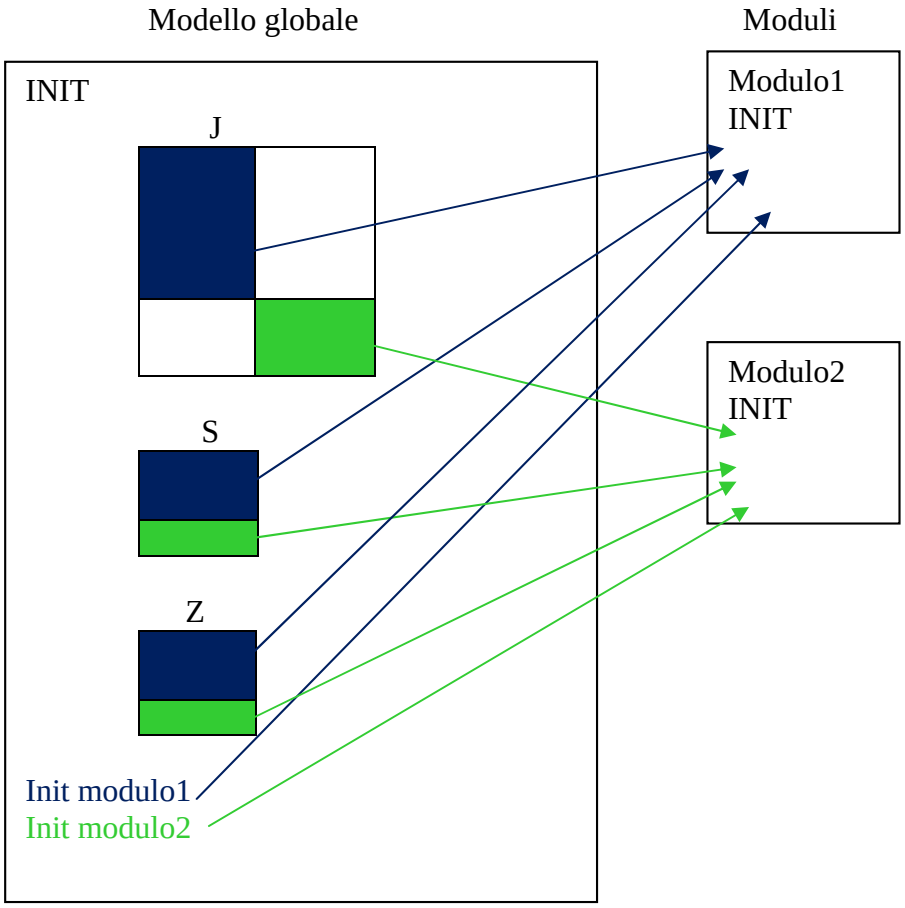

Figura 1.11 Inizializzazione variabili da modello globale ai singoli moduli

Le altre funzioni definiscono l'evoluzione del modello, calcolando lo jacobiano rispetto alle variabili e agli ingressi, i secondi membri delle equazioni, le costanti di tempo e definendo gli ingressi (*SetU*) e le uscite (*SetY*) globali.
Ognuna effettua una chiamata alle corrispondenti funzioni di tutti i moduli presenti nell'impianto.

Per esempio, la funzione setU ha il seguente codice:

```
void setU(Model *M)
{
getInput_sgtw(&M->elem.Mfi1,M->u+0);
getInput_evap(&M->elem.MevA,(double *)NULL);
getInput_evap(&M->elem.MevM,(double *)NULL);
getInput_vghe(&M->elem.Msh1,(double *)NULL);
getInput_vghe(&M->elem.Mrh1,(double *)NULL);
getInput_vghe(&M->elem.Msh2,(double *)NULL);
getInput_vghe(&M->elem.Mrh2,(double *)NULL);
getInput_vghe(&M->elem.MshM,(double *)NULL);
getInput_vghe(&M->elem.MshB,(double *)NULL);
getInput_tval(&M->elem.MtAP,M->u+2);
getInput_tval(&M->elem.MtMP,M->u+3);
getInput_tval(&M->elem.MtBP,M->u+4);
getInput_coal(&M->elem.Mc01,(double *)NULL);
getInput_coal(&M->elem.Mc02,(double *)NULL);
getInput_coal(&M->elem.Mc03,(double *)NULL);
getInput_coal(&M->elem.Mc04,(double *)NULL);
getInput_svhw(&M->elem.Map4,M->u+5);
getInput_svhw(&M->elem.Map5,M->u+7);
getInput_svhp(&M->elem.Mte0,M->u+9);
getInput_svhw(&M->elem.Mte1,M->u+10);
getInput_sgtp(&M->elem.Mf01,M->u+12);
getInput_pval(&M->elem.MpAP,M->u+13);
getInput_pval(&M->elem.MpMP,M->u+15);
getInput_lghe(&M->elem.MeMP,(double *)NULL);
getInput_lghe(&M->elem.MeAp,(double *)NULL);
getInput_lghe(&M->elem.Me3,(double *)NULL);
getInput_lghe(&M->elem.Me2,(double *)NULL);
getInput_lghe(&M->elem.Me1,(double *)NULL);
getInput_evbp(&M->elem.MevBP,(double *)NULL);
}
```

dove

- getInput_# : funzione che definisce gli ingressi del modulo, il simbolo # è sostituito dal nome del modulo corrispondente, ad esempio *vghe* per lo scambiatore vapore, *evbp,* per l'evaporatore a bassa pressione
- il primo parametro della funzione è la struttura che definisce il singolo modulo
- il secondo parametro è la posizione occupata dal modulo nel vettore degli ingressi del modello complessivo

Se un modulo ha due ingressi, che corrispondono al primo e al secondo ingresso globale, il secondo parametro della funzione sarà *M->u+0.* L'assiematore dalla topologia ricava che il modulo ha due ingressi, quindi al modulo successivo verrà passato il riferimento a partire da *M->u+2,* come mostrato in figura 1.12.
Se un modulo non ha ingressi, il secondo parametro vale *NULL.*

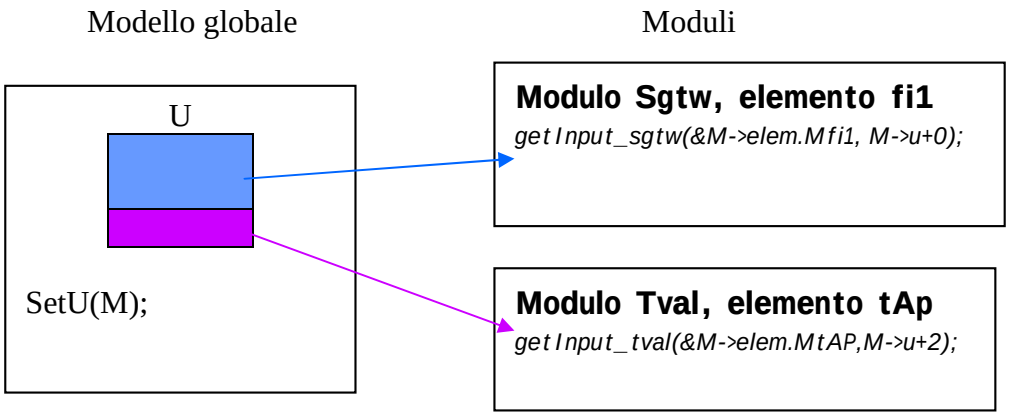

Figura 1.12 Definizione degli ingressi dal modello globale ai singoli moduli

L'ultima funzione, *terminate*, libera la memoria precedentemente allocata, chiamando la funzione *terminate* di tutti i moduli.

Si può concludere che il codice prodotto dall'assiematore comprende una parte relativa all'inizializzazione e una relativa all'evoluzione dell'impianto; entrambe queste parti richiamano i singoli moduli presenti per descrivere il modello globale.

Il codice viene poi integrato nel file ***solutore.c*** che si occupa della simulazione dell'impianto, risolvendo il sistema di equazioni e trovando i valori delle variabili ad ogni passo di tempo.

In figura 1.13 è riportato lo schema delle chiamate tra i vari componenti.

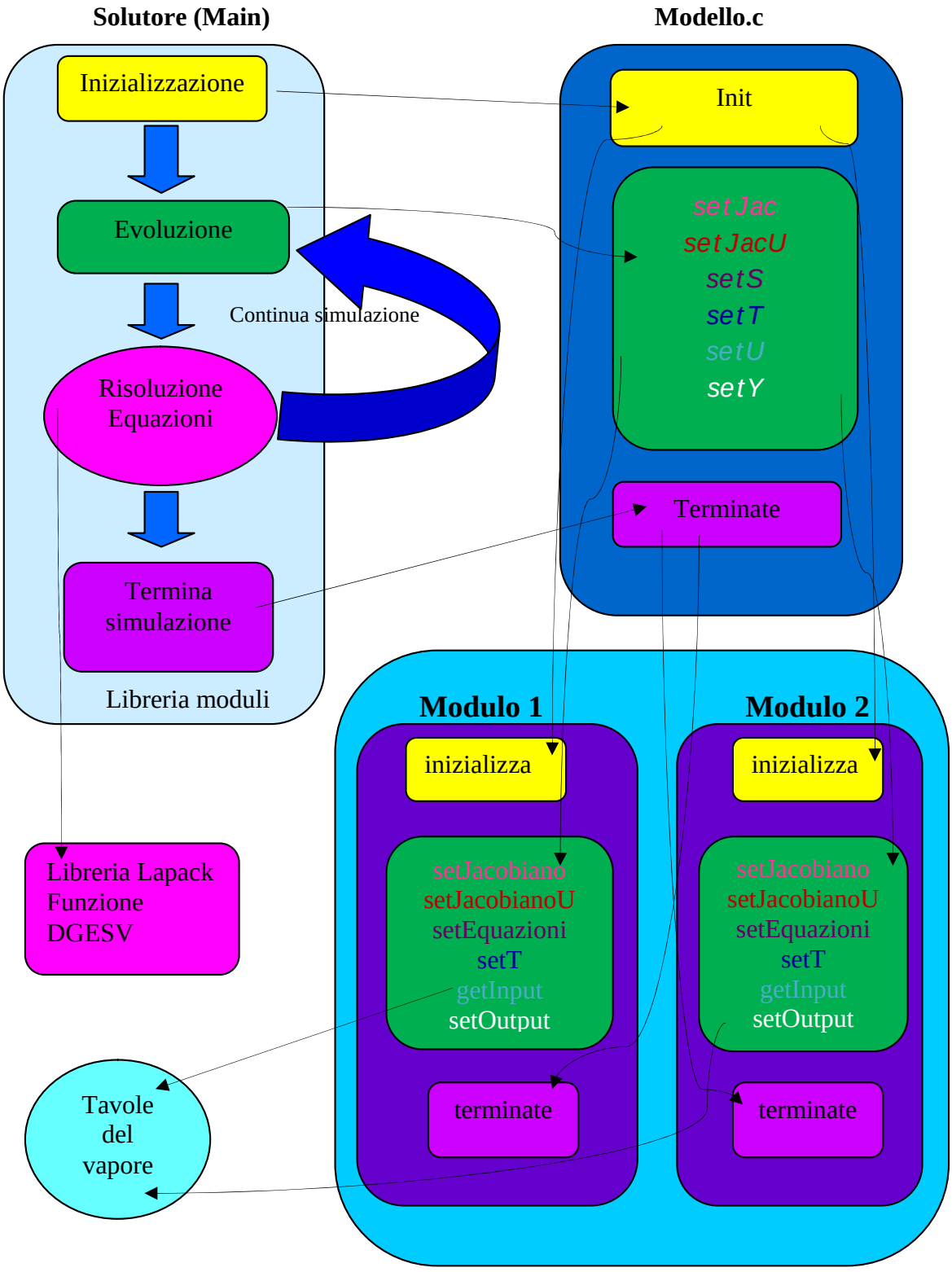

Figura 1.13 Schema delle chiamate tra i componenti del simulatore

La parte relativa al solutore è la stessa per ogni modello che si vuole realizzare, quello che cambia è il file modello.c, prodotto dall'assiematore, a partire dalla topologia descritta nel file SVG.

La struttura basata su chiamate a funzioni permette di usare i file anche in ambienti di simulazione diversi da quello descritto, ad esempio in Simulink, inserendo le chiamate alle funzioni in una S-function.

Si parte con l'inizializzazione dei moduli, quindi si passa alla fase di evoluzione, al termine della quale si incrementa il tempo di simulazione di un intervallo pari al passo di tempo. Se non si è raggiunto il tempo di fine simulazione si prosegue ripetendo l'evoluzione, altrimenti si termina.

Oltre al solutore, al modello e alla libreria dei moduli, sono presenti altre due librerie: le tavole del vapore, usate all'interno dei moduli per calcolare le grandezze termodinamiche e la libreria matematica Lapack, della quale si usa la funzione DGESV, per risolvere il sistema di equazioni.

Nello scrivere i modelli si è usato un approccio *acausale*, in modo da avere relazioni sempre valide, qualunque sia il modo in cui un elemento viene collegato ad altri.
Questo consente di avere modelli riutilizzabili in contesti diversi, ma porta ad avere un sistema di equazioni DAE (Differential algebraic equation), che comprende appunto sia equazioni algebriche che equazioni differenziali, quindi più complesso da risolvere.

Per capire meglio il funzionamento, si può considerare l'applicazione ad un sistema, composto da due elementi, il primo descritto da un'equazione algebrica, il secondo da un'equazione algebrica e da una differenziale.

Si dovrà quindi risolvere il seguente sistema DAE:

$$\begin{aligned} \tau\dot{x} &= ax + bu \\ cx + dz_1 + eu &= 0 \\ fz_2 + gu &= 0 \end{aligned} \quad \text{(S1)}$$

con

- x: variabile di stato
- z: variabili algebriche
- u: ingressi
- τ: matrice contenente i "termini di accumulo" di massa ed energia
- t: tempo

Dopo aver effettuato le chiamate alle funzioni che eseguono l'inizializzazione, sono noti i valori delle costanti e di tutte le variabili all'istante iniziale.
Si passa quindi alla fase di evoluzione, che tramite le chiamate alle opportune funzioni, descritte di seguito, calcola i secondi membri delle equazioni, le costanti di tempo e lo jacobiano del modello completo.

Si hanno quindi tutte le informazioni per risolvere il sistema S1 e trovare i valori degli stati e delle variabili algebriche all'istante successivo.

Il sistema S1 può essere riscritto nella forma compatta

$$\begin{aligned} 0 &= g(z, x, u) \\ \tau\dot{x} &= f(z, x, u) \end{aligned}$$

con

- f e g: vettori dei secondi membri delle equazioni differenziali ed algebriche

Esistono varie tecniche numeriche per risolvere un sistema di equazioni algebriche e differenziali non lineari; molti di questi metodi, in particolare quello usato in questo lavoro, richiedono il calcolo delle matrici jacobiane associate alle equazioni algebriche e differenziali. Lo jacobiano del sistema assiemato si ricava dall'unione dello jacobiano di tutti gli elementi, considerando le uguaglianze tra le variabili terminali di elementi adiacenti e riducendo così la dimensione del sistema.

Dal sistema

$$0 = g(z, x, u)$$
$$\tau\dot{x} = f(z, x, u)$$

si ottiene il sistema S2

$$\frac{\tau(x_{n+1,j+1} - x_n)}{\Delta t} = f\left(x_{n+1,j+1}, z_{n+1,j+1}, u_n\right)$$
$$0 = g(x_{n+1,j+1}, z_{n+1,j+1}, u_n) \qquad \text{(S2)}$$

Si risolve il sistema tramite una procedura iterativa con il metodo di Eulero all'indietro

$x_j = (x_{n+1})_j$ e $x_{j+1} = (x_{n+1})_{j+1}$,

quindi $\Delta x = x_{j+1} - x_j$

$z_j = (z_{n+1})_j$ e $z_{j+1} = (z_{n+1})_{j+1}$,

quindi $\Delta z = z_{j+1} - z_j$

Alla prima iterazione, con j=0, $x_0 = x_n$ e $z_0 = z_n$

Linearizzando S2 si ha

$$\frac{\tau(x_{j+1} - x_n)}{\Delta t} = f\left(x_{j+1}, z_{,j+1}, u_n\right) + \left(\frac{\partial f}{\partial x}\right)_j \Delta x + \left(\frac{\partial f}{\partial z}\right)_j \Delta z$$
$$-g\left(x_j, z_j, u_n\right) = \left(\frac{\partial g}{\partial x}\right)_j \Delta x + \left(\frac{\partial g}{\partial z}\right)_j \Delta z$$

Inoltre

$$x_{j+1} - x_n = \Delta x + x_j - x_n$$

Si deve quindi risolvere il sistema S3

$$\left(\left(\frac{\partial f}{\partial x}\right)_j - \frac{\tau}{\Delta t}\right)\Delta x + \left(\frac{\partial f}{\partial z}\right)_j \Delta z = -f\left(x_j, z_{,j}, u_n\right) - \frac{\tau}{\Delta t}(x_j - x_n)$$
$$\left(\frac{\partial g}{\partial x}\right)_j \Delta x + \left(\frac{\partial g}{\partial z}\right)_j \Delta z = -g\left(x_j, z_j, u_n\right) \qquad \text{(S3)}$$

Da S3 si ricava che la matrice A dei coefficienti è

$$\begin{matrix} \left(\left(\frac{\partial f}{\partial x}\right)_j - \frac{\tau}{\Delta t}\right) & \left(\frac{\partial f}{\partial z}\right)_j \\ \left(\frac{\partial g}{\partial x}\right)_j & \left(\frac{\partial g}{\partial z}\right)_j \end{matrix}$$

Mentre il vettore b dei termini noti è

$$\begin{matrix} -f(x_j, z_{,j}, u_n) - \frac{\tau}{\Delta t}(x_j - \quad x_n) \\ -g(x_j, z_j, u_n) \end{matrix}$$

Come già detto, il sistema è stato risolto usando la routine DGESV, della libreria Lapack.

Le incognite sono Δx e Δz; trovate le soluzioni si avrà

$$x_{j+1} = \Delta x + x_j$$
$$z_{j+1} = \Delta z + z_j$$

Quindi si incrementa j e si passa all'iterazione successiva; si prosegue finchè non si è raggiunto il numero massimo di iterazioni previsto, oppure finchè non si è ottenuta la convergenza, in base a un valore ε fissato, cioè quando $\sum|\Delta x| + \sum|\Delta z| < \varepsilon$

Ad ogni iterazione è necessario ricalcolare i secondi membri delle equazioni e lo jacobiano, a partire dalle soluzioni trovate.

Ottenuta la convergenza, si ripete l'operazione per tutta la durata della simulazione, sommando all'istante corrente il passo di tempo scelto.

Ad ogni passo di tempo vengono memorizzati i valori delle variabili algebriche e degli stati, in modo da poter visualizzare, tramite dei grafici, il loro andamento nella simulazione.

1.5 Regole per la scrittura di modelli di elementi compatibili con l'uso dell'assiematore

Per realizzare dei moduli compatibili con il processo di assiemamento automatico è necessario seguire alcune regole riguardanti la definizione delle variabili e delle funzioni che descrivono il comportamento fisico del modello.

1.5.1 Definizione delle variabili

Le variabili cui si fa riferimento nel codice rappresentante il generico modello di un componente di impianto possono essere o variabili "locali", il cui valore non deve essere conservato tra un passo di tempo ed il successivo, o variabili che determinano l'evoluzione dell'impianto, che quindi devono essere dichiarate all'esterno, in copia univoca per ogni istanza dell'elemento in questione, in modo da poter essere correttamente referenziate ed inoltre conservare il proprio valore tra un passo di tempo e l'altro.

Quest'ultimo gruppo di variabili deve essere dichiarato all'interno di particolari strutture, definite a priori. Per mantenere coinciso il codice generato automaticamente e favorirne la

comprensione, si è deciso di identificare ciascuna struttura con una sola lettera maiuscola seguita da quattro caratteri alfanumerici che identificano univocamente il tipo di componente di impianto relativo. Nelle righe seguenti tale identificativo è rappresentato dal simbolo **#**

Si è inoltre deciso di usare le seguenti strutture, definite come **typedef**:

1. *Struttura Terminali* **T#** : le variabili definite in tale struttura sono rappresentate a loro volta da strutture che identificano quali variabili vengono utilizzate nel collegamento; ad esempio sono definiti terminali per il collegamento lato vapore, con le variabili entalpia, pressione e portata, e terminali lato fumi, con le variabili temperatura, pressione e portata. Ovviamente è possibile collegare tra loro solo elementi che prevedano tipologie comuni di collegamento.
   Prima di cominciare a scrivere il modello, è fondamentale definire un ordine dei terminali, che sia lo stesso utilizzato nella descrizione grafica, in modo che l'assiematore possa eseguire i collegamenti in modo corretto.
   Inoltre ogni elemento può definire al suo interno dei *passanti* sui terminali, cioè imporre che ci sia un'uguaglianza sulla stessa variabile, per due o più suoi terminali.
   Ad esempio se un elemento prevede 2 terminali, t_0 e t_1, con la stessa portata, nel modello, nella fase di inizializzazione, si dovrà specificare questa uguaglianza.

2. *Struttura Parametri* **P#** : raggruppa le variabili che definiscono i parametri dell'elemento, quali ad esempio, coefficienti di scambio o costanti di tempo. Non sono in generali costanti, ma sono calcolate.

3. *Struttura Costanti* **C#** : raggruppa le variabili che definiscono le costanti fisiche e geometriche dell'elemento, quali ad esempio superfici di scambio, volumi, diametri. Il valore di tali variabili verrà letto da un file XML, durante la fase di inizializzazione dell'impianto.

4. *Struttura Stati* **S#**: contiene le variabili dinamiche del modello; occorre anche definire un vettore *stati*, di dimensione pari al numero degli stati, che contiene gli stati dell'elemento e permette di identificarli tra gli stati del modello globale.
   L'ordine in cui compaiono gli stati in questa struttura è lo stesso che verrà usato per il calcolo dello jacobiano.

5. *Struttura variabili* **V#** : raggruppa tutte le variabili non contenute nelle strutture definite in precedenza ed i cui valori devono essere visibili all'esterno. In tale gruppo rientrano ad esempio il vettore dei secondi membri delle equazioni algebriche e differenziali e la matrice dello jacobiano. Per convenzione, si è deciso di organizzare la matrice in questo modo: le righe corrispondono alle equazioni differenziali e algebriche, le colonne si riferiscono alle variabili rispetto alle quali è calcolato lo jacobiano. Si elencano prima tutti gli stati, quindi le variabili algebriche e infine quelle terminali.

6. *Struttura ingressi* **U#** : contiene gli ingressi esterni del modello, se presenti. Può comprendere variabili di controllo o variabili esogene generiche, ad esempio variabili ambientali.

7. *Struttura uscite* **Y#** : contiene le uscite del modello, cioè le variabili che si vogliono misurare

8. Infine si definisce una struttura **M#**, che contiene tutte le strutture descritte sopra, così da semplificare le chiamate alle funzioni del modello, definite in seguito.

1.5.2 Definizione formale delle funzioni

Ogni modello contiene le seguenti funzioni, che descrivono il suo comportamento fisico.

```
void inizializza_#(M# *M);
void setEquazioni_#(M# *M);
void setJacobiano_#(M# *M);
void setJacobianoU_#(M# *M);
void getInput_#(M# *M, double *in);
void setOutput_#(M# *M, double *out);
void setT_#( M# * M);
void terminate_#(M# *M);
```

Si possono individuare due fasi principali, quella di inizializzazione e quella di evoluzione.

Nella fase di inizializzazione, a partire da valori noti, si calcolano i valori dei parametri del modello, usati poi nella fase successiva. Tutte le operazioni di inizializzazione vengono eseguite all'interno di una funzione, la cui definizione formale è standardizzata in questo modo:

```
void inizializza_#(M# *M);
```

con *M#* struttura descritta in precedenza.

Inoltre, a seconda dell'elemento, tale funzione richiede un diverso numero di parametri.

In generale, considerando un elemento con dinamica, la chiamata di inizializzazione è la seguente:

```
void inizializza_#(double **p,M# *M, double *f,
                   double *g,double **T, double **Jdz,
                   double **Jdx, double **Jaz, double **Jax);
```

- double **p: il vettore p contiene i valori letti da un file di inizializzazione, prodotto automaticamente dall'assiematore.
  I valori presenti in questo file corrispondono alle variabili che devono essere assegnate per poter eseguire in modo corretto l'inizializzazione del modello globale.
  Poiché ogni elemento, come già detto, occupa una precisa posizione nel modello globale, il vettore p conterrà la prima posizione associata all'elemento considerato
- M# *M: struttura che definisce l'elemento, al suo interno contiene tutte le strutture descritte precedentemente
- double *f: vettore che contiene tutte le equazioni differenziali
- double *g: vettore che contiene tutte le equazioni algebriche
- double **T: matrice delle costanti di tempo
- double **Jdz: matrice dello jacobiano differenziale, calcolato rispetto alle variabili algebriche
- double **Jdx: matrice dello jacobiano differenziale, calcolato rispetto agli stati
- double **Jaz: matrice dello jacobiano algebrico, calcolato rispetto alle variabili algebriche

- double **Jax: matrice dello jacobiano algebrico, calcolato rispetto agli stati

  Per i parametri dal terzo al nono, alla funzione viene passata la variabile con l'indice che corrisponde alla posizione che l'elemento considerato occupa nel modello globale.

Nella funzione di inizializzazione si dovranno definire i passanti delle variabili terminali previsti dal modello, come detto in precedenza.

Nella fase di evoluzione, si calcolano i secondi membri delle equazioni algebriche e differenziali e si risolve il sistema lineare, con un metodo appropriato, trovando così il valore degli stati e delle variabili algebriche ad ogni istante di tempo.

La funzione ***setEquazioni_#(M# *M)*** setta i secondi membri delle equazioni, memorizzati negli appositi vettori.

setJacobiano_#(M# *M) e ***setJacobianoU_#(M# *M)*** calcolano lo jacobiano rispetto agli stati, alle variabili terminali ed agli ingressi. Per il calcolo dello jacobiano è importante specificare l'ordine delle righe e delle colonne, così da permettere una costruzione corretta dello jacobiano di tutto l'impianto.

getInput_#(M# *M, double *in) riceve come parametri oltre alla struttura M, anche il vettore *in*, che contiene gli ingressi del modello. Ad ogni ingresso è associata la corrispondente variabile.

setOutput_#(M# *M, double *out) complementare rispetto a quella degli ingressi, riceve il vettore out, assegna ad ogni uscita il valore della corrispondente variabile.

void setT_#(M# * M) è presente solo nei modelli di elementi dinamici, setta il valore delle costanti di tempo, memorizzate in una matrice.

1.6 Esempio di assiemamento di un modello di impianto

Per illustrare concretamente il funzionamento dell'assiematore, facciamo riferimento ad un esempio semplice, ma significativo, il collegamento tra un evaporatore di caldaia e una valvola che rappresenta la condizione di contorno tra evaporatore e turbina. L'evaporatore riceve energia termica dai fumi caldi dei gas di combustione.
I modelli fisici dei due elementi saranno descritti nel capitolo successivo.

Riportiamo lo schema del collegamento in Figura 1.14.

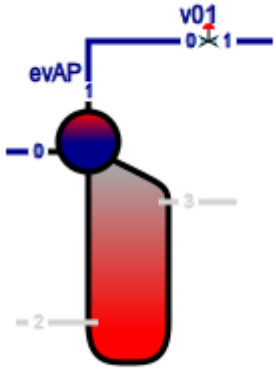

Figura 1.14 Collegamento tra valvola ed evaporatore

Ovviamente le variabili del terminale 1 dell'evaporatore coincidono con le variabili del terminale 0 della valvola.
Nel modello dell'evaporatore si è assunto che la pressione del terminale 0 sia la stessa del terminale 1, ovvero la stessa variabile p_0. In generale, invece, le portate dei due terminali sono diverse, perché si ammettono variazioni di livello nel corpo cilindrico. Per i terminali 2 e 3 (ingresso e uscita fumi), invece, si ha la stessa pressione e la stessa portata, mentre le temperature sono ovviamente diverse, a causa dello scambio termico.
Nel modello della valvola, la portata nel terminale 0 è la stessa di quella nel terminale 1; inoltre sono pure coincidenti le entalpie, dato che si tratta di un modello di valvola isoentalpica.
Il modello globale è costituito dalle equazioni dell'evaporatore e dalle equazioni della valvola, in particolare l'evaporatore presenta una dinamica, mentre la valvola è un elemento algebrico.
Le variabili che interessano sono principalmente la portata di vapore in uscita dall'evaporatore, attraverso la valvola, la pressione dell'evaporatore e il livello.
Tali variabili dipendono dalla posizione della valvola; che è un ingresso (simbolo u) del modello globale, definito dal sistema di controllo.
L'evoluzione delle variabili considerate dipende anche dalla temperatura e dalla portata dei gas di combustione, variabili associate al terminale 2 dell'evaporatore, e dall'entalpia e dalla portata di acqua associate al terminale 0 dell'evaporatore. Si deve anche definire la pressione del terminale 1 della valvola. Tutte queste informazioni rappresentano le condizioni al contorno, che devono essere definite da opportuni elementi con un solo terminale.

Tali elementi si dividono in *rami* e *pozzi*.

L'elemento "pozzo" definisce la portata e l'entalpia, mentre l'elemento "ramo" definisce la pressione.

L'elemento pozzo ha le seguenti equazioni:

$$w_0 - u_w = 0$$
$$h_0 - u_h = 0$$

con
w_0 e h_0 : portata e entalpia del terminale 0 del pozzo
u_w : ingresso del modello globale
u_h: ingresso del modello globale

Ciò significa che fissa sia la portata che l'entalpia dei suoi terminali.

Un elemento ramo, invece, è descritto da un'unica equazione

$$p_0 - u_p = 0$$

con
p_0: pressione del terminale 0 del ramo
u_p : ingresso del modello globale

Quindi tale elemento fissa solo la pressione dei suoi terminale.

Le variabili terminali non fissate da questi elementi verranno imposte dai moduli ai quali sono collegati.
Rami e pozzi completano il modello e permettono di avere un sistema quadrato, che può quindi essere risolto con il metodo descritto al paragrafo 1.4.

Capitolo 2. Descrizione di un impianto a ciclo combinato

Una centrale a ciclo combinato alimentata a gas naturale coniuga in modo ideale i punti di forza di due processi termici: la generazione di corrente elettrica mediante una turbina a gas abbinata a una turbina a vapore. Per indicare questo tipo di centrale viene comunemente impiegato l'acronimo inglese **CCGT** (*Combined Cycle Gas Turbine*).

Circa due terzi dell'intera energia elettrica di una centrale a ciclo combinato sono prodotti dalla turbina a gas. In questa tipologia di centrale viene bruciata una miscela composta da aria e combustibile. I gas caldi generati dalla combustione alimentano la turbina e indirettamente anche il generatore che vi è collegato.
La restante energia elettrica della centrale, ovvero circa un terzo, è prodotta dalla turbina a vapore, che sfrutta i fumi caldi provenienti dalla turbina a gas. Nella caldaia per il recupero del calore, i fumi cedono la propria energia termica all'acqua circolante: l'acqua sotto pressione evapora, provocando un innalzamento della temperatura presente all'interno dell'impianto. Il vapore alimenta la relativa turbina a vapore e conseguentemente anche il generatore ad essa collegato.

I costi di investimento sono relativamente contenuti, poiché i componenti principali sono in larga misura standardizzati. La turbina a gas consente di scegliere una struttura compatta per la centrale e di ridurre al minimo i tempi di costruzione che si aggirano attorno ai due anni e mezzo.
Grazie alla moderna tecnica di combustione è possibile monitorare e ridurre al minimo le emissioni di fumi nocivi. Rispetto a tutti gli altri impianti termici tradizionali, come ad esempio quelli a carbone, le centrali a ciclo combinato sono state progettate per produrre energia elettrica nel massimo rispetto dell'ambiente.

2.1 Funzionamento dell'impianto a ciclo combinato

In figura 2.1 è riportato lo schema di funzionamento di un impianto a ciclo combinato.

L'aria proveniente dall'ambiente viene aspirata con un filtro e compressa nel compressore; arriva quindi alla turbina a gas, dove viene miscelata con gas naturale. A questo punto si verifica la combustione, con la formazione di gas caldi sotto pressione.
I fumi caldi passano poi nella turbina per il recupero del calore e a contatto con l'acqua provocano la sua evaporazione. Il vapore prodotto raggiunge la turbina a vapore e fa girare la turbina, producendo energia meccanica, che viene trasmessa al generatore, che converte l'energia meccanica in corrente elettrica. Il vapore di scarico della turbina passa poi al condensatore e tramite un processo di raffreddamento dell'aria viene trasformato nuovamente in acqua.

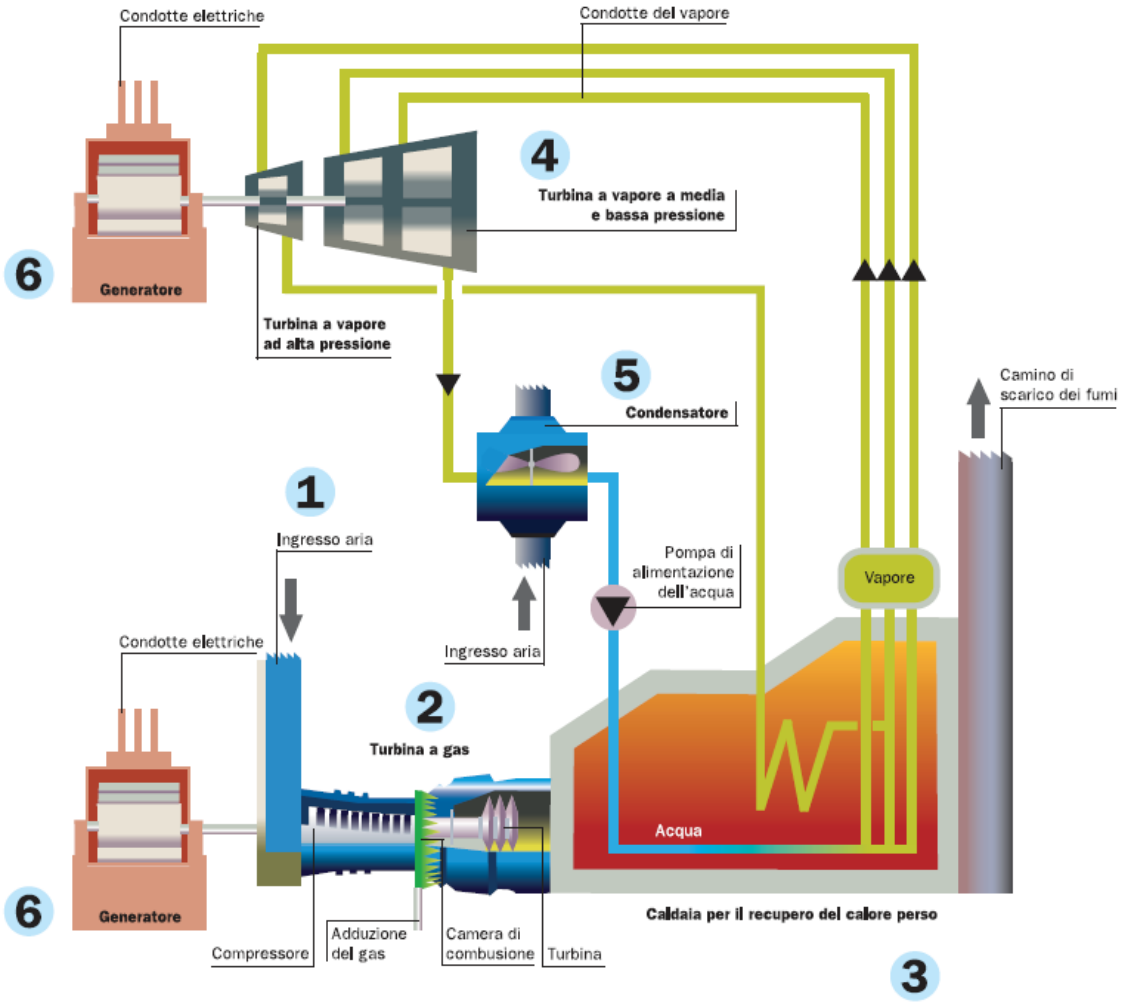

Figura 2.1 Schema di funzionamento di un impianto a ciclo combinato

Gli elementi principali di un ciclo combinato sono

- la turbina a gas
- il generatore di vapore a recupero (GVR)
- la turbina a vapore
- il condensatore
- il degasatore

Nel seguito del capitolo verrà presentata una descrizione di questi componenti.

2.1.1 Turbina a gas

La turbina a gas è considerata il cuore dell'impianto; produce circa i due terzi dell'energia elettrica di una centrale a ciclo combinato (la quantità restante è prodotta dalla turbina a vapore), e rappresenta il primo livello nel processo di produzione di elettricità.

Attraverso un filtro, il compressore della turbina a gas aspira aria dall'ambiente, la comprime, e la convoglia nella camera di combustione. In essa il combustibile confluisce sotto forma di gas naturale e viene bruciato.

Durante questo processo si formano dei gas caldi che, portati alla pressione ambiente nella turbina, si espandono. L'energia liberata viene convertita in un movimento rotatorio meccanico che trasforma l'energia prodotta in corrente elettrica.
Quando lasciano la turbina sotto forma di fumi, i gas caldi raggiungono una temperatura di circa 600°C. Successivamente tale energia termica è ceduta all'acqua nella caldaia per il recupero del calore perso.

2.1.2 Generatore di vapore a recupero

In questa sezione inizia il secondo livello del processo di produzione di elettricità: il ciclo del vapore acqueo. L'acqua sotto pressione viene quindi riscaldata e fatta evaporare.
La caldaia per il recupero del calore comprende tre settori in ognuno dei quali è presente una pressione diversa: una alta, una media e una bassa. La suddivisione in questi tre livelli di pressione consente uno sfruttamento ottimale dell'energia insita nei gas di scarico.
Questo elemento permette di trasferire il calore dei gas di scarico al fluido del circuito acqua/vapore; l'acqua viene riscaldata e fatta evaporare, sfruttando l'energia termica dei fumi di scarico, il vapore acqueo generato passa poi nella turbina e produce elettricità.

Il generatore di calore a recupero è formato da:

- l'*economizzatore,* uno scambiatore di calore che preriscalda l'acqua liquida proveniente dalle pompe di alimento, prima che entri nella parte di generatore destinata alla vaporizzazione, usando il calore dei gas di scarico
- l'*evaporatore* in cui avviene l'evaporazione dell'acqua e la separazione della fase gassosa da quella liquida. È un anello percorso dal fluido con circolazione naturale, cioè circolazione dovuta alla differenza di massa volumica tra l'acqua sottoraffreddata che scende nei tubi di caduta e la miscela di acqua e vapore che risale nei tubi vaporizzanti.
  È composto da:
  - corpo cilindrico (drum), in cui si verifica la separazione tra acqua e vapore; per effetto della gravità la parte liquida, più pesante, si raccoglie sul fondo del cilindro, mentre quella gassosa, più leggera, si concentra nella parte superiore
  - tubi di caduta (downcomers) che costituiscono la prima metà dell'anello di ricircolazione. In essi entra l'acqua sottoraffreddata accumulata sul fondo del drum, che viene poi trasportata all'inizio dei risers
  - tubi vaporizzatori (risers) che costituiscono la seconda metà dell'anello di ricircolazione. Ricevono in ingresso l'acqua uscente dai downcomers e la riportano nel drum. In questi tubi si verifica il passaggio di fase da acqua a vapore, perché sono disposti lungo la parete della camera di combustione e ricevono per irraggiamento il calore prodotto. Nella prima parte dei tubi c'è acqua in condizioni di sottoraffreddamento, nella seconda parte c'è una miscela di vapore e acqua in condizioni di saturazione
- il *surriscaldatore,* scambiatore di calore che serve a portare il vapore uscente dal drum in condizioni di surriscaldamento

In tutti gli elementi lo scambio di calore avviene in controcorrente, per favorire lo scambio energetico.
Nel circuito acqua-vapore, l'acqua, dopo aver aumentato la sua temperatura attraverso l'economizzatore, raggiunge il corpo cilindrico, in uno stato di sottoraffreddamento.

Il corpo cilindrico contiene acqua sul fondo e vapore in cima; al fondo è collegato l'anello di ricircolazione, formato come detto, dai risers e dai downcomers.
L'acqua presente sul fondo entra nei downcomers, per differenza di densità e risale verso il corpo cilindrico; grazie all'energia che viene trasmessa al fluido, si verifica il passaggio da fase liquida a vapore.
L'acqua presente nei risers che non evapora torna al corpo cilindrico, depositandosi sul fondo, mentre il vapore si accumula nella parte superiore.
Il vapore viene prelevato dal corpo cilindrico e passa nel surriscaldatore, dove viene aumentata la temperatura del vapore, prima che entri in turbina.

2.1.3 Turbina a vapore

La turbina a vapore è la macchina termica che chiude il circuito acqua/vapore del ciclo combinato.
È suddivisa, come la caldaia, in tre livelli: alta, media e bassa pressione. Il rispettivo settore della caldaia a vapore fornisce alla turbina il giusto tipo di vapore che viene quindi espanso nella turbina attraverso il passaggio ad un livello di pressione inferiore.
Il vapore fa girare la turbina e produce energia meccanica, convertita poi in corrente elettrica dai generatori.
Attraverso condotte dal diametro di alcuni metri, il vapore viene poi convogliato nel condensatore con raffreddamento ad aria.

2.1.4 Condensatore

Il condensatore è l'elemento terminale del ciclo acqua-vapore, è un serbatoio che opera come uno scambiatore di calore; in esso i gas di scarico provenienti dalla turbina subiscono il processo di condensa e tornano allo stato liquido.
Dei grandi ventilatori apportano aria ambiente che raffredda il vapore fino a trasformarlo in acqua, tramite un processo di condensazione.
Quindi la pompa di alimentazione dell'acqua riporta questo condensato alla caldaia, dove il ciclo ricomincia.

2.1.5 Degasatore

Il degasatore, evaporatore di bassa pressione, serve per separare dall'acqua i gas disciolti, dannosi perché a causa della presenza di ossigeno si potrebbero verificare fenomeni di corrosione nelle tubazioni e nelle zone ad alta temperatura della caldaia.
È un serbatoio a pressione maggiore di quella atmosferica, in cui entra l'acqua di alimento che arriva dal surriscaldatore, quindi in condizioni prossime alla saturazione, e viene miscelata con il vapore spillato, proveniente dalla turbina di media pressione.
Poiché l'acqua ha un peso maggiore, si deposita sul fondo del serbatoio, insieme al vapore che sta condensando, mentre gli eventuali gas disciolti, più leggeri, salgono verso l'alto.
Nella parte superiore c'è poi una valvola di sfogo che permette la fuoriuscita dei gas dannosi.

Nel prossimo capitolo verranno descritti in modo dettagliato i modelli fisici degli elementi realizzati, il generatore di vapore a recupero e la turbina a vapore.

Capitolo 3. Modello del generatore di vapore a recupero e della turbina a vapore di un ciclo combinato

Il modello dell'impianto a ciclo combinato realizzato è riportato in figura 3.1.

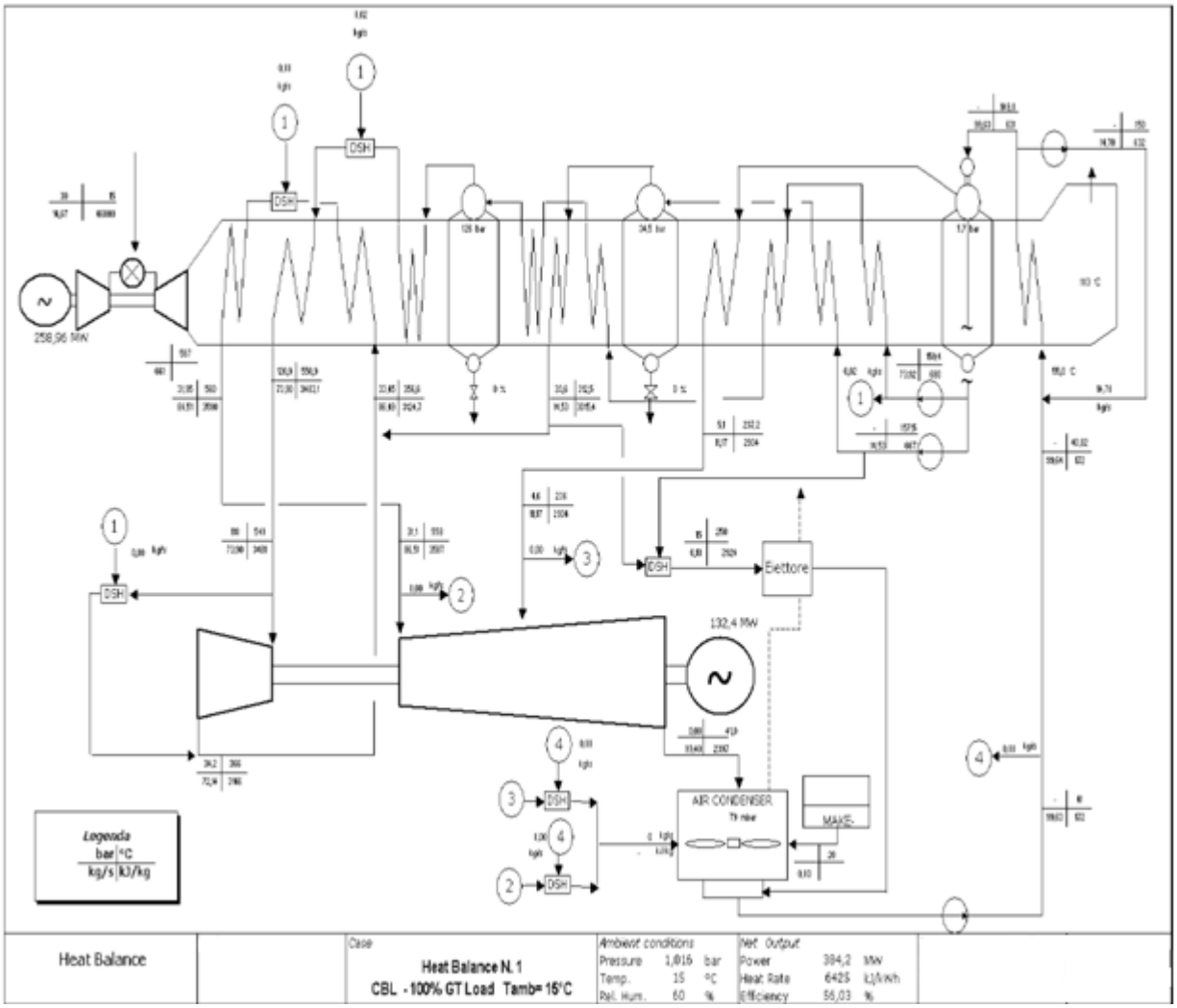

Figura 3.1 Schema dell'impianto a ciclo combinato

Comprende i seguenti elementi:

- scambiatori di calore (con vapore e con acqua liquida)
- evaporatori (per alta-media pressione e per bassa pressione)
- collettore
- sistema pompe-valvole
- turbina

Come già detto, i modelli dei singoli impianti di generazione devono essere abbastanza semplici e tarabili agevolmente a partire da registrazioni delle varie variabili nel normale funzionamento di impianto o nel corso di prove ad hoc richieste dal gestore della rete.

Nel seguito verranno descritti i modelli fisici di ciascun componente.

3.1 Scambiatore di calore di caldaia gas combustione-vapore d'acqua

3.1.1 Preliminari modellistici

Questo modello riguarda scambiatori di calore in caldaia attraversati da vapore surriscaldato (lato acqua) quali surriscaldatori o risurriscaldatori.
Lo scambiatore è costituito da n_t tubazioni (serpentini) attraversate da vapore; nel modello si considera un'unica tubazione equivalente, attraversata dall'intera portata di vapore.

Osserviamo che la massa e l'energia lato gas di combustione e lato vapore sono piccole rispetto all'energia accumulata nelle pareti metalliche delle tubazioni. Ciò significa che la dinamica associata alle variazioni di massa e di energia nel volume dei gas di combustione e nel volume di vapore influenzano il comportamento dinamico del sistema solo con scale temporali ridotte (da frazioni di secondo a secondi). Il modello sviluppato, quindi, trascura sicuramente gli accumuli di massa e di energia nei fluidi: le portate dei gas di combustione si considerano uniformi e l'equazione di conservazione dell'energia viene scritta nell'ipotesi di "quasi stazionarietà", sia lato vapore sia lato fumi.

La dinamica della pressione è sostanzialmente determinata dal modello dell'evaporatore (descritto in seguito), ma in essa si tiene anche conto dell'accumulo di massa in tutti gli scambiatori, tra evaporatore e turbina.

Un altro punto considerato nello sviluppo del modello è la dipendenza del flusso di calore dai gas di combustione al vapore dalla differenza di temperatura tra gas di combustione e vapore secondo una "resistenza termica globale", costituita dalla serie di 3 resistenze:

- resistenza tra gas di combustione e parete metallica, k_g (conduttanza γ_g)
- resistenza della parete metallica, data da $\frac{\lambda_m}{S_m}$
  dove
  - λ_m conducibilità del metallo
  - S_m spessore della tubazione

  Questa resistenza è sicuramente trascurabile

- resistenza tra parete metallica e vapore k_v (conduttanza γ_v)

k_v è molto più alta di k_g, quindi spesso si sviluppano modelli che trascurano la resistenza del flusso di calore tra parete metallica e vapore, assumendo la temperatura del vapore coincidente con la temperatura del metallo. Tali modelli possono essere molto accurati e sono utilizzati suddividendo lo scambiatore in un numero adeguato di celle (volumi di controllo) anche per il progetto del controllo delle temperature vapore dell'impianto [1].

Il modello proposto, invece, considera un unico volume di controllo, che si ritiene adeguato a descrivere la dinamica dei transitori di temperatura nelle scale temporali che interessano.

Gli scambiatori di calore del generatore di vapore a recupero si possono sostanzialmente considerare in controcorrente, anche se la disposizione geometrica è quella di *cross flow.*

L'andamento delle temperature dei gas di combustione e del vapore lungo la coordinata spaziale della tubazione è del tipo mostrato nella figura 3.2.

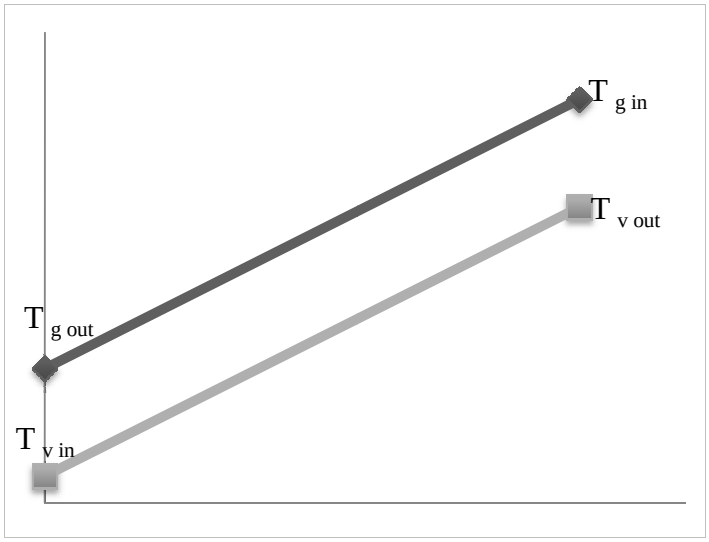

Figura 3.2 Andamento delle temperature di gas ($T_{g\ out}$ all'uscita e $T_{g\ in}$) e di vapore (T_{vin} all'ingresso e $T_{v\ out}$ all'uscita) lungo la tubazione

Si può ritenere quindi che il flusso di calore tra gas di combustione e vapore sia ben approssimato dalla seguente relazione:

$$\dot{Q} = k_g S(T_{g\ in} - T_{v\ out})$$

dove

- S superficie di scambio
- $T_{g\ in}$ temperatura gas di combustione in ingresso
- $T_{v\ out}$ temperatura vapore in uscita
- k_g coefficiente di scambio lato fumi

Ovviamente si assume la temperatura del vapore in uscita coincidente con la temperatura media del metallo della tubazione, come già detto.

Per quanto riguarda l'equazione di conservazione della quantità di moto, si eliminano i termini inerziali e quelli gravitazionali e si descrive la caduta di pressione in funzione della portata, con le classiche relazioni di moto turbolento di un fluido in un condotto. È stato aggiunto un termine di caduta di pressione proporzionale alla portata, significativo e utile per ragioni numeriche quando la portata è molto piccola.

Si ha così un modello di scambiatore con un'equazione differenziale e tre equazioni algebriche.

3.1.2 Equazioni del modello

Equazione differenziale

$$\tau_m \dot{T}_v = w_g(h_{g\ in} - h_{g\ out}) - MAX(0, w_v)(h_{v\ out} - h_{v\ in})$$

dove

- T_v temperatura vapore in uscita
- w_g portata gas di combustione
- $h_{g\ in}$ entalpia gas di combustione in ingresso
- $h_{g\ out}$ entalpia gas di combustione in uscita
- w_v portata vapore
- $h_{v\ out}$ entalpia vapore in uscita
- $h_{v\ in}$ entalpia vapore in ingresso
- τ_m costante di tempo del metallo, il valore di questo parametro è calcolato tramite la procedura di identificazione dinamica, descritta in seguito

Nell'equazione si usa la funzione *MAX* che restituisce il valore maggiore tra i due parametri, 0 e w_v.
Nel modello non è ammessa inversione di portata, quindi se la portata è positiva, la funzione *MAX* restituisce tale valore, se la portata è negativa, restituisce 0, annullando così il suo contributo.

Equazioni algebriche

$$h_{v\ out} - h_{vap} = 0$$
$$(p_{v\ in} - p_{v\ out}) - kw_v = 0 \qquad (3.1)$$
$$\sigma_f(T_{g\ in} - T_v) - w_g(h_{g\ in} - h_{g\ out}) = 0 \qquad (3.2)$$

dove

- h_{vap} entalpia calcolata tramite le tavole del vapore, a partire dal valore della pressione del vapore in uscita $p_{v\ out}$ e dal valore della temperatura del vapore in uscita T_v.
- $p_{v\ in}$ pressione del vapore in ingresso
- $p_{v\ out}$ pressione del vapore in uscita
- k coefficiente calcolato in fase di inizializzazione
- σ_f coefficiente di scambio lato fumi, calcolato nella fase di inizializzazione
- $T_{g\ in}$ temperatura fumi in ingresso

3.2 Scambiatore di calore di caldaia gas combustione-acqua liquida

3.2.1 Preliminari modellistici

Questo modello riguarda scambiatori di calore in caldaia attraversati da acqua liquida (economizzatori).
Mentre nel modello degli scambiatori di vapore è possibile trascurare l'accumulo di energia nel vapore, nel modello dell'economizzatore è aggiunto all'accumulo di energia nel metallo, l'accumulo di energia nell'acqua liquida.

Restano valide le altre ipotesi del modello dello scambiatore di calore, in particolare si assume ancora che la temperatura di liquido in uscita coincida con la temperatura media del metallo quindi si ha un coefficiente di scambio tra metallo e acqua liquida molto elevato.

3.2.2 Equazioni del modello

Il modello di economizzatore realizzato prevede una sola equazione differenziale e due equazioni algebriche.

Equazione differenziale

$$\tau_m \dot{T}_l = w_g(h_{g\,in} - h_{g\,out}) - MAX(0, w_l)(h_{l\,out} - h_{l\,in})$$

dove

- T_l temperatura liquido in uscita
- w_g portata gas di combustione
- $h_{g\,in}$ entalpia gas di combustione in ingresso
- $h_{g\,out}$ entalpia gas di combustione in uscita
- w_l portata acqua sottoraffreddata
- $h_{l\,out}$ entalpia liquido in uscita
- $h_{l\,in}$ entalpia liquido in ingresso
- τ_m costante di tempo del metallo, il valore di questo parametro è calcolato tramite la procedura di identificazione dinamica, descritta nel capitolo successivo

Nell'equazione si usa la funzione *MAX* che restituisce il valore maggiore tra i due parametri, 0 e w_l.
Nel modello non è ammessa inversione di portata, quindi se la portata è positiva, la funzione *MAX* restituisce tale valore, se la portata è negativa, restituisce 0, annullando così il suo contributo.

Equazioni algebriche

$$h_{l\,out} - h_{vap} = 0$$
$$\sigma_f(T_{g\,in} - T_l) - w_g(h_{g\,in} - h_{g\,out}) = 0$$

dove

- h_{vap} entalpia calcolata tramite le tavole del vapore, a partire dal valore della pressione del liquido in uscita $p_{l\,out}$ e dal valore della temperatura del liquido in uscita T_l.
- $T_{g\,in}$temperatura gas di combustione in ingresso

3.3 Collettore

3.3.1 Preliminari ed equazioni del modello

In un collettore il volume è molto piccolo rispetto al volume di fluido degli scambiatori, quindi l'accumulo di energia e di massa è trascurabile.

Si è deciso di realizzare un modello di collettore a tre terminali, ciascuno con portata, pressione e entalpia; in particolare si assume che i terminali *0* e *1* siano degli ingressi, mentre il terminale *2* sia un'uscita per il modello.

Si scrivono le equazioni di conservazione della massa e dell'energia nell'ipotesi di *quasi stazionarietà*.

Per quanto detto sopra, le variabili con pedice *in*, considerate positive in ingresso, sono precedute dal segno +, mentre quelle con pedice *out*, considerate positive in uscita, sono precedute dal segno -

$$w_{in\,0} + w_{in\,1} - w_{out\,2} = 0$$
$$w_{in\,0}\,h_{in\,0} + w_{in\,1}\,h_{in\,1} - w_{out\,2}\,h_{out\,2} = 0$$

con

- $w_{in\,0}$ portata vapore ingresso terminale 0
- $w_{in\,1}$ portata vapore ingresso terminale 1
- $w_{out\,2}$ portata vapore uscita terminale 2
- $h_{in\,0}$ entalpia vapore ingresso terminale 0
- $h_{in\,1}$ entalpia vapore ingresso terminale 1
- $h_{out\,2}$ entalpia vapore uscita terminale 2

L'elemento ha un'unica pressione, comune ai tre terminali, fissata dagli elementi a cui è collegato, mentre variano la portata e l'entalpia di ogni terminale.

3.4 Evaporatore

3.4.1 Preliminari modellistici

In un evaporatore la portata d'acqua proveniente dall'economizzatore, ricevendo calore dal condotto contenente il gas di combustione, evapora; si ha quindi una produzione di vapore (portata Ψ) , somma di due termini:

- il primo termine è relativo all'energia proveniente dai gas di combustione

- il secondo termine è relativo all'energia accumulata nella massa d'acqua e nella massa di vapore in condizioni di saturazione e all'energia accumulata nel metallo della zona adiacente (che è sostanzialmente alla temperatura di saturazione). Questo termine è puramente transitorio ed è proporzionale alla derivata $\dot{p}$ della pressione (con il segno negativo)

$$\Psi = \frac{\sigma_g \Omega_g L_t \left(T_{gi} - T_{sat}(p)\right) - \sigma_g \Omega_g L_w \left(T_{gi} - \frac{T_d + T_{sat}(p)}{2}\right)}{H_{vs}(p) - H_{ls}(p)} - a\,\dot{p}$$

con

$$a = \left(\frac{M_{ls}}{H_{vs}(p) - H_{ls}(p)} Ts \frac{dSls}{dP} + \frac{M_{vs}}{H_{vs}(p) - H_{ls}(p)} T_s \frac{dS_{vs}}{dP} + \frac{C_m A_m \rho_m L_{ev}}{H_{vs}(p) - H_{ls}(p)} \frac{dT_s}{dP}\right)$$

Per la determinazione di questa formula si rimanda a [1]

Nell'evaporatore si ha circolazione naturale: l'acqua senza contenuto di vapore e quindi con densità alta scende dai *downcomers* (portata w_d) e risale nei *riser*, ove si ha la produzione di vapore e quindi una densità media molto inferiore a quella dell'acqua dei *downcomers*; questa differenza di densità è la *driving force* della circolazione naturale.

Notiamo che in ingresso ai *risers* l'acqua può essere in condizione di sottoraffreddamento e quindi nel primo tratto dei *risers* di lunghezza L_w si ha il riscaldamento dell'acqua fino a portarla in condizioni di acqua satura.

Negli evaporatori dei cicli combinati spesso l'acqua proveniente dall'economizzatore ha una entalpia h_{eco} maggiore dell'entalpia del liquido saturo H_{ls} dell'evaporatore. In questo caso l'acqua proveniente dall'economizzatore evapora rapidamente (*flash*) producendo vapore che entra direttamente nella fase gassosa dell'evaporatore.

In conclusione si hanno due casi:

i. $H_{eco} < H_{ls}$: non si ha *flash* dell'acqua proveniente dall'economizzatore; nei *risers* parte dell'energia proveniente dai gas di combustione viene spesa per portare l'acqua in condizione di saturazione. Ovviamente in questo caso la lunghezza di acqua sotto raffreddata nei *risers,* L_w, è maggiore di zero anche se in genere è molto piccola rispetto alla lunghezza dei *risers* ove avviene l'evaporazione, L_{ev}.
ii. $H_{eco} > H_{ls}$ si ha una produzione di vapore pari a: $w_{ali}\ \frac{(h_{eco}-H_{ls})}{H_{vs}-H_{ls}}$. In questo caso la lunghezza di acqua sottoraffreddata è nulla; i *risers* sono completamente riempiti di miscela bifase (liquido saturo di massa M_{ls} e volume V_{ls} e vapore saturo di massa M_{vs} e volume V_{vs}).

Da notare che anche nel caso i) si ha una situazione analoga al caso ii) ma solo nel tratto di *risers* di lunghezza L_{ev} ove avviene l'evaporazione.

Le equazioni di conservazione della massa e dell'energia nella zona contenente acqua sottoraffreddata e dell'energia nella massa metallica corrispondente forniscono l'equazione dinamica della lunghezza L_w in cui si ha acqua sottoraffreddata:

$$\frac{(\rho_d + \rho_{ls})}{2}(H_{ls} - h_d)\dot{L_w} - A_r L_w T_s \frac{dS_{ls}}{dp}\dot{p} = w_{ali}(H_{ls} - h_d) - \sigma_g \Omega_g L_w \left(T_{gi} - \frac{T_d + T_{sat}(p)}{2}\right)$$

dove si è considerata la relazione:

$$h_d w_d = w_{ali} h_{ali} + (w_d - w_{ali}) H_{ls}$$

Poiché nei generatori di vapore dei cicli combinati si è spesso nella situazione ii) in cui $L_w = 0$ l'equazione corrispondente non compare. Anche nel caso i) si elimina la dinamica di L_w e nell'equazione relativa al calcolo di Ψ si sostituisce il termine di regime $w_{ali}(H_{ls} - h_d)$ al termine $\sigma_g \Omega_g L_w \left(T_{gi} - \frac{T_{al}+T_{sat}(p)}{2}\right)$.
Come si è detto questa dinamica pur avendo una scala temporale non piccola viene trascurata, dato che il termine energetico ad esso associato è piccolo o addirittura assente come nel caso ii).

Consideriamo ora l'equazione di conservazione della massa di liquido saturo e vapore saturo:

$$\dot{M_{ls}} = w_d - w_{lr} - \Psi$$
$$\dot{M_{vs}} = \Psi - w_{vr}$$

Ove w_{lr} e w_{vr} sono rispettivamente la portata di liquido saturo e vapore saturo uscenti dai *riser*.

Tenendo conto che $M_{ls} = V_{ls} \cdot \rho_{ls}$ e $M_{vs} = V_{vs} \cdot \rho_{vs}$ (essendo ρ_{ls} e ρ_{vs} rispettivamente le densità del liquido saturo e del vapore saturo), con opportune manipolazioni algebriche si ottengono le due equazioni:

$$\left(V_{vs}\frac{d\rho_{vs}}{dp} + \mu V_{ls}\frac{d\rho_{ls}}{dp}\right)\dot{p} = \mu(w_d - w_{lr}) + (1-\mu)\Psi - w_{vr}$$

con

$$\mu = \frac{\rho_{vs}}{\rho_{ls}}$$

$$\frac{dM_{ls}}{dt} + \frac{dM_{vs}}{dt} = w_d - w_{lr} - w_{vr} - \rho_{ls} A_r \frac{dL_w}{dt} \qquad (3.3)$$

Esamineremo in seguito le conseguenze di quest'ultima equazione; per quanto riguarda l'equazione precedente, invece, sostituendo in essa l'espressione della produzione di vapore Ψ si ottiene:

$$\left(V_{vs}\frac{d\rho_{vs}}{dp} + \mu V_{ls}\frac{d\rho_{ls}}{dp} + (1-\mu)a\right)\dot{p} = \mu(w_d - w_{lr}) - w_{vr} + \frac{\sigma_g \Omega_g L_t\left(T_{gi} - T_{sat}(p)\right) - w_{ali}(H_{ls} - h_d)}{H_{vs}(p) - H_{ls}(p)}$$

Poiché μ è molto minore di uno si può porre nell'equazione precedente $\mu = 0$. Inoltre tenendo conto dell'equazione di conservazione della massa M_{vD} di vapore saturo al di sopra del livello (avente volume V_{vD}) si ottiene l'equazione finale della pressione dell'evaporatore:

$$\left(V_{vs}\frac{d\rho_{vs}}{dp} + a + V_v D\frac{d\rho_{vs}}{dp}\right)\dot{p} = -w_v + \frac{\sigma_g \Omega_g L_t\left(T_{gi} - T_{sat}(p)\right) - w_{ali}(H_{ls} - h_d)}{H_{vs}(p) - H_{ls}(p)} \qquad (3.4)$$

essendo w_v la portata di vapore saturo in uscita all'evaporatore.

Il comportamento dinamico del livello si può ottenere considerando l'equazione di conservazione della massa nell'anello di circolazione e nella fase liquida del corpo cilindrico:

$$\rho_{ls} A \dot{y} = w_{ali} - w_{vr} + \frac{dM_{ls}}{dt} + \frac{dM_{vs}}{dt}$$

dove

- *A* area del pelo libero del livello *y*

Quindi le variazioni di livello sono dovute al termine $w_{ali} - w_{vr}$ che ha un significato ovvio: il livello varia a causa di una differenza tra la portata acqua alimento entrante e la portata vapore uscente e a causa di un termine puramente transitorio che rappresenta la derivata di massa totale di fluido nella zona evaporante.

Ebbene indicando questa massa con la variabile *z*, l'equazione 3.3 si può scrivere:

$$\dot{z} = w_d - w_{lr} - w_{vr} - \rho_{ls} A_r \frac{dL_w}{dt}$$

Esaminiamo ora i vari termini del secondo membro di questa equazione.

L'equazione di conservazione della quantità di moto per l'anello di circolazione fornisce

$$K_d w_d^2 = \rho_l g L_{ev} - \left(\frac{z}{A_r L_{ev}}\right) L_{ev} g$$

con

- $L_{ev} = L_t - L_w$
- g: accelerazione di gravità
- $\left(\frac{z}{A_r L_{ev}}\right)$: densità media della miscela

Da questa equazione si ha la valutazione della portata w_d dell'anello di circolazione

$$w_d = \sqrt{\frac{\rho_l g L_{ev} - \frac{zg}{A_r}}{K_d}}$$

Indicando con S il rapporto tra la velocità del liquido saturo e la velocità del vapore saturo in uscita ai *risers* si ha:

$$\frac{W_{vr}}{W_{lr}} = \frac{\alpha_u \cdot \mu}{(1-\alpha_u)s}$$

essendo α_u il grado di vuoto in uscita dai *risers*, ossia il rapporto tra la sezione dei *risers* in uscita occupata dal vapore rispetto alla sezione totale.
Il grado di vuoto medio nella zona evaporante si ritiene legato al grado di vuoto in uscita dalla relazione $\alpha_m = \xi \cdot \alpha_u$ con ξ valutabile dal regime permanente (per esempio ξ=2 se l'andamento del grado do vuoto è lineare tra 0 ed α_u).

$$\alpha_m = \frac{V_{vs}}{A_r L_{ev}}$$

Tutte queste relazioni portano all'equazione:

$$\dot{z} = \sqrt{\frac{\rho_l g L_{ev} - \frac{zg}{A_r}}{K_d}} - w_{vr}\left(1 - \frac{S}{\mu} + \frac{(1-\mu)S}{\mu}\frac{1}{\xi}\frac{1}{1-\frac{z}{\rho_{ls} A_r L_{ev}}}\right)$$

Osserviamo che

$$z = \rho s_{ls} V_{ls} + \rho_{vs} V_{vs}$$

$$A_r L_{ev} = V_{ls} + V_{vs}$$

Queste sono due equazioni lineari nelle incognite V_{ls} *e* V_{vs} che risolte danno V_{ls} *e* V_{vs} in funzione di *z, p* e $A_r L_{ev}$.

In particolare se si sostituisce la relazione che fornisce l'incognita V_{vs} si ha:

$$\dot{z} = f(p, z, A_r L_{ev}, w_{vr})$$

Linearizzando quest'ultima equazione si ha:

$$\tau_l \dot{z} = -z + K_l w_{vr}$$

dove in generale τ_l e K_l dipendono da p e da $A_r L_{ev}$.

Quest'ultima equazione è conveniente in un modello semplificato e fornisce z e quindi $\dot{z}$ da inserire nell'equazione del livello precedentemente scritta.

$$\rho_{ls} A \dot{y} = w_{ali} - w_{vr} + \frac{-z + K_l w_{vr}}{\tau_l}$$

La dinamica di z è importante per rappresentare lo *shink and swell* del livello.
Quindi si ritiene utile identificare i parametri τ_l e K_l da prove sperimentali che mettono in evidenza il fenomeno dello *shink and swell* del livello, quali ad esempio un transitorio di variazione di posizione della valvola di turbina e quindi una variazione rapida di V_{vr}.

3.4.2 Equazioni del modello

Equazioni differenziali

$$\tau_p \dot{P}_c = -w_{vr} + x_v w_{ali} + \frac{\sigma_f (T_{g\,in} - T_v) - w_{v\,in}\, MAX(0, H_{ls} - h_{v\,in})}{H_{sat} - H_{ls}}$$

$$\tau_y \dot{y} = w_{ali} - w_{vr}$$

dove

- P_c pressione corpo cilindrico
- y livello
- w_{vr} portata vapore in uscita
- w_{ali} portata liquido in ingresso
- $x_v = \frac{MAX(0, h_{v\,in} - H_{ls})}{H_{sat} - H_{ls}}$ titolo del vapore. Si usa la funzione *MAX*, perché il vapore è presente solo se $h_{v\,in}$ è maggiore di H_{ls}, altrimenti si ha liquido sottoraffreddato.
- $T_{g\,in}$ temperatura gas combustione in ingresso
- T_v temperatura di saturazione alla pressione P_c
- H_{ls} entalpia del liquido saturo alla pressione P_c
- H_{sat} entalpia del vapore saturo alla pressione P_c
- $h_{v\,in}$ entalpia liquido in ingresso
- σ_f coefficiente di scambio lato fumi
- τ_p costante di tempo della pressione
- τ_y costante di tempo del livello

Equazioni algebriche

$$h_{v\,out} - H_{sat} = 0$$
$$p_{v\,in} - P_c = 0$$

$$w_g\left(h_{g\,in} - h_{g\,out}\right) - \sigma_f\left(T_{g\,in} - T_v\right) = 0$$

dove

- $h_{v\,out}$ entalpia vapore in uscita
- $p_{v\,in}$ pressione vapore in ingresso
- w_g portata gas di combustione
- $h_{g\,in}$ entalpia gas di combustione in ingresso
- $h_{g\,out}$ entalpia gas di combustione in uscita

3.5 Evaporatore di bassa pressione (degasatore)

3.5.1 Preliminari modellistici

Il modello dell'evaporatore a bassa pressione è stato realizzato con gli stessi criteri del modello evaporatore descritto in precedenza, ha quindi un'equazione dinamica per la pressione P_c e una per il livello y.
Rispetto al modello precedente, ha quattro terminali di tipo vapore, quindi quattro portate.
La portata w_0 è un ingresso, la portata w_1 è fornita dal modello turbina, mentre le portate w_4 e w_5 sono fornite dai modelli delle pompe alta e bassa pressione.

3.5.2 Equazioni del modello

Equazione differenziale

$$\tau_p \dot{P_c} = \sigma_f\left(T_{g\,in} - T_v\right) - w_1\left(h_1 - h_0\right) - \left(w_4 + w_5\right)\left(h_4 - h_0\right)$$
$$\tau_y \dot{y} = w_0 - w_1 - w_4 - w_5$$

dove

- P_c pressione corpo cilindrico
- y livello
- $T_{g\,in}$ temperatura gas di combustione in ingresso
- T_v temperatura di saturazione alla pressione P_c
- w_0 portata in uscita terminale 0
- w_1 portata in uscita terminale 1
- w_4 portata in uscita terminale 4
- w_5 portata in uscita terminale 5
- h_1 entalpia in uscita terminale 1
- h_0 entalpia in ingresso terminale 0
- h_4 entalpia in uscita terminale 4
- σ_f coefficiente di scambio lato fumi
- τ_p costante di tempo relativa alla pressione
- τ_y costante di tempo relativa al livello

Equazioni algebriche

$$h_1 - H_{sat} = 0$$
$$p_0 - P_c = 0$$
$$w_g\left(h_{g\,in} - h_{g\,out}\right) - \sigma_f\left(T_{g\,in} - T_v\right) = 0$$
$$h_4 - H_{ls} = 0$$

dove

- H_{sat} entalpia del vapore saturo alla pressione P_c
- p_0 pressione ingresso terminale 0
- w_g portata gas di combustione
- $h_{g\,in}$ entalpia gas di combustione in ingresso
- $h_{g\,out}$ entalpia gas di combustione in uscita
- h_4 entalpia uscita terminale 4
- H_{ls} entalpia del liquido saturo alla pressione P_c

3.6 Pompa centrifuga e valvola

3.6.1 Preliminari ed equazione del modello

Il modello prevede un accoppiamento tra due elementi, la pompa centrifuga e la valvola. Sono quindi presenti due equazioni, la prima riferita alla pompa, la seconda alla valvola.

I numeri adimensionali relativi al funzionamento delle pompe centrifughe sono:

- $flow\ number\ \frac{w}{\rho\omega A_p D_p}$

  dove
  - A_p e D_p parametri costitutivi della pompa

- $work\ number\ \frac{p_1 - p_0}{\rho\omega^2 D_p^2}$

Assumendo che il $flow\ number$ dipenda dal $work\ number$ secondo una parabola e conglobando in soli tre parametri le variabili ρ, A_p e D_p si ottiene l'equazione:

$$p_2 - p_0 - c\omega^2 + aw_0^2 + bw_0 = 0 \qquad (3.5)$$

Il modello ha anche un'altra equazione algebrica, relativa alla valvola

$$\theta^2(p_1 - p_0) - \theta^2 c\omega^2 + \left(a\theta^2 + \sigma_{ft}\right)w_0^2 + \left(b\theta^2 + \sigma_{fl}\right)w_0 = 0$$

dove

- p_1 pressione uscita terminale 1
- p_2 pressione terminale 2
- p_0 pressione ingresso terminale 0
- w_0 portata ingresso terminale 0
- ω velocità angolare della pompa
- c, a, b coefficienti calcolati in fase di inizializzazione
- θ apertura della valvola
- σ_{ft} e σ_{fl} coefficienti di attrito turbolento e laminare

3.7 Turbina

3.7.1 Equazioni del modello

$$w_0 - k_v\theta\sqrt{\frac{\rho k_t^2(p_0^2 - p_1^2)}{(k_t^2 + k_v^2\theta^2)p_0}} = 0$$

$$h_1 - h_0 + \eta(h_0 - h_{iso}) = 0 \qquad (3.6)$$

dove

- w_0 portata in ingresso
- p_0 pressione in ingresso
- $p1$ pressione uscita
- h_0 entalpia in ingresso
- h_1 entalpia in uscita
- ρ densità vapore
- h_{iso} entalpia isoentalpica, calcolata a partire dal valore della pressione p_0 e della temperatura del vapore T_v
- θ apertura della valvola di turbina
- k_v coefficiente calcolato in fase di inizializzazione
- k_t coefficiente calcolato in fase di inizializzazione
- η rendimento della turbina, calcolato in fase di inizializzazione

Capitolo 4. Realizzazione dell'impianto a ciclo combinato

Nel capitolo precedente sono stati descritti i modelli fisici dei vari componenti dell'impianto. Per ogni componente è stato realizzato un modello in codice C, che prevede una parte di evoluzione, con le equazioni descritte nel capitolo 3 e una parte di inizializzazione, necessaria per assegnare alle variabili i valori all'istante iniziale della simulazione.
Nel seguito verrà presentata la fase di inizializzazione di ogni componente, quindi si passerà a descrivere la procedura di assiemamento, per ottenere il modello del sistema globale.

4.1 Fase di inizializzazione

Per prima cosa si inizializzano le strutture dello jacobiano e delle costanti di tempo e i vettori delle variabili algebriche, degli stati e delle equazioni, relativi al modello globale.

Si passa quindi all'inizializzazione dei singoli componenti dell'impianto, chiamando le funzioni C, relative a questa fase, di ciascun elemento.

Tali funzioni assegnano il valore iniziale agli stati dell'elemento, se presenti, e alle variabili terminali.

Queste variabili fanno riferimento a particolari strutture, dette interfacce, e permettono i collegamenti tra i diversi elementi che compongono l'impianto: quando due elementi sono collegati, le variabili dell'interfaccia di uscita del primo coincideranno con le variabili dell'interfaccia di ingresso del secondo.

Sono state definite due diverse interfacce, una per il vapore e una per i fumi.

L'interfaccia vapore ha le tre seguenti variabili terminali

- portata w_v
- pressione p_v
- entalpia h_v

mentre l'interfaccia fumi è definita da

- portata w_g
- pressione p_g
- temperatura T_g

Un modello può definire uno o più cortocircuiti o passanti sulle grandezze dei terminali che hanno la stessa interfaccia. Ad esempio se per un elemento la portata del terminale 0 coincide con la portata del terminale 1, nella funzione di inizializzazione si dovrà definire il cortocircuito

$$t_0.w = t_1.w$$

dove

- t_0 e t_1 indicano i due terminali
- w indica la variabile sulla quale è definito il cortocircuito, in questo caso la portata

Di seguito è presentata l'inizializzazione dei componenti.

4.1.1 Scambiatore

Uno scambiatore ha quattro terminali, come mostrato in figura 4.1, due terminali (2 e 3 grigi) relativi al lato fumi (ingresso e uscita) e due (0 e 1 blu) relativi al lato vapore (ingresso e uscita)

Figura 4.1 Rappresentazione (1:1) di uno scambiatore e dei suoi terminali

Per le ipotesi usate nella realizzazione del modello, nei terminali lato fumi si ha un'uguaglianza tra portata di ingresso e di uscita e tra pressione di ingresso e di uscita.
Per le temperature, invece, si hanno valori diversi, quindi si considera una $T_{g\ in}$, temperatura gas di combustione in ingresso, e una $T_{g\ out}$, temperatura gas di combustione in uscita.

Anche per terminali lato vapore si ha uguaglianza tra portata in ingresso e portata in uscita, non essendoci accumulo di massa, mentre per le pressioni e per le entalpie si distinguono due valori, $p_{v\ in}$ e $p_{v\ out}$, $h_{v\ in}$ e $h_{v\ out}$.
Noti i valori di queste variabili, è possibile calcolare il valore dei parametri, cioè del coefficiente di scambio σ e del coefficiente k.
Il coefficiente k è determinato a partire dall'equazione di conservazione della quantità di moto descritta nel capitolo 3

$$k = \frac{(p_{v\ in} - p_{v\ out})}{w_v}$$

Il coefficiente di scambio, σ, dipende dalla portata di gas di combustione w_g secondo la seguente relazione

$$\sigma_f = \sigma_{f0}\left(0.95\left(\sqrt{\frac{w_g}{w_{g\ n}}}\right) + 0.05\right) \qquad (4.1)$$

dove

- $w_{g\ n}$ portata nominale di gas di combustione
- σ_{f0} coefficiente di scambio iniziale, ricavato dall'equazione

$$\sigma_{f0} = \frac{w_g(h_{g\ in} - h_{g\ out})}{(T_{g\ in} - T_v)} \qquad (4.2)$$

Dalla relazione (4.1), si nota che il coefficiente di scambio diventa maggiore se si aumenta la portata di gas di combustione in ingresso.

4.1.2 Economizzatore

L'economizzatore ha quattro terminali, due terminali relativi al lato fumi (ingresso e uscita) e due relativi al lato acqua liquida (ingresso e uscita), come lo scambiatore.

Per le ipotesi usate nella realizzazione del modello, nei terminali lato fumi si ha un'uguaglianza tra portata di ingresso e di uscita e tra pressione di ingresso e di uscita.

Per le temperature, invece, si hanno valori diversi, quindi si considera una $T_{g\ in}$ temperatura gas di combustione in ingresso e una $T_{g\ out}$, temperatura gas di combustione in uscita.
Anche per i terminali alto vapore si ha uguaglianza tra portata in ingresso e portata in uscita, e tra pressione in ingresso e pressione in uscita, non essendoci accumulo di massa.
Per le entalpie, invece, si distinguono due valori, $h_{l\ in}$ e $h_{l\ out}$.
Noti i valori di queste variabili, è possibile calcolare il valore del coefficiente di scambio σ_f.
Tale parametro dipende dalla portata di gas di combustibile w_g secondo le relazioni (4.1) e (4.2).

4.1.3 Collettore

Il collettore ha tre terminali di tipo vapore, come mostrato in figura 4.2.

Figura 4.2 Rappresentazione (2:1) di un collettore e dei suoi terminali

Si assume che i terminali *0* e *1* siano degli ingressi, mentre il terminale *2* sia un'uscita per il modello.
Ogni terminale ha una propria portata e una propria entalpia, mentre la pressione è comune a tutti e tre.
Poiché nel collettore non sono presenti parametri da calcolare, la sua funzione di inizializzazione assegna solo i valori iniziali a tutte le variabili.

4.1.4 Evaporatore

L'evaporatore ha quattro terminali, due terminali (2 e 3 grigi) relativi al lato fumi (ingresso e uscita) e due (0 e 1 blu) relativi al lato vapore (ingresso e uscita) come mostrato in figura 4.3.

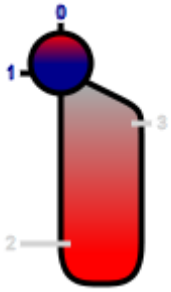

Figura 4.3 Rappresentazione (1:1) di un evaporatore e dei suoi terminali

Per le ipotesi usate nella realizzazione del modello, per i terminali lato fumi, si ha un'uguaglianza tra portata di ingresso e di uscita e tra pressione di ingresso e di uscita.
Per le temperature, invece, si hanno valori diversi, quindi si considera una $T_{g\ in}$ temperatura gas di combustione in ingresso e una $T_{g\ out}$, temperatura gas di combustione in uscita.

In questo modello, a differenza del modello dello scambiatore e dell'economizzatore, l'accumulo di massa non è trascurabile, quindi per i terminali lato vapore oltre ad avere due valori dell'entalpia, $h_{v\ in}$ e $h_{v\ out}$, si avranno due valori anche per le portate, $w_{v\ in}$ e $w_{v\ out}$.
Le pressioni, invece, sono uguali e a regime coincidono con il valore dello stato, P_c.

Anche per l'evaporatore, il coefficiente di scambio, σ_f, dipende dalla portata di gas di combustione w_g secondo le relazioni (4.1) e (4.2).

4.1.5 Evaporatore di bassa pressione (degasatore)

L'evaporatore bassa pressione ha sei terminali, due terminali (2 e 3 grigi) relativi al lato fumi (ingresso e uscita) e quattro (0,1, 4 e 5 blu) relativi al lato vapore, come mostrato in figura 4.4.

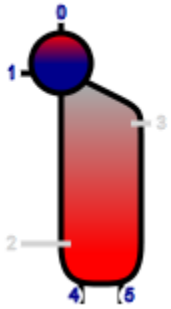

Figura 4.4 Rappresentazione (1:1) di un evaporatore bassa pressione e dei suoi terminali

Per le ipotesi usate nella realizzazione del modello, nei terminali lato fumi si ha un'uguaglianza tra portata di ingresso e di uscita e tra pressione di ingresso e di uscita.
Per le temperature, invece, si hanno valori diversi, quindi si considera una $T_{g\ in}$ temperatura gas di combustione in ingresso e una $T_{g\ out}$, temperatura gas combustibile in uscita.

Nei terminali lato vapore, si considera l'accumulo di massa; si hanno quindi quattro portate diverse: w_0, w_1, w_4, w_5.
Le quattro pressioni sono uguali e a regime coincidono con il valore dello stato, P_c.
Per le entalpie, si ha un'uguaglianza tra i terminali 4 e 5, quindi si avranno tre valori diversi: h_0, h_1 e h_4.

Anche per l'evaporatore bassa pressione, il coefficiente di scambio, σ_f, dipende dalla portata di gas di combustione w_g secondo le relazioni (4.1) e (4.2).

4.1.6 Pompa centrifuga e valvola

L'elemento, rappresentato nella figura 4.5, ha quattro terminali di tipo lato vapore (ingresso e uscita), due riferiti all'elemento pompa (0 e 2) e due all'elemento valvola (1 e 3), che nel modello sono accoppiati.

Figura 4.5 Rappresentazione (1:1) di una pompa alimento e dei suoi terminali

Per le ipotesi usate nella realizzazione del modello, si ha un'uguaglianza tra tutte le portate e tra le entalpie h_0 e h_2 e h_1 e h_3, e tra le pressioni p_2 e p_3, mentre sono definite due pressioni, p_1 e p_0 .

Dopo aver definito i valori di queste variabili è possibile ricavare i valori dei coefficienti *a*, *b* e *c*, partendo dalle equazioni algebriche del modello, descritte nel capitolo 3

$$c = (\frac{p_2 - p_0}{\omega_0^2})\xi$$

$$a = \frac{(p_0 - p_2) + c\omega^2}{w_0^2 \left(1 + \frac{1}{100}\right)}$$

$$b = \frac{a\, w_0}{100}$$

dove

- ω_0 velocità angolare iniziale
- ξ valore assegnato, per default è uguale a 1.3

Si ricavano inoltre i valori dei coefficienti di attrito

$$\sigma_{ft} = \frac{\theta^2(p_2 - p_1)}{w_0^2 \left(1 + \frac{1}{100}\right)}$$

$$\sigma_{fl} = \frac{\sigma_{ft}\, w_0}{100}$$

4.1.7 Turbina

La turbina ha due terminali di tipo vapore (ingresso e uscita)

Figura 4.6 Rappresentazione (1:1) di una turbina e dei suoi terminali

Si ha un'unica portata tra ingresso e uscita, mentre sono definite due pressioni, p_0 e p_1, e due entalpie h_0 e h_1.
Assegnati i valori alle variabili terminali, si ricava il parametro k_v

$$k_v = \frac{w_0}{\theta\sqrt{(p_0^2 - p_{ug}^2)\frac{\rho}{p_0}}}$$

Noto il suo valore è possibile ricavare k_t

$$k_t = k_v\theta\sqrt{\frac{p_0^2 - p_{ug}^2}{p_{ug}^2 - p_1^2}}$$

con p_{ug} valore dato in ingresso.

Dall'equazione algebrica relativa alle entalpia si ricava inoltre il valore di η, rendimento della turbina.

$$\eta = \frac{h_0 - h_1}{h_0 - h_{iso}}$$

dove h_{iso} è l'entalpia isoentropica, calcolata tramite le tavole del vapore.

4.2 Codice C del modello scambiatore di calore di caldaia gas combustione-vapore d'acqua

Per chiarire le idee, viene presentato il codice C del modello dello scambiatore di calore di caldaia gas combustione-vapore d'acqua, chiamato *vghe*.

Per prima cosa si definisce una struttura, *Mvghe*, che contiene tutte le strutture che definiscono il modello

```
typedef struct Mvghe{

Cvghe c;
Pvghe p;
Svghe s;
Vvghe v;
Uvghe u;
Yvghe y;
Tvghe t;
Dvghe d;

}Mvghe;
```

con

```
typedef struct interfacciaVap{

double
       *w,
       *p,
       *h;

}interfacciaVap;
```

InterfacciaVap è la struttura che definisce un'interfaccia di tipo vapore.
Ha tre elementi: w, p e h, che si riferiscono rispettivamente alla portata, alla pressione e all'entalpia del terminale.
Le variabili sono dichiarate come puntatori, per permettere il collegamento tra due elementi. Quando il terminale di un elemento è collegato a quello di un altro, le variabili dei due terminali coincidono. Poiché tali variabili sono dichiarate come puntatori, gli elementi collegati scrivono nello stesso indirizzo di memoria, quindi qualsiasi modifica fatta da un elemento viene vista dall'altro, senza perdita di informazioni.

```
typedef struct interfacciaGas{

double
       *w,
       *p,
       *T;

}interfacciaGas;
```

InterfacciaGas è la struttura che definisce un'interfaccia di tipo fumi.
Ha tre elementi: w, p e T, che si riferiscono rispettivamente alla portata, alla pressione e alla temperatura del terminale.
Le variabili sono dichiarate come puntatori, come detto sopra.

```
typedef struct Tvghe{

interfacciaVap t0, t1;
interfacciaGas t2, t3;

}Tvghe;
```

Tvghe è la struttura che ha per elementi i terminali dello scambiatore.
Come già detto lo scambiatore ha quattro terminali, due t_0 e t_1, lato vapore, *interfacciaVap* e due t_2 e t_3, lato fumi, *interfacciaGas*.

```
typedef struct Pvghe{
     double
            k,
            sigma,
            sigma0;
     double
            gas_compo[NCOMP];
     thermoprop
                 thermo_vap_in,
                 thermo_vap_out;
     gasprop
             gasin,
             gasout;
}Pvghe;
```

Pvghe è la struttura che contiene i parametri del modello.
- *k:* parametro definito dall'equazione della quantità di moto
- *sigma*$_0$ e *sigma*: coefficienti di scambio calcolati rispettivamente in fase di inizializzazione e in fase di evoluzione
- *gas_compo[NCOMP]:* vettore che contiene la composizione dei gas, *NCOMP* corrisponde al numero di componenti dei gas
- *thermo_vap_in* e *thermo_vap_out:* strutture di tipo *thermoprop,* contengono le variabili termodinamiche del vapore, tra le quali pressione, temperatura, entalpia, entropia, densità…
  Ciascuna delle due strutture è riferita a uno dei terminali lato vapore.
- *gasin* e *gasout*: strutture di tipo *gasprop,* contengono le variabili termodinamiche dei gas, tra le quali pressione, temperatura, densità…
  Ciascuna delle due strutture è riferita a uno dei terminali lato fumi.

```
typedef struct Cvghe{
     double massa_m;
}Cvghe;
```

Cvghe è la struttura che contiene le costanti del modello.

- massa_m: massa del metallo

```
typedef struct Svghe{
     double *Tv;
     double *stati[nstati_vghe];
}Svghe;
```

Svghe è la struttura che definisce gli stati del modello.

- **Tv*: puntatore allo stato, temperatura del vapore. Si utilizza un puntatore per eseguire correttamente il passaggio da stato locale dell'elemento a stato del modello globale.
- **stati[nstati_vghe]* : riferimento al vettore che contiene tutti gli stati dell'elemento. La sua dimensione è pari a *nstati_vghe*, numero degli stati dell'elemento

```
typedef struct Vvghe{
     double **Jdz, **Jdx, **Jax, **Jaz,
            *f[nstati_vghe],
            *g[nalg_vghe],
            **tau,
            rho,
            rho_init,
            T_init,
            p_init,
            wn;
}Vvghe;
```

Vvghe è la struttura che contiene le variabili globali dell'elemento, cioè le variabili che devono essere visibili all'etserno del modulo.

- ***Jdz, **Jdx, **Jax, **Jaz :* riferimenti ai blocchi dello jacobiano
- *f[nstati_vghe]* : vettore che contiene le equazioni differenziali
- *g[nalg_vghe]:* vettore che contiene le equazioni algebriche,
- ***tau:* matrice delle costanti di tempo
- *rho* e *rho_init:* densità e densità iniziale
- *T_init:* temperatura iniziale
- *p_init:* pressione iniziale
- *wn:* portata nominale dei gas

```
typedef struct Yvghe{
     double
          Tvapout,
          Qvap;
}Yvghe;
```

Yvghe è la struttura che contiene le uscite del modello.

- *Tvapout* : temperatura del vapore in uscita, coincide con lo stato del modello
- *Qvap* : calore del vapore

```
typedef struct Dvghe{
     int
          nstati,
          nalg,
```

```
        nti,
        nout,
        nrJ,
        ncJ;}Dvghe;
```

Dvghe è la struttura che contiene informazioni sul numero degli stati, delle variabili algebriche, dei terminali, delle uscite e delle dimensioni dello jacobiano.

- *nstati* : numero degli stati
- *nalg* : numero delle equazioni algebriche
- *nti* : numero delle variabili terminali senza cortocircuiti, cioè indipendenti
- *nout* : numero delle uscite
- *nrJ* : numero di righe della matrice dello jacobiano
- *ncJ* : numero di colonne della matrice dello jacobiano

Il modello ha 7 funzioni, la prima per la parte di inizializzazione, le restanti per l'evoluzione.

La funzione ***inizializza_vghe*** riceve come parametri un riferimento al vettore che contiene i valori da assegnare alle variabili dell'elemento, un riferimento alla struttura *Mvghe* descritta sopra, i riferimenti ai vettori che contengono i secondi membri delle equazioni differenziali e algebriche, *s* e *g*, alla matrice delle costanti di tempo, *T*, e ai diversi blocchi in cui è divisa la matrice *J* dello jacobiano.

L'assiematore si occupa di passare ad ogni elemento le variabili corrette per effettuare le chiamate alle funzioni, in base alla posizione dell'elemento nel modello globale.

Si inizializza il numero degli stati dell'elemento, il numero delle equazioni algebriche, dei terminali privi di cortocircuiti, degli ingressi, delle uscite e le dimensioni dello jacobiano.

```
M->d.nstati = nstati_vghe;
M->d.nalg = nalg_vghe;
M->d.nti = nti_vghe;
M->d.nin = nin_vghe;
M->d.nout = nout_vghe;
M->d.nrJ = nstati_vghe + nalg_vghe;
M->d.ncJ = nstati_vghe + nti_vghe;
```

Successivamente si impostano i cortocircuiti e si assegnano i valori delle variabili lette dal file XML, contenuti nel vettore *p*, primo parametro della funzione *inizializza_vghe*.

```
M->t.t0.w = M->t.t1.w;
M->t.t2.w = M->t.t3.w;
M->t.t2.p = M->t.t3.p;

*M->t.t0.w = *p[0];
*M->t.t0.p = *p[1];
*M->t.t0.h = *p[2];
*M->t.t1.p = *p[3];
*M->t.t1.h = *p[4];
*M->t.t2.w = *p[5];
*M->t.t2.p = *p[6];
*M->t.t2.T = *p[7];
*M->t.t3.T = *p[8];
*M->s.stati[0] = *p[9];
M->c.massa_m = *p[10];
```

```
M->s.Tv = M->s.stati[0];

M->v.wn = *p[5];
```

Si effettuano le chiamate alle tavole del vapore, quindi si calcolano i parametri e si inizializza lo jacobiano, i vettori delle equazioni differenziali e algebriche e la matrice delle costanti di tempo.
Quest'ultima inizializzazione è fondamentale, perché consente all'elemento di modificare solo le variabili del modello globale che gli corrispondono, garantendo il corretto aggiornamento di tutte le variabili.

```
calc_vap_pT(*M->t.t1.p, *M->s.Tv, &M->p.thermo_vap_out);
calc_vap_ph(*M->t.t0.p, *M->t.t0.h, &M->p.thermo_vap_in);

                                    M->p.gasin.h =
(1150*Riferimenti.Trif/Riferimenti.Hrif)*(*M->t.t2.T-(298.15/Riferimenti.Trif));

                                   M->p.gasout.h =
(1150*Riferimenti.Trif/Riferimenti.Hrif)*(*M->t.t3.T-(298.15/Riferimenti.Trif));


M->v.p_init = *M->t.t0.p;
M->v.T_init = *M->s.Tv;

M->v.rho_init = M->p.thermo_vap_in.rho;

M->v.rho = (M->v.rho_init*M->v.T_init**M->t.t0.p)/(*M->s.Tv*M->v.p_init);

                                    M->p.sigma0 =
(*M->t.t2.w*(M->p.gasin.h - M->p.gasout.h))/(*M->t.t2.T-*M->s.Tv);

*M->t.t1.h = M->p.thermo_vap_out.h;

M->p.k = (*M->t.t0.p - *M->t.t1.p)/(*M->t.t0.w);

M->v.Jdz = Jdz;
M->v.Jaz = Jaz;
M->v.Jdx = Jdx;
M->v.Jax = Jax;

M->v.tau = T;

M->v.f[0] = s;
M->v.g[0] = g;
M->v.g[1] = g+1;
M->v.g[2] = g+2;
```

La fase di evoluzione comprende le seguenti funzioni:

- *setEquazioni_vghe* : calcola le equazioni algebriche e differenziali, descritte nel capitolo 3. Il vettore *f* contiene l'unica equazione differenziale, il vettore *g* contiene le equazioni algebriche.

```
M->v.rho = (M->v.rho_init*M->v.T_init**M->t.t0.p)/(*M->s.Tv*M->v.p_init);

M->p.sigma = M->p.sigma0*(0.95*sqrt(*M->t.t2.w/M->v.wn)+0.05);

                                    M->p.gasin.h =
(1150*Riferimenti.Trif/Riferimenti.Hrif)*(*M->t.t2.T-(298.15/Riferimenti.Trif));

                                    M->p.gasout.h=
(1150*Riferimenti.Trif/Riferimenti.Hrif)*(*M->t.t3.T-(298.15/Riferimenti.Trif));

                                     *M->v.f[0]=
*M->t.t2.w*(M->p.gasin.h-M->p.gasout.h)-*M->t.t0.w*(*M->t.t1.h-*M->t.t0.h);

*M->v.g[0] = *M->t.t1.h - M->p.thermo_vap_out.h;
*M->v.g[1] = (*M->t.t0.p-*M->t.t1.p)-M->p.k**M->t.t0.w;
```

```
                                        *M->v.g[2] =
M->p.sigma*(*M->t.t2.T-*M->s.Tv)-*M->t.t2.w*(M->p.gasin.h-M->p.gasout.h);
```

- *setT_vghe* : calcola le costanti di tempo
- *setJacobiano_vghe* : calcola i diversi blocchi dello jacobiano
- *setOutput_vghe* : assegna le uscite del modello al vettore *out* delle uscite del sistema globale

```
M->y.Tvapout = *M->s.Tv;
M->y.Qvap = *M->t.t0.w*(*M->t.t1.h - *M->t.t0.h);
out[0] = M->y.Tvapout;
out[1] = M->y.Qvap;
```

Infine è stata definita la funzione *terminate_vghe* che libera la memoria allocata per il modello.

In appendice è riportato il codice C completo di alcuni dei componenti descritti nel capitolo 3.

4.3 Assiemamento

Dopo aver realizzato i modelli degli elementi in codice C, è possibile passare alla fase di collegamento, realizzata con l'assiematore, per ottenere il modello globale di un impianto.
Come prima verifica del corretto funzionamento del software, si considera l'assiemamento del modello evaporatore-valvola, già descritto nel paragrafo 1.6.

4.3.1 Analisi del modello evaporatore-valvola

In figura 4.7 è riportato il disegno SVG del modello.

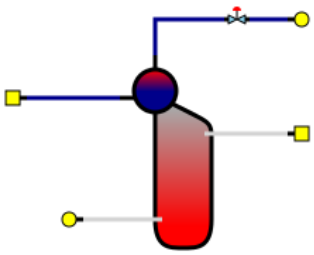

Figura 4.7 Disegno SVG: collegamento tra evaporatore e valvola

Come si può vedere dalla figura 4.7, il modello comprende l'evaporatore, la valvola e quattro elementi terminali, due rami e due pozzi.

I rami e i pozzi sono importanti perché permettono di definire le condizioni al contorno dell'impianto.

Un pozzo lato vapore ha le seguenti equazioni

$$w_0 - u_w = 0$$
$$h_0 - u_h = 0$$

con

- w_0 e h_0 : portata e entalpia del terminale 0 del pozzo
- u_w : ingresso del modello globale
- u_h: ingresso del modello globale

mentre un ramo lato vapore ha un'unica equazione

$$p_0 - u_p = 0$$

con

- p_0: pressione del terminale 0 del ramo
- u_p : ingresso del modello globale

La valvola è collegata all'evaporatore e a un pozzo lato vapore, mentre l'evaporatore è collegato alla valvola e a un ramo lato vapore.

Un pozzo lato fumi ha le seguenti equazioni

$$w_0 - u_w = 0$$
$$T_0 - u_T = 0$$

con

- w_0 e T_0 : portata e temperatura del terminale 0 del pozzo
- u_w : ingresso del modello globale
- u_T: ingresso del modello globale

Un ramo lato fumi ha un'unica equazione

$$p_0 - u_p = 0$$

con

- p_0: pressione del terminale 0 del ramo
- u_p : ingresso del modello globale

I terminali lato fumi dell'evaporatore sono collegati a sinistra con un elemento terminale pozzo, a destra con un elemento terminale ramo.

Considerando i modelli di ciascun elemento si avranno

- 2 equazioni differenziali, poiché solo l'evaporatore ha variabili differenziali
- 10 equazioni algebriche

In totale si hanno quindi 12 equazioni.

Per il numero di variabili, ci sono 2 variabili differenziali.
Inoltre l'evaporatore ha 12 variabili terminali e 3 cortocircuiti, quindi in totale 9 variabili algebriche.
La valvola ha 6 variabili terminali e 2 cortocircuiti, quindi in totale 4 variabili algebriche.
I rami e i pozzi hanno 3 variabili terminali, senza cortocircuiti, quindi in totale 3 variabili algebriche.

Sommando i valori ottenuti si ottiene il numero delle variabili algebriche totali, pari a 9+4+3+3+3+3, 25.
Ogni collegamento tra elementi fa coincidere le variabili dei terminali.
Ogni elemento prima del collegamento ha 3 variabili terminali, quindi in totale in un collegamento tra due elementi si hanno 6 variabili terminali. Il collegamento fa coincidere le variabili terminali corrispondenti, che quindi diventano 3.
Si può concludere che ogni collegamento diminuisce di 3 il numero delle variabili algebriche.
Considerando il modello in figura 4.7, si hanno 5 collegamenti, quindi dal numero totale delle variabili algebriche si dovranno sottrarre 5*3, cioè 15 variabili.

Il numero totale delle variabili algebriche del modello complessivo è quindi

$$25-15 = 10$$

Il numero complessivo delle variabili algebriche e differenziali è

$$10+2 = 12$$

Poiché il numero di equazioni è uguale al numero complessivo di variabili, il sistema DAE che descrive il modello è quadrato e può essere risolto, usando la procedura descritta nel capitolo 1.
Si può quindi concludere che l'assiematore ha collegato in modo corretto i singoli elementi, realizzando un unico modello complessivo che li rappresenta.
In figura 4.8 viene riportata la mappa della matrice dello jacobiano del modello, prodotta dall'assiematore.

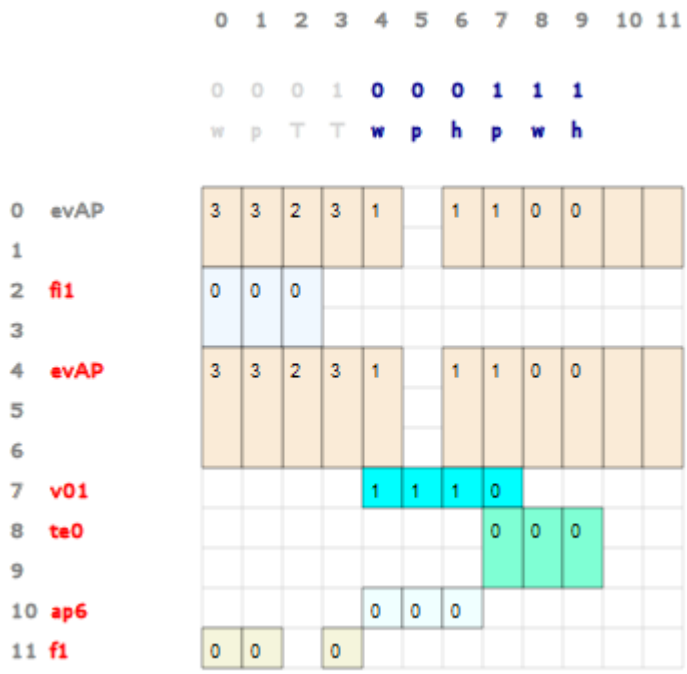

Figura 4.8 Matrice jacobiana del modello assiemato

Come detto, la matrice ha dimensione 12*12.
Le righe corrispondono alle equazioni degli elementi, all'inizio sono posizionate le equazioni differenziali, quindi quelle algebriche.

Ad esempio all'evaporatore, elemento evAP, sono assegnate le uniche righe delle equazioni differenziali, essendo l'unico elemento del modello con dinamica, e 3 righe della parte algebrica, per un totale di 5 righe, pari al numero effettivo delle sue equazioni.
Le colonne corrispondono alle variabili algebriche, da 0 a 9, e alle variabili differenziali, 10 e 11.
Ad ogni colonna è associata una delle variabili, nel seguente modo:

- la lettera specificata sopra l'intestazione della colonna indica la variabile
    a) w: portata
    b) p: pressione
    c) T: temperatura
    d) h: entalpia

- il colore corrisponde invece al tipo di collegamento
    a) blu: vapore
    b) grigio: fumi

- il numero sopra la lettera indica il terminale a cui si riferisce la variabile

- le colonne senza numero e variabile corrispondono agli stati.

I numeri specificati all'interno delle colonne assegnate agli elementi indicano a quale terminale di quell'elemento appartiene la variabile.
Ogni elemento può modificare solo le variabili del modello globale corrispondenti alle colonne che gli vengono assegnate dall'assiematore.
Quando due elementi vengono collegati, l'assiematore assegna ad essi la stessa colonna per ogni variabile terminale del collegamento, effettuando così l'uguaglianza tra le variabili.
Considerando la valvola e l'evaporatore, il terminale 0 della valvola è collegato al terminale 1 dell'evaporatore.
La portata e l'entalpia del terminale 0 della valvola coincidono con la portata e l'entalpia del terminale 1.
La pressione del terminale 1 dell'evaporatore coincide con la pressione del terminale 0.
Definite queste uguaglianze, dalla figura 4.8 si vede che l'evaporatore e la valvola hanno in comune le colonne, 4, 6 e 7, che corrispondono alle variabili del loro collegamento.

4.3.2 Analisi del modello ciclo combinato

Dopo aver verificato la correttezza di un modello semplice, si può realizzare il modello completo dell'impianto che vogliamo studiare.

Lo schema SVG dell'impianto è riportato in figura 4.9

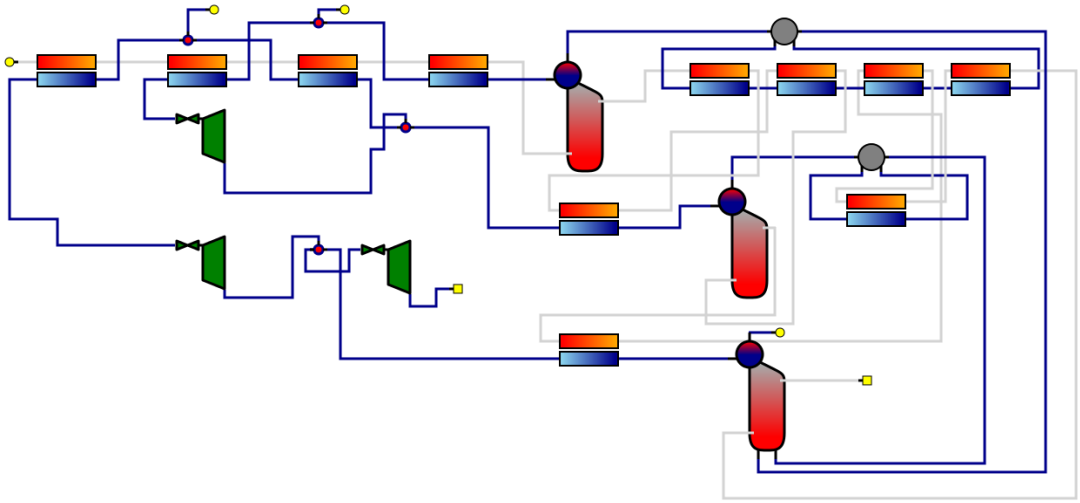

Figura 4.9 Disegno SVG del ciclo combinato

Sono presenti gli scambiatori, gli economizzatori, gli evaporatori, lo scambiatore di bassa pressione, 3 turbine di alta, media e bassa pressione, i collettori, il sistema pompa-valvola e l'evaporatore di bassa pressione, oltre agli elementi di contorno, rami e pozzi lato vapore e lato fumi.

Le dimensioni del sistema da risolvere sono aumentate, infatti in totale si hanno 83 equazioni in 83 variabili, con 17 stati.

È in casi come questo che si vede l'utilità dell'assiematore, che si occupa di gestire i vari collegamenti e assegnare ad ogni elemento la propria posizione all'interno della matrice dello jacobiano, che nel modello analizzato ha 6889 elementi.

Un altro vantaggio per l'uso dell'assiematore è la frequente ripetizione di componenti di impianto dello stesso tipo, scambiatori, evaporatori,turbine…
Quindi con la procedura descritta si può ottenere una struttura complessa a partire da componenti semplici, combinandoli rispettando le regole di un linguaggio grafico, l'SVG.

4.3.3 Costruzione del sistema linearizzato

Legato al problema del controllo, c'è la realizzazione del sistema linearizzato associato all'impianto globale: il modello assiemato fornisce le equazioni, algebriche e differenziali e lo jacobiano globale. Con queste informazioni è possibile ricavare il sistema linearizzato dell'impianto e la funzione di trasferimento, utile per eseguire degli studi di controllo sull'impianto.

Nel sistema

$$\begin{cases} 0 = g(z, x, u) \\ \tau\dot{x} = f(z, x, u) \end{cases} \quad (4.3)$$

con

- z: vettore delle variabili algebriche
- x: vettore degli stati
- u: vettore degli ingressi
- g: vettore delle equazioni algebriche

f: vettore delle equazioni differenziali
τ: matrice costanti di tempo

sono noti il vettore f, il vettore g e i blocchi dello jacobiano, calcolati dai singoli moduli.

Si considerano i valori delle variabili e degli ingressi a regime, x_0, z_0, u_0, per i quali vale

$$\begin{cases} 0 = f(x_0, z_0, u_0) \\ 0 = g(x_0, z_0, u_0) \end{cases}$$

Si calcolano le variazioni delle variabili, rispetto al valore di regime

$$\Delta x = x(t) - x_0$$
$$\Delta z = z(t) - z_0$$
$$\Delta u = u(t) - u_0$$

Se si considerano piccole variazioni delle variabili, il sistema (4.3) può essere riscritto nella forma

$$\begin{cases} 0 = \left(\frac{dg}{dz}\right)_0 \Delta z + \left(\frac{dg}{dx}\right)_0 \Delta x + \left(\frac{dg}{du}\right)_0 \Delta u \\ \tau \dot{\Delta x} = \left(\frac{df}{dx}\right)_0 \Delta x + \left(\frac{df}{dz}\right)_0 \Delta z + \left(\frac{df}{du}\right)_0 \Delta u \end{cases} \quad (4.4)$$

considerando come incognite le variazioni delle variabili algebriche e degli stati.

Risolvendo il sistema (4.4) si trovano le variazioni rispetto ai valori di regime; l'andamento delle variabili algebriche e differenziali si ricava da

$$x(t) = \Delta x + x_0$$
$$z(t) = \Delta z + z_0$$

Per piccole variazioni degli ingressi, il sistema può essere studiato considerando il sistema linearizzato (4.4), poiché per tali condizioni il sistema globale e quello linearizzato si comportano allo stesso modo.
La costruzione del sistema linearizzato è stata affrontata solo teoricamente, la sua implementazione è stata tralasciata per considerare il problema dell'identificazione, descritto nel capitolo successivo.
Infatti per simulare il funzionamento dell'impianto è necessario conoscere anche i valori dei parametri, relativi sia al funzionamento statico, come i coefficienti di scambio, sia al funzionamento dinamico, come le costanti di tempo.
Verrà descritta, quindi, l'identificazione statica e dinamica dei parametri, a partire da prove sperimentali e dati di progetto (bilanci termici), senza considerare le misure geometriche, come la lunghezza dei tubi, il numero dei tubi, il loro diametro…
Verranno inoltre riportati i transitori, ottenuti dalla simulazione del modello creato con l'assiematore.

Capitolo 5. Identificazione dei parametri

Dopo aver realizzato il modello dell'impianto di ciclo combinato con l'assiematore, si vogliono trovare i valori dei parametri di funzionamento statici e dinamici, per effettuare la simulazione del sistema.
In particolare per il funzionamento statico occorrono i parametri della turbina e i coefficienti di scambio degli scambiatori, degli economizzatori e degli evaporatori, mentre per la parte "dinamica" le costanti di tempo degli scambiatori, degli economizzatori e degli evaporatori.
Si è deciso di effettuare l'identificazione a partire dai dati di progetto descritti in seguito e dalle misurazioni di alcune grandezze a disposizione, mentre non vengono considerati dati geometrici, come ad esempio il numero di tubi in parallelo o il loro diametro e valori fisici come la massa del metallo.
Per prima cosa sono stati identificati i parametri del funzionamento statico, quindi noti questi valori si è proceduto con la seconda parte dell'identificazione, relativa alla dinamica.
Nei paragrafi seguenti vengono descritti i dati di progetti usati nell'identificazione, quindi viene presentata l'identificazione e vengono mostrati i risultati ottenuti con il modello completo del ciclo combinato, ottenuto con l'assiematore.

5.1 Dati di progetto

L'obiettivo è trovare i valori dei parametri di funzionamento statici e dinamici a partire da dati di progetto e dalla misurazione di alcune grandezze.
Per i dati di progetto si hanno a disposizione gli schemi di bilancio termico dell'impianto nella condizione di riferimento nominale con turbogas a pieno carico e a carico variabile, 85%, 70% e 60%.
Sono inoltre disponibili le misurazioni di alcune grandezze, tra le quali la potenza meccanica prodotta dall'impianto, le temperature vapore degli scambiatori, le pressioni del corpo cilindrico degli evaporatori, ottenute dal funzionamento dell'impianto in condizioni nominali, con variazione di portata e temperatura dei gas in ingresso.
Per i parametri che si vogliono calcolare, è importante conoscere l'andamento delle temperature degli scambiatori e degli economizzatori, e le pressioni degli evaporatori.
Purtroppo tra i dati a disposizione non erano presenti tutte le grandezze necessarie, quindi per l'identificazione di alcuni parametri tali grandezze sono state sostituite da quelle ottenute dalla simulazione del modello, come verrà precisato nei prossimi paragrafi.

5.2 Identificazione statica

L'identificazione statica riguarda i parametri delle turbine e i coefficienti di scambio di scambiatori, economizzatori ed evaporatori.
Mentre per le turbine si avevano a disposizione tutte le grandezze necessarie per il calcolo dei parametri, per i coefficienti di scambio mancavano delle informazioni, quindi è stata effettuata un'identificazione basata sul bilancio termico, considerando i valori delle variabili a regime.

5.2.1 Identificazione statica turbina

Si considera l'equazione algebrica della portata della turbina, descritta nel capitolo 3 e riportata di seguito

$$w_0 = k_v\theta\sqrt{\frac{\rho k_t^2(p_0^2-p_1^2)}{(k_t^2+k_v^2\theta^2)p_0}} \quad (5.1)$$

Si pone $\frac{\rho}{p_0} = \frac{\rho_0 T_0}{T p_0}$

con

- ρ_0 valore densità calcolato nella fase di inizializzazione
- T_0 valore iniziale della temperatura
- p_0 valore iniziale della pressione
- T valore della temperatura nel tempo

L'equazione (5.1) può essere quindi riscritta nella forma seguente

$$w_0 = K_{globale}\sqrt{\frac{{p_0}^2-{p_1}^2}{T}} \quad (5.2)$$

dalla quale si ottiene

$$K_{globale} = \frac{k_v k_t \theta}{\sqrt{(k_t^2 + k_v^2\theta^2}}\sqrt{\frac{\rho_0 T_0}{p_0}}$$

Conoscendo l'andamento nel tempo delle seguenti grandezze:

- w_0
- p_0
- p_1
- T

è stato possibile costruire un grafico, considerando in ordinata la portata in ingresso, w_0, e in ascissa il prodotto $\sqrt{\frac{{p_0}^2-{p_1}^2}{T}}$
I calcoli sono stati effettuati sulle tre turbine presenti nell'impianto.

In figura 5.1 è riportato il grafico relativo ai risultati ottenuti per la turbina di alta pressione.

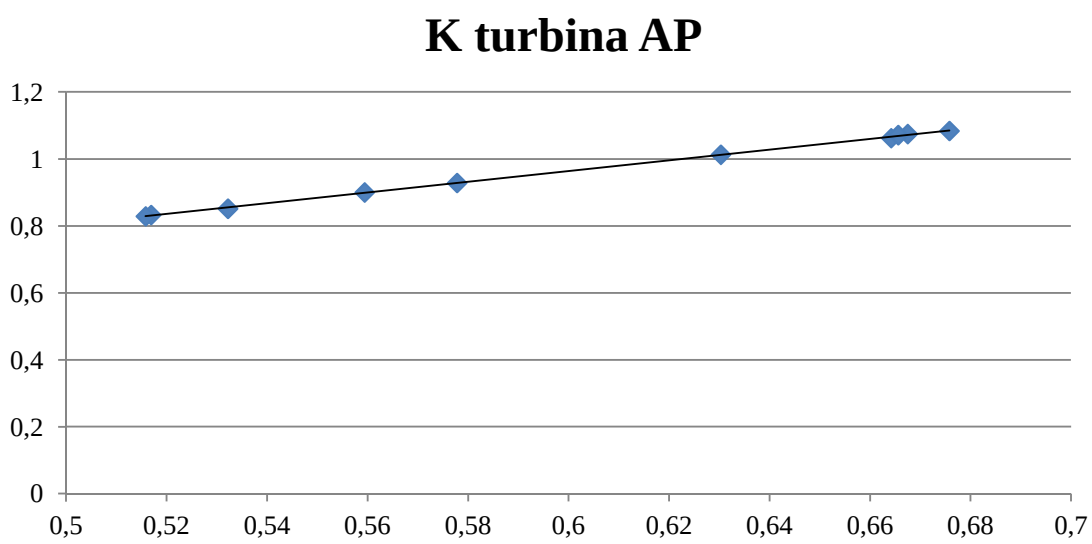

Figura 5.1 Variazione del parametro della turbina al variare della pressione, delle portate e della temperatura

In ascissa sono riportati i valori normalizzati della portata in ingresso, in ordinata il prodotto $\sqrt{\frac{{p_0}^2 - {p_1}^2}{T}}$ sempre considerando i valori normalizzati delle grandezze.

Come si può vedere dalla figura 5.1 tra w_0 e $\sqrt{\frac{{p_0}^2 - {p_1}^2}{T}}$ esiste un legame lineare, espresso nell'equazione (5.2); il coefficiente angolare della retta corrisponde al valore di $K_{globale}$, parametro cercato.
Per ottenere tale valore, si può usare una regressione lineare tra i diversi punti calcolati.

Considerando un valore in ascissa e il corrispondente valore in ordinata, $K_{globale}$ si ottiene dalla seguente equazione

$$y = 1{,}6015x + 0{,}0028$$

Il valore del coefficiente R^2 è 0,9995, quindi la regressione definisce in modo corretto la relazione tra w_0 e $\sqrt{\frac{{p_0}^2 - {p_1}^2}{T}}$.
Lo stesso procedimento è stato applicato alla turbina di media pressione e a quella di bassa.
Anche in questi due casi il valore di R^2 è prossimo a 1, quindi la regressione spiega bene la correlazione tra le grandezze.

5.2.2 Identificazione statica coefficienti di scambio

Per identificare i coefficienti di scambio è possibile usare la stesso procedimento applicato alla turbina.

Si parte dall'equazione di bilancio termico a regime

$$w_v(h_{v\,out} - h_{v\,in}) = \sigma(T_{g\,in} - T_v) = w_g(h_{g\,in} - h_{g\,out})$$

con

- w_v portata di vapore
- $h_{v\,in}$ e $h_{v\,out}$ entalpia di vapore in ingresso e in uscita
- $T_{g\,in}$ temperatura fumi in ingresso
- T_v temperatura vapore in uscita
- w_g portata fumi
- $h_{g\,in}$ e $h_{g\,out}$ entalpia fumi in ingresso e in uscita
- σ coefficiente di scambio, parametro da calcolare

È noto che aumentando la portata fumi, il coefficiente di scambio aumenta, mentre diminuendo la portata fumi il coefficiente di scambio diminuisce. Si vuole, però, trovare una relazione che descriva meglio il legame tra queste due grandezze.

Dall'equazione scritta sopra, considerando le prime due uguaglianze, si può calcolare il parametro σ, dipendente da w_g

$$\sigma(w_g) = \frac{w_v(h_{v\,out} - h_{v\,in})}{(T_{g\,in} - T_v)} \quad (5.3)$$

Conoscendo l'andamento della portata di vapore, dell' entalpia di vapore in ingresso e in uscita e della temperatura dei fumi in ingresso e del vapore in uscita, dalla (5.3) è possibile ricavare l'andamento del coefficiente di scambio.
Riportando poi in un grafico i valori del coefficiente di scambio in ordinata e della portata fumi in ascissa, si può ricavare la relazione cercata.

Purtroppo non è stato possibile procedere in questo modo, in quanto tra i dati a disposizione mancava l'andamento di alcune grandezze.
Si è deciso allora di ricavare i valori dei coefficienti di scambio tramite un bilancio termico, sostituendo nell'equazione (5.3) i valori delle grandezze a regime, ricavati dallo schema dell'impianto.
Poiché è nota solo la temperatura iniziale dei fumi e non i valori intermedi, è stato necessario risolvere in cascata le equazioni di bilancio, a partire dal primo elemento presente nello schema, lo scambiatore RH2. Dall'equazione di bilancio si ricava l'entalpia fumi in uscita, unica incognita e, a partire da questa, la temperatura fumi in uscita, tramite le relazioni delle tavole dei fumi.
La temperatura dei fumi in uscita all'RH2 coincide con quella in ingresso all'elemento successivo, lo scambiatore SH2; si ripete quindi lo stesso procedimento per ogni elemento presente, fino ad ottenere tutti i valori del circuiti fumi.

Il valore di σ è sempre ricavato dalla (5.3) in condizioni di regime.

Per non trascurare la variazione di σ al variare della portata fumi, si è deciso di considerare il valore ricavato dal metodo come un valore iniziale, σ_0.
Nell'evoluzione, a partire dal valore della portata fumi, si ricava il valore del coefficiente di scambio usando la seguente relazione, fornita dalla letteratura

$$\sigma = \sigma_0 \left(0.95 \sqrt{\frac{w_g}{w_n}} + 0.05 \right)$$

dove w_n è il valore della portata fumi in condizioni nominali, quando il coefficiente di scambio assume valore pari a σ_0.
Trovati i parametri "statici", è stata effettuata una prima simulazione, per verificare la validità dell'identificazione fatta.
I valori dei parametri dinamici non erano ancora noti, quindi si è deciso di ricavarli a partire da dati geometrici e fisici di impianti simili. ([4])

Noti i valori di regime, è stata effettuata una simulazione variando la temperatura fumi e la portata fumi in ingresso.
La loro variazione nel tempo è riportata in figura 5.2 e 5.3.

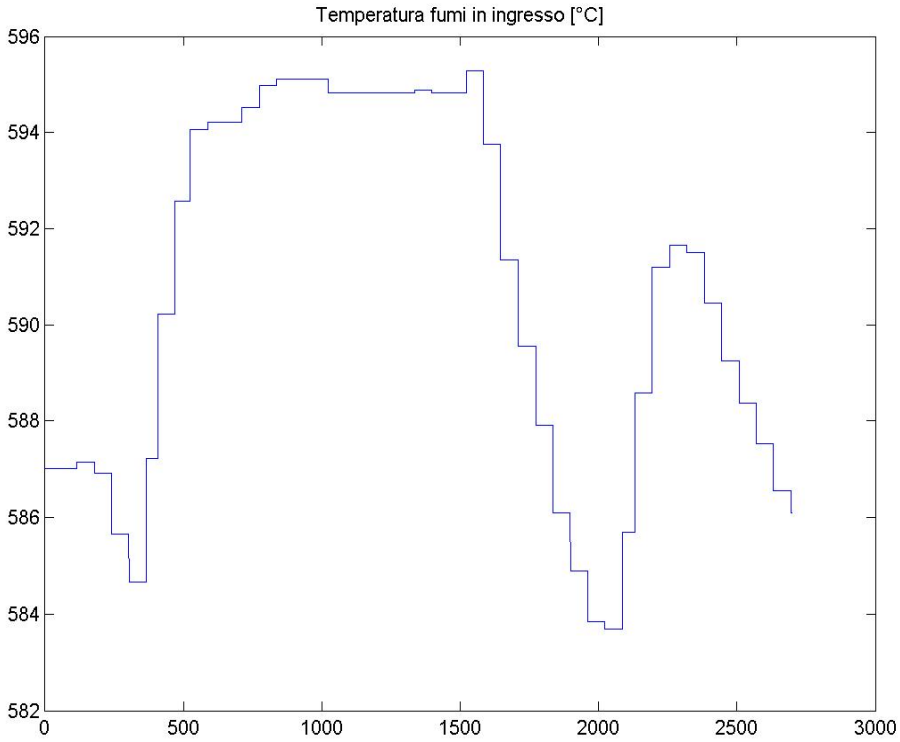

Figura 5.2 Andamento nel tempo della temperatura fumi in ingresso

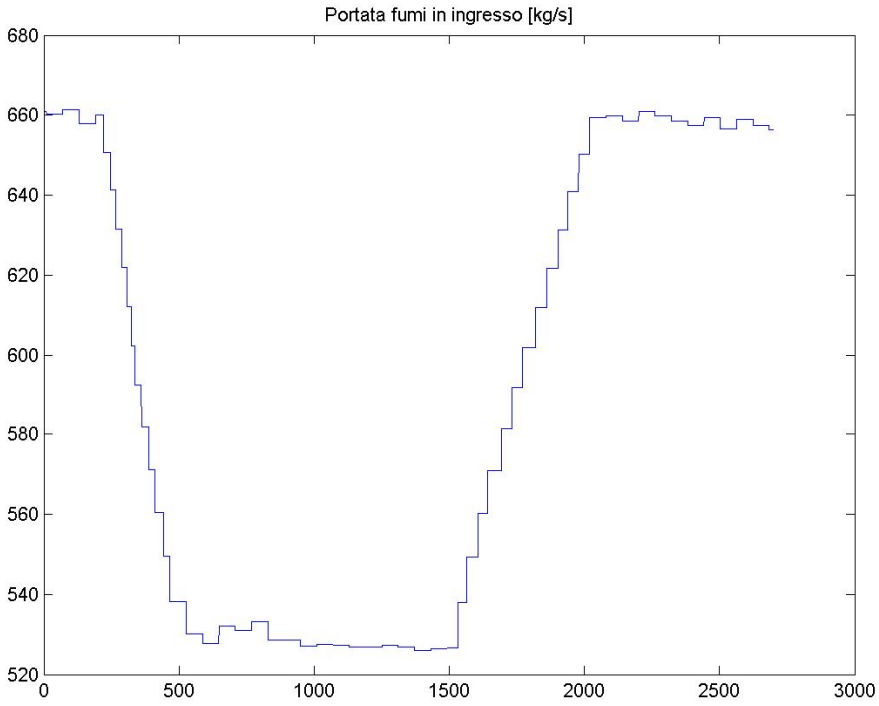

Figura 5.3 Andamento nel tempo della portata fumi in ingresso

Nelle figure 5.4-5.6 sono riportati i confronti tra i valori sperimentali e quelli ottenuti con la simulazione, relativi alla pressione del corpo cilindrico nell'evaporatore di alta pressione e nell'evaporatore di media pressione e alla potenza meccanica prodotta dall'impianto.
In rosso è riportato l'andamento sperimentale, mentre in blu quello ottenuto dalla simulazione.

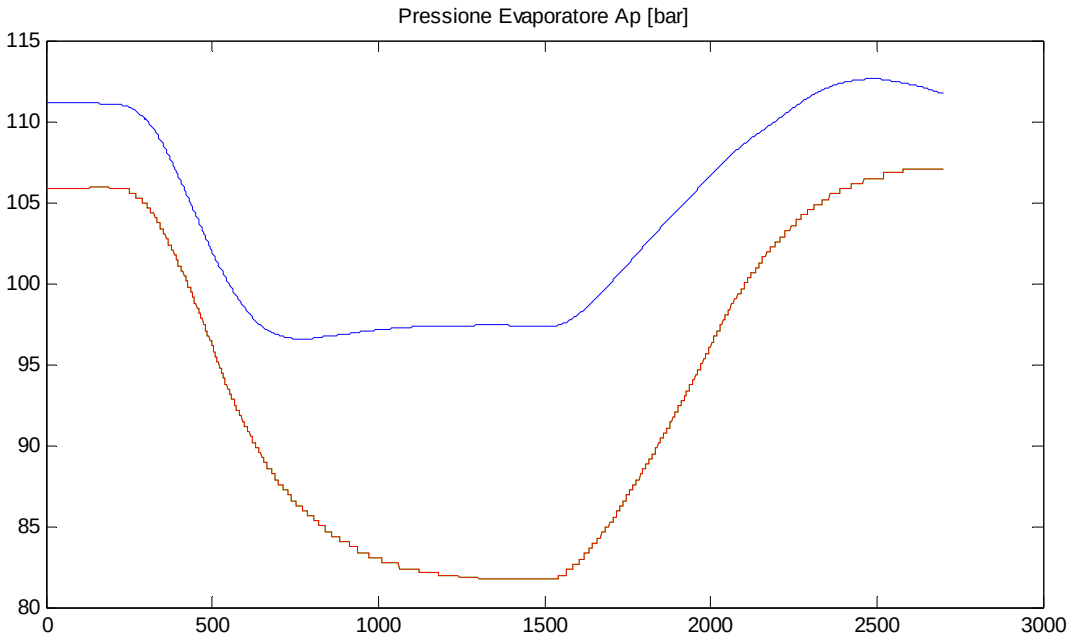

Figura 5.4 Confronto tra andamento sperimentale (rosso) e andamento simulato (blu) della pressione del corpo cilindrico dell'evaporatore di alta pressione

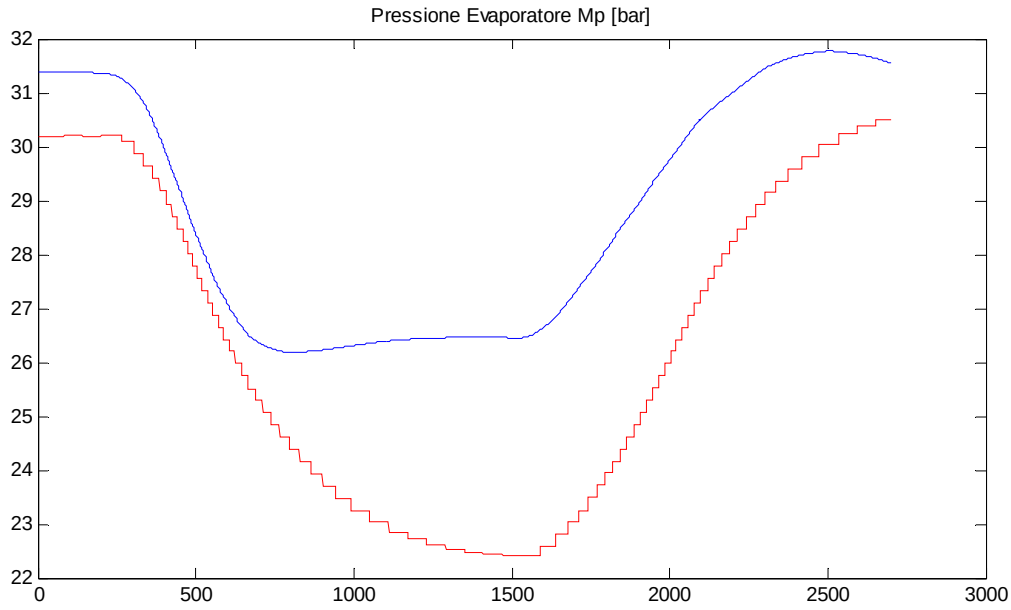

Figura 5.5 Confronto tra andamento sperimentale (rosso) e andamento simulato (blu) della pressione del corpo cilindrico dell'evaporatore di media pressione

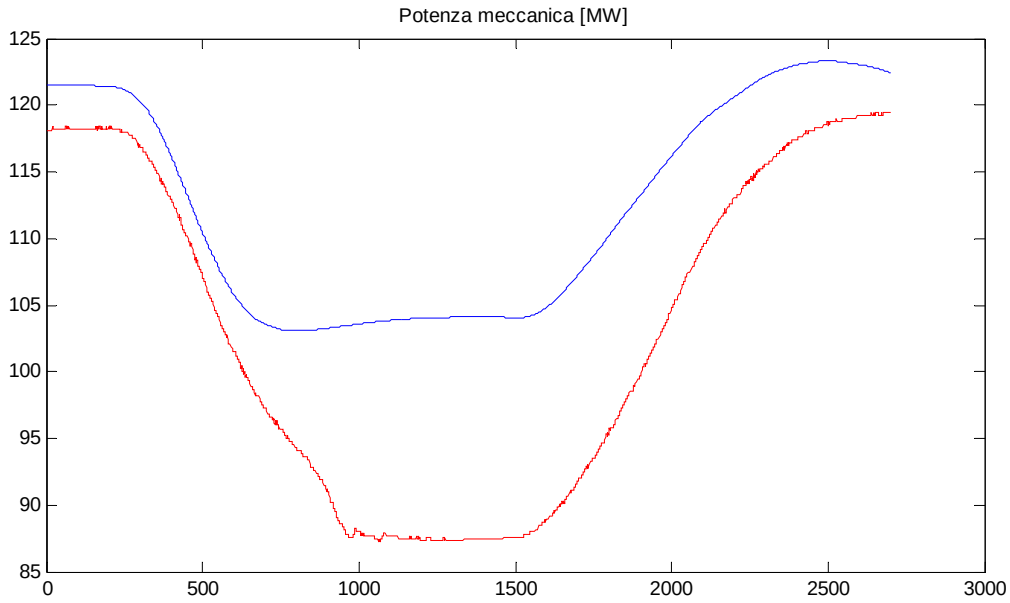

Figura 5.6 Confronto tra andamento sperimentale (rosso) e andamento simulato (blu) della potenza meccanica prodotta dall'impianto

Come si può vedere osservando i grafici riportati, l'andamento delle grandezze sperimentali e di quelle ottenute dalla simulazione è simile; ci sono differenze rispetto ai valori iniziali.
Questo può dipendere dal fatto che nell'identificazione dei coefficienti di scambio, non avendo a disposizione tutti i dati, sono stati usati i valori di regime, cioè i valori di progetto, che potrebbero essere diversi dai valori raggiunti nel funzionamento reale.

5.3 Identificazione dinamica

Nella fase di identificazione statica dei parametri, per simulare il funzionamento del sistema sono stati usati dei valori delle costanti di tempo ricavati da considerazioni su impianti simili a quello testato.
Per trovare in modo più accurato i valori delle costanti di tempo, è necessario realizzare una procedura di identificazione dinamica.

Il modello è descritto dalle seguenti equazioni

$$0 = g(z, x, u, p)$$
$$\tau\dot{x} = f(z, x, u, p)$$

con

- p parametri che influenzano il regime, identificati nel paragrafo precedente
- τ parametri che influenzano la dinamica

Dalle prove sperimentali si conoscono l'andamento nel tempo degli ingressi, T_g e w_g, delle pressioni degli evaporatori, e di alcune temperature degli scambiatori e degli economizzatori, cioè degli stati del sistema.
Mancano i dati degli scambiatori SH2 e RH2 e degli economizzatori ECO1, ECO2 ed ECO3.
Per poter applicare il metodo di identificazione, i valori reali di queste variabili sono stati sostituiti con quelli sperimentali, ottenuti dalla simulazione assegnando le costanti di tempo come detto sopra.

L'identificazione prevede tre "fasi":

a. a partire dall'andamento degli stati, è possibile ricavare la variazione della loro derivata nel tempo
b. noti i valori nel tempo degli stati *x*, degli ingressi *u* e dei parametri *statici*, si possono risolvere le equazioni algebriche ed ottenere l'andamento nel tempo delle variabili algebriche *z*
c. con i dati ottenuti dalle prime due fasi, è possibile trovare per ogni istante di tempo il valore delle *f*, cioè delle equazioni differenziali.

A questo punto è possibile procedere con l'identificazione dei parametri.

Infatti conoscendo, per ogni stato *x*, il valore di f e il valore della sua derivata $\dot{x}$, è possibile ricavare il valore delle τ, usando la seguente relazione

$$\tau = \frac{\int_0^T f\ dt}{\int_0^T \dot{x}\ dt}$$
$$\tau > 0$$

con [0,T]: intervallo in cui si calcolano i valori di f e di $\dot{x}$.

La procedura descritta è ragionevole se le derivate nei transitori sperimentali sono relativamente grandi: la derivata compare al denominatore, quindi un valore piccolo di questa grandezza determina un valore grande di τ.

Per questo motivo, dai dati ottenuti si scartano i valori delle derivate troppo piccole o i valori che danno un τ negativo.

Per capire meglio il procedimento, vediamo la sua applicazione nel calcolo della costante di tempo dell'evaporatore di alta pressione.

Si è scelto l'intervallo da 0 a 500 secondi; ad ogni Δt di 20 secondi è stato calcolato il valore della f e della derivata; dal loro quoziente si ottiene il valore di τ, in ogni Δt.
Il valore finale della costante di tempo si ottiene come media dei valori trovati in ogni intervallo, scartando i valori non accettabili.

Il calcolo della derivata è una fase fondamentale: occorre scartare i valori di derivata troppo piccoli e i valori che determinano una costante di tempo negativa.

In figura 5.7 è riportato l'andamento nel tempo della τ dell'evaporatore.

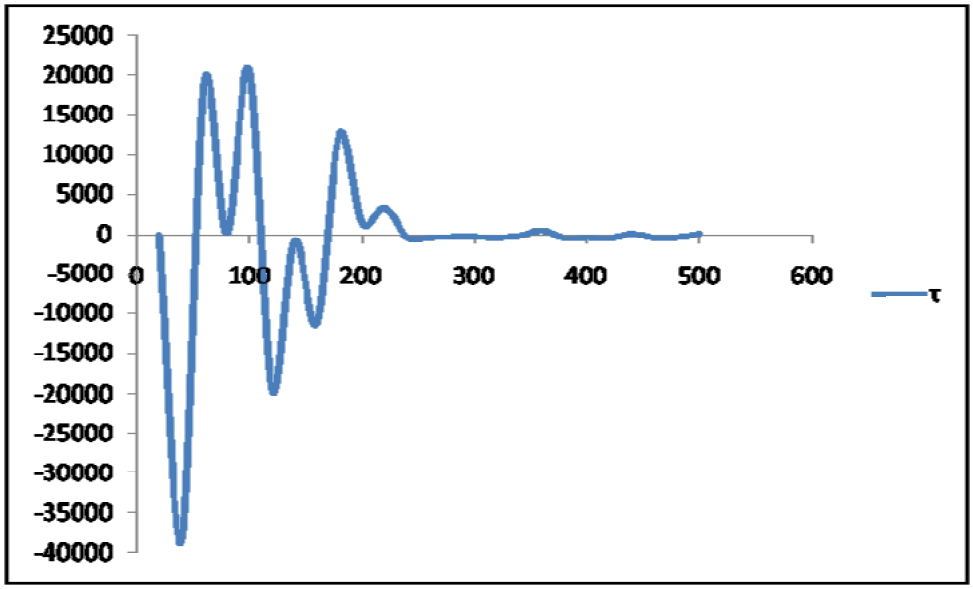

Figura 5.7 Andamento nel tempo della τ dell'evaporatore alta pressione

Sono stati scartati 19 valori, su un totale di 25, perché non rispettavano le condizioni richieste. Dei 6 valori rimasti, per calcolare la media sono stati considerati valori vicini tra loro, in modo da avere una media significativa.
Si è ottenuto così un valore di $\tau = 169{,}5\ secondi$.

Il metodo di identificazione descritto è stato applicato a tutti gli stati del modello globale, ottenendo i risultati riportati di seguito.

$$\tau_{\ pressione\ evAp} = 169{,}5\ secondi$$
$$\tau_{\ pressione\ evMp} = 164{,}21\ secondi$$
$$\tau_{\ SH1} = 1225{,}562\ secondi$$
$$\tau_{\ RH1} = 1490{,}818\ secondi$$
$$\tau_{\ SH2} = 160{,}645\ secondi$$
$$\tau_{\ RH2} = 218{,}385\ secondi$$
$$\tau_{\ SHMP} = 302{,}48\ secondi$$

$$\tau_{SHBP} = 119{,}53\ secondi$$
$$\tau_{ECO\ MP} = 1705{,}45\ secondi$$
$$\tau_{ECO\ AP} = 204\ secondi$$
$$\tau_{ECO\ 3} = 324{,}246\ secondi$$
$$\tau_{ECO\ 2} = 366{,}168\ secondi$$
$$\tau_{ECO\ 1} = 758{,}4275\ secondi$$
$$\tau_{pressione\ evBp} = 536{,}78\ secondi$$

I valori delle costanti di tempo ipotizzate erano invece i seguenti

$$\tau_{scambiatori\ vapore} = 135\ secondi$$
$$\tau_{scambiatori\ acqua\ liquida} = 139\ secondi$$
$$\tau_{evaporatore\ alta\ pressione} = 213\ secondi$$
$$\tau_{evaporatore\ mediapressione} = 151\ secondi$$
$$\tau_{evaporatore\ bassa\ pressione} = 430\ secondi$$

Come si può vedere, erano stati ipotizzati valori di un ordine di grandezza superiore per lo scambiatore SH1, lo scambiatore RH1 e per l'economizzatore di media pressione, gli altri valori sono simili.
Infine è stata effettuata la simulazione usando questi valori delle costanti di tempo; il transitorio ottenuto appare avere una oscillazione poco smorzata, di periodo pari a circa 3000 secondi; sembra quindi che il sistema sia instabile.

Il risultato ottenuto dipende da un *errore* di calcolo delle costanti di tempo a causa delle poche informazioni disponibili.
Nelle prove sperimentali le variazioni sono relativamente lente, quindi appare difficile poter valutare in modo corretto i valori delle costanti di tempo a partire da tale prova, che è l'unica fornita.
Inoltre tra le condizioni da inserire nella valutazione dei parametri, occorre aggiungere anche quella di avere i poli del sistema linearizzato sufficientemente smorzati, come è in effetti nella prova sperimentale d'impianto.

5.4 Prova di validazione dei parametri di funzionamento statico

Come prova di validazione dei parametri di funzionamento statico, è stato fatto un confronto tra i valori delle temperature a regime ottenuti dalla simulazione, e quelli descritti nel bilancio, considerando il funzionamento dell'impianto al 70% del carico nominale.
A partire dalla condizione di regime a pieno carico, sono stati variati gli ingressi di portata e temperatura fumi, usando i valori definiti nel bilancio al 70% del carico.
La portata fumi varia da 661 kg/s a 541 kg/s, mentre la temperatura da 587 °C a 569 °C.
Per i parametri di funzionamento statici e dinamici sono stati usati i valori trovati applicando il metodo descritto prima.
Nella tabella 5.1 vengono riportati i valori di bilancio e quelli ottenuti dalla simulazione per le temperature di vapore in uscita dello scambiatore SH2, dello scambiatore RH2, e degli scambiatori di media e bassa pressione.

Variabile	Valore bilancio	Valore simulazione
Temperatura Sh2	539 °C	530 °C
Temperatura Rh2	548 °C	540 °C
Temperatura Sh media pressione	301 °C	290 °C
Temperatura Sh bassa pressione	225 °C	239 °C

Tabella 5.1 Confronto tra i valori delle temperature a regime

I valori ottenuti a regime sono simili, quindi i parametri sono stati identificati in modo corretto.

Si può ritenere che l'identificazione è valida, ma occorre avere maggiori informazioni a disposizione per ottenere dei risultati migliori, come già detto nel paragrafo 5.3.

Conclusioni

In questa tesi, è stata proposta una procedura per la costruzione automatica di impianti complessi di tipo termo-idraulico a partire dai modelli dei singoli componenti, estendibile anche a processi chimici. Grazie al software realizzato, è stato possibile costruire il simulatore di un impianto a ciclo combinato (come noto, una quota rilevante dell'energia nella rete elettrica italiana e' fornita da impianti di questo tipo).

Un problema rilevante che si incontra nella modellizzazione è quello dell'identificazione dei parametri statici e dinamici di funzionamento, usando dati sperimentali e/o di progetto (per esempio bilanci termici di caldaia). Data la scarsità di prove disponibili, si raccomanda di approfondire la validazione del modello in successivi lavori.
La metodologia proposta si considera comunque adeguata in relazione all'obiettivo di realizzare un modello dinamico di ciclo combinato a tre livelli di pressione, a partire solo dai dati sperimentali.

Riguardo al simulatore, si possono ipotizzare dei possibili sviluppi futuri.

In particolare va approfondita la parte di Human-Machine Interface, integrando il modello realizzato dall'assiematore con il codice C del solutore e fornendo un'interfaccia grafica utente (GUI) per permettere ad esempio di creare i grafici delle variabili memorizzate o modificare i valori degli ingressi durante la simulazione.

Un cenno infine al problema della simulazione della regolazione.
A questo fine i controllori possono essere realizzati come particolari modelli assiemabili con gli altri moduli dei vari componenti.

I modelli sviluppati sono per lo più non lineari. Per lo studio del controllo dell'impianto, è opportuno disporre anche del sistema linearizzato associato al modello globale. Nel capitolo 4 è stato affrontato questo aspetto, ma occorre verificare che il sistema linearizzato e il modello globale abbiano effettivamente lo stesso comportamento per piccole perturbazioni degli ingressi.

Bibliografia

[1] S. Bittanti, M. Bottinelli, A. De Marco, M. Facchetti, W. Prandoni "Performance assessment of the control system of once-through boilers". 13th Conference process control '01, Strbske Pleso (Slovakia), 2001.

[2] "*Guida SVG*" http://xml.html.it/guide/leggi/54/guida-svg/

[3] A. Comi "Costruzione automatica di modelli di impianti relativi a processi termoidraulici". Tesi di laurea triennale, Settembre 2007.

[4] F. Camozzi "Strategie di avviamento per un esercizio flessibile di impianti termo elettrici a ciclo combinato". Tesi di laurea magistrale, Aprile 2008.

Appendice

A1. File SVG del modello ciclo combinato

Il file SVG del modello è diviso in due parti: la prima comprende la libreria dei moduli, la seconda lo schema del disegno dell'impianto e i collegamenti tra gli elementi.
Di seguito si riporta il file SVG del modello dell'impianto a ciclo combinato, relativo solo alla parte dello schema.

```
<g id="SCHEMA">
 <rect x="0" y="0" width="100%" height="100%" fill="white"/>
  <g id="COLLEGAMENTI" fill="none" stroke-width="3">
    <polyline id="L00" points="511,100 430,100" stroke="lightgray"/>
    <polyline id="L01" points="361,100 280,100" stroke="lightgray"/>
   <polyline id="L02" points="211,100 130,100" stroke="lightgray"/>
   <polyline id="L03" points="61,100 40,100" stroke="lightgray"/>
   <polyline id="L04" points="580,100 620,100 620,205 646,205"
     stroke="lightgray"/>
    <polyline id="L05" points="721,145 760,145 760,110 811,110"
     stroke="lightgray"/>
   <polyline id="L06" points="646,120 580,120" stroke="darkblue"/>
   <polyline id="L07" points="511,120 460,120  460,55 395,55"
     stroke="darkblue"/>
   <polyline id="L08" points="220,165 185,165 185,120 211,120 "
     stroke="darkblue"/>
   <polyline id="L09" points="277,216 277,250 445,250 445,200 460,200
     460,160 485,160 485,165" stroke="darkblue"/>
   <polyline id="L10" points="361,120 330,120 330,75 245,75"
     stroke="darkblue"/>
   <polyline id="L11" points="225,75 155,75 155,120 130,120"
     stroke="darkblue"/>
   <polyline id="L12" points="61,120 30,120 30,280 85,280 85,310 220,310"
     stroke="darkblue"/>
   <polyline id="L13" points="277,361 277,370 355,370 355,300 385,300
     385,305" stroke="darkblue"/>
   <polyline id="L14" points="235,65 235,40 255,40" stroke="darkblue"/>
   <polyline id="L15" points="385,45 385,40 405,40" stroke="darkblue"/>
   <polyline id="L16" points="280,120 305,120 305,55 375,55"
     stroke="darkblue"/>
   <polyline id="L17" points="475,175 445,175 445,120 430,120"
     stroke="darkblue"/>
   <polyline id="L18" points="1000,209 905,209 860,209 860,235"
     stroke="darkblue"/>
   <polyline id="L19" points="835,265 800,265  800,290 730,290"
     stroke="darkblue"/>
   <polyline id="L20" points="663,290  580,290 580,175 495,175"
     stroke="darkblue"/>
   <polyline id="L21" points="380,315 370,315 370,340 420,340 420,315
     433,315 " stroke="darkblue"/>
   <polyline id="L22" points="490,366 490,380 520,380 520,360 535,360"
     stroke="darkblue"/>
   <polyline id="L23" points="395,315 410,315 410,440 662,440"
     stroke="darkblue"/>
   <polyline id="L24" points="729,440 855,440" stroke="darkblue"/>
   <polyline id="L25" points="900,65 720,65 671,65 671,90"
     stroke="darkblue"/>
   <polyline id="L26" points="730,270 790,270 790,180 900,180 900,110
     912,110" stroke="lightgray"/>
   <polyline id="L27" points="662,420 640,420 640,390 909,390 909,289"
     stroke="lightgray"/>
   <polyline id="L28" points="729,420 1100,420 1100,160 1005,160 1005,110
     1011,110" stroke="lightgray"/>
   <polyline id="L29" points="879,130 912,130" stroke="darkblue"/>
   <polyline id="L30"  points="979,130 1012,130" stroke="darkblue"/>
   <polyline id="L31"  points="1079,130 1112,130" stroke="darkblue"/>
```

```
 <polyline id="L32"  points="1180,130 1212,130 1212,85 931,85 931,82"
   stroke="darkblue"/>
 <polyline id="L33"  points="811,130 780,130 780,85 909,85 909,82"
   stroke="darkblue"/>
 <polyline id="L34"  points=" 991,280 950,280 950,230 1009,230 1009,226"
   stroke="darkblue"/>
 <polyline id="L35"  points=" 1060,280 1130,280 1130,230 1031,230
   1031,226" stroke="darkblue"/>
 <polyline id="L36" points="879,110 890,110 890,230 650,230 650,270
   663,270" stroke="lightgray"/>
 <polyline id="L37" points="980,110 990,110 990,180 930,180 930,400
   830,400 830,350 835,350" stroke="lightgray"/>
 <polyline id="L38" points="1080,110 1090,110 1090,245 980,245 980,260
   991,260" stroke="lightgray"/>
 <polyline id="L39" points="1060,260 1105,260 1105,110 1111,110"
  stroke="lightgray"/>
   <polyline id="L40" points="1180,110 1255,110 1255,600 850,600 850,525
   855,525" stroke="lightgray"/>
 <polyline id="L41" points="930,465 1005,465" stroke="lightgray"/>
 <polyline id="L42" points="879,410 905,410" stroke="darkblue"/>
 <polyline id="L43" points="940,65 1220,65 1220,570 890,570 890,555"
   stroke="darkblue"/>
 <polyline id="L44" points="1040,209 1055,209 1150,209 1150,560 910,560
   910,555" stroke="darkblue"/>
</g>
<g id="ELEMENTI">
 <g id="fi1" x="20" y="90">
  <use xlink:href="#B_pozFuDx"/>
  <use xlink:href="#I_pozFuDx"/>
 </g>
 <g id="evA" x="641" y="85">
  <use xlink:href="#B_evaporatore"/>
  <use xlink:href="#I_evaporatore"/>
 </g>
 <g id="evM" x="830" y="230">
  <use xlink:href="#B_evaporatore"/>
  <use xlink:href="#I_evaporatore"/>
 </g>
 <g id="sh1" x="500" y="80">
  <use xlink:href="#B_scambiatore"/>
  <use xlink:href="#I_scambiatore"/>
 </g>
 <g id="rh1" x="350" y="80">
  <use xlink:href="#B_scambiatore"/>
  <use xlink:href="#I_scambiatore"/>
 </g>
 <g id="sh2" x="200" y="80">
  <use xlink:href="#B_scambiatore"/>
  <use xlink:href="#I_scambiatore"/>
 </g>
 <g id="rh2" x="50" y="80">
  <use xlink:href="#B_scambiatore"/>
  <use xlink:href="#I_scambiatore"/>
 </g>
 <g id="shM" x="650" y="250">
  <use xlink:href="#B_scambiatorek"/>
  <use xlink:href="#I_scambiatorek"/>
 </g>
 <g id="shB" x="650" y="400">
  <use xlink:href="#B_scambiatore"/>
  <use xlink:href="#I_scambiatore"/>
 </g>
 <g id="tAP" x="217" y="150">
  <use xlink:href="#B_turbina"/>
  <use xlink:href="#I_turbina"/>
 </g>
 <g id="tMP" x="217" y="295">
  <use xlink:href="#B_turbina"/>
  <use xlink:href="#I_turbina"/>
 </g>
```

```
<g id="tBP" x="430" y="300">
 <use xlink:href="#B_turbina"/>
 <use xlink:href="#I_turbina"/>
</g>
<g id="c01" x="370" y="40">
 <use xlink:href="#B_collettore"/>
 <use xlink:href="#I_collettore"/>
</g>
<g id="c02" x="220" y="60">
 <use xlink:href="#B_collettore"/>
 <use xlink:href="#I_collettore"/>
</g>
<g id="c03" x="470" y="160">
 <use xlink:href="#B_collettore"/>
 <use xlink:href="#I_collettore"/>
</g>
<g id="c04" x="370" y="300">
 <use xlink:href="#B_collettore"/>
 <use xlink:href="#I_collettore"/>
</g>
<g id="ap4" x="400" y="30">
 <use xlink:href="#B_pozVaSx"/>
 <use xlink:href="#I_pozVaSx"/>
</g>
<g id="ap5" x="250" y="30">
 <use xlink:href="#B_pozVaSx"/>
 <use xlink:href="#I_pozVaSx"/>
</g>
<g id="te0" x="530" y="350">
 <use xlink:href="#B_termiVapSx"/>
 <use xlink:href="#I_termiVapSx"/>
</g>
<g id="te1" x="900" y="400">
 <use xlink:href="#B_pozVaSx"/>
 <use xlink:href="#I_pozVaSx"/>
</g>
<g id="f01" x="1000" y="455">
 <use xlink:href="#B_termiFumSx"/>
 <use xlink:href="#I_termiFumSx"/>
</g>
<g id="pAP" x="900" y="45">
 <use xlink:href="#B_pompa-valvola"/>
 <use xlink:href="#I_pompa-valvola"/>
</g>
<g id="pMP" x="1000" y="189">
 <use xlink:href="#B_pompa-valvola"/>
 <use xlink:href="#I_pompa-valvola"/>
</g>
<g id="eMP" x="980" y="240">
 <use xlink:href="#B_economizzatore"/>
 <use xlink:href="#I_economizzatore"/>
</g>
<g id="eAp" x="800" y="90">
 <use xlink:href="#B_economizzatore"/>
 <use xlink:href="#I_economizzatore"/>
</g>
<g id="e3" x="900" y="90">
 <use xlink:href="#B_economizzatore"/>
 <use xlink:href="#I_economizzatore"/>
</g>
<g id="e2" x="1000" y="90">
 <use xlink:href="#B_economizzatore"/>
 <use xlink:href="#I_economizzatore"/>
</g>
<g id="e1" x="1100" y="90">
 <use xlink:href="#B_economizzatore"/>
 <use xlink:href="#I_economizzatore"/>
</g>
<g id="evBP" x="850" y="405">
 <use xlink:href="#B_evaporatore BP"/>
```

```
    <use xlink:href="#I_evaporatore BP"/>
   </g>
  </g>
</g>
```

Il tag <g> con id **COLLEGAMENTI** racchiude le linee di collegamento tra gli elementi dello schema; il tag <g> con id **ELEMENTI**, invece, definisce tutti gli elementi presenti; per ogni elemento è specificato il nome, la posizione nello schema e il riferimento alla libreria corrispondente.

A2. Codice di risoluzione del sistema non lineare

Il file **solutore.c** contiene il codice del solutore.

Tale codice può essere diviso in tre parti:

- iniziliazzazione del modello globale
- evoluzione del modello globale
- risoluzione del sistema non lineare associato al modello globale

Di seguito si riporta il codice della funzione **linsolve,** che risolve sistema lineare.
Il parametro A contiene i coefficienti del sistema, B i termini noti, ny è la dimensione del sistema da risolvere.
Si usa la funzione **dgesv**, definita nella libreria matematica Lapack

```
int linsolve(double *A, double *B, int ny, double *delta) {

int N, NRHS, LDA, LDB, INFO, *IPIV;
int i;

      N = LDA = LDB = ((int)ny);
      NRHS = 1;

      IPIV = (int*)malloc(N * sizeof(int));

      dgesv(&N, &NRHS, A, &LDA, IPIV, B, &LDB, &INFO);

      *delta = 0;

      for (i = 0; i < ny; i++)
            *delta += B[i] * B[i];

      free(IPIV);

      return (int)INFO;
};
```

A3. Libreria dei moduli

Il modello del ciclo combinato comprende i seguenti moduli:

- scambiatori di calore (con vapore e con acqua liquida)
- evaporatori (per alta-media pressione e per bassa pressione)
- collettore
- sistema pompe-valvole
- turbina

Per ogni modulo è stato realizzato un file .c, che definisce il singolo elemento, e il corrispondente header file, con la dichiarazione delle strutture.
Tutti i file sono stati inseriti nella libreria dei moduli.

Di seguito si riportano i file realizzati, degli elementi più importanti.

A3.1 Scambiatore di calore con vapore

Lo scambiatore di calore con vapore è identificato dalla sigla **vghe**.
Si riporta l'header file dell'elemento e il corrispondente codice C.

```
/** @file vghe_strutture.h
 * @brief Scambiatore di calore: Strutture
 * @author Arianna Comi
 * Componente VGHE (scambiatore di calore)
 * Strutture
 */

#ifndef _VGHE_STRUTTURE_H
#define _VGHE_STRUTTURE_H    1

#include "interfacce.h"
#include "common.h"
#include "sim_utils.h"
#include "riferimenti.h"

#define nstati_vghe    1
#define nalg_vghe 3
#define nti_vghe  9
#define nout_vghe 2

#define NCOMP     7

/* Tvghe contiene i terminali del modulo, con assegnata la corrispondente
 * intefaccia.
 * In particolare, lo scambiatore ha 4 terminali, due lato vapore e due
 * lato fumi
 */

typedef struct Tvghe {
  interfacciaVap t0, t1;
  interfacciaGas t2, t3;
} Tvghe;

/* Pvghe contiene i parametri dell'elemento
 *  - k: coefficiente equazione portata dello scambiatore
 *  - sigma e sigma0: coefficiente di scambio, fase di evoluzione e fase di
 *    inizializzazione
 *  - gas_compo[NCOMP]: vettore contenente la composizione dei fumi
 *  - thermo_vap_in, thermo_vap_out: struttura thermoprop, contenente le
 *    variabili termodinamiche del vapore, associate ai due terminali lato
 *    vapore
 *  - gasin, gasout: strutture gasprop, associate ai due terminali lato
 *    fumi, contengono le variabili termodinamiche del gas
 */

typedef struct Pvghe{
      double
            k,
            sigma,
            sigma0;
      double
            gas_compo[NCOMP];
      thermoprop
            thermo_vap_in,
            thermo_vap_out;
```

```
      gasprop
            gasin,
            gasout;
}Pvghe;

/* Cvghe contiene le costanti dello scambiatore
 *  - massa_m: massa del metallo
 */

typedef struct Cvghe{
      double massa_m;
}Cvghe;

/* Svghe contiene gli stati dello scambiatore
 *  - Tv: temperatura del vapore
 *  - stati[nstati_vghe]: vettore che contiene l'unico stato;
 *    nstati_vghe: numero degli stati
 */

typedef struct Svghe{
      double *Tv;
      double * stati[nstati_vghe];
}Svghe;

/* Vvghe contiene le variabili che devono essere visibili all'esterno del
 *  modulo e che non appartengono ad altre strutture.
 *  - **Jdz,**Jdx, **Jax, **Jaz: blocchi dello jacobiano
 *  - *f[nstati_vghe]: vettore delle equazioni differenziali
 *  - *g[nalg_vghe]: vettore delle equazioni algebriche
 *  - **tau: matrice delle costanti di tempo
 *  - rho: densità del vapore
 *  - rho_init: densità iniziale del vapore
 *  - T_init: temperatura iniziale del vapore
 *  - p_init: pressione iniziale del vapore
 *  - wn: valore nominale portata fumi
 */

typedef struct Vvghe{
      double  **Jdz, **Jdx, **Jax, **Jaz,
              *f[nstati_vghe],
              *g[nalg_vghe],
              **tau,
              rho,
              rho_init,
              T_init,
              p_init,
              wn;
}Vvghe;

/* Yvghe contiene le uscite del modulo.
 *  - Tvapout: temperatura del vapore in uscita, coincide con lo stato
 *  - Qvap: calore del vapore
 */

typedef struct Yvghe{
      double
            Tvapout,
            Qvap;
}Yvghe;

/* Dvghe contiene le informazioni riguardo al numero di equazioni, di
 *  variabili algebriche, di stati e di terminali del modulo.
 *  - nstati: numero degli stati
 *  - nalg: numero delle equazioni algebriche
 *  - nti: numero dei terminali, esclusi i corto circuiti
 *  - nout: numero delle uscite
 *  - nrJ: numero di righe dello jacobiano
 *  - ncJ: numero di colonne dello jacobiano
 */
```

```
typedef struct Dvghe{
      int
            nstati,
            nalg,
            nti,
            nout,
            nrJ,
            ncJ;}Dvghe;

/* Mvghe contiene le strutture definite sopra */

typedef struct Mvghe{
      Cvghe c;
      Pvghe p;
      Svghe s;
      Vvghe v;
      Yvghe y;
      Tvghe t;
      Dvghe d;
}Mvghe;

/* Prototipo delle funzioni */

void inizializza_vghe(double **p, Mvghe *M, double *s, double *g,double
**T, double **Jdz, double **Jdx, double **Jaz, double **Jax);
void setEquazioni_vghe(Mvghe *M);
void setJacobiano_vghe(Mvghe *M, int tipo);
void setT_vghe(Mvghe *M);
void getInput_vghe(Mvghe *M, double *in);
void setOutput_vghe(Mvghe *M, double *out);
void terminate_vghe(Mvghe *M);

#endif      // _VGHE_STRUTTURE_H


/** @file vghe_modello.c
 * @brief Scambiatore di calore: Funzioni
 * @author Arianna Comi
 * componente VGHE (scambiatore di calore)
 * Modello
 */

#define _VGHE_MODELLO_C 1

#include "vghe_strutture.h"
#include <math.h>
#include "vapo_n.h"
#include "fumi_n.h"
#include "common.h"
#include <stdlib.h>
#include <stdio.h>

#define CENT 100

/**
 *    ordine stati:
 *    0
 *    Tv
 *
 *    ordine terminali:
 *    0: vapore in
 *    1: vapore out
 *    2: gas in
 *    3: gas out
 *
 *    ordine variabili terminali:
 *    0       1    2    3     4     5          6          7       8
 *    w0:=w1  p0   h0   p1    h1    w2:=w3     p2:=p3    T2      T3
 *    t0                  t1          t2                        t3
 */
```

```
/* La funzione inizializza_vghe riceve i seguenti parametri:
 *  - **p: vettore che contiene i valori, letti da file, da assegnare alle
 *    variabili
 *  - *M: struttura Mvghe, che contiene tutte le strutture che definiscono
 *    il modulo
 *  - *s: vettore che contiene le equazioni differenziali
 *  - *g: vettore che contiene le equazioni algebriche
 *  - **T: matrice delle costanti di tempo
 *  - **Jdz, **Jdx, **Jaz, **Jax: matrici dei 4 blocchi dello jacobiano
 *
 *  Tutti i parametri, a parte il secondo, vengono passati con l'indice
 *  che corrisponde alla posizione occupata dal modulo
 *  nel modello globale che si vuole costruire.
 *
 * La funzione inizializza le variabili locali del modulo e imposta i corto
 * circuiti dei terminali.
 * Inoltre calcola il valore iniziale del coefficiente di scambio.
 */

void inizializza_vghe(double **p,Mvghe * M, double *s, double *g, double
**T, double ** Jdz, double **Jdx, double **Jaz, double **Jax) {

int i;

M->d.nstati = nstati_vghe;
M->d.nalg = nalg_vghe;
M->d.nti = nti_vghe;
M->d.nout = nout_vghe;
M->d.nrJ = nstati_vghe + nalg_vghe;
M->d.ncJ = nstati_vghe + nti_vghe;

// Imposto corto circuiti

M->t.t0.w = M->t.t1.w;
M->t.t2.w = M->t.t3.w;
M->t.t2.p = M->t.t3.p;

/* Assegno i valori letti dal file XML */

*M->t.t0.w = *p[0];
*M->t.t0.p = *p[1];
*M->t.t0.h = *p[2];
*M->t.t1.p = *p[3];
*M->t.t1.h = *p[4];
*M->t.t2.w = *p[5];
*M->t.t2.p = *p[6];
*M->t.t2.T = *p[7];
*M->t.t3.T = *p[8];
*M->s.stati[0] = *p[9];
M->c.massa_m = *p[10];

M->s.Tv = M->s.stati[0];

M->v.wn = *p[5];

M->p.gas_compo[0] = 0.03503;  // CO2
M->p.gas_compo[1] = 0.00894;  // H2
M->p.gas_compo[2] = 0.7461;         // N2
M->p.gas_compo[3] = 0.0;            // CO
M->p.gas_compo[4] = 0.0;            // CH4
M->p.gas_compo[5] = 0.07923;  // H2O
M->p.gas_compo[6] = 0.1307;         // O2

// inizializza parametri

calc_vap_pT(*M->t.t1.p, *M->s.Tv, &M->p.thermo_vap_out);
calc_vap_ph(*M->t.t0.p, *M->t.t0.h, &M->p.thermo_vap_in);

M->p.gasin.h =(1150*Riferimenti.Trif/Riferimenti.Hrif)*(*M->t.t2.T+
-298.15/Riferimenti.Trif);
```

```
M->p.gasout.h =((1150*Riferimenti.Trif)/Riferimenti.Hrif)*(*M->t.t3.T+
-298.15/Riferimenti.Trif);

M->v.p_init = *M->t.t0.p;
M->v.T_init = *M->s.Tv;

M->v.rho_init = M->p.thermo_vap_in.rho;

M->v.rho = (M->v.rho_init*M->v.T_init**M->t.t0.p)/(*M->s.Tv*M->v.p_init);

M->p.sigma0 = (*M->t.t2.w * (M->p.gasin.h - M->p.gasout.h))/(*M->t.t2.T+
-*M->s.Tv);

*M->t.t1.h = M->p.thermo_vap_out.h;

M->p.k = (*M->t.t0.p - *M->t.t1.p)/(*M->t.t0.w);

// inizializza Jacobiano

M->v.Jdz = Jdz;
M->v.Jaz = Jaz;
M->v.Jdx = Jdx;
M->v.Jax = Jax;

M->v.tau = T;

M->v.f[0] = s;
M->v.g[0] = g;
M->v.g[1] = g+1;
M->v.g[2] = g+2;
}

/* La funzione setEquazioni_vghe riceve come parametro la struttura Mvghe,
 *  che definisce l'elemento, e calcola le equazioni del modulo
 */

void setEquazioni_vghe(Mvghe * M) {

M->v.rho = (M->v.rho_init*M->v.T_init**M->t.t0.p)/(*M->s.Tv*M->v.p_init);

M->p.sigma = M->p.sigma0*(0.95*sqrt(*M->t.t2.w/M->v.wn)+0.05);

M->p.gasin.h = ((1150*Riferimenti.Trif)/Riferimenti.Hrif)*(*M->t.t2.T+
-298.15/Riferimenti.Trif);
M->p.gasout.h =((1150*Riferimenti.Trif)/Riferimenti.Hrif)*(*M->t.t3.T+
-298.15/Riferimenti.Trif);

// Controllo temperature

if((*M->t.t2.T - *M->s.Tv)<0)
    printf("ERRORE: Temperatura fumi in ingresso troppo bassa!\n");

// Equazioni differenziali

*M->v.f[0] =
*M->t.t2.w*(M->p.gasin.h-M->p.gasout.h)-*M->t.t0.w*(*M->t.t1.h-*M->t.t0.h);

// Equazioni algebriche

*M->v.g[0] = *M->t.t1.h-M->p.thermo_vap_out.h;
*M->v.g[1] = (*M->t.t0.p-*M->t.t1.p)-M->p.k**M->t.t0.w;
*M->v.g[2] =
M->p.sigma*(*M->t.t2.T-*M->s.Tv)-*M->t.t2.w*(M->p.gasin.h-M->p.gasout.h);

}

/* La funzione setT_vghe riceve come parametro la struttura Mvghe, calcola
*  le costanti di tempo del modulo
*/
void setT_vghe(Mvghe *M){
```

```
SETELEM(M->v.tau, 0, 0,
10*(450*3000*Riferimenti.Trif)/(Riferimenti.Hrif*Riferimenti.Wrif));

}

/* La funzione setJacobiano_vghe riceve come parametri la struttura Mvghe e
 *  un intero.
 *  Calcola lo jacobiano del modulo, diviso in blocchi, il parametro intero
 *  permette di calcolare uno solo dei 4 blocchi.
 *  Se
 *  - tipo = 0: la funzione calcola lo jacobiano completo
 *  - tipo = 1: la funzione calcola lo jacobiano delle equazioni
 *    differenziali rispetto alle variabili algebriche
 *  - tipo = 2:la funzione calcola lo jacobiano delle equazioni
 *    differenziali rispetto agli stati
 *  - tipo = 3: la funzione calcola lo jacobiano delle equazioni
 *    algebriche rispetto alle variabili algebriche
 *  - tipo = 4: la funzione calcola lo jacobiano delle equazioni
 *    algebriche rispetto agli stati
 */

void setJacobiano_vghe(Mvghe *M, int tipo) {

int i, j;

M->p.sigma = M->p.sigma0*(0.95*sqrt(*M->t.t2.w/M->v.wn)+0.05);

M->v.rho = (M->v.rho_init*M->v.T_init**M->t.t0.p)/(*M->s.Tv*M->v.p_init);

/* Calcolo i diversi blocchi dello jacobiano */

if(tipo == 0 || tipo == 1){

if (*M->t.t0.w > 0)

SETELEM(M->v.Jdz,0,0,+*M->t.t0.h-*M->t.t1.h);

else

SETELEM(M->v.Jdz,0,0,0);

SETELEM(M->v.Jdz,0,1,+MAX(*M->t.t0.w,0));

SETELEM(M->v.Jdz, 0, 4,-MAX(*M->t.t0.w,0));

SETELEM(M->v.Jdz, 0, 6,+M->p.gasin.h-M->p.gasout.h);

SETELEM(M->v.Jdz,0,5,+*M->t.t2.w*(1150*Riferimenti.Trif/Riferimenti.Hrif);

SETELEM(M->v.Jdz,0,8,-*M->t.t2.w*(1150*Riferimenti.Trif/Riferimenti.Hrif);

            }

if(tipo == 0 || tipo == 2){

                  }

if(tipo == 0 || tipo == 3){

SETELEM(M->v.Jaz,0,4,+1);

SETELEM(M->v.Jaz,1,2,-M->p.k);

SETELEM(M->v.Jaz,1,0,1);

SETELEM(M->v.Jaz,1,3,-1);

SETELEM(M->v.Jaz,2,6,-(M->p.gasin.h-M->p.gasout.h));
SETELEM(M->v.Jaz,2,5,M->p.sigma+
-*M->t.t2.w*1150*Riferimenti.Trif/Riferimenti.Hrif);
```

```
SETELEM(M->v.Jaz,2,8,*M->t.t2.w*1150*Riferimenti.Trif/Riferimenti.Hrif);

            }

if(tipo == 0 || tipo == 4){

SETELEM(M->v.Jax,0,0,-M->p.thermo_vap_out.dh_dT_p);

SETELEM
(M->v.Jax,1,0,M->p.thermo_vap_out.drho_dT_p*(*M->t.t0.p-*M->t.t1.p));

SETELEM(M->v.Jax,2,0,-M->p.sigma);

}

}

/* La funzione setOutput_vghe riceve come parametri la struttura Mvghe e un
 *  vettore, contenente le uscite del modello complessivo.
 *  Assegna ad ogni uscita del modulo la corrispondente uscita del sistema
 *  complessivo
 */

void setOutput_vghe(Mvghe * M, double * out){
      M->y.Tvapout = *M->s.Tv;
      M->y.Qvap = *M->t.t0.w*(*M->t.t1.h-*M->t.t0.h);
      out[0] = M->y.Tvapout;
      out[1] = M->y.Qvap;
}

/* La funzione getInput_vghe riceve come parametro la struttura Mvghe e un
 *  vettore in, contenente un riferimento al vettore degli ingressi
 *  del modello globale.
 *  Poichè lo scambiatore non ha ingressi, la funzione non ha un corpo e
 *  riceverà come secondo parametro di chiamata il puntatore NULL;
 *  viene comunque definita per rispettare lo schema comune delle funzioni
 *  dei moduli presenti nella libreria
 */

void getInput_vghe(Mvghe * M, double * in){

}

/* La funzione terminate_vghe riceve come parametro la struttura Mvghe;
 *  dealloca tutte le strutture definite dal modulo
 */

void terminate_vghe(Mvghe *M){

      free((Vvghe*)&M->v);
      free((Tvghe*)&M->t);
      free((Dvghe*)&M->d);
      free((Cvghe*)&M->c);
      free((Pvghe*)&M->p);
      free((Svghe*)&M->s);
      free((Yvghe*)&M->y);
      free((Mvghe *)M);
};
```

A3.2 Evaporatore di alta-media pressione

L'evaporatore di alta-media pressione è identificato dalla sigla **evap**.
Si riporta l'header file dell'elemento e il corrispondente codice C.

```
/** @file evap_strutture.h
 * @brief Evaporatore: Strutture e dichiarazioni
 * @author Arianna Comi
 * Componente EVAP (evaporatore)
 * Strutture
 */

#ifndef _EVAP_STRUTTURE_H
#define _EVAP_STRUTTURE_H    1

#include "interfacce.h"
#include "riferimenti.h"
#include "common.h"
#include "sim_utils.h"

#define nstati_evap 2
#define nalg_evap 3
#define nti_evap 9
#define nout_evap 3

#define NCOMP 7

/* Tevap contiene i terminali del modulo, con assegnata la corrispondente
*  intefaccia.
*  In particolare, l'evaporatore ha 4 terminali, due lato vapore e due lato
*  fumi
*/

typedef struct Tevap{
      interfacciaVap t0, t1;
      interfacciaGas t2, t3;
}Tevap;

/* Pevap contiene i parametri dell'evaporatore
 *  - sigma e sigma0: coefficiente di scambio, fase di evoluzione e fase di
 *    inizializzazione
 *  - vapsat: struttura satprop, contenente le variabili termodinamiche del
 *    vapore e del liquido alla saturazione
 *  - gasin e gasout: strutture gasprop, associate ai due terminali lato
 *    fumi, contengono le variabili termodinamiche del gas
 *  - propd: struttura thermoprop, contenente le variabili termodinamiche
 *    del vapore
 *  - gas_compo[NCOMP]: vettore contenente la composizione dei fumi
 */

typedef struct Pevap{

      double
                  sigma,
                  sigma0;

      satprop
                  vapsat;

      gasprop
                  gasin,
                  gasout;

      thermoprop
                  propd;

      double
                  gas_compo[NCOMP];}Pevap;
```

```
/* Cevap contiene le costanti dell'evaporatore
 * - mm_risers: massa metallica risers
 * - vliq_risers: volume liquido saturo risers
 * - v_drum: volume del drum
 * - ro:metallo: densità del metallo
 * - c_metallo: calore specifico del metallo
 * - ro_liq: densità del liquido
 * - drum_lsection: area del pelo libero del liquido
 */

typedef struct Cevap{
      double
             mm_risers,
             vliq_risers,
             v_drum,
             ro_metallo,
             c_metallo,
             ro_liq,
             drum_lsection;
}Cevap;

/* Sevap contiene gli stati dell'evaporatore
 * - Pc: pressione del corpo cilindrico
 * - y:livello
 * - stati[nstati_evap]: vettore che contiene i due stati, definiti sopra;
 *   nstati_evap: numero degli stati
 */

typedef struct Sevap{
      double
              *Pc,
              *y;
      double *stati[nstati_evap];
}Sevap;

/* Vevap contiene le variabili che devono essere visibili all'esterno del
 * modulo e che non appartengono ad altre strutture.
 * - **Jdz,**Jdx, **Jax, **Jaz: blocchi dello jacobiano rispetto alle
 *   variabili
 * - *f[nstati_evap]: vettore delle equazioni differenziali
 * - *g[nalg_evap]: vettore delle equazioni algebriche
 * - **tau: matrice delle costanti di tempo
 * - x: titolo
 * - wn: valore nominale portata fumi
 */

typedef struct Vevap{
      double  **Jdz,**Jdx, **Jax, **Jaz,
              *f[nstati_evap],
              *g[nalg_evap],
              **tau,
              x,
              wn;
}Vevap;

/* Yevap contiene le uscite del modulo.
 * - Pressione: stato, pressione del corpo cilindrico
 * - Livello: stato
 * - Tgasout: temperatura del terminale di uscita, lato fumi, t3
 */

typedef struct Yevap{
      double  Pressione,
              Livello,
              Tgasout;
}Yevap;


/* Devap contiene le informazioni riguardo al numero di equazioni, di
 * variabili algebriche, di stati e di terminali del modulo.
```

```
 * - nstati: numero degli stati
 * - nalg: numero delle equazioni algebriche
 * - nti: numero dei terminali, esclusi i corto circuiti
 * - nout: numero delle uscite
 * - nrJ: numero di righe dello jacobiano
 * - ncJ: numero di colonne dello jacobiano
 */

typedef struct Devap{
      int
            nstati,
            nalg,
            nti,
            nout,
            nrJ,
            ncJ;
}Devap;

/* Mevap contiene le strutture definite sopra */

typedef struct Mevap{
      Cevap c;
      Pevap p;
      Sevap s;
      Vevap v;
      Yevap y;
      Devap d;
      Tevap t;
}Mevap;

/* Prototipo delle funzioni */

void inizializza_evap(double **p,Mevap * M, double *s, double *g,double
**Tx, double **Jdz, double **Jdx, double **Jaz, double **Jax);
void setEquazioni_evap(Mevap * M);
void setJacobiano_evap(Mevap * M, int tipo);
void setJacobianoU_evap(Mevap * M);
void getInput_evap(Mevap * M, double * in);
void setT_evap( Mevap * M);
void setOutput_evap(Mevap * M, double * out);
void terminate_evap(Mevap *M);

#endif      // _EVAP_STRUTTURE_


/** @file evap_modello.c
 * @brief Evaporatore: Funzioni
 * @author Arianna Comi
 * Componente EVAP (evaporatore)
 * Modello
 */

#define _EVAP_MODELLO_C 1

#include "evap_strutture.h"
#include "fumi_n.h"
#include "vapo_n.h"
#include <math.h>
#include "common.h"
#include <stdlib.h>
#include <stdio.h>
#define MAX(a, b) (((a) > (b)) ? (a) : (b))


/**
 *    ordine stati:
 *    0: Pc
 *    1: Lc
 *
 *    ordine terminali:
```

```
 *    0: vapore in
 *    1: vapore out
 *    2: gas in
 *    3: gas out
 *
 *    ordine variabili terminali:
 *    0    1       2    3   4   5       6       7      8
 *    w0   p0:=p1  h0   w1  h1  w2:=w3  p2:=p3  T2     T3
 *    t0               t1      t2                     t3
 *
 */

/* La funzione setEquazioni_evap riceve come parametro la struttura Mevap,
 * che definisce l'elemento, e calcola le equazioni del modulo
 */

void setEquazioni_evap(Mevap * M){

calc_vap_sat_p(*M->s.Pc, &M->p.vapsat);
M->p.sigma = M->p.sigma0*(0.95*sqrt(*M->t.t2.w/M->v.wn)+0.05);

M->v.x=(MAX(0,*M->t.t0.h-M->p.vapsat.hls))/(M->p.vapsat.hgs+
-M->p.vapsat.hls);

M->p.gasin.h =(1150*Riferimenti.Trif/Riferimenti.Hrif)*(*M->t.t2.T+
-(298.15/Riferimenti.Trif));
M->p.gasout.h =(1150*Riferimenti.Trif/Riferimenti.Hrif)*(*M->t.t3.T+
-(298.15/Riferimenti.Trif));

//Controllo su temperature

if((*M->t.t2.T - M->p.vapsat.T)<0)
      printf("ERRORE: temperatura fumi in ingresso troppo bassa!\n");

//Equazioni differenziali

// Equazione pressione,

*M->v.f[0]=-*M->t.t1.w+M->v.x**M->t.t0.w+
+(M->p.sigma*(*M->t.t2.T-M->p.vapsat.T)+
-*M->t.t0.w*(MAX(0,M->p.vapsat.hls-*M->t.t0.h)))/(M->p.vapsat.hgs+
-M->p.vapsat.hls);

// Equazione livello

*M->v.f[1] = *M->t.t0.w - *M->t.t1.w;

//Equazioni algebriche

*M->v.g[0] =*M->t.t1.h-M->p.vapsat.hgs;
*M->v.g[1] =*M->t.t0.p-*M->s.Pc;
*M->v.g[2] =*M->t.t3.w*(M->p.gasin.h-M->p.gasout.h)+
-M->p.sigma*(*M->t.t2.T-M->p.vapsat.T);

};

/* La funzione setJacobiano_evap riceve come parametri la struttura Mevap e
 * un intero.
 * Calcola lo jacobiano del modulo, diviso in blocchi, il parametro intero
 * permette di calcolare uno solo dei 4 blocchi.
 * Se
 * - tipo = 0: la funzione calcola lo jacobiano completo
 * - tipo = 1: la funzione calcola lo jacobiano delle equazioni
 *   differenziali rispetto alle variabili algebriche
 * - tipo = 2: la funzione calcola lo jacobiano delle equazioni
 *   differenziali rispetto agli stati
 * - tipo = 3: la funzione calcola lo jacobiano delle equazioni algebriche
 *   rispetto alle variabili algebriche
 * - tipo = 4: la funzione calcola lo jacobiano delle equazioni algebriche
 *   rispetto agli stati */
```

```
void setJacobiano_evap(Mevap * M, int tipo){

calc_vap_sat_p(*M->s.Pc, &M->p.vapsat);

M->p.sigma = M->p.sigma0*(0.95*sqrt(*M->t.t2.w/M->v.wn)+0.05);

if(tipo == 0 || tipo == 1){

SETELEM
(M->v.Jdz,0,0,-MAX(0,(M->p.vapsat.hls-*M->t.t0.h))/(M->p.vapsat.hgs+
-M->p.vapsat.hls)+M->v.x);

SETELEM(M->v.Jdz,0,1,*M->t.t0.w/(M->p.vapsat.hgs-M->p.vapsat.hls));

SETELEM(M->v.Jdz,0,2,-1);

SETELEM(M->v.Jdz,1,0,1);

SETELEM(M->v.Jdz,1,2,-1);

}

if(tipo == 0 || tipo == 2){

SETELEM(M->v.Jdx,0,0,
((-M->p.sigma*M->p.vapsat.dT_dp-*M->t.t0.w*M->p.vapsat.dhls_dp)*
*(M->p.vapsat.hgs-M->p.vapsat.hls)-(M->p.sigma*(*M->t.t2.T+
-M->p.vapsat.T)*M->t.t0.w*(M->p.vapsat.hls-*M->t.t0.h))*
*(M->p.vapsat.dhgs_dp-M->p.vapsat.dhls_dp))/
(pow((M->p.vapsat.hgs-M->p.vapsat.hls),2)));

            }

if(tipo == 0 || tipo == 3){


SETELEM(M->v.Jaz,0,4,1);

SETELEM(M->v.Jaz,1,3,1);

SETELEM(M->v.Jaz,2,6,M->p.gasin.h - M->p.gasout.h);

SETELEM(M->v.Jaz,2,5,*M->t.t2.w*1150*Riferimenti.Trif/Riferimenti.Hrif+
-M->p.sigma);

SETELEM(M->v.Jaz,2,8,-*M->t.t2.w*1150*Riferimenti.Trif/Riferimenti.Hrif);

                  }


if(tipo == 0 || tipo == 4 ){

SETELEM(M->v.Jax,0,0,-M->p.vapsat.dhgs_dp);

SETELEM(M->v.Jax,1,0,-1);

SETELEM(M->v.Jax,2,0,M->p.sigma*M->p.vapsat.dT_dp);


            }

};

/* La funzione setOutput_evap riceve come parametri la struttura Mevap e un
 * vettore, contenente le uscite del modello complessivo.
 * Assegna ad ogni uscita del modulo la corrispondente uscita del sistema
* complessivo
*/
```

```
void setOutput_evap(Mevap * M, double * out){

calc_vap_ph(*M->t.t0.p, *M->t.t0.h, &M->p.propd);

M->y.Pressione = *M->s.Pc;
M->y.Livello = *M->s.y;
M->y.Tgasout = *M->t.t3.T;

out[0] = M->y.Pressione;
out[1] = M->y.Livello;
out[2] = M->y.Tgasout;

};

/* La funzione setT_evap riceve come parametro la struttura Mevap, calcola
 * le costanti di tempo del modulo
 */

void setT_evap(Mevap *M){

calc_vap_sat_p(*M->s.Pc, &M->p.vapsat);

SETELEM(M->v.tau, 0, 0,
10*((Riferimenti.Trif*450*10000*M->p.vapsat.dT_dp)/
(Riferimenti.Wrif*Riferimenti.Hrif*(M->p.vapsat.hgs-M->p.vapsat.hls))+
+(2000*M->p.vapsat.dhls_dp)/
((M->p.vapsat.hgs-M->p.vapsat.hls)*Riferimenti.Wrif)+
+(M->p.vapsat.rhogs*20*M->p.vapsat.drhogs_dp)/Riferimenti.Wrif));

SETELEM(M->v.tau, 1, 1, 100*(Riferimenti.Mrif/Riferimenti.Wrif));

}

/* La funzione inizializza_evap riceve i seguenti parametri:
 * - **p: vettore che contiene i valori, letti da file, da assegnare alle
 *   variabili
 * - *M: struttura Mevap, che contiene tutte le strutture che definiscono
 *   il modulo
 * - *s: vettore che contiene le equazioni differenziali
 * - *g: vettore che contiene le equazioni algebriche
 * - **Tx: matrice delle costanti di tempo
 * - **Jdz, **Jdx, **Jaz, **Jax: matrici dei 4 blocchi dello jacobiano

 * Tutti i parametri, a parte il secondo, vengono passati con l'indice che
 * corrisponde alla posizione occupata dal modulo nel modello globale che
 * si vuole costruire.
 *
 * La funzione inizializza le variabili locali del modulo e imposta i corto
 * circuiti dei terminali.
 * Inoltre calcola il valore iniziale del coefficiente di scambio.
 */

void inizializza_evap(double **p, Mevap *M, double *s, double *g, double
**Tx, double **Jdz, double **Jdx, double **Jaz, double **Jax)
{
M->d.nstati = nstati_evap;
M->d.nalg = nalg_evap;
M->d.nti = nti_evap;
M->d.nout = nout_evap;
M->d.nrJ = nstati_evap + nalg_evap;
M->d.ncJ = nstati_evap + nti_evap;

// Imposto corto circuiti

M->t.t0.p = M->t.t1.p;
M->t.t2.w = M->t.t3.w;
M->t.t2.p = M->t.t3.p;

/* Assegno i valori letti dal file XML */
```

```
*M->t.t0.p = *p[0];
*M->t.t0.w = *p[1];
*M->t.t0.h = *p[2];
*M->t.t1.w = *p[3];
*M->t.t1.h = *p[4];
*M->t.t2.w = *p[5];
*M->t.t2.p = *p[6];
*M->t.t2.T = *p[7];
*M->t.t3.T = *p[8];
*M->s.stati[0] = *p[9];
*M->s.stati[1] = *p[10];
M->c.mm_risers = *p[11];
M->c.vliq_risers = *p[12];
M->c.v_drum = *p[13];
M->c.ro_metallo = *p[14];
M->c.ro_liq = *p[15];
M->c.c_metallo = *p[16];
M->c.drum_lsection = *p[17];

/* assegno agli stati dell'elemento la posizione corretta nel vettore degli
stati globali */

M->s.Pc = M->s.stati[0];
M->s.y = M->s.stati[1];

M->v.wn = *p[5];

calc_vap_sat_p(*M->s.Pc, &M->p.vapsat);

M->p.gasin.h =(1150*Riferimenti.Trif/Riferimenti.Hrif)*(*M->t.t2.T+
-298.15/Riferimenti.Trif);

M->p.gasout.h =(1150*Riferimenti.Trif/Riferimenti.Hrif)*(*M->t.t3.T+
-298.15/Riferimenti.Trif);

/*Calcolo coefficiente di scambio Gas-H2O*/

M->p.sigma0 =(*M->t.t3.w)*(M->p.gasin.h-M->p.gasout.h)/(*M->t.t2.T+
-M->p.vapsat.T);

M->v.Jax = Jax;
M->v.Jdx = Jdx;
M->v.Jaz = Jaz;
M->v.Jdz = Jdz;

M->v.tau  = Tx;

M->v.f[0] = s;
M->v.f[1] = s+1;

M->v.g[0] = g;
M->v.g[1] = g+1;
M->v.g[2] = g+2;

};

/* La funzione getInput_evap riceve come parametro la struttura Mevap e un
 * vettore in, contenente un riferimento al vettore degli ingressi
 * del modello globale.
 * Poichè l'evaporatore non ha ingressi, la funzione non ha un corpo e
 * riceverà come secondo parametro di chiamata il puntatore NULL;
 * viene comunque definita per rispettare lo schema comune delle funzioni
 * dei moduli presenti nella libreria
 */
void getInput_evap(Mevap * M, double * in) {

};

/* La funzione terminate_evap riceve come parametro la struttura Mevap;
 * dealloca tutte le strutture definite dal modulo
```

```
 */

void terminate_evap(Mevap *M){

      free((Vevap*)&M->v);
      free((Tevap*)&M->t);
      free((Devap*)&M->d);
      free((Cevap*)&M->c);
      free((Pevap*)&M->p);
      free((Sevap*)&M->s);
      free((Yevap*)&M->y);
      free((Mevap *) M);

};
```

A3.3 Degasatore (Evaporatore di bassa pressione)

Il degasatore è identificato dalla sigla **evbp**.
Si riporta l'header file dell'elemento e il corrispondente codice C.

```
/* @file evbp_strutture.h
 * @brief evbporatore bassa pressione: Strutture e dichiarazioni
 * @author Comi Arianna
 *
 * Componente EVBP (evbporatore bassa pressione)
 * Strutture
 *
 */

#ifndef _EVBP_STRUTTURE_H
#define _EVBP_STRUTTURE_H    1

#include "interfacce.h"
#include "common.h"
#include "Riferimenti.h"
#include "sim_utils.h"

#define nstati_evbp 2
#define nalg_evbp 4
#define nti_evbp 12
#define nin_evbp 0
#define nout_evbp 0

#define NCOMP7

/* Tevbp contiene contiene i terminali del modulo, con assegnata la
 * corrispondente intefaccia.
 * In particolare, l'evaporatore bassa pressione ha 6 terminali, quattro
 * lato vapore e due lato fumi
*/

typedef struct Tevbp{
      interfacciaVap t0, t1, t4, t5;
      interfacciaGas t2, t3;
}Tevbp;

/* Pevbp contiene i parametri dell'evaporatore bassa pressione
 * - sigma e sigma0: coefficiente di scambio, fase di evoluzione e fase di
 *   inizializzazione
 * - vapsat: struttura satprop, contenente le variabili termodinamiche del
 *   vapore e del liquido alla saturazione
 * - gasin e gasout: strutture gasprop, associate ai due terminali lato
 *   fumi, contengono le variabili termodinamiche del gas
 * - propd: struttura thermoprop, contenente le variabili termodinamiche
 *   del vapore
 * - gas_compo[NCOMP]: vettore contenete la composizione dei fumi
 */
```

```
typedef struct Pevbp{

      double
                  sigma,
                  sigma0;

      satprop
                  vapsat;
      gasprop
                  gasin,
                  gasout;
      thermoprop
                  propd;

      double
                  gas_compo[NCOMP];
}Pevbp;

/* Cevbp contiene le costanti dell'evaporatore bassa pressione
 * - mm_risers: massa metallica risers
 * - vliq_risers: volume liquido saturo risers
 * - v_drum: volume del drum
 * - ro_metallo: densità del metallo
 * - c_metallo: calore specifico del metallo
 * - ro_liq: densità del liquido
 * - drum_lsection: area del pelo libero del liquido
 */

typedef struct Cevbp{
      double
             mm_risers,
             vliq_risers,
             v_drum,
             ro_metallo,
             c_metallo,
             ro_liq,
             drum_lsection;
}Cevbp;

/* Sevbp contiene gli stati dell'evaporatore bassa pressione
 * - Pc: pressione del corpo cilindrico
 * - y: livello
 * - stati[nstati_evap]: vettore che contiene i due stati, definiti sopra;
 *   nstati_evap: numero degli stati */

typedef struct Sevbp{
      double
             *P,
             *y;
      double *stati[nstati_evbp];
}Sevbp;

/* Vevbp contiene le variabili che devono essere visibili all'esterno del
 * modulo e che non appartengono ad altre strutture.
 * - **Jdz,**Jdx, **Jax, **Jaz: blocchi dello jacobiano rispetto alle
 *   variabili
 * - *f[nstati_evbp]: vettore delle equazioni differenziali
 * - *g[nalg_evbp]: vettore delle equazioni algebriche
 * - **tau: matrice delle costanti di tempo
 * - Dh: differenza di entalpia tra terminale 4 e terminale 0
 * - wn: valore nominale portata fumi
 */
typedef struct Vevbp{
      double  **Jdz,**Jdx, **Jax, **Jaz,
              *f[nstati_evbp],
              *g[nalg_evbp],
              **tau,
              Dh,
              wn;
}Vevbp;
```

```
/* Yevbp contiene le uscite del modulo.
 * - Pressione: stato, pressione del corpo cilindrico
 * - Livello: stato
 * - Tgasout: temperatura del terminale di uscita, lato fumi, t3
 */

typedef struct Yevbp{
      double  Pressione,
              Livello,
              gasout;
}Yevbp;

/* Devbp contiene le informazioni riguardo al numero di equazioni, di
 * variabili algebriche, di stati e di terminali del modulo.
 * - nstati: numero degli stati
 * - nalg: numero delle equazioni algebriche
 * - nti: numero dei terminali, esclusi i corto circuiti
 * - nout: numero delle uscite
 * - nrJ: numero di righe dello jacobiano
 * - ncJ: numero di colonne dello jacobiano
 */

typedef struct Devbp{
      int
            nstati,
            nalg,
            nti,
            nout,
            nrJ,
            ncJ;
}Devbp;

/* Mevbp contiene le strutture definite sopra */

typedef struct Mevbp{
      Cevbp c;
      Pevbp p;
      Sevbp s;
      Vevbp v;
      Yevbp y;
      Devbp d;
      Tevbp t;
}Mevbp;

/* Prototipo delle funzioni */

void inizializza_evbp(double **p,Mevbp * M, double *s, double *g,double
**Tx, double **Jdz, double **Jdx, double **Jaz, double **Jax);
void setEquazioni_evbp(Mevbp * M);
void setJacobiano_evbp(Mevbp * M, int tipo);
void getInput_evbp(Mevbp * M, double * in);
void setT_evbp( Mevbp * M);
void setOutput_evbp(Mevbp * M, double * out);
void terminate_evbp(Mevbp *M);

#endif
```

```
/** @file evbp_modello.c
 * @brief Evaporatore bassa pressione: Funzioni
 * @author Arianna Comi
 * Componente EVBP (evaporatore bassa pressione)
 * Modello
 */

#define _EVBP_MODELLO_C 1

#include "evbp_strutture.h"
#include "fumi_n.h"
#include "vapo_n.h"
#include <math.h>
#include "common.h"
#include <stdlib.h>
#include <stdio.h>
#define MAX(a, b) (((a) > (b)) ? (a) : (b))


/**
 *    ordine stati:
 *    0: Pc
 *    1: y
 *
 *    ordine terminali:
 *    0: vapore in
 *    1: vapore out
 *    2: gas in
 *    3: gas out
 *    4: t4
 *    5: t5
 *
 *    ordine variabili terminali:
 *    0   1                2  3  4  5       6       7  8    9   10     11
 *    w0  p0:=p1:=p4:=p5   h0 w1 h1 w2:=w3  p2:=p3  T2 T3   w4  h4:=h5 w5
*     t0                      t1    t2                t3   t4         t5
 *
*/


/* La funzione setEquazioni_evbp riceve come parametro la struttura Mevbp,
 * che definisce l'elemento, e calcola le equazioni del modulo
 */

void setEquazioni_evbp(Mevbp * M){

calc_vap_sat_p(*M->s.P, &M->p.vapsat);

M->p.gasin.h =(1150*Riferimenti.Trif/Riferimenti.Hrif)*(*M->t.t2.T+
-298.15/Riferimenti.Trif);
M->p.gasout.h =(1150*Riferimenti.Trif/Riferimenti.Hrif)*(*M->t.t3.T+
-298.15/Riferimenti.Trif);

M->p.sigma = M->p.sigma0*(0.95*sqrt(*M->t.t2.w/M->v.wn)+0.05);

// Controllo temperature

if((*M->t.t2.T - M->p.vapsat.T)<0)
      printf("ERRORE: temperatura fumi in ingresso troppo bassa!\n");

//Equazioni algebriche

*M->v.g[0]=*M->t.t1.h-M->p.vapsat.hgs;
*M->v.g[1]=*M->t.t0.p-*M->s.P;
*M->v.g[2]=*M->t.t3.w*(M->p.gasin.h-M->p.gasout.h)+
-M->p.sigma*(*M->t.t2.T-M->p.vapsat.T);
*M->v.g[3]=*M->t.t4.h-*M->t.t0.h-M->v.Dh;

//Equazioni differenziali
```

```
// Equazione pressione

*M->v.f[0]=M->p.sigma*(*M->t.t2.T-M->p.vapsat.T)+
-*M->t.t1.w*(M->p.vapsat.hgs-*M->t.t0.h)+
-(*M->t.t4.w+*M->t.t5.w)*(*M->t.t4.h-*M->t.t0.h);

// Equazione livello

*M->v.f[1]=*M->t.t0.w-*M->t.t1.w-*M->t.t4.w-*M->t.t5.w;

};

/* La funzione setJacobiano_evbp riceve come parametri la struttura Mevbp e
 * un intero.
 * Calcola lo jacobiano del modulo, diviso in blocchi, il parametro intero
 * permette di calcolare uno solo dei 4 blocchi.
 * Se
 * - tipo = 0: la funzione calcola lo jacobiano completo
 * - tipo = 1: la funzione calcola lo jacobiano delle equazioni
 *   differenziali rispetto alle variabili algebriche
 * - tipo = 2: la funzione calcola lo jacobiano delle equazioni
 *   differenziali rispetto agli stati
 * - tipo = 3: la funzione calcola lo jacobiano delle equazioni algebriche
 *   rispetto alle variabili algebriche
 * - tipo = 4: la funzione calcola lo jacobiano delle equazioni algebriche
 *   rispetto agli stati
 */

void setJacobiano_evbp(Mevbp * M, int tipo){

calc_vap_sat_p(*M->s.P, &M->p.vapsat);

M->p.sigma = M->p.sigma0*(0.95*sqrt(*M->t.t2.w/M->v.wn)+0.05);

if(tipo == 0 || tipo == 1){

SETELEM(M->v.Jdz,0,2,(*M->t.t4.w-*M->t.t5.w+*M->t.t1.w));

SETELEM(M->v.Jdz,0,3,(-(M->p.vapsat.hgs-*M->t.t0.h)));

SETELEM(M->v.Jdz,0,5,M->p.sigma);

SETELEM(M->v.Jdz,0,9,-(*M->t.t4.h-*M->t.t0.h));

SETELEM(M->v.Jdz,0,11,-(*M->t.t4.w+*M->t.t5.w));

SETELEM(M->v.Jdz,0,10,-(*M->t.t4.h-*M->t.t0.h));

SETELEM(M->v.Jdz,1,0,1);

SETELEM(M->v.Jdz,1,3,-1);

SETELEM(M->v.Jdz,1,9,-1);

SETELEM(M->v.Jdz,1,10,-1);

}

if(tipo == 0 || tipo == 2){

SETELEM(M->v.Jdx,0,0,
-M->p.sigma*M->p.vapsat.dT_dp-*M->t.t1.w*M->p.vapsat.dhgs_dp);

            }

if(tipo == 0 || tipo == 3){

SETELEM(M->v.Jaz,0,4,1);

SETELEM(M->v.Jaz,1,1,1);
```

```
SETELEM(M->v.Jaz,2,6,(M->p.gasin.h-M->p.gasout.h));

SETELEM(M->v.Jaz,2,5,
-M->p.sigma+*M->t.t3.w*1150*Riferimenti.Trif/Riferimenti.Hrif);

SETELEM(M->v.Jaz,2,8,
-*M->t.t3.w*(1150*Riferimenti.Trif)/Riferimenti.Hrif);

SETELEM(M->v.Jaz,3,11,1);

SETELEM(M->v.Jaz,3,2,-1);

      }


if(tipo == 0 || tipo == 4 ){

SETELEM(M->v.Jax,0,0,-M->p.vapsat.dhgs_dp);

SETELEM(M->v.Jax,1,0,-1);

SETELEM(M->v.Jax,2,0,M->p.sigma*M->p.vapsat.dT_dp);

                  }

};

/* La funzione setOutput_evbp riceve come parametri la struttura Mevbp e un
 * vettore, contenente le uscite del modello complessivo.
 * Assegna ad ogni uscita del modulo la corrispondente uscita del sistema
 * complessivo
 */

void setOutput_evbp(Mevbp * M, double * out)
{

calc_vap_ph(*M->t.t0.p, *M->t.t0.h, &M->p.propd);

M->y.Pressione = *M->s.P;
M->y.Livello = *M->s.y;
M->y.Tgasout = *M->t.t3.T;

out[0] = M->y.Pressione;
out[1] = M->y.Livello;
out[2] = M->y.Tgasout;

};

/* La funzione setT_evbp riceve come parametro la struttura Mevbp; calcola
 * le costanti di tempo del modulo
 */

void setT_evbp(Mevbp *M){

calc_vap_sat_p(*M->s.P, &M->p.vapsat);

SETELEM(M->v.tau, 0, 0,
10*((Riferimenti.Trif*450*10000*M->p.vapsat.dT_dp)/
(Riferimenti.Wrif*Riferimenti.Hrif*(M->p.vapsat.hgs-M->p.vapsat.hls))+
+(2000*M->p.vapsat.dhls_dp)/
((M->p.vapsat.hgs-M->p.vapsat.hls)*Riferimenti.Wrif)+
+(M->p.vapsat.rhogs*20*M->p.vapsat.drhogs_dp)/Riferimenti.Wrif));

SETELEM(M->v.tau, 1, 1, 100*(Riferimenti.Mrif/Riferimenti.Wrif));

};
```

```
/* La funzione inizializza_evbp riceve i seguenti parametri:
 * - **p: vettore che contiene i valori, letti da file, da assegnare alle
 *   variabili
 * - *M: struttura Mevbp, che contiene tutte le strutture che definiscono
 *   il modulo
 * - *s: vettore che contiene le equazioni differenziali
 * - *g: vettore che contiene le equazioni algebriche
 * - **Tx: matrice delle costanti di tempo
 * - **Jdz, **Jdx, **Jaz, **Jax: matrici dei 4 blocchi dello jacobiano
 *
 * Tutti i parametri, a parte il secondo, vengono passati con l'indice che
 * corrisponde alla posizione occupata dal modulo nel modello globale che
 * si vuole costruire.
 *
 * La funzione inizializza le variabili locali del modulo e imposta i corto
 * circuiti dei terminali.
 * Inoltre calcola il valore iniziale del coefficiente di scambio.
 */

void inizializza_evbp(double **p,Mevbp *M, double *s, double *g, double
**Tx, double **Jdz, double **Jdx, double **Jaz, double **Jax)
{

M->d.nstati = nstati_evbp;
M->d.nalg = nalg_evbp;
M->d.nti = nti_evbp;
M->d.nout = nout_evbp;
M->d.nrJ = nstati_evbp + nalg_evbp;
M->d.ncJ = nstati_evbp + nti_evbp;

// Imposto corto circuiti

M->t.t1.p = M->t.t0.p;
M->t.t4.p = M->t.t0.p;
M->t.t5.p = M->t.t0.p;
M->t.t2.w = M->t.t3.w;
M->t.t2.p = M->t.t3.p;

M->t.t4.h = M->t.t5.h;

/* Assegno i valori letti dal file XML */

*M->t.t0.p = *p[0];
*M->t.t0.w = *p[1];
*M->t.t0.h = *p[2];
*M->t.t1.w = *p[3];
*M->t.t1.h = *p[4];
*M->t.t2.w = *p[5];
*M->t.t2.p = *p[6];
*M->t.t2.T = *p[7];
*M->t.t3.T = *p[8];
*M->t.t4.w = *p[9];
*M->t.t4.h = *p[10];
*M->t.t5.w = *p[11];
*M->s.stati[0] = *p[12];
*M->s.stati[1] = *p[13];
M->c.mm_risers = *p[14];
M->c.vliq_risers = *p[15];
M->c.v_drum = *p[16];
M->c.ro_metallo = *p[17];
M->c.ro_liq = *p[18];
M->c.c_metallo = *p[19];
M->c.drum_lsection = *p[20];

/* assegno agli stati dell'elemento la posizione corretta nel vettore degli
 * stati globali
 */

M->s.P = M->s.stati[0];
M->s.y = M->s.stati[1];
```

```
M->v.wn = *p[5];

calc_vap_sat_p(*M->s.P, &M->p.vapsat);

*M->t.t1.h = M->p.vapsat.hgs;

M->v.Dh = *M->t.t4.h - *M->t.t0.h;

M->p.gasin.h =(1150*Riferimenti.Trif/Riferimenti.Hrif)*(*M->t.t2.T+
-298.15/Riferimenti.Trif);

M->p.gasout.h =(1150*Riferimenti.Trif/Riferimenti.Hrif)*(*M->t.t3.T+
-298.15/Riferimenti.Trif);

/*Calcolo coefficiente di scambio Gas-H2O*/

M->p.sigma0 =*M->t.t3.w*(M->p.gasin.h-M->p.gasout.h)/(*M->t.t2.T+
-M->p.vapsat.T);

M->v.Jax = Jax;
M->v.Jdx = Jdx;
M->v.Jaz = Jaz;
M->v.Jdz = Jdz;
M->v.tau = Tx;

M->v.f[0] = s;
M->v.f[1] = s+1;

M->v.g[0] = g;
M->v.g[1] = g+1;
M->v.g[2] = g+2;
M->v.g[3] = g+3;

};

/* La funzione getInput_evbp riceve come parametro la struttura Mevbp e un
 * vettore in, contenente un riferimento al vettore degli ingressi
 * del modello globale.
 * Poichè l'evaporatore non ha ingressi, la funzione non ha un corpo e
 * riceverà come secondo parametro di chiamata il puntatore NULL;
 * viene comunque definita per rispettare lo schema comune delle funzioni
 * dei moduli presenti nella libreria
 */

void getInput_evbp(Mevbp* M, double * in) {
};

/* La funzione terminate_evbp riceve come parametro la struttura Mevbp,
 * dealloca tutte le strutture definite dal modulo
 */

void terminate_evbp(Mevbp *M){

      free((Vevbp*)&M->v);
      free((Tevbp*)&M->t);
      free((Devbp*)&M->d);
      free((Cevbp*)&M->c);
      free((Pevbp*)&M->p);
      free((Sevbp*)&M->s);
      free((Yevbp*)&M->y);
      free((Mevbp *) M);

};
```