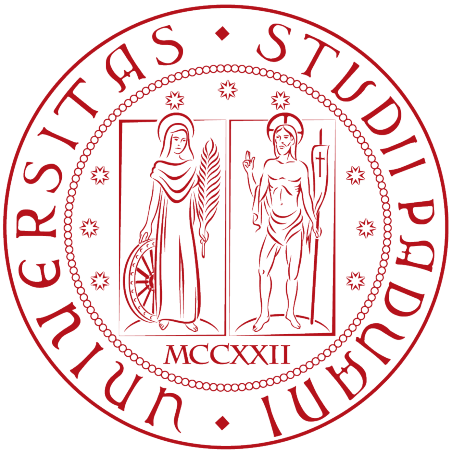
Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica



Fluidprops: una libreria di calcolo per la termodinamica dei fluidi

Tesi di laurea

Relatore

Prof. Luigi De Giovanni

Laureando

Marco Odinotte 1170564

Anno Accademico 2022-2023

ALLA MIA FAMIGLIA
E A QUELLA CHE MI SONO SCELTO

Sommario

Il presente elaborato rappresenta il resoconto dell'attività di stage svolta dallo studente Odinotte Marco presso l'azienda Novaeka s.r.l. per un totale di trecentotrenta ore complessive.
Lo scopo primario del lavoro era la comprensione delle librerie di calcolo adottate attualmente per rilevare le proprietà termodinamiche legate ai fluidi refrigeranti, evidenziarne le criticità e possibilmente scovare alternative più convenienti dai punti di vista di efficacia ed efficienza.
Il precipitato di questa indagine è stata Fluidprops: una libreria software che, sfruttando le equazioni di stato dei fluidi espresse nella forma dell'energia di Helmholtz, ha reso possibile il calcolo di numerose proprietà fisiche a partire dai soli dati di temperatura e densità.
L'utilizzo del linguaggio di programmazione Rust è stato congeniale non solo per l'integrazione del software con il progetto di più ampio respiro dell'azienda, ma ha facilitato la formazione di un sistema di tipi coeso, limitando le possibili ambiguità all'interno dei processi di calcolo sfruttati dal software.
I risultati evidenziano un aumento delle prestazioni rispetto ai concorrenti sul mercato, nonché maggiore precisione nell'accuratezza dei calcoli in alcune zone critiche di rilevazione.
Il progetto non si considera di per sé conclusivo, bensì un coraggioso apripista nel settore, strutturato per agevolare integrazioni e miglioramenti nel lungo periodo.

Convenzioni tipografiche:

Riguardo la stesura del testo, relativamente al documento sono state adottate le seguenti convenzioni tipografiche:

- gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati vengono definiti nel glossario, situato alla fine del presente documento;
- per la prima occorrenza dei termini riportati nel glossario sono evidenziati con il colore blu e contengono un link diretto alla sezione apposita;
- i termini in lingua straniera o facenti parti del gergo tecnico sono evidenziati con il carattere *corsivo*.

Ringraziamenti

In primis, vorrei esprimere la mia gratitudine al Prof. Luigi De Giovanni, relatore della mia tesi, per l'aiuto e il sostegno fornitomi durante la stesura del lavoro.

Un doveroso ringraziamento anche a Dennis ed Eduard, tutor dello stage, per la preziosa esperienza vissuta all'interno del team di Novaeka.

Ringrazio la mia famiglia per avermi permesso di volare e i miei amici per non avermi fatto cadere.

Padova, Settembre 2023 Marco Odinotte

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione e contesto dello stage

1.1 L'idea

Nell'orizzonte della produzione e dell'impiantistica industriale, vi è largo utilizzo di circuiti refrigeranti che permettono il corretto svolgimento delle attività produttive o di ricerca. Progettare un impianto di tale natura ha come passaggio obbligato lo studio delle varie tipologie di fluidi refrigeranti che vi scorrono all'interno, che variano nelle caratteristiche e nel comportamento, in seguito a stimoli fisici e condizioni precise. Per poter avere un'idea chiara di queste variabili, è necessario dotarsi di supporti software che permettano il calcolo agile e il più possibile accurato delle varie proprietà termodinamiche del fluido impiegato. Non si tratta di semplice diletto di calcolo, ma ottenere misurazioni o previsioni precise permette di modellare al miglior modo l'apparato meccanico ed ingegneristico entro cui il fluido verrà utilizzato.
Nel pieno dell'era dell'informatizzazione, risulta naturale affidare questo tipo di onere a librerie software specifiche, non solo per la complessità e la lungaggine dei calcoli necessari, ma anche per l'utilizzo ausiliario di metodi di calcolo numerico a supporto della computazione principale. Il progetto Fluidprops nasce dalla necessità di studiare una possibile alternativa alle librerie di calcolo presenti oggigiorno sul mercato e utilizzate dall'azienda che ospita lo stage. Questa necessità è frutto di numerose lacune riscontrate durante il loro utilizzo, non solo dal punto di vista del calcolo, bensì anche dalla prospettiva dell'organizzazione del codice. Avere del codice inefficiente non solo ritarda il calcolo, ma predispone ad un peggior inserimento della libreria in modelli o strutture software che ne fanno uso.
A tal proposito, il progetto si è posto l'audace ma necessario obiettivo di ispezionare la giungla di opportunità esistenti, addentrarsi sufficientemente nella meccanica di calcolo sottostante, e produrre come risultato una nuova via esplorativa nel panorama del calcolo delle proprietà termodinamiche dei fluidi.

1.2 Il profilo aziendale

Nata nel 2017, l'azienda Novaeka, il cui logo appare in Figura 1.1, si sta rivelando una realtà innovativa nell'ambito dei settori aerospaziale e industriale relativo ad impianti di raffreddamento. Al suo interno riunisce un team estremamente variegato fra matematici,

informatici ed ingegneri, non solo per garantire collaborazione e condivisione delle conoscenze all'interno del team, ma anche per fornire soluzioni olistiche e poliedriche ai proprio clienti. Il suo apporto principale nel mercato riguarda la formulazione di "innovativi modelli matematici per gestire e raffinare l'utilizzo dell'energia migliorando le performance" [38]. Questa metodologia nuova di sperimentazione e, conseguentemente, di lavoro, permette al cliente finale di utilizzare questi modelli sia in ottica progettuale, sia come mezzo per migliorare l'impiantistica attualmente in produzione e limarne le eventuali criticità.

Figura 1.1: Logo rappresentativo dell'azienda Novaeka

1.3 L'applicativo Novaeka

Questo progetto di stage si inquadra all'interno di un progetto che l'azienda Novaeka sta sviluppando da diverso tempo, a supporto non solo di attività industriali, ma anche come risorsa fondamentale per il lavoro di punta dell'azienda.
L'obiettivo che Novaeka si è posta è la creazione di un prodotto software che permettesse la progettazione di impianti di refrigerazione e che fornisse agli sviluppatori, oltre alla controparte grafica di disegno, anche un apparato di calcoli e note accessorie per prendere decisioni progettuali istantanee, simulando immediatamente il comportamento di un determinato fluido all'interno dell'impianto in fase di progetto.
Per compiere questa attività, l'applicativo dovrà disporre di tutti gli strumenti necessari per effettuare questi calcoli, con la maggiore velocità ed efficienza possibili. In questo contesto si inquadra il presente lavoro, come una possibile opzione alle librerie attualmente in uso dall'azienda e come un ulteriore passo del progetto madre verso l'obiettivo prestabilito.

1.4 La metodologia di lavoro

Novaeka è un'azienda di piccole/medie dimensioni, che permette una comunicazione diretta tra i membri del team e un ambiente di lavoro funzionale. Il comparto informatico dell'azienda conta soli due membri, quindi ha permesso un contatto diretto e reattivo con il tutor aziendale sia in caso di necessità, che per la pianificazione dei task. La giovane età dell'azienda fisiologicamente favorisce un ambiente interno in rapida crescita, che necessita di dinamicità e disposizione al cambiamento adattivo; da intendersi non come instabilità, ma come risorsa di flessibilità da sfruttare nell'organizzazione del lavoro e nella ripartizione dei compiti. Il progetto sviluppato in questo

contesto, avendo durata predeterminata, dimensioni contenute, e valenza a sé stante rispetto al macro-progetto in fase di sviluppo, non ha potuto che sfruttare le proprietà sopracitate.
Di comune accordo con l'azienda, si è deciso di adottare come metodologia di sviluppo il modello di lavoro Agile [27], utilizzando il framework Scrum [37] all'interno del micro-team di lavoro nel seguente modo:

1. Scrum master: ruolo ricoperto dal tutor aziendale con cui si discutevano i task da svolgere rimasti nel backlog, secondo le priorità, e le milestones desiderabili;
2. Sprint: periodo della durata di una settimana normalmente, ad eccezione di rari casi in cui è stato esteso a due settimane se il lavoro lo richiedeva;
3. Daily Scrum: incontri quotidiani effettuati al mattino e, se necessario, nel primo pomeriggio per monitorare l'avanzamento del lavoro e le eventuali criticità riscontrate;
4. Sprint Review: a cadenza settimanale per reimpostare il backlog dei task e migliorare l'organizzazione del lavoro se necessario.

1.5 Piano di lavoro: le premesse e gli obiettivi

Il progetto ha previsto un impiego di trecentotrenta ore complessive dal 16 Gennaio al 14 Marzo 2023, suddivise in nove settimane di lavoro, cinque giorni (40 ore) a settimana. Accanto alla realizzazione del vero e proprio prodotto software, uno degli scopi principali dello stage era la familiarizzazione con il linguaggio di programmazione Rust [26] e la sua toolchain, senza tralasciare le basi teoriche che sarebbero state necessarie per la comprensione del contesto di lavoro e la scrittura del codice definitivo. A tal proposito, gran parte del tempo effettivo è stato necessario per affinare e comprendere il substrato teorico delle equazioni di stato dei fluidi e l'analisi approfondita dei software di riferimento per il loro calcolo, soprattutto nelle prime settimane, per lasciare spazio, in ultimo, ad una pesante fase di testing sistematico sul prodotto completo. Gli obiettivi dello stage, come delineati in fase di pianificazione, sono stati sintetizzati nella Tabella 1.1.
La pianificazione settimanale preventiva delle attività visibile in Tabella 1.2, annunciata nel piano di lavoro e, successivamente, rispettata durante l'attività di stage è risultata sufficientemente attendibile e ha permesso il completamento di tutti gli obiettivi obbligatori e anche di quelli desiderabili.

Tipologia	Codice	Descrizione obiettivo
Obbligatori	O01	Analisi del perimetro delle funzionalità richieste dall'applicativo Novaeka
	O02	Definizione di una Application Program Interface più robusta e semplificata rispetto alle rispettive parti già esistenti (Refrprop e Coolprop)
	O03	Implementazione in Rust della libreria
Desiderabili	D01	Ottimizzazione delle soluzioni implementate per ottenere il massimo dal contesto di applicazione della libreria
Facoltativi	F01	Report sulle performance delle librerie esistenti e quella sviluppata
	F02	Integrazione della libreria nell'applicativo software Novaeka

Tabella 1.1: Tabella degli obiettivi suddivisi per tipologie

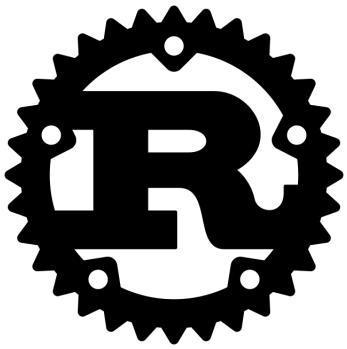
Descrizione dell'attività	Ore Preventivate
Formazione preliminare	40 ore
Studio della documentazione e delle soluzioni software esistenti	40 ore
Definizione dell'API contenuta nella libreria	70 ore
Stesura della relativa documentazione	10 ore
Implementazione della libreria	80 ore
Aggiornamento documentazione	10 ore
Ottimizzazione algoritmi	40 ore
Testing finale relativo alle performance	25 ore
Redazione finale della documentazione	5 ore
Integrazione dei progressi con l'applicativo Novaeka esistente	10 ore
Totale	330 ore

Tabella 1.2: Prospetto orario delle attività con il relativo preventivo orario

1.6 Tecnologie utilizzate

1.6.1 Rust

Sin dalla sua prima release stabile (1.0) nel 2015, il linguaggio compilato Rust ha sempre fatto di performance, affidabilità e produttività i suoi principi, adattandosi bene a paradigmi di programmazione imperativo-procedurale, funzionale, e persino *object-oriented*. Si è posto come obiettivo nel tempo la creazione di sistemi complessi altamente sicuri e ha investito nel favorire elevate prestazioni, buon controllo dell'allocazione di memoria e alta disposizione di documentazione, continuamente alimentata e aggiornata dalla fedele community sottostante [26]. Grazie a queste caratteristiche, sta diventando uno dei linguaggi emergenti più amati dagli utenti, nonché versatile per la programmazione nei contesti di sviluppo come quello in cui questo stage si inquadra. Maggiori informazioni riguardo la scelta e l'utilità in itinere di questo linguaggio verranno date nel Capitolo 3.



1.6.2 Visual Studio Code

Anche chiamato comunemente VS Code, è un IDE sviluppato da Microsoft nel 2015 per i sistemi operativi Linux, Microsoft e macOS. In aggiunta offre uno stabile supporto nella fase di debugging. È una scelta congeniale in fatto di sintassi del codice, poiché predispone dei plugin utili alla suddivisione, leggibilità della sintassi e correzione del codice che si sta scrivendo [47].

1.6.3 Discord

Con la necessità di dover comunicare e collaborare in ambito lavorativo, anche a distanza, si è ritenuto necessario in determinate occasioni l'utilizzo di servizi di VoIP e messaggistica come Discord. È un servizio lanciato nel mercato nel 2015 che permette agli utenti di effettuare chiamate vocali, videochiamate, inviare messaggi di testo, media e file in chat private o come membri di un server Discord.
Si è reso utile non solo in situazioni di smart-working di alcuni membri del team, ma anche per i meeting giornalieri con i superiori o in occasione di impedimenti non programmabili [14].

1.6.4 Azure DevOps Server

Per il versionamento del codice e l'organizzazione del progetto si è deciso di optare per il server nato in seno a Microsoft già in uso dall'azienda, dato che la successiva destinazione ed uso della libreria all'interno dell'applicativo Novaeka. L'interfaccia intuitiva e minimale ha facilitato l'organizzazione e l'ispezione del codice, sottolineando con chiarezza le modifiche operate con i commit in fase di aggiornamento del codice redatto [2].

1.6.5 Microsoft Office Excel

Oltre al suo naturale utilizzo come foglio di calcolo, questo software presente nel pacchetto Office ha giocato un ruolo fondamentale del controllo incrociato dei calcoli prodotti dalla libreria, data la disponibilità della libreria Coolprop a lavorare all'interno dell'ambiente Excel. Inoltre è stato sfruttato per l'organizzazione personale, il conteggio delle ore di stage e, per la sua predisposizione tabulare, per rappresentare graficamente alcune proprietà dei fluidi presi in esame.

1.7 Organizzazione del testo

Il secondo capitolo descrive l'apparato teorico alla base della libreria.

Il terzo capitolo si incentra sul prodotto software *Fluidprops*.

Nel quarto capitolo si traggono le conclusioni del prodotto e dell'esperienza di stage.

Capitolo 2

Helmholtz e le equazioni di stato

In questo capitolo teorico, verrà approfondito il significato delle equazioni di stato di fluidi puri o pseudo-puri, il calcolo che ne permette di ricavare le proprietà e le varie forme che queste assumono nel passaggio dai dati sperimentali alla forma matematica compiuta. Lo studio di queste nozioni ha permesso di acquisire competenze utili per giustificare le scelte progettuali compiute nella costruzione della libreria di calcolo e agevolare l'analisi dei prodotti già esistenti sul mercato.
Spesso si farà riferimento ai soli fluidi puri o pseudo-puri, in quanto principale oggetto dell'indagine e dell'elaborato, ma non va dimenticata l'applicabilità di tali principi anche ad altre forme della materia o a miscele di fluidi in qualsiasi proporzione. Le nozioni riportate in questo capitolo sono state tratte principalmente dal testo [35] e dalla documentazione relativa a Coolprop disponibile online [9].

2.1 Le equazioni di stato nella forma di Helmholtz

Nel campo della fisica, le equazioni di stato, in inglese Equation Of State (EOS), descrivono con la maggior accuratezza possibile il comportamento della materia sotto determinate condizioni quali pressione, volume, temperatura, o energia interna. Le EOS risultano utili per prevedere numerose proprietà di fluidi o mix di fluidi nei tre stati della materia, liquido, solido e gassoso, in modo da consentirne l'impiego nei più svariati ambiti applicativi, di ricerca o industriali [35].
Attualmente, non esiste un'unica equazione di stato che possa descrivere tutti i comportamenti possibili delle sostanze conosciute, ma spesso si ricorre all'interpolazione e approssimazione di dati sperimentali accumulati nel tempo, per produrre un'equazione adatta al fluido in oggetto e che ne descriva il comportamento fisico col minor errore possibile. Questa procedura operativa è largamente utilizzata con risultati notevoli sulla maggior parte dei fluidi esaminati al momento corrente e in letteratura scientifica si trovano numerosi articoli che aggiornano, limano e correggono le EOS ogni qual volta vengano aggiunti risultati sperimentali nuovi o migliori [11].
L'energia di Helmholtz, cui spesso si fa riferimento nel corso di questa tesi, è stata elaborata per la prima volta nel 1882 dal fisico tedesco Hermann von Helmholtz all'interno della conferenza intitolata *On the thermodynamics of chemical processes* [18] ed è rappresentata dalla grandezza 2.1:

$$A = U - TS \tag{2.1}$$

dove A rappresenta l'energia libera di Helmholtz (spesso rappresentata anche come a o F) in forma adimensionale, U è l'energia interna del sistema, T la temperatura espressa in gradi Kelvin e S l'entropia del sistema.
Sempre più spesso, le EOS utilizzano la forma dell'energia di Helmholtz per essere rappresentate, in funzione di due grandezze indicate con τ e δ come unici parametri indipendenti. Questa scelta è frutto del grande vantaggio che ne deriva, ovvero la possibilità di calcolare tutte le altre proprietà termodinamiche da questa particolare forma, sfruttandone le derivate parziali [11]. Tau (τ) e delta (δ) vengono adottati come forme particolari di temperatura e densità del fluido, e sono denominate *Reduced variables*, poiché rapportate con le temperature e densità critiche nella maniera della formula 2.2.

$$\tau = \frac{T_c}{T} \quad \delta = \frac{\rho}{\rho_c} \tag{2.2}$$

Con queste premesse, l'equazione di stato nella forma dell'energia di Helmholtz per un dato fluido risulta come:

$$\alpha(\tau, \delta) = \frac{a(T, \rho)}{RT} = \frac{a^0(T, \rho) + a^r(T, \rho)}{RT} = \alpha^0(\tau, \delta) + \alpha^r(\tau, \delta) \tag{2.3}$$

dove $a(T, \rho)$ è l'energia di Helmholtz del fluido, R la costante universale dei gas pari a $8,314462 \frac{J}{molK}$, T la temperatura in Kelvin e ρ la densità (molare o meno).
Nel termine destro della formula vediamo le due componenti principali delle EOS in questa forma, che descrivono il comportamento della sostanza come somma del suo aspetto di gas ideale e della porzione residuale. α^0 o *Ideal term* è legato all'energia di Helmholz per i gas ideali, quindi descrive il comportamento del fluido quando assume la forma di molecole con massa puntiforme, indipendenti e a sé stanti, senza alcun tipo di interazione tra di loro. La porzione residuale α^r o *Residual term* invece si occupa di descrivere le interazioni tra le molecole, in caso vi siano; il termine agisce quindi da correttore per la formula nel caso in cui il fluido a date τ e δ non si trovi nel suo stato di gas ideale. Quest'ultima porzione è quella più legata a dati fisici sperimentali, sui quali le EOS sono state costruite. È quella che assume le forme più variegate di calcolo, mentre se considerassimo il fluido a densità estremamente basse, quindi vicino al suo comportamento di gas ideale, potremmo utilizzare la sola parte ideale α^0.
Questi due blocchi costituiscono i due addendi principali entro cui possiamo descrivere le EOS per i vari fluidi e, di conseguenza, il loro comportamento. I blocchi sono indipendenti tra loro e presentano una struttura fluido-dipendente, quindi il passo seguente nella comprensione di questo schema è l'individuazione delle strutture elementari che li compongono, come diversi pezzi di un mosaico che possono generare una rosa di formule ampia tanto quanto il numero delle loro combinazioni [35].

2.1.1 La formula per i gas ideali

La formula dell'energia di Helmholtz per i gas ideali, o *Ideal Term*, assume la forma dell'equazione generalista 2.4. Questa forma esprime il comportamento del fluido nel suo stato di gas ideale, ovvero quando è idealmente formato da molecole puntiformi che non interagiscono fisicamente tra loro, ma ammettono solo mutamenti di traslazione, rotazione e vibrazione.
La Formula 2.4 in letteratura è suddivisa in diverse porzioni, in inglese *terms*, composte da uno o più addendi, la cui presenza non è assicurata nell'equazione di stato specifica per ogni fluido. Ognuno dei fluidi, infatti, gode di una particolare combinazione di questi *terms*, che comporranno l'*Ideal Term* su misura per quel fluido e ne esprimeranno le peculiarità fisiche secondo le variabili indipendenti τ e δ fissate.
Ogni coefficiente o esponente delle formula, descritti meglio nelle sezioni dedicate successivamente ad ogni *term*, è fornito in input da appositi dati sperimentali [9] e rappresenta un set unico di dati che sarà valido soltanto per il fluido in esame.

$$\alpha^0 = \ln(\delta) + a_1 + a_2\tau + n_1 \ln(\tau) + \sum_{i=1}^{I_{Pol}} c_i \tau^{t_i} + \sum_{k=1}^{K_{PE}} m_k \ln(1 - e^{-\vartheta_k \tau}) \tag{2.4}$$

Dall'analisi di altre librerie similari [11] [30] e del manuale teorico [35], è emerso che i *terms* principali da cui può essere costituito α^0 sono cinque, come di seguito descritti.

Lead Term

$$\ln(\delta) + a_1 + a_2\tau \tag{2.5}$$

Il *Lead Term* è formato dalla manipolazione di due coefficienti a_1 e a_2, che variano a seconda del fluido e vengono forniti dai vari paper in base ai dati sperimentali. Questo termine è fondamentale per la maggior parte dei fluidi e appare quasi nella totalità dei casi. In alcuni casi, ai coefficienti si aggiunge uno scarto denominato Entropy-Enthalpy Offset (EEO) che descrive lo scostamento dallo stato di riferimento del fluido come definito dalla Formula 2.6, anche se nella maggior parte dei casi questo scostamento è nullo.

$$EEO = \frac{\Delta s}{R_u/M} + \frac{\Delta h}{(R_u/M)T}\tau \tag{2.6}$$

Logtau Term

$$n_1 \ln \tau \tag{2.7}$$

Il *Logtau Term* necessita di un solo coefficiente esterno per il calcolo denominato n_1 (a volte indicato anche come c_0). Anche questo compare in quasi tutte le EOS analizzate.

Polynomial Term

$$\sum_{i=1}^{I_{pol}} c_i \tau^{t_i} \tag{2.8}$$

Il *Polynomial Term* è un termine polinomiale dato dalla somma di un certo numero di termini nella stessa forma: $c_i\tau^{t_i}$. Il numero i delle iterazioni sulla sommatoria, e quindi di coefficienti ed esponenti, non è prevedibile a priori e varia drasticamente da caso a caso, o per meglio dire, da fluido a fluido. Per questo motivo il numero di addendi della sommatoria è indicato con I_{pol}.

Planck-Einstein Term

$$\begin{gathered}\sum_{k=1}^{K_{PE}} m_k \ln[c_k + d_k e^{\vartheta_k \tau}] \\ \text{con } c_k = 1 \text{ e } d_k = -1 \\ \sum_{k=1}^{K_{PE}} m_k \ln[1 - e^{\vartheta_k \tau}]\end{gathered} \tag{2.9}$$

Il *Planck-Einstein Term* è un termine piuttosto articolato rispetto ai precedenti con la presenza di un termine polinomiale contaminato da altri coefficienti e un'esponenziale, dove i coefficienti c_k e d_k assumono sempre lo stesso valore, $c_k = 1$, $d_k = -1$.
Come nel caso del *Polynomial Term* anche qui la dimensione della sommatoria è diversa a seconda dei casi e, per tal motivo, è indicata genericamente con K_{PE}.

Cp0Poly Term

$$\begin{cases} \sum_{i=1}^{I_{CP}} c_{p_i} - c_{p_i}\frac{\tau}{\tau_0} + c_{p_i}\ln(\frac{\tau}{\tau_o}), & \text{se } |t_i| = 0 \\ \sum_{i=1}^{I_{CP}} c_{p_i}\frac{\tau}{T_c}\ln(\frac{\tau_0}{\tau}) + \frac{c_{p_i}}{T_c}(\tau - \tau_0), & \text{se } |t_i + 1| = -1 \\ \sum_{i=1}^{I_{CP}} -c_{p_i}T_c^{t_i}\frac{\tau^{-t_i}}{t_i(t_i+1)} - c_{p_i}T_0^{t_i+1}\frac{\tau}{T_c(t_i+1)} + c_{p_i}\frac{T_0^{t_i}}{t_i}, & \text{altrimenti} \end{cases} \tag{2.10}$$

In ultimo si trova il *Cp0Poly Term*, un termine scarsamente utilizzato dai fluidi. Non fa parte abitualmente della formula generale dell'*Ideal Term* 2.4, ma può servire da fattore di correzione per alcuni fluidi.
È in stretta relazione con la capacità termica del fluido (c_p), da cui è generato anche il set di coefficienti c_{p_i} e varia la sua struttura allo scandire dei valori t_i.
I singoli c_{p_i} rappresentano dei coefficienti forniti in input, mentre τ_0 è il risultato di T_c/T_0, dove T_0 è la temperatura del fluido di riferimento e T_c la temperatura critica.

2.1.2 La formula per l'energia residuale

A differenza dell'*Ideal term*, l'energia residuale registra le interazioni tra le molecole del fluido, ovvero tutti quegli scostamenti fisici dal suo stato di gas ideale, pur non disponendo in letteratura di modelli fisici che descrivano universalmente e accuratamente questo comportamento. In mancanza di modelli generali adeguati, si è ricorso all'interpolazione di misurazioni sperimentali per costruire questa porzione dell'equazione [9].
Come per l'*Ideal Term*, anche il presente, sebbene naturalmente più variegato come composizione e suscettibile di variazioni significative tra fluidi diversi, è frutto della combinazione di *terms* diversi che ne definiscono la peculiare struttura [35].
Mentre nell'*Ideal Term* si nota più regolarità tra le combinazioni, nel *Residual Term*, dove disponiamo esclusivamente di modelli sperimentali, ricorre frequentemente un termine di base, presente nella quasi totalità dei fluidi analizzati, accompagnato da altre porzioni di formula pressoché scarsamente ripetute.
Qui i *terms* da investigare sono ben otto e possono essere sintetizzati dalla formula generale 2.11 relativa all'energia residuale di Helmholtz.

$$\alpha^r(\tau,\delta) = \sum_{i=1}^{I_{pol}} n_i \delta^{d_i} \tau^{t_i} + \sum_{i=I_{pol}+1}^{I_{pol}+I_{exp}} n_i \delta^{d_i} \tau^{t_i} e^{-\gamma_i \delta^{p_i}} \tag{2.11}$$

La Formula 2.11 è composta da due addendi principali: un termine polinomiale e un termine misto polinomiale ed esponenziale. I due addendi possono assumere forme diverse al variare del fluido preso in esame. In alcuni casi sono presenti entrambi così come nella formula generale, in altri mutano solamenti degli indici dell'esponenziale, in altri ancora cambia completamente forma il secondo addendo.
Entrambe le sommatorie si muovono su un range di indici i che variano in numero e valore a seconda del fluido. A tal proposito, trattandosi di formulazione generali, non è possibile stabilire a priori la grandezza delle sommatorie, costringendoci a fissarne i limiti superiori a I_{pol} e $I_{pol} + I_{exp}$.
Qui di seguito verranno elencati i possibili otto tasselli che compongono le varie formule dell'energia residuale, mentre le loro specifiche saranno indicate nelle corrispondenti sezioni.

Power Term

$$\begin{cases} \sum_{i=1}^{I_{pol}} n_i \delta^{d_i} \tau^{t_i} & l_i = 0 \\ \sum_{i=I_{pol}+1}^{I_{pol}+I_{exp}} n_i \delta^{d_i} \tau^{t_i} e^{-\delta^{l_i}} & l_i \neq 0 \end{cases} \tag{2.12}$$

Il *Power Term* è il termine che ricorre più frequentemente fra le varie combinazioni che definiscono l'energia residuale. Ad ogni iterazione, il *Power Term* si arricchisce di un membro polinomiale o polinomiale ed esponenziale a seconda del valore dell'esponente l_i associato, mentre la variabile γ della Formula 2.11 permane sempre al valore 1. La forma ricorda molto quella della formula generale, e per raggruppamento ne possiamo ancora delineare la struttura. Gli altri elementi (n_i, d_i, t_i) rappresentano coefficienti ed esponenti modellati per interpolare il comportamento del fluido secondo le variabili δ e τ.

Exponential Term

$$\sum_{i=1}^{I_{exp}} n_i \delta^{d_i} \tau^{t_i} e^{-\gamma_i \delta^{l_i}} \tag{2.13}$$

L'*Exponential Term* risulta spesso abbinato al *Power Term* nelle varie combinazioni. Essa apporta delle modifiche sostanziali all'addendo esponenziale della formula generale permettendo alla variabile γ_i di variare. Sarà necessario, di conseguenza, includerla nella rosa degli input forniti.

Lemmon Term

$$\sum_{i=1}^{I_{lem}} n_i \delta^{d_i} \tau^{t_i} e^{-\delta^{l_i} - \tau^{m_i}} \tag{2.14}$$

Prende il nome dall'articolo scritto proprio da Eric W. Lemmon nel 2005 [22], che forniva nuove forme di rappresentazione delle equazioni di stato per il fluido refrigerante R-125. Quest'ultimo è anche l'unico fluido ad utilizzare questo termine. Lemmon propone una variazione sul secondo addendo modificando il contenuto dell'esponenziale: il termine γ_i rimane ad 1, mentre si aggiunge in coda un'altra porzione che valuta anche τ accompagnato da una serie di esponenti m_i forniti in input.

Gaussian Term

$$\sum_{i=1}^{I_{gau}} n_i \delta^{d_i} \tau^{t_i} e^{-\eta_i (\delta - \epsilon_i)^2 - \beta_i (\tau - \gamma_i)^2} \tag{2.15}$$

Il *Gaussian Term* è il secondo termine più frequente tra i fluidi di interesse e presenta un notevole numero di coefficienti ed esponenti forniti in input. Anch'esso è una variante del secondo addendo che ne modifica la porzione esponenziale sfruttando nuovi parametri $(\eta_i, \epsilon_i, \beta_i)$ per la costruzione di un termine gaussiano.

GERG Gaussian Term

$$\sum_{i=1}^{I_{gerg}} n_i \delta^{d_i} \tau^{t_i} e^{-\eta_i (\delta - \epsilon_i)^2 - \beta_i (\delta - \gamma_i)} \tag{2.16}$$

Utilizzato esclusivamente per le EOS che rappresentano miscele di fluidi, il *GERG Gaussian Term* prende spunto dal lavoro di O. Kunz and W. Wagner per le equazioni di stato dei gas naturali [21]. Il termine viene riportato per completezza, ma trattando esclusivamente fluidi puri o pseudo-puri, non sarà oggetto di questa tesi. La sua forma rispecchia molto il *Gaussian Term*, dal quale prende parte del nome, ma è manchevole di una potenza sull'argomento dell'esponenziale.

NonAnalytic Term

$$\begin{aligned}
&\sum_{i=1}^{I_{na}} n_i \Delta^{b_i} \delta\psi \\
&\Delta = \theta^2 + B_i[(\delta-1)^2]^{a_i} \\
&\theta = (1-\tau) + A_i[(\delta-1)^2]^{1/(2\beta_i)} \\
&\psi = \exp(-C_i(\delta-1)^2 - D_i(\tau-1)^2)
\end{aligned} \tag{2.17}$$

Il *NonAnalytic Term* non fa frequentemente parte delle equazioni di stato, ma è segmento essenziale per il calcolo dell'energia residuale dell'acqua.
La sua forma è completamente stravolta rispetto ai casi appena elencati, ma spesso si accompagna al primo addendo in sostituzione del termine esponenziale.

SAFTAssociation Term

$$\begin{aligned}
&a \cdot m \left(\ln X - \frac{X}{2} + \frac{1}{2} \right) \\
&\text{dove} \\
&X = \frac{2}{\sqrt{1 + 4\bar{\Delta}\delta} + 1} \\
&\bar{\Delta} = g(\eta(\delta))(e^{\bar{\epsilon}\tau} - 1)\bar{\kappa} \\
&\eta(\delta) = \bar{V}_n \delta \\
&g(\eta) = \frac{1}{2}\frac{(2-\eta)}{(1-\eta)^3}
\end{aligned} \tag{2.18}$$

Il *SAFTAssociation Term* risulta parte integrante solamente dell'equazione di stato del metanolo. Deve il suo nome alle equazioni di stato fondate sulla base della SAFT [32] e dispone di una serie di variabili e funzioni variegate per il calcolo del risultato.

GaoB Term

$$\begin{aligned}
&\sum_{i=1}^{I_{gaob}} n_i F_\tau F_\delta \\
&F_\tau = \tau^{t_i} e^{\frac{1}{b_i + \beta_i(-\gamma_i+\tau)^2}} \\
&F_\delta = \delta^{d_i} e^{\eta_i(\delta_i - \epsilon_i)^2}
\end{aligned} \tag{2.19}$$

In ultimo, il *GaoB Term* risulta sfruttato solo dall'equazione di stato dell'ammoniaca e prende il nome dai suoi autori K. Gao e I. H. Bell [16].
È l'ennesimo caso di un termine completamente diverso dalla struttura della formula generale, anche se permane il coefficiente n_i iniziale e la presenza di τ^{t_i} e δ^{d_i}. Le restanti variabili sono tutte fornite in input dal codice [9] e ricavati dalle misurazioni sperimentali.

2.2 L'utilizzo delle derivate nel calcolo delle proprietà

Il punto di forza nell'utilizzo dell'energia di Helmholtz sta proprio, come già accennato nelle sezioni precedenti, nella possibilità di calcolare tutte le altre proprietà termodinamiche, tramite la conoscenza dei due soli parametri di input, δ e τ, sfruttando le derivate parziali delle equazioni di stato. Questa caratteristica deriva direttamente da una Formula 2.20 molto nota nell'ambito della termodinamica che descrive un sistema che mantiene una costante quantità di sostanza [35]. Ciò permette di raccogliere una grande mole di dati riguardanti lo stato del fluido, con il minimo utilizzo di variabili indipendenti e abbassando notevolmente la complessità di calcolo oltre che di gestione della formula. Tale formulazione permette, inoltre, di sfruttare al meglio la potenza di calcolo e l'efficienza di un calcolatore, permettendo di creare un'unica libreria software capace di svolgere questo compito.

$$\begin{aligned} \partial a(T,v) &= -s\partial T - p\partial v \\ a &= \text{ energia di Helmholtz} \\ T &= \text{ temperatura} \\ v &= \text{ volume} \\ s &= \text{ entropia} \\ p &= \text{ pressione} \end{aligned} \tag{2.20}$$

Per lo svolgimento del progetto, sono state analizzate le nove proprietà dei fluidi che risultavano più utili agli scopi dell'azienda e che, comunemente, sono sfruttate più frequentemente. Nella Tabella 2.1 sono riportate le proprietà in oggetto, accostate alla formula che tipicamente è utilizzata nella loro definizione fisica, mentre nella colonna di destra si sviluppa la loro formula utilizzando l'energia di Helmholtz.
Per comodità di calcolo, tutte le formule verranno espresse nella forma delle *reduced variables* δ e τ. Nell'indicare le varie derivate si utilizzerà la grafia α_N^A per indicare la derivata parziale di α^A rispetto a N, con A pari a 0 (*Ideal Term*) oppure r (*Residual Term*). La presenza di un doppio simbolo a pedice indica il numero di volte con cui avviene la derivazione, ad esempio $\alpha_{\tau\tau}^0$ indicherà l'*Ideal Term* (α^0) derivato due volte rispetto a τ.

Proprietà	Formula classica	Formula in relazione ad α
Pressione	$p(T,\rho) = -(\partial a/\partial v)_T$	$\frac{p}{\rho RT} = 1 + \delta\alpha^r_\delta$
Entropia	$s(T,\rho) = -(\partial a/\partial T)_v$	$\frac{s}{R} = \tau(\alpha^0_\tau + \alpha^r_\tau) - \alpha^0 - \alpha^r$
Energia interna	$u(T,\rho) = a + Ts$	$\frac{u}{RT} = \tau(\alpha^0_\tau + \alpha^r_\tau)$
Capacità termica isocora	$c_v(T,\rho) = (\partial u/\partial T)_v$	$\frac{c_v}{R} = -\tau^2(\alpha^0_{\tau\tau} + \alpha^r_{\tau\tau})$
Entalpia	$h(T,p) = u + pv$	$\frac{h}{RT} = 1 + \tau(\alpha^0_\tau + \alpha^r_\tau) + \delta\alpha^r_\delta$
Capacità termica isobara	$c_p(T,p) = (\partial h/\partial T)_p$	$\frac{c_p}{R} = -\tau^2(\alpha^0_{\tau\tau} + \alpha^r_{\tau\tau}) + \frac{(1+\delta\alpha^r_\delta - \delta\tau\alpha^r_{\delta\tau})^2}{1+2\delta\alpha^r_\delta + \delta^2\alpha^r_{\delta\delta}}$
Energia di Gibbs	$g(T,p) = h - Ts$	$\frac{g}{RT} = 1 + \alpha^0 + \alpha^r + \delta\alpha^r_\delta$
Velocità del suono	$w(T,p) = \sqrt{(\partial p/\partial\rho)_s}$	$\frac{w^2}{RT} = 1 + 2\delta\alpha^r_\delta + \delta^2\alpha^r_{\delta\delta} - \frac{(1+\delta\alpha^r_\delta - \delta\tau\alpha^r_{\delta\tau})^2}{\tau^2(\alpha^0_{\tau\tau} + \alpha^r_{\tau\tau})}$
Fattore di comprimibilità	$Z(T,\rho) = \frac{p}{\rho RT}$	$Z = 1 + \delta\alpha^r_\delta$

Tabella 2.1: Proprietà dei fluidi sfuttando l'energia di Helmholtz [35]

2.3 L'equilibrio liquido-vapore e i fluidi bi-fase

Nel calcolo delle proprietà termodinamiche di un fluido non si possono di certo ignorare le fasi che la materia assume a determinate condizioni fisiche. La fase è una porzione di un sistema termodinamico che presenta stato fisico e composizione chimica essenzialmente uniformi, mentre altre grandezze possono essere non uniformi. Il concetto di fase non equivale allo stato di aggregazione della materia (solido, liquido e gassoso), bensì all'interno dello stesso sistema possono essere presenti ad esempio più fasi liquide o più fasi solide [35].
Fino ad ora abbiamo considerato il calcolo delle proprietà termodinamiche in una fase omogenea, ovvero in tutte quelle quelle situazioni in cui la materia è in un'unica fase alla volta. A particolari condizioni fisiche, però, si possono riscontrare delle zone di equilibrio, ovvero quando assistiamo alla presenza di due fasi che si trovano in una situazione di equilibrio termodinamico tra loro [35].
Nello specifico ci occuperemo ora dell'equilibrio liquido-vapore proprio perché riguarda maggiormente la materia di cui stiamo trattando: i fluidi. All'interno dei confini delle fasi di equilibrio, rappresentate frequentemente come due curve che si congiungono a formare una campana (Figura 2.1), vi è una porzione fisica dove le fasi non sono omogenee, bensì risultano coesistere due fasi contemporaneamente (liquida e gassosa).

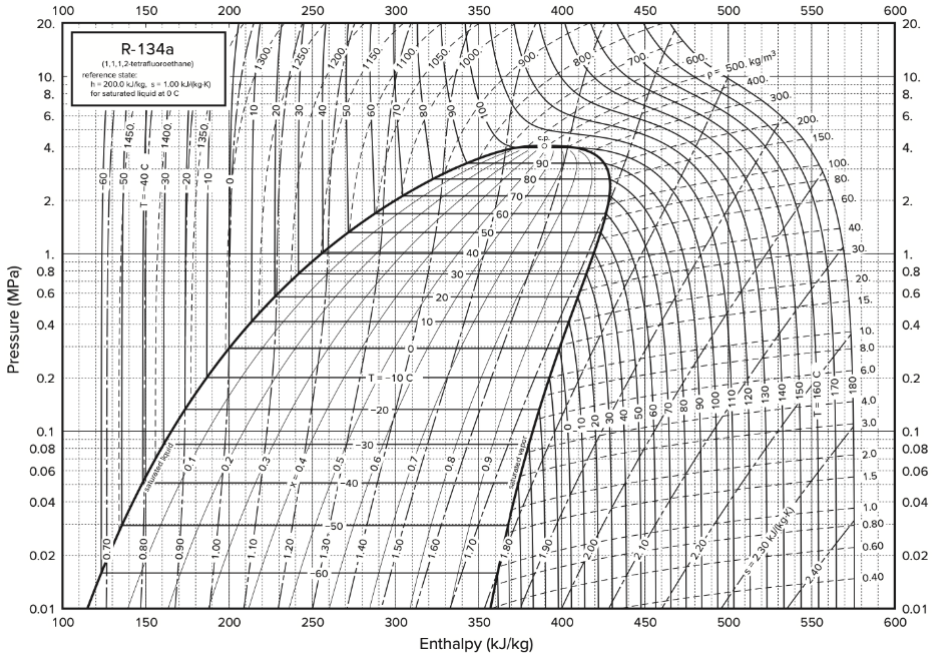


Figura 2.1: Rappresentazione del fluido R-134a sul piano di entalpia (H) e pressione (p) [17]

Per individuare queste aree dobbiamo innanzitutto delineare i confini delle fasi di equilibrio, dette *curve di saturazione*, che delimitano la zona bi-fase (*Two phase region*) ove il calcolo delle proprietà nella forma enunciata precedentemente non può essere più valido.
L'equilibrio delle fasi liquida ($'$) e gassosa ($"$) può essere descritto dall'equivalenza, nelle due fasi, di dati quali temperatura, pressione ed energia di Gibbs, come indicato dalle equivalenze 2.21.

$$\begin{aligned} T' &= T" \\ p' &= p" \\ g' &= g" \end{aligned} \tag{2.21}$$

Le quantità di densità ove sussistono queste condizioni sono calcolabili tramite metodi numerici, come il metodo di Newton, ma per questa operazione è sufficiente individuare quei due punti nello spazio termodinamico che, a parità di temperatura, si trovano in fasi di equilibrio diverse, delimitando la campana, con differenti densità. Una volta individuati, si procede con il calcolo della proprietà in oggetto utilizzando l' input di temperatura comune e come densità, rispettivamente, i punti (T, ρ) calcolati attraverso il metodo numerico. Si proseguirà con il calcolo della *vapour fraction* sulla base dei due risultati come indicato dalla Formula 2.22; per esprimere la quantità di massa presente in ognuna delle fasi, che corrisponderà a 1 per stati sulla linea del gas saturo e a 0 per quelli sulla linea del liquido saturo.

$$x = \frac{\frac{1}{\rho(T,x)} - \frac{1}{\rho'(T)}}{\frac{1}{\rho''(T)} - \frac{1}{\rho'(T)}} = \frac{y(T,x) - y'(T)}{y''(T) - y'(T)} \tag{2.22}$$

Il tutto per poter finalmente ottenere il valore della proprietà desiderata anche in condizioni bi-fase o al limite dell'equilibrio, come indicato dalla Formula 2.23.

$$y(T,x) = y'(T) + x(y''(T) - y'(T)) \tag{2.23}$$

Fortunatamente questo passaggio ulteriore è necessario solo in determinate condizioni, che possono rappresentare anche una consistente porzione del piano termodinamico per alcuni fluidi. Durante l'operazione di calcolo ci si preoccuperà preventivamente di verificare in quale fase il fluido si trova, per poi procedere, eventualmente, al calcolo nelle condizioni appena descritte. In tutti gli altri casi sarà sufficiente sfruttare le formule così come riportate nella Tabella 2.1.

2.4 Le proprietà di transport

Tra le proprietà dei fluidi che tratteremo ne esistono due, non ancora menzionate, che necessitano di un trattamento diverso per il loro calcolo: viscosità e conduttività termica. Queste due quantità fanno parte del macro-insieme delle proprietà di trasporto (*transport properties*), ovvero tutti quei meccanismi di trasporto di quantità fisiche che presentano analogie a livello molecolare e nella loro descrizione come modello matematico [35].
Sebbene si servano ugualmente dell'energia di Helmholtz per la costruzione delle loro formule di calcolo, queste proprietà non possono essere calcolate tramite derivazione dalle equazione di stato, ma necessitano di forme proprie oltre che set di dati appositi. Sia che si tratti di viscosità o di conduttività, le librerie esistenti mettono a disposizione il calcolo diretto delle proprietà, che chiameremo *classic model*, solo per un gruppo limitato di fluidi, mentre per gli altri ci si avvarrà di metodi alternativi basati sulla stima della quantità a confronto con fluidi di riferimento, oppure su formule costruite ad hoc per quel determinato fluido ed innestate direttamente nel codice.

2.4.1 La Viscosità

La viscosità viene definita come il coefficiente di scambio di quantità di moto e, a livello microscopico, rappresenta l'attrito tra le varie molecole di un fluido. Tale quantità è espressa in maniera generale dalla formula 2.24 che separa in due addendi il contributo che ciascuno di due membri apporta al valore finale [35].

$$\begin{aligned} \eta(T,\rho) &= \eta^*(T) + \Delta\eta(T,\rho) = \\ &= \eta^*(T) + \Delta\eta_0(T_0,\rho_0)F_\eta(T,\rho) \end{aligned} \tag{2.24}$$

Il primo addendo è detto *Dilute viscosity*, ovvero la descrizione della viscosità nella condizione particolare in cui il fluido presenti densità pari a 0, rappresentante quindi lo stato base del fluido. Il secondo addendo è, invece, la *Residual contribution*: lo scostamento dallo stato ideale di densità nulla e l'apporto più suscettibile di variazioni

durante il calcolo. Il pedice 0 indica la stima della quantità di viscosità sulla base di un fluido standard di riferimento, campionato a temperatura T_0 e densità ρ_0, considerando le dovute manipolazioni in carico alle *shape functions* (Formula 2.25) nell'analisi microscopica della forma delle molecole [19].

$$\begin{aligned} T_0 &= \frac{T}{f} \\ \rho_0 &= \rho h \\ f &= \frac{T_c}{T_{C_0}}\theta \\ h &= \frac{\rho_{c_0}}{\rho_c}\Phi \end{aligned} \tag{2.25}$$

Nella sua forma più completa, il calcolo classico della viscosità si struttura come la somma di tre termini (Formula 2.26) [19]:

$$\eta(T,\rho) = \eta^*(T)[1 + B_\eta(T)\rho] + \Delta_H\eta(T,\rho) + \Delta_c\eta(T,\rho) \tag{2.26}$$

- *Dilute Term*: valore della viscosità in uno stato ideale a densità nulla;
- *Initial density Term*: rappresenta il secondo coefficiente viriale di viscosità, ovvero la variazione della temperatura in correlazione alla densità;
- *Higher-order density Term*: misura la viscosità residuale e il contributo all'avvicinarsi del punto critico. A tal proposito, quest'ultimo termine è comunemente scisso in *Residual Term* e *Critical Term*.

Dilute Term

$$\eta^*(T) \tag{2.27}$$

Dall'analisi della letteratura, ma soprattutto del codice a disposizione nelle librerie maggiormente note, questo termine sembra assumere forme diverse strettamente legate al fluido che si va analizzando [15]. Molte forme sono generate a partire dall'integrale di collisione (*collision integral*) o dalla sezione d'urto del fluido (*effective cross section*), oppure ancora dalla teoria cinetica dei gas [19] [7]. In rari casi vi sono delle formule appositamente costruite per il calcolo della viscosità di determinati fluidi, che chiameremo *hardcoded*, che non rientravano nelle categorie precedenti e non potevano essere trattati altrimenti.
Nella Tabella 2.2 sono stati elencate le forme che il *Dilute Term* può assumere nel calcolo della viscosità. Tali forme sono fluido-dipendenti e i loro coefficienti sono forniti direttamente dai dati sperimentali. Nella scrittura delle formule spesso si presenteranno

gli elementi descritti nella Formula 2.28.

$$\begin{aligned} \sigma, \frac{\epsilon}{k} &= \text{parametri Lennard-Jones} \\ M &= \text{massa molare } (kg/mol) \\ T_c &= \text{temperatura critica } (K) \\ A, C_i, S_\eta, a_i, d_i, t_i &= \text{opportune variabili in input} \end{aligned} \tag{2.28}$$

Forma	Calcolo	
Collision integral	$\frac{A\sqrt{MT}}{\sigma^2 \vartheta^*(T^*)}$	$\vartheta(T^*) = e^{\sum_i a_i [\ln T^*]^{t_i}}$ $T^* = \frac{T}{\frac{\epsilon}{k}}$
Collision integral powers of T	$\frac{A\sqrt{T}}{\sum_{i=1}^{I_{CPT}} a_i T^{*t_i}}$	$T^* = \frac{T}{T_c}$
Kinetic theory	$\frac{26.692 \cdot 10^{-9}\sqrt{MT}}{\sigma^2 \vartheta^*(T^*)}$	
Powers of T	$\sum_{i=1}^{I_{PT}} a_i T_{t_i}$	
Powers of Tr	$\sum_{i=1}^{I_{PTr}} a_i (\frac{T}{T_c})^{t_i}$	
Ethane	$12.0085 \cdot \sqrt{T^*} \frac{\Omega^{(2,2)}}{10^6}$	$\Omega^{(2,2)} = \sum_i C_i T^{* \frac{(i-1)}{3-1}}$
Cyclohexane	$0.19592 \cdot \frac{\sqrt{T}}{S_\eta} \frac{1}{10^6}$	

Tabella 2.2: Le varie forme del Dilute Term nel calcolo della viscosità [15]

Initial density Term

$$B_\eta(T) \tag{2.29}$$

Per questo termine, le forme ottenibili sono solamente due, come mostrato dalla Tabella 2.3, ma entrambe mirano a rappresentare l'apporto della dipendenza dalla densità iniziale contenuto nella formula, o attraverso la teoria di *Rainwater-Friend* oppure

attraverso il *modello empirico* [15].

Forma	Calcolo	
Rainwater-Friend	$\sum_{i=1}^{I_R F} b_i (T^*)^{t_i} N_A \sigma^3$	N_A = numero di Avogadro b_i = opportuno coefficiente
Empirical	$\sum_{i=1}^{I_E} n_i \delta^{d_i} \tau^{t_i}$	n_i = coefficiente

Tabella 2.3: Le varie forme dell'Initial density Term nel calcolo della viscosità [15]

Higher-order Term

$$\Delta_{H\eta}(\rho, T) \tag{2.30}$$

La funzione che copre questa porzione di formula è quella di esprimere la viscosità del fluido a livelli alti di densità utilizzando, a seconda dei casi: la forma modificata del modello *Batschinski-Hildebrand*, la *Friction Theory* (teoria dell'attrito legata al comportamento microscopico delle molecole), oppure ancora, come anche i casi precedenti, prevede una discreta quantità di formulazioni personalizzate per i fluidi che le richiedono [15].
Nella Tabella 2.4 si elencano le varie forme dell'*Higher-order Term* nel calcolo della viscosità.
Per il calcolo della forma *Batschinski-Hildebrand* saranno necessari un certo numero di parametri $(a_i, d1_i, t1_i, \gamma_i, l_i, p_i, q_i, g_i, h_i, d2_i, f_i, t2_i)$ forniti dalla libreria e dipendenti dall'indice i che ha limite superiore variabile a seconda del fluido in esame. Il risultato finale, inoltre, dipenderà dal fattore δ_0 presente nella Tabella 2.4 e scritto in forma estesa nella Formula 2.31.

$$\delta_0 = \sum_{i=1}^{I_{d0}} \frac{g_i \tau^{h_i}}{p_i \tau^{q_i}} \tag{2.31}$$

La forma della *Friction theory*, invece, sfrutta una serie di coefficienti k generati attraverso diverse serie di dati forniti in input e di accurate misure della pressione del fluido (p) come stabilito dal metodo stesso.
Infine incontriamo le varie forme *hardcoded* dell'*Higher-order Term* che sfruttano delle formule personalizzate solamente per quel particolare fluido. I vari c_i che si incontrano corrispondono a coefficienti codificati per interpolare le forme che i metodi precedenti non potevano esprimere.

Forma	Calcolo
Batschinski-Hildebrand	$\sum_{i=1}^{I_S} a_i \delta^{d1_i} \tau^{t1_j} e^{\gamma_i \delta^{l_i}} + \left(\sum_{i=1}^{I_F} f_i \delta^{d2_i} \tau^{t2_i} \right) \left(\frac{1}{\delta_0(\tau) - \delta} - \frac{1}{\delta_0(\tau)} \right)$
Friction theory	$\kappa_a p_a + \kappa_r \Delta p_r + \kappa_i p_{id} + \kappa_{aa} p_a^2 + \kappa_{drdr} \Delta p_r^2 + \kappa_{rr} p_r^2 + \kappa_{ii} p_{id}^2 + \kappa_{rrr} p_r^3 + \kappa_{aaa} p_a^3$
Hexane	$\rho^{2/3} \sqrt{T_r} [\frac{c_0}{T_r} + \frac{c_1}{c_2 + T_r + c_3 \rho^2} + c_4 \frac{1+\rho}{c_5 + c_6 T_r + c_7 \rho + \rho^2 + c_8 \rho T_r}]$
Hydrogen	$c_1 + \rho^2 e^{c_2 T_r + \frac{c_3}{T_r} + c_4 \frac{\rho^2}{c_5 + T_r} + c_6 \rho^6}$
Heptane	$\rho^{2/3} \sqrt{T_r} [c_1 \rho + c_2 \rho^2 + c_3 \rho^3 + c_4 \frac{\rho}{c_5 + c_6 Tr + c_7 \rho + \rho^2 + c_8 \rho Tr}]$
Ethane	$15.977 \cdot \frac{\sum_{i=1}^{9} g_i \delta^{r_i} \tau^{s_i}}{(1 + \sum_{i=10}^{11} g_i \delta^{r_i} \tau^{s_i})} \frac{1}{10^6}$
Benzene	$10^{-6} \cdot \rho^{2/3} \sqrt{T_r} [c_0 \rho^2 + c_1 \frac{\rho}{c_2 + c_3 T_r + c_4 \rho} + \frac{c_5 \rho + c_6 \rho^2}{c_7 + c_8 \rho^2}]$
Toulene	$10^{-6} \cdot \rho^{2/3} \sqrt{T_r} [\frac{c_0 \rho + c_1 \rho^4}{T_r} + c_2 \frac{\rho^3}{\rho^2 + c_3 + c_4 T_r} + c_5 \rho]$

Tabella 2.4: Le varie forme dell'Higher-order Term nel calcolo della viscosità [15]

Modelli alternativi di calcolo

Nell'approcciarsi al calcolo delle proprietà termodinamiche attraverso lo strumento del calcolatore, possiamo decidere, quando necessario, di abbandonare i metodi analitici e utilizzare quelli numerici per approssimare il più possibile soluzioni che non sarebbero ottenibili con i modelli classici analitici. Se ciò non bastasse, si potrà anche decidere di adottare delle formule confezionate nello specifico per alcuni fluidi, personalizzate sul loro comportamento, poiché un calcolo di tipo tradizionale non sarebbe sufficientemente accurato.
Nel caso della viscosità, si fa largo uso nelle librerie esistenti di modelli numerici o alternativi di calcolo a causa della scarsa presenza di formule analitiche attendibili che rappresentino il vasto insieme dei fluidi analizzabili. Nel corso del progetto sono stati studiati alcuni di questi metodi per estendere il range dei fluidi trattabili e mimare il comportamento delle librerie disponibili sul mercato:

- *Modelli hardcoded*: sono modelli che brutalmente implementano formule analitiche completamente personalizzate per il fluido all'interno del codice. Nel caso della viscosità, sei fluidi tra quelli disponibili utilizzano questo tipo di calcolo;
- *Modello ECS*: è l'acronimo di Extended Corresponding States e trae spunto dai lavori di Bell, I. H., Wronski, J., Quoilin, S. e Lemort, V. [19] [28]. Il principio si

basa sulla simulazione del comportamento del fluido sulla base di un fluido di riferimento. Spesso si utilizza il fluido refrigerante R-134a come modello perché è uno dei più largamente utilizzati e su chi si dispone di numerosi dati sperimentali [7];

- *Modello di Chung*: prende il nome dall'omonimo creatore che ne cita il funzionamento all'interno del'articolo *Generalized multiparameter correlation for nonpolar and polar fluid transport properties* [4];
- *Modello Rho-Sr*: sfrutta ugualmente il principio del fluido di riferimento, ma manipolando i dati a partire dal calcolo delle densità e innestando anche porzioni appartenenti al modello classico come la *Kinetic theory* [15].

2.4.2 La Conduttività termica

Pur rappresentando una grandezza totalmente differente dalla viscosità, la conduttività termica, nel calcolo termodinamico, ha grandissime somiglianze con la prima, soprattutto per le meccaniche di calcolo e la struttura della formula che viene adottata all'interno delle librerie software. A tal proposito, data proprio la grande somiglianza di funzionamento, si metteranno in luce solamente le differenze salienti.
La conduttività termica (λ), o conducibilità, misura l'attitudine di una sostanza a trasmettere calore attraverso la conduzione termica e risulta strettamente legata alla natura del materiale [35]. Anche in questo caso, come con la viscosità, la formula classica (relativa al *classic model*) è composta da quattro termini principali che registrano i vari apporti che concorrono alla misura della quantità [15] (Formula 2.32).

$$\lambda(T,\rho) = \lambda^{*}(T) + \lambda^{init}(T) + \lambda^{r}(T,\rho) + \lambda^{crit}(T,\rho) \tag{2.32}$$

Il primi due addendi sono detti *Dilute Term* $\lambda^{*}(T)$ e *Initial density Term* λ^{init}, ed entrambi utilizzano la temperatura (T) come parametro. I secondi, invece, vengono denominati *Residual Term* $\lambda^{r}(T,\rho)$ e *Critical Term* $\lambda^{crit}(T,\rho)$ ed oltre alla temperatura necessitano anche della densità (ρ) per il calcolo.
La grande differenza con la viscosità, però, sta nella predilezione delle librerie nell'utilizzo di metodi alternativi di calcolo rispetto alla formula tradizionale. In tutti i casi analizzati, si preferisce, ove possibile, utilizzare metodi numerici come l'*ECS Model* o modelli *hardcoded* perché risultano più attendibili per gli utilizzi che la conduttività termica deve svolgere. Nel processo di decisione del metodo utilizzabile, dunque, si procederà a controllare se il metodo ECS è disponibile per il dato fluido, poi si passerà al controllo dei metodi hardcoded. In caso di risposta negativa dei due, e solo in questo caso, si utilizzerà il metodo classico di calcolo [15].
Nelle Tabelle 2.5, 2.6, 2.7 verranno elencate le varie forme che i termini principali possono assumere nel calcolo della conduttività seguendo il *classic model* [15] [28].

Dilute Term $\lambda^{*}(T)$

Le varie forme che il *Dilute Term* può assumere per il calcolo della conduttività sono rappresentate nella Tabella 2.5.
Nelle formule, oltre alle variabili già note come T_r (*Reducing temperature*) e η_0 (*Dilute Term* relativo alla viscosità), vi sono anche numerosi coefficienti ed esponenti $(A_i, n_i, m_i, t_i, l_i)$ che vengono forniti dalle librerie al momento del calcolo.
Come nel caso della viscosità, anche qui il numero degli indici non è possibile stabilirlo a priori, ma varierà in funzione del fluido da interrogare.
Da notare come in Tabella 2.5 sia presente l'opzione *Nullo*, poichè per alcuni fluidi il quantitativo che il *Dilute Term* conserva è 0.

Forma	Calcolo
Ratio of Polynomials	$\sum_{i=1}^{I_A} A_i T_r^{n_i} / \sum_{i=1}^{I_B} B_i T_r^{m_i}$
Eta0 and Polynomials	$A_0 \eta_0(T) + \sum_{i=1}^{I_{\eta 0}} A_i \tau^{t_i}$
C02 Hardcoded	$[\frac{\tau^{-0.5}}{l_1\tau + l_2\tau^2 + l_3\tau^3}]/1000$
Ethane Hardcoded	$0.276505 \cdot 10^{-3}(\eta_0 \cdot T \cdot 10^6) \cdot 3.75 - f(\tau^2 \alpha^0_{\tau\tau} + 1.5)$
Nullo	0

Tabella 2.5: Le varie forme del Dilute Term nel calcolo della conduttività [15]

Initial density Term $\lambda^{init}(T)$

All'interno dei casi analizzati il termine risulta sempre nullo o viene sfruttato come contenitore per agevolare altri calcoli come l'ECS.

Residual Term $\lambda^r(T, \rho)$

Il terzo addendo della formula generale 2.32 è rappresentato dal *Residual Term* che può assumere solamente due forme differenti, come indicato dalla Tabella 2.6.

Forma	Calcolo
Polynomial	$\sum_{i=1}^{I_P} B_i \tau^{t_i} \delta^{d_i}$
Polynomial and exponential	$\sum_{i=1}^{I_{PE}} A_i \tau^{t_i} \delta^{d_i} e^{-\gamma_i \delta^{l_i}}$

Tabella 2.6: Le varie forme del Residual Term nel calcolo della conduttività [15]

Lo schema è tipico del calcolo residuale, con un termine polinomiale e un altro contaminato da un esponenziale. Tutte le variabili letterali, $B_i, t_i, d_i, A_i, \gamma_i, l_i$ rappresentano, come sempre, variabili esterne provenienti dai vari paper sperimentali.

Critical Term

L'ultimo addendo contiene il contributo del *Critical Term* all'interno del calcolo classico della conduttività. Anche in questo caso, l'apporto potrebbe essere nullo oppure rientrare in un modesto paniere di forme, come mostrato nella Tabella 2.7.

Forma	Calcolo
Olchowy-Sengers	$\rho_{molar} c_p R_0 k \frac{T}{6\pi\mu\zeta}(\tilde{\omega} - \tilde{\omega_0})$
Hardcoded	utilizzato nel calcolo di R123, Ammoniaca e CO2
Nullo	0

Tabella 2.7: Le varie forme del Critical Term nel calcolo della conduttività [15]

In particolare il calcolo della forma *Olchowy-Sengers* necessita di alcuni calcoli intermedi come mostrato dalla Formula 2.33.

$$\begin{aligned}
\tilde{\omega} &= \frac{2}{\pi}\left[\left(\frac{c_p - c_v}{c_p}\right)\arctan(q_d\zeta) + \frac{c_v}{c_p}q_d\zeta\right] \\
\tilde{\omega_0} &= \frac{2}{\pi}\left[1 - \exp\left(-\frac{1}{(q_d\zeta)^{-1} + (q_d\zeta\rho_c/\rho)^2/3}\right)\right] \\
\zeta &= \zeta_0\left(\frac{p_c\rho}{\Gamma\rho_c^2}\right)^{\nu/\gamma}\left[\left.\frac{\partial\rho(T,\rho)}{\partial p}\right|_T - \frac{T_R}{T}\left.\frac{\partial\rho(T_R,\rho)}{\partial p}\right|_T\right]^{\nu/\gamma}
\end{aligned} \tag{2.33}$$

con

k = costante di Boltzman

R_0 = universal amplitude

ν, γ = esponenti critici

T_R = temperatura di riferimento

Capitolo 3

Fluidprops

In questo capitolo verrà analizzato il processo di sviluppo del prodotto software Fluidprops*, parte centrale dell'attività di stage e obiettivo primario fissato dall'azienda. Si tratta di una libreria software realizzata ex novo in linguaggio Rust che svolge compiti di calcolo di proprietà termodinamiche dei fluidi. Lo scopo a lungo termine è il suo inserimento all'interno dell'applicativo Novaeka per sostituire le oramai note librerie presenti sul mercato. Questo non solo per coerenza di linguaggio tra applicativo e libreria, ma anche per favorire una migliore trasparenza e organizzazione del substrato di calcolo contenuto nel prodotto. Il lavoro è partito da uno studio approfondito delle opzioni attualmente utilizzate ed è proseguito nella progettazione e scrittura del software in oggetto.*

3.1 Refprop e Coolprop: lo stato dell'arte

Coolprop e *Refprop* sono le uniche alternative presenti attualmente sul mercato per il calcolo di proprietà dei fluidi, specialmente fluidi refrigeranti, come anche i loro nomi suggeriscono [10] [30]. Sono entrambi largamente utilizzati in ambito industriale e rappresentano le fonti più attendibili per la termodinamica dei fluidi.
Refprop, acronimo di Reference fluid Properties, è un software scritto in linguaggio Fortran e distribuito attraverso lo *Standard Reference Data program* del NIST, l'istituto nazionale americano per gli standard. Sin dalla sua prima release nel 1989, ha avuto l'obiettivo di rendere accessibili, attraverso l'implementazione di formule e modelli computazionali, il calcolo di proprietà termodinamiche e proprietà di trasporto per gli scopi più disparati, dalla ricerca all'industria. Il nucleo iniziale permetteva solo calcoli relativi a 15 fluidi refrigeranti, ma si è espanso negli anni fino a comprendere al momento una grande varietà di fluidi, miscele e unità di misura utilizzabili. [30]
La sua controparte open-source, nonché concorrente al momento, è *Coolprop*, una libreria di calcolo primariamente ideata da Ian Bell, Jorrit Wronski, Sylvain Quoilin e Vincent Lemort e scritta in linguaggio C++. Sommariamente questa libreria svolge gli stessi scopi di *Refprop* e si adatta alla maggior parte di sistemi presenti sul mercato. È stato fin dall'inizio un progetto collettivo, partito dagli sviluppatori, ma sempre a disposizione degli utenti per apportare modifiche e miglioramenti [10]. Oltre alla sua gratuità, gode anche del vantaggio di avere una grande varietà di wrapper per il suo utilizzo all'interno di ambienti di vario genere, addirittura proponendo una sua versione ad estensione delle potenzialità di Microsoft Excel [6].
Durante d'attività di stage, è stato prediletto l'utilizzo di *Coolprop* come punto di

riferimento e base software di partenza, poiché la natura di libreria si confaceva meglio agli scopi dell'azienda e permetteva la sua futura integrazione nell'applicativo Novaeka. Ha giocato un peso notevole nella scelta anche la maggiore familiarità con il linguaggio C++ rispetto a Fortran, permettendo di navigare più agevolmente il codice e leggerlo con maggiore chiarezza. Lo studio del caso, infatti, è proprio partito dall'orientamento all'interno del codice, l'individuazione della sua struttura e gerarchia, per poi raffrontare la coerenza di tipi e metodi esistenti nel codice con le definizioni teoriche riscontrate nei riferimenti bibliografici. La libreria *Fluidprops* si pone l'obiettivo di diventare l'alternativa embrionale a *Coolprop* non solo per l'imitazione delle funzionalità e princìpi, ma soprattutto per i miglioramenti a livello architetturale e nella struttura di tipi che garantisce maggiore sicurezza dei dati e limita gli errori di manipolazione, volontari o accidentali.

3.2 Analisi dei requisiti e funzioanlità

Durante la prima settimana erano già emerse alcune necessità e funzionalità specifiche che il prodotto dovesse avere. Le maggiori urgenze erano già presenti in fase di presentazione del progetto e, come inserito anche nel piano di lavoro, era chiaro il contesto entro cui la libreria dovesse lavorare. È stato solo dopo l'analisi delle librerie di riferimento, però, che si è potuto procedere con la schematizzazione dei casi d'uso e la definizione dei requisiti che il prodotto dovesse soddisfare. Lo studio approfondito della letteratura scientifica a disposizione, inoltre, si è reso fondamentale per sezionare i macro-casi e suddividere il lavoro in piccoli task operativi come specificato nel Capitolo 1.
La funzione *PropSI* all'interno di *Coolprop* rappresenta l'interfaccia che l'utente utilizza per interrogare la libreria come nell'esempio mostrato dalla Figura 3.1. Prende come parametri una serie di *char* e valori *double* per poi restituire come risultato il valore della proprietà richiesta sul fluido in oggetto, secondo i parametri indicati.

```
double PropsSI(const char *Outputs, const char *Name1, double Prop1, const char *Name2, double Prop2, const char *Ref);

// esempio di utilizzo

double ris = PropsSI('T', 'P', 101325, 'Q', 0, 'Water');
```

Figura 3.1: Firma della funzione PropSI presente all'interno di Coolprop

3.2.1 Casi d'uso

Gli utilizzi di questo prodotto sono estremamente essenziali proprio perché il suo scopo è incentrato sul calcolo delle proprietà termodinamiche. L'attore del sistema è l'utente che interroga la libreria o un software esterno, come nel caso dell'applicativo Novaeka, che richiama la libreria per eseguire i calcoli necessari.
Individuiamo, dunque, i seguenti casi d'uso (Figura 3.2):

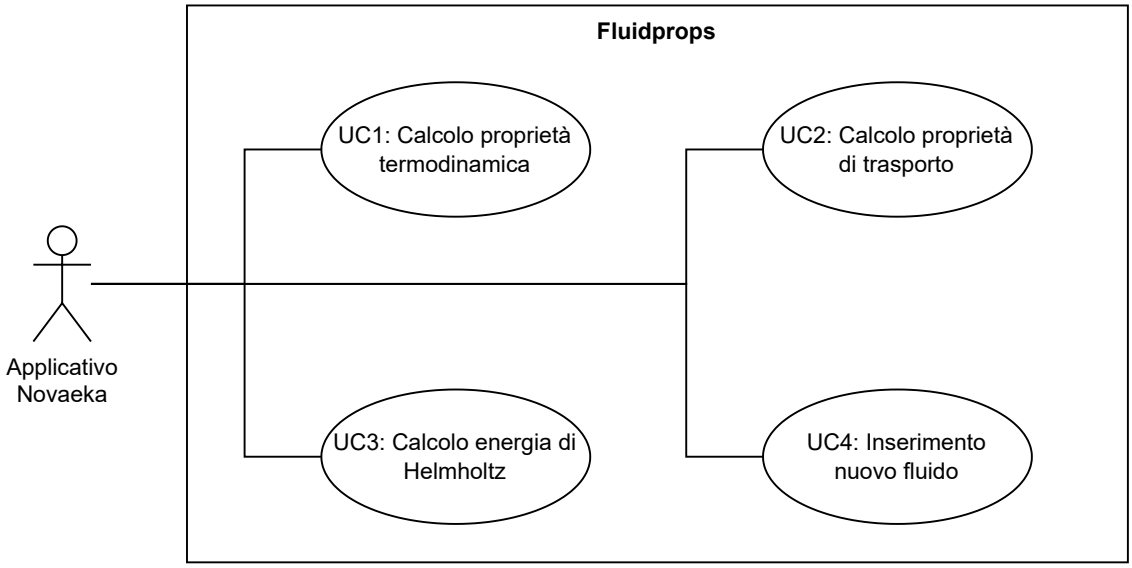


Figura 3.2: Casi d'uso della libreria

UC1: Calcolo di una proprietà termodinamica secondo determinati valori in input

- Attori: Applicativo Novaeka;
- Precondizione: Aver calcolato l'energia di Helmholtz per il fluido in esame;
- Postcondizione: Il risultato ritornato è il valore della proprietà prescelta alle condizioni di input;
- Scenario principale: L'applicativo necessita del calcolo di una particolare proprietà, fornisce i dati necessari e la calcola attraverso la libreria.

UC2: Calcolo di una proprietà di transport (Viscosità o conduttività)

- Attori: Applicativo Novaeka;
- Precondizione: Aver calcolato l'energia di Helmholtz per il fluido in esame;;
- Postcondizione: La libreria ritorna il valore di viscosità o conduttività termica per i dati inseriti;
- Scenario principale: Si necessita della misurazione di una certa proprietà di transport, si forniscono le informazioni e si ottiene il risultato sperato.

UC3: Calcolo dell'energia di Helmholtz

- Attori: Applicativo Novaeka;
- Precondizione: Essere all'interno dell'applicativo e aver selezionato il fluido in esame, con relativi input δ e τ;

- Postcondizione: *Fluidprops* ritorna come risultato il valore dell'energia di Helmholtz;
- Scenario principale: L'applicativo necessita dell'energia di Helmholtz per un fluido ed interroga la libreria.

UC4: Inserimento di dati riguardanti un nuovo fluido non presente nel database

- Attori: Applicativo Novaeka;
- Precondizione: Si è all'interno dell'applicativo e la libreria è sprovvista dei dati sul fluido di interesse;
- Postcondizione: Si completa l'inserimento dei dati per il particolare fluido e, ora, si dispone di un nuovo fluido in memoria;
- Scenario principale: L'applicativo non trova il fluido selezionato tra quelli disponibili e si procede alla creazione di un nuovo record.

3.2.2 Requisiti

I requisiti che il prodotto software necessita di soddisfare, come riassunti dalla Tabella 3.1, non erano tutti completamente chiari fin dall'inizio del progetto. Nella prima fase ci si è concentrati sullo studio teorico e del materiale a disposizione per poi delineare le direttrici da seguire nel prodotto. In fase conclusiva, infine, una volta concluso il calcolo delle proprietà come da piano, si è deciso di operare una valutazione del tempo rimanente e si è deciso di estendere l'elenco dei requisiti con i punti GR12, GR13 e GR15, poiché si era stimato il loro soddisfacimento entro la data conclusiva delle attività.

Utilizzeremo la seguente notazione:
GR = *General requirement*, requisiti generali di funzionamento;
QR= *Quality requirement*, requisiti riguardanti la qualità del codice prodotto.

Codice	Priorità	Requisito
GR01	Obbligatorio	Calcolo dell'energia di Helmholtz dati δ e τ
GR02	Obbligatorio	Calcolo della pressione attraverso le EOS
GR03	Obbligatorio	Calcolo dell'entropia attraverso le EOS
GR04	Obbligatorio	Calcolo dell'energia interna attraverso le EOS
GR05	Obbligatorio	Calcolo della capacità termica isocora attraverso le EOS
GR06	Obbligatorio	Calcolo dell'entalpia attraverso le EOS
GR07	Obbligatorio	Calcolo della capacità termica isobara attraverso le EOS
GR08	Obbligatorio	Calcolo dell'energia di Gibbs attraverso le EOS
GR09	Obbligatorio	Calcolo della velocità del suono attraverso le EOS
GR10	Obbligatorio	Calcolo del fattore di comprimibilità attraverso le EOS
GR11	Obbligatorio	Calcolo delle proprietà nella regione bi-fase
GR12	Desiderabile	Calcolo della viscosità di un fluido
GR13	Desiderabile	Calcolo della conducibilità termica di un fluido
GR14	Obbligatorio	Popolamento del set dei fluidi analizzabili
GR15	Desiderabile	Rappresentazione grafica dell'accuratezza del calcolo
QR01	Obbligatorio	I risultati devono combaciare con le librerie di riferimento
QR02	Obbligatorio	La struttura del codice deve avere un'organizzazione chiara e funzionale

Tabella 3.1: Tabella dei requisiti della libreria Fluidprops

3.2.3 I fluidi selezionati

La libreria *Coolprop* mette a disposizione dell'utente un database interno, scritto in linguaggio Json, ricco di informazioni e dati riguardanti centoventidue fluidi refrigeranti e non; se ne può osservare un estratto alla Figura 3.3. Questi dati sono la summa di tutte le informazioni necessarie per operare il calcolo delle proprietà e vanno da banali caratteristiche fisiche del fluido, come la massa molare, ad array di coefficienti per alimentare le formule interne.


```
"EOS": [
  {
    "BibTeX_CP0": "",
    "BibTeX_EOS": "TillnerRoth-JPCRD-1994",
    "STATES": {
      "hs_anchor": {
        "T": 411.598,
        "T_units": "K",
        "hmolar": 45468.086192270304,
        "hmolar_units": "J/mol",
        "p": 6858750.808053112,
        "p_units": "Pa",
        "rhomolar": 4480.9471539000015,
        "rhomolar_units": "mol/m^3",
        "smolar": 172.44139866136513,
        "smolar_units": "J/mol/K"
      },
      "reducing": {
        "T": 374.18,
        "T_units": "K",
        "hmolar": 39803.894110114015,
        "hmolar_units": "J/mol",
        "p": 4059280,
        "p_units": "Pa",
        "rhomolar": 4978.830171000001,
        "rhomolar_units": "mol/m^3",
        "smolar": 159.51273279856932,
```


Figura 3.3: Estratto del file Json relativo ai dati del fluido R-134a

Dato che non sarebbe stato possibile implementare tutti i fluidi di *Coolprop* all'interno di *Fluidprops*, si è deciso di optare per un range di fluidi che coprissero tutte le casistiche di calcolo operabili dalla libreria, in modo da poterla testare in tutte le sue componenti. Il risultato è un insieme di diciotto fluidi che rappresentano tutte le combinazioni di formule calcolabili dalla libreria e ne danno così una panoramica generale di funzionamento.
I dati riguardanti le composizioni delle varie formule sono tratti principalmente dalla libreria di *Coolprop* [9], ma sono stati tutti sottoposti ad un controllo incrociato con i paper a cui la stessa libreria *Coolprop* fa riferimento [8]. Nelle Tabelle 3.2 e 3.3 sono indicati i fluidi prescelti e le loro equazioni di stato secondo una rappresentazione

tabellare. La presenza di un segno di spunta (✓) significa che l'equazione di stato relativa a quel fluido presenta all'interno il termine contrassegnato, la sua assenza, di conseguenza, rappresenta la mancanza di quel termine all'interno dell'equazione.
Lo stesso ragionamento è stato applicato anche alla rappresentazione tabellare delle varie formule per il calcolo di viscosità (Tabelle 3.4, 3.5 e 3.6) e conduttività termica (Tabelle 3.7, 3.8 e 3.9).
Nella Tabella 3.2, la prima della serie, ogni nome di fluido sarà, inoltre, accompagnato dal riferimento all'articolo dal quale le EOS sono state ricavate.

Fluido	Lead	EEO	Logtau	Polynomial	Planck-Einstein	Cp0PolyT
R134a [43]	✓		✓	✓		
Water [49]	✓		✓		✓	
R125 [22]	✓		✓	✓	✓	
Ammonia [16]	✓		✓		✓	
Methanol [32]	✓	✓	✓	✓	✓	
Acetone [25]	✓	✓	✓		✓	
Fluorine [12]	✓		✓	✓	✓	
HydrogenSulfide [25]	✓		✓	✓	✓	
Ethanol [36]	✓	✓	✓		✓	
CycloHexane [51]	✓		✓		✓	
nPentane [42]	✓		✓		✓	
Ethane [3]	✓	✓	✓		✓	
R13 [33]	✓	✓	✓			✓
nPropane [23]	✓		✓		✓	
Isopentane [25]	✓		✓		✓	
R152A [31]	✓	✓	✓			✓
R32 [44]	✓		✓		✓	
Air [24]	✓	✓	✓	✓	✓	

Tabella 3.2: Tabella dei fluidi selezionati e delle loro composizioni per l'Ideal Term

Fluido	Power	Exp	Lemmon	Gauss	GERG	NonAnalytic	SAFTA	GaoB
R134a	✓							
Water	✓			✓		✓		
R125			✓					
Ammonia	✓			✓				✓
Methanol	✓						✓	
Acetone	✓							
Fluorine	✓	✓						
HydrogenSulfide	✓							
Ethanol	✓			✓				
CycloHexane	✓			✓				
nPentane	✓			✓				
Ethane	✓			✓				
R13	✓	✓						
nPropane	✓			✓				
Isopentane	✓							
R152A	✓							
R32	✓							
Air	✓							

Tabella 3.3: Tabella dei fluidi selezionati e delle loro composizioni per il Residual Term [8]

Fluido	C. Integral	C. power T	Kinetic T.	Powers of T	Powers of Tr	Hardcoded
R134a	✓					
Water						
R✓25			✓			
Ammonia	✓					
Methanol						
Acetone						
Fluorine						
HydrogenSulfide		✓				
Ethanol			✓	✓		
CycloHexane						✓
nPentane					✓	
Ethane						✓
R13						
nPropane	✓					
Isopentane						
R152A						
R32						
Air	✓					

Tabella 3.4: Fluidi e loro composizioni per il Dilute Term nella viscosità [8]

Fluido	Rainwater-Friend	Empirical	Batschinski-Hildebrand	Friction T.	Hardcoded
R134a	✓		✓		
Water					
R125	✓		✓		
Ammonia	✓		✓		
Methanol					
Acetone					
Fluorine					
HydrogenSulfide	✓			✓	
Ethanol	✓		✓		
CycloHexane		✓	✓		
nPentane				✓	
Ethane					✓
R13					
nPropane	✓		✓		
Isopentane					
R152A					
R32					
Air			✓		

Tabella 3.5: Fluidi e loro composizioni per l'Initial e l'Higher-order terms nella viscosità [8]

Fluido	Hardcoded	No viscosity	ECS	Chung	RhoSr
R134a					
Water	✓				
R125					
Ammonia					
Methanol	✓				
Acetone		✓			
Fluorine		✓			
HydrogenSulfide					
Ethanol					
CycloHexane					
nPentane					
Ethane					
R13			✓		
nPropane					
Isopentane				✓	
R152A					✓
R32			✓		
Air					

Tabella 3.6: Fluidi e loro utilizzo di metodi alternativi per il calcolo della viscosità [8]

Fluido	Ratio of Poly	Eta0 and Poly	Hardcoded	Nullo
R134a	✓			
Water				
R125	✓			
Ammonia	✓			
Methanol	✓			
Acetone				
Fluorine				
HydrogenSulfide				
Ethanol	✓			
CycloHexane				
nPentane	✓			
Ethane			✓	
R13				
nPropane	✓			
Isopentane	✓			
R152A	✓			
R32				
Air		✓		

Tabella 3.7: Fluidi e loro composizioni per il Dilute Term nella conduttività [8]

Fluido	Polynomial	Poly and Exp	Olchowy-Sengers	Hardcoded	Nullo
R134a	✓		✓		
Water					
R125	✓		✓		
Ammonia	✓			✓	
Methanol	✓		✓		
Acetone					
Fluorine					
HydrogenSulfide					
Ethanol	✓		✓		
CycloHexane					
nPentane	✓		✓		
Ethane	✓		✓		
R13					
nPropane	✓		✓		
Isopentane	✓		✓		
R152A	✓		✓		
R32					
Air		✓			

Tabella 3.8: Fluidi e loro composizioni per i Residual e Critical terms nella conduttività [8]

Fluido	Hardcoded	No conductivity	ECS
R134a			
Water	✓		
R125			
Ammonia			
Methanol			
Acetone		✓	
Fluorine		✓	
HydrogenSulfide		✓	
Ethanol			
CycloHexane		✓	
nPentane			
Ethane			
R13			✓
nPropane			
Isopentane			
R152A			
R32			✓
Air			

Tabella 3.9: Fluidi e loro loro utilizzo di metodi alternativi per il calcolo della conduttività [8]

3.3 La scelta di Rust

Può sorgere spontanea a questo punto la domanda per cui si sia adottato un linguaggio diverso, e in particolare il linguaggio Rust, per la codifica di *Fluidprops* rispetto a quelli utilizzati nelle librerie di riferimento; e quali siano stati i vantaggi effettivi nell'attuazione di questa scelta. Questo linguaggio non rappresentava l'unica alternativa per raggiungere il risultato sperato poiché esistono librerie codificate in altri linguaggi che svolgono lo stesso compito, di conseguenza, l'equilibrio della bilancia è stato influenzato sia da scelte di preferenza aziendale sia da caratteristiche tecniche che vedono nel linguaggio Rust una scelta vincente per costruire questo tipo di architettura software.
Come già accennato, l'applicativo Novaeka si basa quasi interamente sul linguaggio Rust, quindi per la scrittura di una libreria che si inserisse perfettamente in questo contesto l'appetibilità della scelta è stata abbastanza chiara. A livello tecnico, però, vi sono alcune caratteristiche proprie del linguaggio che hanno facilitato la scrittura di un codice estremamente sicuro e con il minor rischio di manipolazione possibile nella gestione della memoria [41]:

- Sintassi simile ai linguaggi C: le graffe racchiudono i blocchi, i tipi primitivi sono analoghi anche se con nomenclature completamente diverse. Questo ha consentito maggiore leggibilità e velocità di apprendimento del linguaggio;

- Non c'è garbage collector: consente una deallocazione deterministica della memoria e la memoria allocata dinamicamente viene deallocata automaticamente grazie al compilatore che inserisce le opportune istruzioni quando le variabili escono dal proprio ambito di visibilità, ovvero quando il proprietario ("owner") della variabile esce fuori dal suo scope;

- Vieta costrutti dal comportamento indefinito: non sono possibili ad esempio la lettura di variabili non ancora scritte, l'accesso agli array oltre i limiti, la dereferenziazione di puntatori nulli o non validi, l'uso di iteratori invalidanti;

- Inferenza di tipo: ogni volta che si dichiara una variabile locale, è facoltativo dichiararne il tipo. Quest'ultimo viene desunto dall'espressione di inizializzazione o dalle successive istruzioni di assegnamento a tale variabile;

- Inizializzazione variabili: non è obbligatorio inizializzare le variabili, ma viene generato un errore di compilazione se una variabile viene usata prima di ricevere un valore, o se le vengono assegnati valori di tipi diversi;

- Le classi: dette Struct possono essere parametrizzate da tipi in modo analogo ai template del C++. Tuttavia questi tipi parametrici devono essere vincolati a dei trait per poterne usare i metodi o gli operatori.

Vi sono inoltre delle differenze sostanziali rispetto ad altri linguaggi *object-oriented* che si traducono in determinate scelte progettuali [41]:

- Ownership: esistono i concetti di proprietà (“ownership”) e di "prestito" ("borrowing") degli oggetti per decidere quali istruzioni possono leggere un oggetto e quali lo possono scrivere. Ogni valore ha un owner e questo è unico, quindi si evita la sovrapposizione del possesso e la contesa della variabile tra più proprietari;

- Side-effect: le variabili nel loro passaggio a metodo vengono automaticamente considerate dotate di *const*. Per poter operare modifiche sui parametri passati in input è necessario dotarli della keyword *mut*;

- Lifetime: si consente, o richiede talvolta, che si specifichi il "tempo di vita" ("lifetime") di oggetti gestiti tramite puntatore, per decidere quando dovranno essere distrutti;

- Exception: non si usa il concetto di eccezione per segnalare o comunicare errori a runtime;

- Ereditarietà: non viene utilizzata di implementazione, ovvero ogni classe può ereditare solo da interfacce, non da altre classi;

- Metaclassi: le varie *struct*, analogo delle classi in C++, non sono oggetti e non esistono metaclassi;

- Overload delle funzioni: devono strettamente avere nomi diversi all’interno dello stesso ambito.

3.4 Architettura della libreria

Dato che è di una libreria che si tratta, è naturale immaginare che tutto ruoti attorno al file primario *lib.rs*, contenente l'elenco di tutti moduli necessari a svolgere i suoi compiti di calcolo. In Rust, ogni modulo è raggruppato graficamente entro una cartella che contiene, oltre ai file di cui è composta, anche un file speciale denominato sempre *mod.rs* che rappresenta il punto di accesso al modulo nonché fulcro di tutte le attività. Può contenere dichiarazioni relative all'utilizzo di altri moduli o packages all'interno dello stesso, o anche dichiarazioni di *struct* o funzioni. Il suo scopo è tenere coeso il modulo e dichiararne esplicitamente ed univocamente il contenuto o utilizzo.
Nel diagramma ad albero espresso nella Figura 3.4, si nota come il codice sorgente sia composto da numerosi moduli e altri file non raggruppati in modulo. Oltre al file *lib.rs* già incontrato, vi sono altri file che svolgono funzioni di calcolo e altri ancora che lo supportano con l'implementazione di metodi numerici per quelli che ne fanno utilizzo.

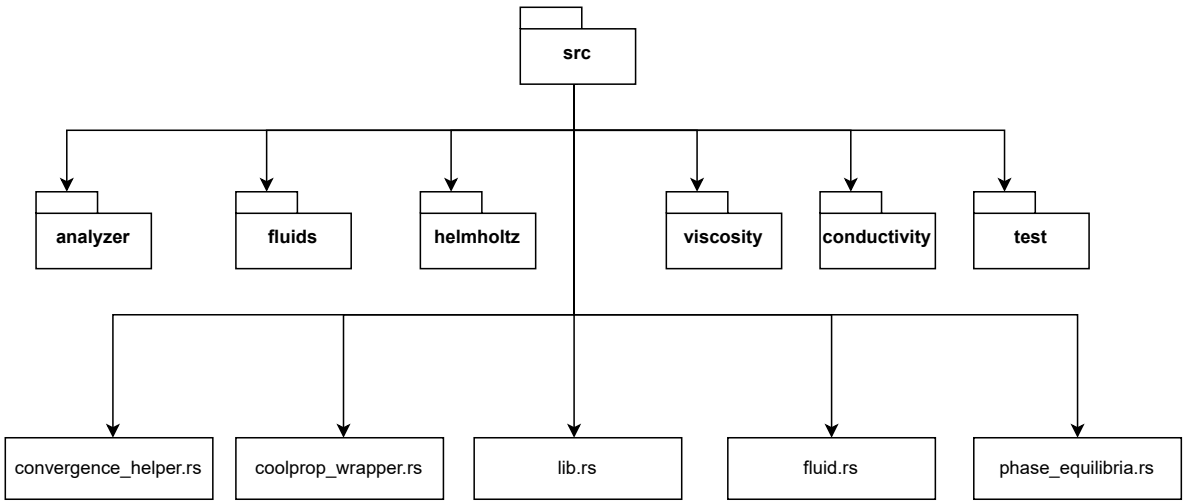


Figura 3.4: Albero del modulo *src* rappresentate lo scheletro principale della libreria

3.4.1 Modulo sorgente

Verrà elencato di seguito il contenuto del *crate Fluidprops*, raggruppando le varie componenti per moduli, ove possibile, e descrivendone i contributi per il funzionamento del prodotto software.

lib.rs

È il file principale della libreria che raccoglie tutti i moduli che la compongono, o quelli di cui ha bisogno per svolgere la sua funzione. In Figura 3.5 si può notare tutto il contenuto del file in oggetto.

```rust
#[cfg(test)]
mod analyzer;
mod conductivity;
mod convergence_helper;
mod coolprop_wrapper;
pub mod fluid;
pub mod fluids;
mod helmholtz;
mod phase_equilibria;
#[cfg(test)]
mod test;
#[cfg(test)]
mod test_two_phase;
#[cfg(test)]
mod test_utils;
mod viscosity;
```

Figura 3.5: Codice della libreria principale lib.rs

fluid.rs

È il cuore delle proprietà di un fluido e del loro calcolo. Al suo interno viene dichiarata ed implementata la struttura dei tipi (Figura 3.6) che gestiranno il fluido e la maggior parte dei suoi dati. Contiene la *struct* che definisce gli attributi e le caratteristiche di un fluido nonché alcuni metodi propri.

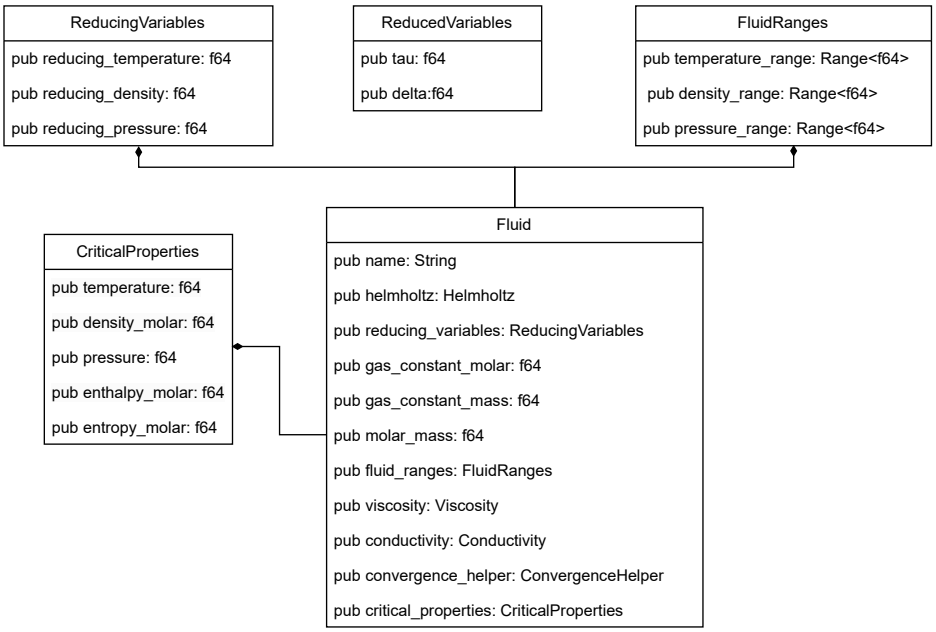


Figura 3.6: Diagramma dei tipi contenuti del file fluid.rs

Ogni fluido è un contenitore di tutte le sue caratteristiche fisiche e di tutte le *struct* necessarie per il calcolo delle sue proprietà. Ogni funzione incaricata del calcolo delle proprietà ha una sua versione per il calcolo in fase omogenea o monofase e una versione aggiuntiva per il controllo e il calcolo dello stato bi-fase.

phase-equilibria.rs

Contiene tutti i moduli e le funzioni essenziali per calcolare i punti di equilibrio di fase o bi-fase. Utilizza *create* ausiliari come *sh-numerical-methods* già sviluppato dall'azienda per implementare il metodo di Newton e svolgere i calcoli non risolvibili in maniera analitica. Nel file è presente anche la dichiarazione della struct *PhaseEquilibria* progettata per rappresentare le variabili di equilibrio di fase, con la loro temperatura (T) comune e le due densità.

convergence-helper.rs

È utile nel calcolo polinomiale e nel calcolo di proprietà in maniera numerica. Contiene anche *struct* utili a raggiungere la convergenza per i metodi appartenenti alla famiglia dei metodi di Newton.

coolprop-wrapper.rs

Contiene il wrapper che incapsula la libreria *Coolprop* all'interno della libreria *Fluidprop*. Tutto ciò è stato necessario in fase di testing per riuscire ad avere un confronto utile tra le due librerie sia in termini di attendibilità di calcolo che di performance.
Nella sezione dei test si troveranno dei moduli di test appositi che richiamano le due librerie e ne confrontano i risultati.

3.4.2 Modulo *fluids*

Questo modulo costituisce un'ampia raccolta di tutti i fluidi che sono stati presi in esame e testati durante l'attività di stage. Ogni file, nominato con il nome del fluido a cui si riferisce, contiene una funzione pubblica *get-nomeFluido* (Figura 3.7) che ritorna un oggetto di tipo *Fluid* e che, dunque, funge da costruttore per l'oggetto. La funzione fabbrica i fluidi con i vari dati di cui necessitano e li popola di tutti i coefficienti necessari al momento del calcolo.

```
Fluid::new(
    "Acetone".to_string(),
    Helmholtz {
        ideal: ideal_helmholtz,
        residual: residual_helmholtz,
    },
    ReducingVariables {
        reducing_temperature: 508.1,  //K
        reducing_density: 272.971958, //kg/m3
        reducing_pressure: 4700000.,  //Pa
    },
    8.314472,
    0.05807914, //kg/mol
    FluidRanges {
        temperature_range: Range {
            start: 200.,
            end: 550.,
        },
        density_range: Range {
            start: 100.,
            end: 400.,
        },
        pressure_range: Range {
            start: 0.1,
            end: 70_000_000.,
        },
    },
    viscosity,
    conductivity,
    convergence_helper,
    critical_properties,
)
```

Figura 3.7: Risultato di costruzione del fluido acetone attrverso il metodo di get

3.4.3 Modulo *helmholtz*

Questo modulo racchiude le varie formule per il calcolo dell'energia di Helmholtz e le varie derivazioni utili nel calcolo delle proprietà. Il file *mod.rs* al suo interno dichiara la *struct* principale *Helmholtz* che contiene come membri i due addendi dellle EOS espresse nella forma dell'energia di Helmholtz (Figura 3.8). Entrambi i membri sono contenitori di tutte le loro possibili forme e composizioni, che poi verranno impiegate nel calcolo dell'energia.

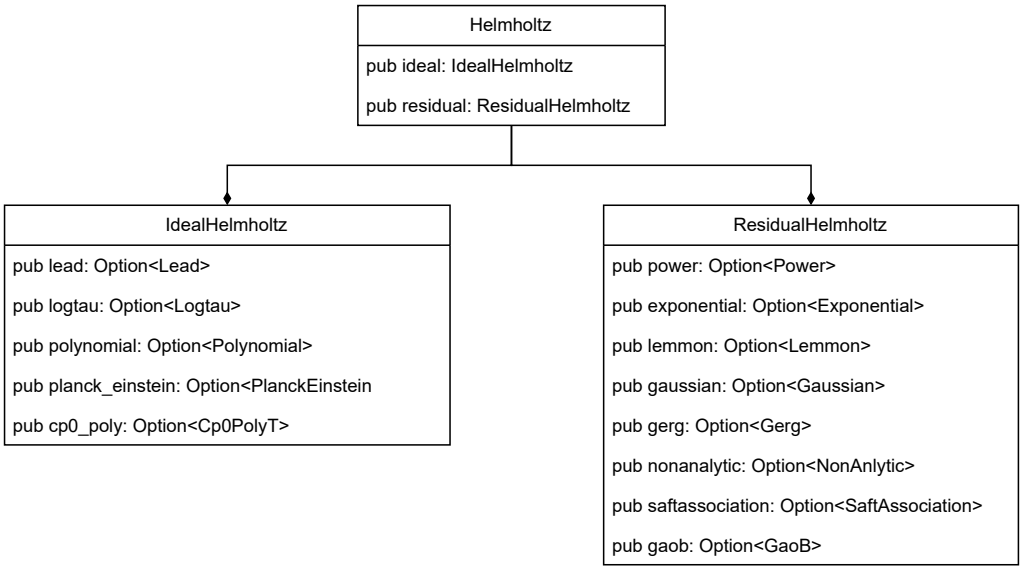


Figura 3.8: Diagramma dei tipi per il modulo helmholtz

Il calcolo della formula è descritto all'interno dei file *mod.rs* dei rispettivi moduli, così come le formule per il calcolo delle derivate parziali per ogni porzione di formula. Si è deciso di mantenere il calcolo separato per i due addendi poiché in molte altre parti del codice sarà necessario utilizzare i risultati (α^0 e α^r) singolarmente, dell'una e dell'altra parte, così come i risultati delle derivate. Le nomenclature utilizzate per le funzioni sono altamente descrittive, affinchè sia più facile individuare il loro scopo nel codice e migliorarne la leggibilità.
Ogni segmento di formula utilizza delle *struct* apposite per immagazzinarne i dati o coefficienti di calcolo che sono strettamente fluido-dipendenti. A volte si è reso necessario l'utilizzo di strutture nested quando l'organizzazione delle *struct* avrebbe risentito troppo del grande numero di membri interni.
Sia il modulo *ideal* che quello *residual* contengono un file di test interno denominato *test.rs* che si è reso utile per il testing delle singole porzioni di formula, quando era possibile raffrontare i calcoli con le librerie di riferimento. *Coolprop*, infatti, mette a disposizione delle funzioni apposite per il calcolo delle singole porzioni ed è stato utile sfruttare questa caratteristica per partizionare il lavoro e scremare subito eventuali errori di trascrizione o di interpretazione. Purtroppo non è stato possibile testare tutte le porzioni separatamente con *Coolprop* perchè la libreria non disponeva di alcune funzioni per il calcolo delle singole derivate, ma come si vedrà nell'ultimo capitolo, i risultati di comparazione sono stati ugualmente soddisfacenti.
I test prodotti confrontano il calcolo di *Coolprop* con quello di *Fluidprop* nei vari fluidi sia per punti singoli, quindi inserendo in input temperatura e densità desiderate, sia per punti multipli, generando un set di 10000 punti entro dei range fisici preimpostati.

3.4.4 Modulo *viscosity*

Come nei casi precedenti, è il file *mod.rs* a contenere le strutture e le informazioni essenziali per il calcolo, in questo caso, della viscosità. Il modulo si trova al di fuori del modulo *helmholtz* perchè certamente utilizza anch'esso l'energia di helmholtz e le sue derivate, ma la viscosità costituisce un insieme separato dalle normali proprietà termodinamiche, essendo appunto una proprietà di trasporto. Al contempo, la sua struttura non rispecchia quella delle altre proprietà e necessita di spazi e metodi organizzativi adeguati.

```rust
#[derive(Debug)]
pub enum Viscosity {
    Classic { model: ClassicModel },
    ECS { data: ECSModelData },
    Chung { data: ChungModelData },
    RhoSR { data: RhoSRModelData },
    Water,
    Methanol { viscosity_params: ViscosityParams },
    NotAvailable,
}
```

Figura 3.9: Codice dell'*enum* che rappresenta il calcolo della viscosità

In particolare, possiamo notare nella Figura 3.9 che la classe della viscosità non è più definita come *struct*, ma come *enum*. Mentre le *struct* permettono di raggruppare assieme dati correlati e di creare tipi inediti dalla loro unione, gli *enum* in Rust vengono utilizzati come delle particolari *struct* che permettono di memorizzare solo uno dei campi indicati.
Data la peculiare struttura della formula della viscosità, dove ogni addendo poteva assumere solo una forma tra quelle elencate, si è deciso di optare per questa scelta ed evitare l'ambiguità che una *struct*, al suo posto, avrebbe creato. Avere a disposizione un oggetto *Viscosity* sotto forma di *struct* avrebbe permesso all'utente o all'applicativo Novaeka di istanziare un calcolo di viscosità valicando qualsiasi regola che la formula prevedeva; paradossalmente anche creando formule che non corrispondevano a nessuna possibilità fisica.
È necessario sottolineare, inoltre, che il tipo *enum* in Rust non dispone delle stesse potenzialità che possiede in altri linguaggi di programmazione, tipo C++. La peculiarità è la capacità di creare un elenco di elementi all'interno del tipo contenitore, che possano effettivamente contenere dei valori oltre ad elencarli, potendo, di conseguenza, contenere altri tipi. In questo aspetto è molto simile alle *struct* di Rust o alle classi in C++, anche se, come già sottolineato, gli elementi non sono campi bensì opzioni di contenuto. Questo è uno degli esempi in cui il linguaggio Rust ha permesso di aumentare il livello di sicurezza del codice e di facilitarne l'utilizzo esterno.
Nell'architettura della classe troviamo tutte le casistiche che avevamo elencato nella Sezione 2.4, con una *struct* che definisce il calcolo con il modello classico (*ClassicModel*) e altre opzioni che implementano le tecniche alternative di calcolo (Figura 3.10); sempre comprendendo tra i casi l'impossibilità di effettuare il calcolo della viscosità per quel determinato fluido.

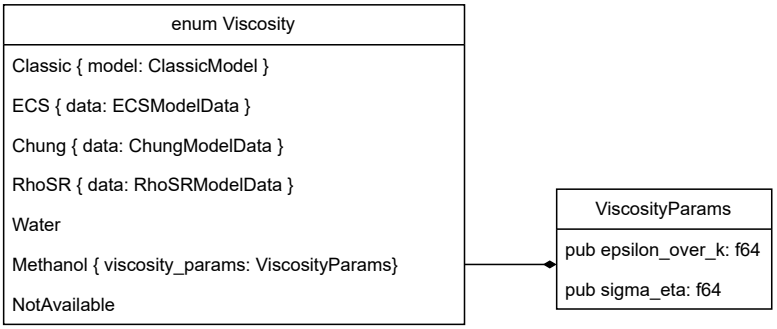


Figura 3.10: Diagramma dei tipi per il modulo *viscosity*

Anche il *ClassicModel* è costruito dalle sue componenti essenziali, i suoi addendi, che assumono varia forma in base alla formula del fluido corrispondente. Ogni componente è costituito da una *struct*, come mostrato in Figura 3.11, che ne contiene i dati oppure sfrutta un modello *hardcoded* di calcolo già implementati all'interno del file.

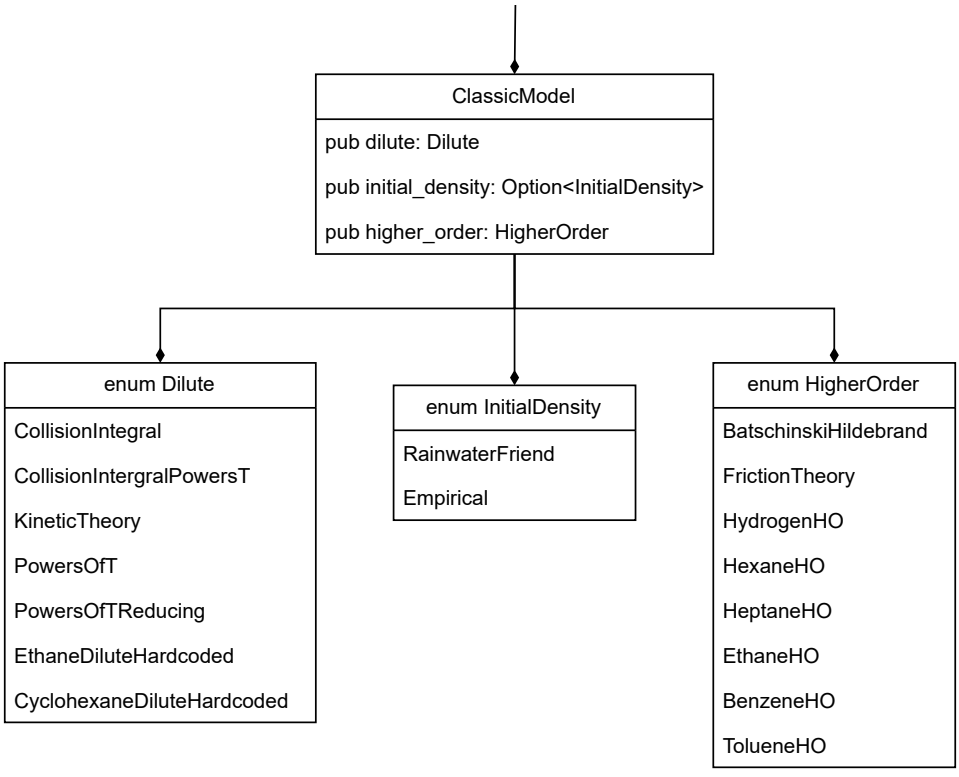


Figura 3.11: Diagramma dei tipi per il modulo *ClassicModel*

L'implementazione dell'*enum Viscosity* (Figura 3.12) prevede una funzione *calc* di calcolo che si occupa, in base al set di informazioni del fluido e sfruttando un'operazione Rust di *match*, di selezionare le varie formule che il fluido utilizza per il calcolo, computare tutti gli addendi e fornire in output il risultato desiderato.

```rust
impl Viscosity {
    pub fn calc(&self, fluid: &Fluid, temperature: f64, density: f64) -> Result<f64, String> {
        match self {
            Viscosity::Classic { model } => {
                model.calc(fluid, temperature, density / fluid.molar_mass, density)
            }

            Viscosity::NotAvailable => {
                Err("Viscosity Calculation not available for this fluid".to_string())
            }
            Viscosity::ECS { data } => calc_ecs(
                fluid,
                temperature,
                density / fluid.molar_mass,
                density,
                data,
            ),
            Viscosity::Chung { data } => calc_chung(temperature, density / fluid.molar_mass, data),
            Viscosity::RhoSR { data } => calc_rho_sr(
                fluid,
                temperature,
                density / fluid.molar_mass,
                density,
                data,
            ),
```

Figura 3.12: Accenno di implementazione dell'*enum Viscosity*

3.4.5 Modulo *conductivity*

Il modulo *conductivity* delle caratteristiche pressochè identiche alla controparte della viscosità.
Si ricorda che la libreria *Coolprop* forza l'utente, quando possibile, all'utilizzo di modelli alternativi di calcolo rispetto al *ClassicModel*, soprattutto sfrutta notevolmente le potenzialità dell'ECS e l'utilizzo di fluidi di riferimento per il calcolo della conduttività termica.
In Figura 3.13 si trova la dichiarazione dell'*enum Conductivity* che si esprime nella struttura di tipi evidenziata in Figura 3.14

```rust
#[derive(Debug)]
pub enum Conductivity {
    Classic { model: ClassicModel },
    ECS { data: ECSModelData },
    Water,
    HeavyWater,
    R23,
    Helium,
    Methane,
    NotAvailable,
}
```

Figura 3.13: Enum che rappresenta la classe della conduttività

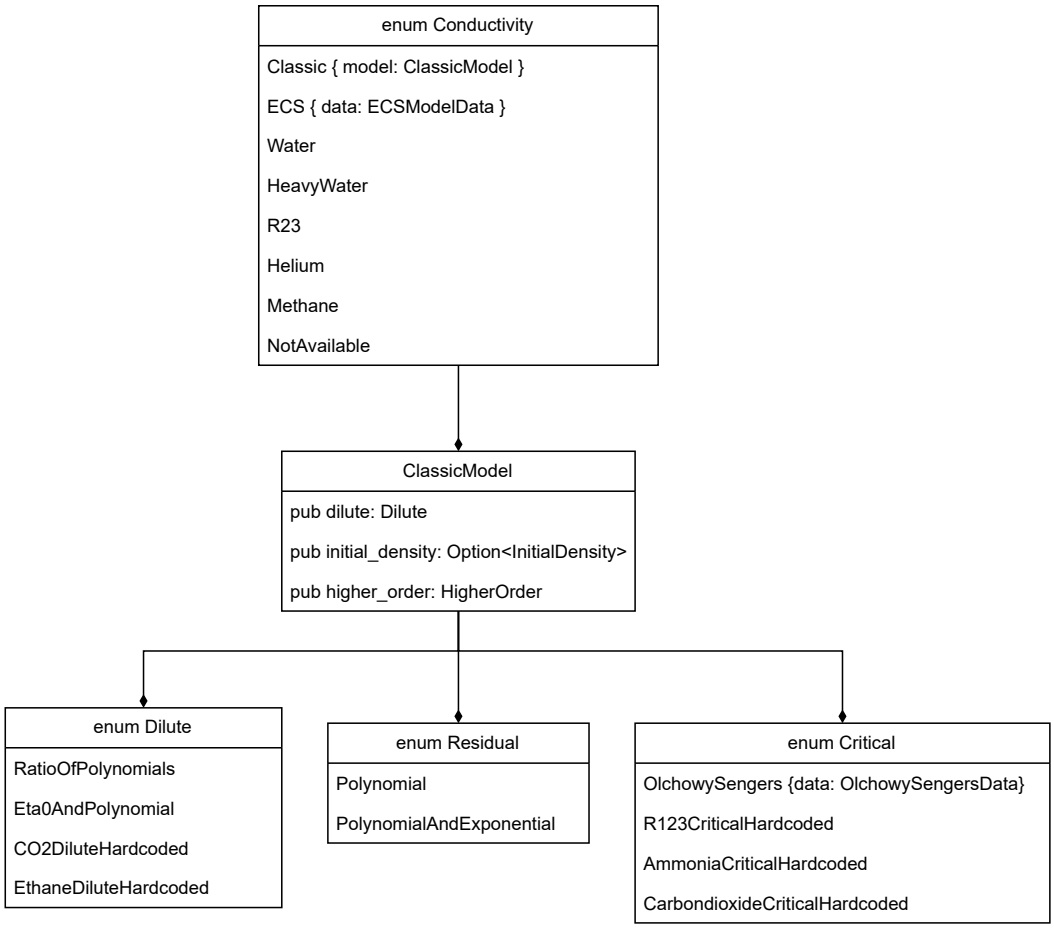


Figura 3.14: Diagramma dei tipi del modulo *conductivity*

3.4.6 Modulo *test*

Il modulo *test* fornisce tutte le istruzioni necessarie per eseguire delle suite di test sui risultati prodotti da *Fluidprops*. Rust mette a disposizione delle strutture di codice apposite (*#[cfg(test)]*) per generare dei moduli di test e separarli dal resto del codice; conferendo la possibilità di azionarli tutti in contemporanea tramite il comando *cargo test*.
Come vedremo nel Capitolo 4, si è deciso di produrre differenti tipologie di test in base alle varie funzionalità da testare e alla loro forma di calcolo. Sono presenti così test *single point* (come in Figura 3.15), altri a *multiple points* (Figura 3.16), altri ancora che testano le singole porzioni di formula o condizioni fisiche particolarmente delicate che meritavano un trattamento separato.

```rust
#[cfg(test)]
pub fn test_properties_single_point(
    fluid: &Fluid,
    temperature: f64,
    density: f64,
    relative_tolerance: f64,
) {
    use crate::{fluid::FluidError, phase_equilibria::calc_phase_equilibria};
    use core::panic;

    let expected_liquid_density = props("D", "T", temperature, "Q", 0., &fluid.name);
    let expected_gas_density = props("D", "T", temperature, "Q", 1., &fluid.name);
```

Figura 3.15: Accenno di codice del test single point

```rust
#[cfg(test)]
pub fn test_properties_multiple_points(
    fluid: &Fluid,
    temperature: &Range<f64>,
    pressure_coolprop: &Range<f64>,
    n_points: usize,
) {
    let relative_tolerance = 1e-08;

    let samples = genereate_samples(n_points, temperature, pressure_coolprop);

    for sample in samples {
        let density = props("D", "T", sample.x, "P", sample.y, &fluid.name);

        test_properties_single_point(fluid, sample.x, density, relative_tolerance);
    }
}
```

Figura 3.16: Accenno di codice del test multiple points

3.4.7 Modulo *analyzer*

Rappresenta un modulo di supporto alla rappresentazione grafica dei risultati. Contiene una serie di funzioni che si occupano di rappresentare su grafico i risultati di alcuni test e dar loro una veste grafica appetibile oltre che funzionale alla comprensione.
Si occupano di disegnare gli assi, tracciare le curve di saturazione, posizionano la legenda e gestiscono la colorazione dei punti nelle heatmap. Vi sono anche le strutture dati necessarie a contenere tali dati e, eventualmente, immagazzinarli in supporti diversi da quello strettamente grafico.

3.5 Testing e plotting

Trattandosi di una libreria di calcolo, risulta fisiologico considerare che una grossa porzione del tempo di lavoro è stata impiegata nel testing delle formule e dei calcoli, per assicurarsi che quelli prodotti da *Fluidprops* fossero coerenti con le librerie di riferimento. A tal proposito, per essere certi di contenere qualsiasi possibilità di errore o svista nella scrittura delle formule, che risultavano abbastanza intricate e facili da codificare erroneamente, si è adottata una tecnica di testing a più step e seriale rispetto a più fluidi, in modo da individuare subito in quale porzione della formula vi fossero delle inesattezze.
Si è cominciato testando inizialmente la prima parte codificata, ovvero la porzione dell'*Ideal Term*, dato che *Coolprop* mette a disposizione una variante della funzione *PropSI* che ritorna come risultato unicamente l' α^0 desiderato. Successivamente si è proseguito con la porzione del *Residual Term* e le derivate dei due addendi. In questa fase si è dovuto fare molta attenzione alle unità di misura utilizzate nel calcolo, poiché molti dati relativi alla densità dei fluidi (e di conseguenza anche la densità critica) nel codice di *Coolprop* erano rappresentati intercambiabilmente come kg/m^3 oppure mol/m^3, generando così non pochi dubbi.
La giusta cautela si è anche dovuta riservare per il numero delle cifre decimali da considerare nel calcolo. Molto spesso le conversioni $kg - mol$ cambiavano l'ordine di grandezza delle misurazioni e, di conseguenza, cambiava la quantità di cifre significative. Nonostante ciò, si è deciso ugualmente di proseguire con l'omogeneizzazione dell'unità di misura kg/m^3, non solo perché è quella adottata dal Sistema Internazionale, ma anche per garantire una più sicura usabilità da parte di qualunque software o utente, senza doversi porre il dubbio di attuare conversioni. In ogni caso, la scelta non ha compromesso le performance di calcolo poiché è stato sufficiente considerare, di caso in caso, il giusto numero di cifre decimali che permettessero una rappresentazione fedele del numero.
Le grandezze macroscopiche come la pressione, ad esempio, necessitano di minore precisione decimale, dato che le cifre decimali del risultato verrebbero comunque troncate, permettendoci di avere anche un range di errore più elevato. Altre misurazioni, invece, necessitano di più cura data la loro portata microscopica; riuscendo ad arrivare in quasi tutte le proprietà anche ad errori molto piccoli, dell'ordine di 10^{-15} (si veda esempio in Figura 3.18).

3.5.1 Single point tests

La prima iterazione sulla suite di test finale è stata la creazione di test definiti *single point*, ovvero che potessero testare il calcolo delle varie proprietà in serie su un singolo fluido a determinate condizioni di temperatura e densità, impostate a priori. Possiamo vedere un esempio di *single point test* relativo al fluido R-134a in Figura 3.17 e il suo output a terminale nella Figura 3.18.
Per i dati in input, si è deciso di utilizzare diversi valori, di volta in volta, in base alle necessità o alle aree da esplorare all'interno del piano del fluido. Inizialmente si sono adottati valori di prova di cui si sapeva con certezza il risultato, per poi passare se necessario a valori precisi per sondare la zona bi-fase o quella nelle vicinanze del punto critico, punto di intersezione superiore delle due curve di saturazione.
Questa tipologia di test non è servita per verificare la correttezza del calcolo in generale, ma per individuare e testare quei punti che risultano ostici o estremamente suscettibili a variazioni.

```rust
#[test]
fn single_point_test_r134a() {
    let fluid = get_r134a();
    let temperature = 300.0; //K
    let density = 17.8592; //kg/m3

    test_properties_single_point(&fluid, temperature, density, 1e-10);
}
```

Figura 3.17: Esempio di single point test relativo al fluido R-134a

```
running 1 test
alpha0 results are calculated = -5.47771639052955 expected = -5.47771639052955 difference = +0e0
alphar results are calculated = -0.08436072349461887 expected = -0.08436072349461883 difference = +4.163336342344337e-17
helmholtz results are calculated = -135974.19103119898 expected = -135974.19103119895 difference = +2.9103830456733704e-11
pressure results are calculated = 400000.5574915032 expected = 400000.5574915032 difference = +0e0
entropy results are calculated = 1779.6550401235434 expected = 1779.6550401235436 difference = +2.2737367544323206e-13
internal_energy results are calculated = 397922.32100586407 expected = 397922.321005864 difference = +5.820766091346741e-11
enthalpy results are calculated = 420319.7720390292 expected = 420319.7720390292 difference = +0e0
gibbs_energy results are calculated = -113576.73999803381 expected = -113576.73999803381 difference = +0e0
isobaric_heat_capacity results are calculated = 920.7443197160605 expected = 920.7443197160607 difference = +1.1368683772161603e-13
isochoric_heat_capacity results are calculated = 792.6677166327695 expected = 792.6677166327695 difference = +0e0
compressibility_factor results are calculated = 0.9161764365670038 expected = 0.9161764365670038 difference = +0e0
test test::single_point_test_r134a ... ok

test result: ok. 1 passed; 0 failed; 0 ignored; 0 measured; 153 filtered out; finished in 0.11s
```

Figura 3.18: Risultato del single point test relativo al fluido R-134a

3.5.2 Multiple points tests

Per sopperire alla necessità di operare test su più larga scala, sono stati introdotti anche dei test denominati *multiple points tests* (Figura 3.19), che hanno validato definitivamente la correttezza dei calcoli e l'ottimizzazione della libreria.
La prima necessità del test consiste nella generazione di un numero elevato di punti,

nel nostro caso 10000, che rientrassero in determinati range di temperatura e densità. Tale operazione è risultata abbastanza triviale poiché la documentazione consultata forniva solamente dei range di temperatura e pressione entro cui il fluido è stato testato sperimentalmente ed entro cui la formula confezionata risulta adatta. Questa decisione deriva dal fatto che sperimentalmente è più immediato indicare tali range rispetto a misurare quello della densità, che risulta maggiormente volubile. Per ovviare a questo problema, si sono adottati nei *multiple tests* i range di temperatura e pressione riscontrati nella documentazione e negli articoli, per poi ricavare attraverso *Coolprop* la misurazione della densità e usarla nel raffronto finale.
Ogni *sample* era dotato, a questo punto, di due coordinate (T, ρ) che potevano essere inserite nelle funzioni di calcolo e testate per tutte le proprietà. Alla fine del test, risultava nel terminale il conteggio delle riuscite e gli eventuali valori errati. Se tutto risultava corretto, si procedeva con il fluido successivo e così via.
In caso di errore, il terminale indica il test che l'ha segnalato e la differenza nelle misurazioni (Figura 3.20).

```rust
#[test]
fn multiple_points_test_r134a() {
    let fluid = get_r134a();
    let n_points = 10_000;
    let fluid_ranges = fluid.get_ranges();
    let temperature = fluid_ranges.temperature_range;
    let pressure = fluid_ranges.pressure_range;

    test_properties_multiple_points(&fluid, &temperature, &pressure, n_points);
}
```

Figura 3.19: Esempio di multiple points test relativo al fluido R-134a

```
running 1 test
thread 'test::multiple_points_test_r134a' panicked at 'phase_equilibria.density_gas at T = 206.86856359099122
 calculated = 0.6231158200447428 expected = 0.6231159409735719 difference = +1.2092882917258407e-7', src\test_utils.rs
:67:13
stack backtrace:
   0: std::panicking::begin_panic_handler
             at /rustc/2c8cc343237b8f7d5a3c3703e3a87f2eb2c54a74/library\std\src\panicking.rs:575
   1: core::panicking::panic_fmt
             at /rustc/2c8cc343237b8f7d5a3c3703e3a87f2eb2c54a74/library\core\src\panicking.rs:64
   2: fluidprops::test_utils::test_properties_single_point
             at .\src\test_utils.rs:67
   3: fluidprops::test_utils::test_properties_multiple_points
             at .\src\test_utils.rs:49
   4: fluidprops::test::multiple_points_test_r134a
             at .\src\test.rs:273
```

Figura 3.20: Esempio di errore durante l'esecuzione di un multiple test

3.5.3 Plotting

Si è ritenuto utile, per un progetto come questo, l'impiego di strumenti grafici per la rappresentazione dei risultati ottenuti nella comparazione del prodotto *Fluidprops* con i prodotti in uso attualmente dall'azienda e, più diffusamente, nel mondo. Dato il

formato numerico e potenzialmente verboso dei risultati e dei raffronti, si è deciso di rappresentare attraverso delle *heatmap* (o mappe di calore) le differenze di calcolo tra le due librerie.
Le *heatmap* sono delle forme di rappresentazione ispirate alle immagini termiche dove i dati assumono una colorazione specifica in base a determinate caratteristiche prefissate. Nel nostro caso, dovendo rappresentare la differenza di calcolo tra due risultati, si è deciso di generare un insieme di 30000 *samples* entro i range di temperatura e pressione disponibili e rappresentarli su piano come dei punti aventi coordinate (T, ρ) oppure (H, P). Queste scelte derivano dal fatto che l'utilità maggiore del calcolo si ottiene avendo come input temperatura (T) e densità (ρ), ma a livello grafico, in letteratura, si utilizza più frequentemente la rappresentazione attraverso entalpia (H) e pressione (P). Ogni punto, come mostrato nelle Figure 3.21 e 3.22, dopo essere stato posizionato sul piano, assume una propria colorazione in base alla quantità di errore che il calcolo di quel valore riscontra con il riferimento di *Coolprop*: più il colore è tendente al rosso, più l'errore è alto e, al contrario, il blu rappresenta la presenza di errore minimo o nullo.
Per facilitare la visualizzazione si sono dotate le *heatmap* di una legenda che ne rappresenta i limiti associati alle colorazioni; mentre nel grafico si notano due linee nere, indice delle curve di saturazione di cui si accennava nella Sezione 2.3.
Come si può notare, i risultati di comparazione sono molto buoni e garantiscono un'elevata accuratezza di calcolo; simbolo non solo della correttezza delle formule, ma anche di una curata organizzazione e strutturazione del codice.
Negli esempi delle Figure 3.23 e 3.24, che rappresentano gli errori nel calcolo della pressione per l'R-134a, si è adottata, diversamente da prima, la costruzione del piano secondo temperatura (T) e densità (ρ). Qui la campana assume un diverso orientamento, ma le stime dell'errore assoluto e relativo rimangono ugualmente buone.

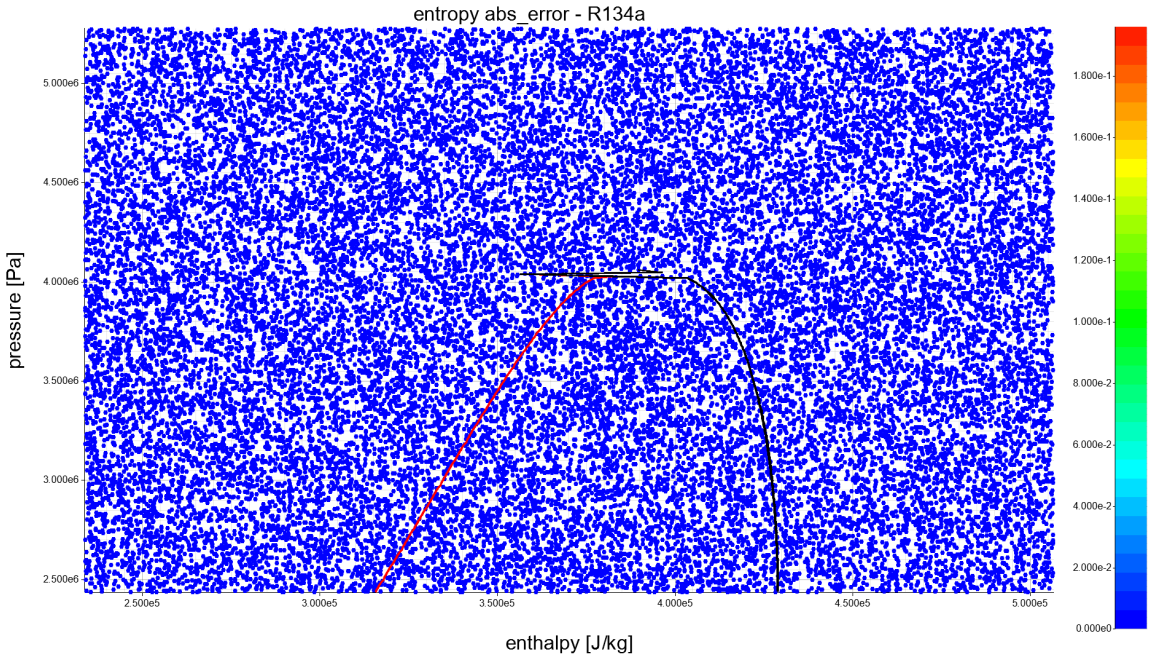


Figura 3.21: Heatmap che misura l'errore assoluto nel calcolo dell'entropia del fluido R-134a

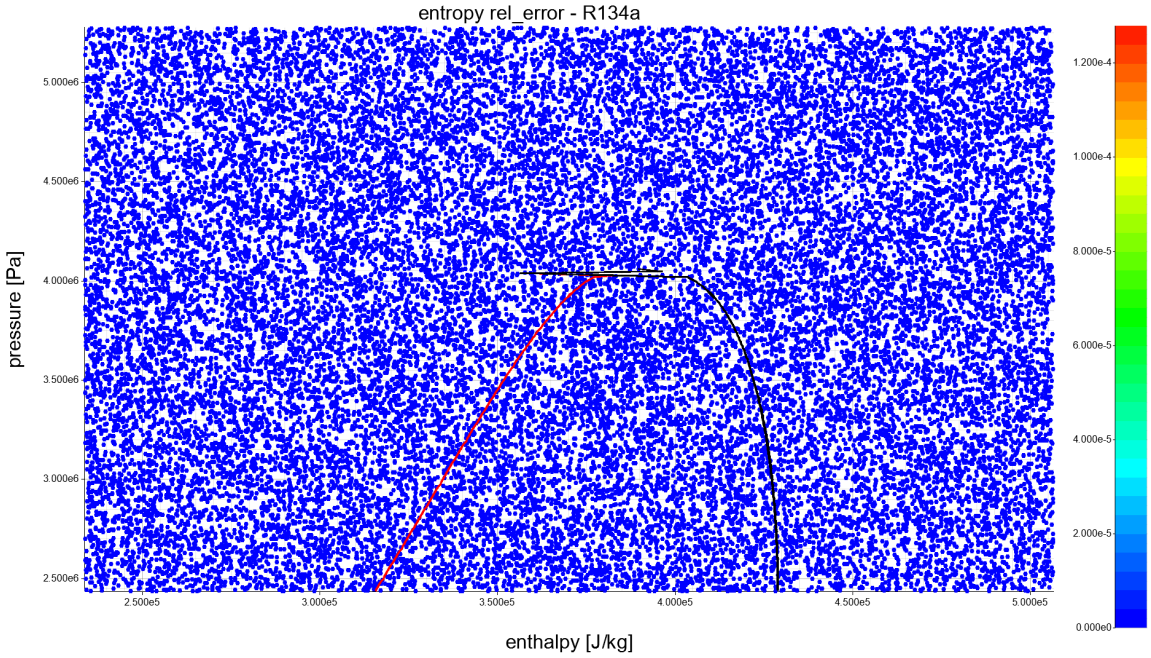


Figura 3.22: Heatmap che misura l'errore relativo nel calcolo dell'entropia del fluido R-134a

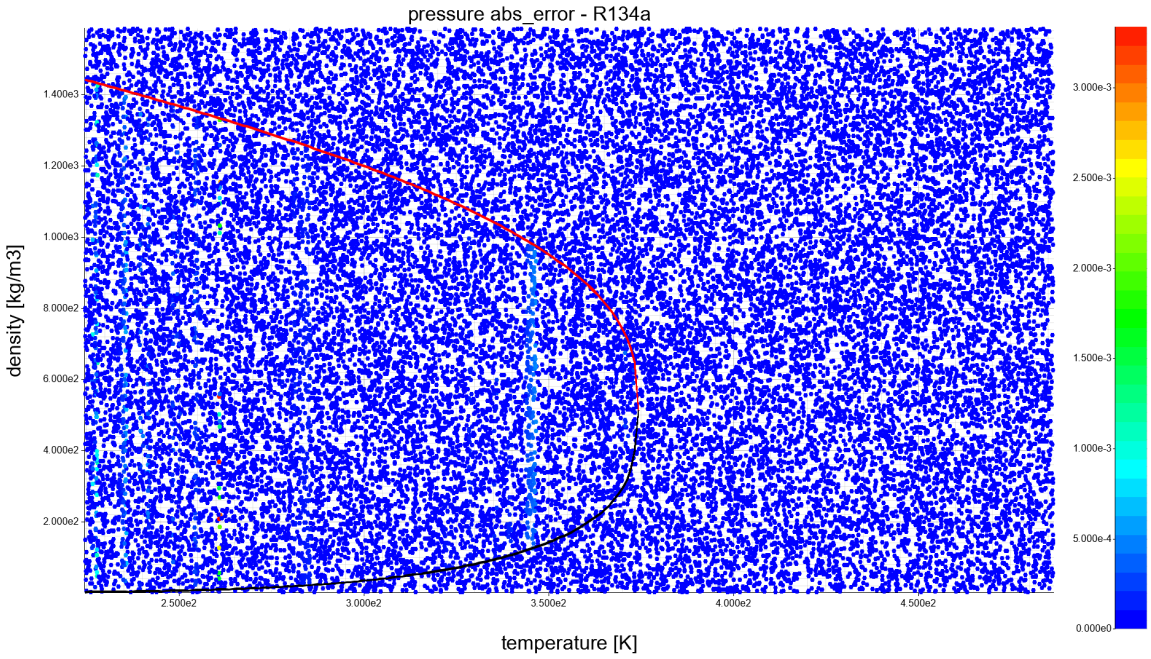


Figura 3.23: Heatmap che misura l'errore assoluto nel calcolo della pressione per l'R-134a

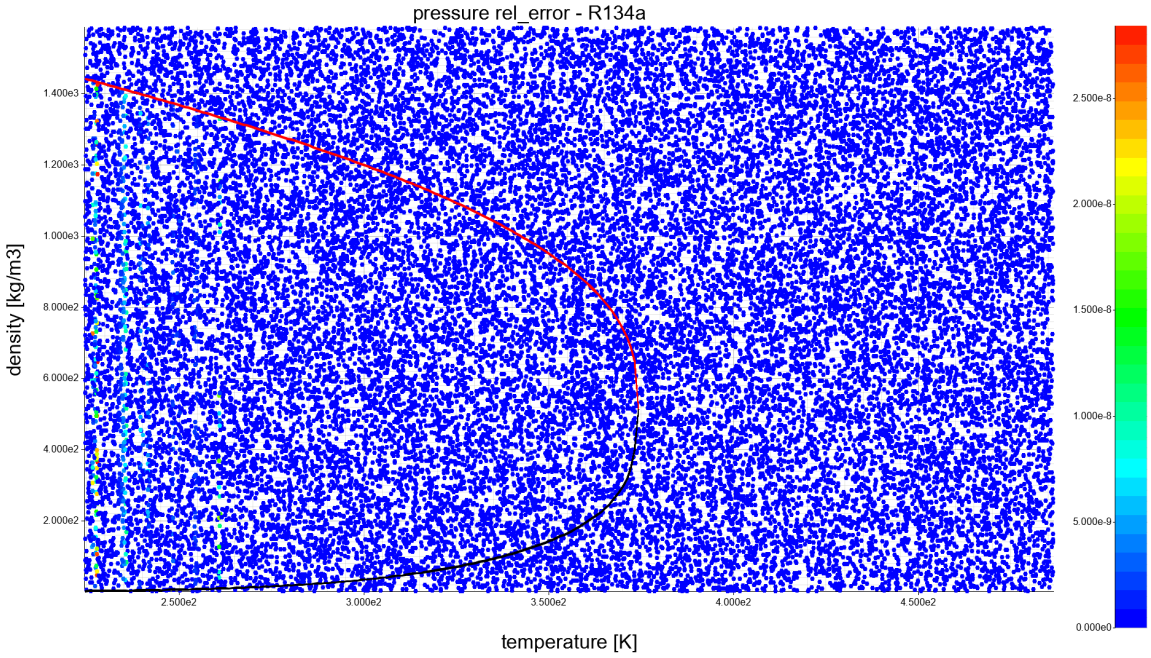


Figura 3.24: Heatmap che misura l'errore relativo nel calcolo della pressione per l'R-134a

Per rappresentare i range di calcolo tra le due librerie, si possono osservare anche le Figure 3.25 e 3.26, che rappresentano il calcolo dell'entropia del fluido R-134a, rispettivamente, attraverso le librerie *Coolprop* e *Fluidprops*. Il range di colori è relativo alla grandezza della proprietà in quel determinato punto del piano, quindi non è particolarmente rilevante in quest'ultimo paragone. Ciò che risulta subito evidente all'occhio, però, è la coerenza tra le due *heatmap* nella disposizione e dispersione dei colori e, di conseguenza, delle computazioni.

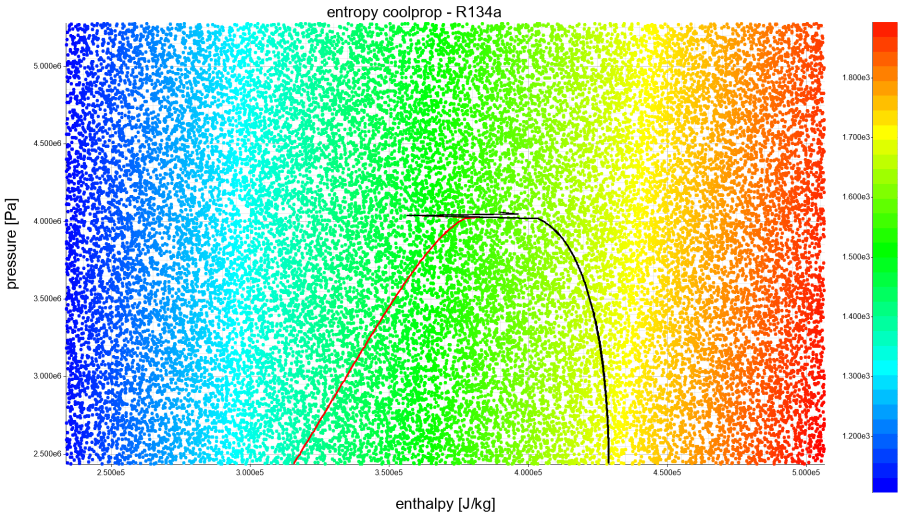


Figura 3.25: Range di misurazioni di *Coolprop* nel calcolo dell'entropia per l'R-134a

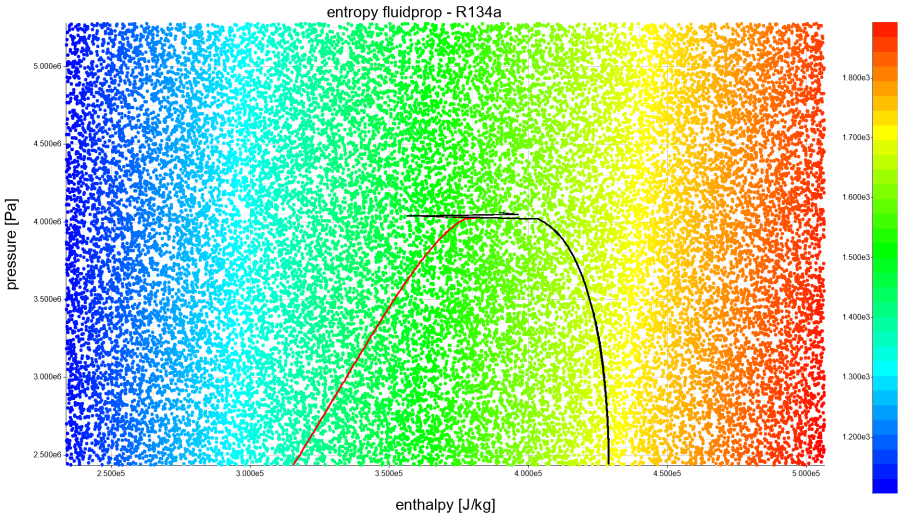


Figura 3.26: Range di misurazioni di *Fluidprops* nel calcolo dell'entropia per l'R-134a

Capitolo 4

Conclusioni

L'obiettivo dell'esperienza di stage prevedeva la creazione di una libreria software, redatta in linguaggio Rust, per il calcolo di proprietà termodinamiche dei fluidi. A completamento dell'attività, riteniamo che il progetto abbia soddisfatto le aspettative iniziali e, oltre alla creazione del prodotto, ha costituito un'esperienza formativa a tutto tondo.
Il risultato è una libreria che incarna i principi delle librerie di riferimento sul mercato, svolge parte delle loro funzioni e si prospetta, nei suoi sviluppi futuri, un loro valido concorrente in termini di efficienza ed efficacia. A ciò si accompagna un ampio lavoro di ricostruzione delle informazioni riguardanti le librerie di calcolo esistenti e delle metodologie interne che queste adottano. Chiaramente un progetto così ambizioso necessiterebbe di numerose ore di lavoro e persone coinvolte per il suo finale completamento, ma *Fluidprops* si conferma un embrione con buone potenzialità sia per la sua valenza di elemento software, sia per gli impieghi futuri nei più svariati campi.

4.1 Consuntivo finale

Il preventivo orario è stato rispettato e l'intera attività si è svolta nelle trecentotrenta ore previste. La ripartizione interna dei task rispetto alle settimane di lavoro ha subito delle leggere variazioni, dato che molti compiti fissati hanno richiesto meno tempo di completamento rispetto a quello indicato. Tutto ciò è andato a vantaggio di quei task che si sono rivelati più ostici, riuscendo a controbilanciare le tempistiche e completare il tutto nei limiti indicati.
Nella Tabella 4.1 si può vedere il consuntivo delle attività.

Descrizione dell'attività	Preventivo	Consuntivo	Differenza
Formazione preliminare	40 ore	30 ore	−10
Studio della documentazione e delle soluzioni software esistenti	40 ore	50 ore	+10
Definizione dell'API contenuta nella libreria	70 ore	70 ore	–
Stesura della relativa documentazione	10 ore	5 ore	−5
Implementazione della libreria	80 ore	100 ore	+20
Aggiornamento documentazione	10 ore	5 ore	−5
Ottimizzazione algoritmi	40 ore	40 ore	–
Testing finale relativo alle performance	25 ore	25 ore	–
Redazione finale della documentazione	5 ore	5 ore	–
Integrazione dei progressi con l'applicativo	10 ore	–	−10
Totale	330 ore	330 ore	0 ore

Tabella 4.1: Prospetto orario delle attività con il relativo preventivo orario

L'iniziale fase di formazione è risultata meno dispendiosa del previsto, mentre i maggiori aumenti sono da imputarsi all studio di *Coolprop*, del manuale [35] e all'implementazione vera e propria.
La mancanza del tempo necessario per raggiungere gli obiettivi facoltativi, F01 e F02 (vedi Tabella 4.2), tra cui anche l'integrazione di *Fluidprops* all'interno dell'applicativo Novaeka, ha fatto ricadere un cospicuo numero di ore nella fase implementativa, ammortando così la differenza creatasi.

4.2 Raggiungimento degli obiettivi

L'esperienza di stage ha raggiunto tutto gli obbligatori (O01, O02, O03) e desiderabili (D01) stabiliti a priori e descritti nel piano di lavoro. Gli obiettivi facoltativi (F01, F02) risultavano troppo costosi per la quantità di tempo che era rimasta a disposizione, una volta completati gli obiettivi primari. Tuttavia si è deciso di sfruttare ugualmente il tempo rimanente per inserire nel progetto delle features che potessero espandere le funzionalità di *Fluidprops* e che fossero realizzabili nel tempo rimasto a disposizione.
La strategia Agile è stata essenziale per mantenere alta l'attenzione e scomporre un progetto così articolato in segmenti più facilmente processabili. Hanno giocato anche un ruolo fondamentale la continua comunicazione tra i membri del team, lo scambio di idee e il confronto aperto sulle modalità di prosecuzione del lavoro o sulle decisioni progettuali operabili.
Nella Tabella 4.2 vengono riassunti gli obiettivi iniziali e il loro finale soddisfacimento o meno.

Tipologia	Codice	Descrizione obiettivo	Soddisfatto
Obbligatori	O01	Analisi del perimetro delle funzionalità richieste dall'applicativo Novaeka	Sì
	O02	Definizione di una Application Program Interface più robusta e semplificata rispetto alle librerie concorrenti	Sì
	O03	Implementazione in Rust della libreria	Sì
Desiderabili	D01	Ottimizzazione delle soluzioni implementate	Sì
Facoltativi	F01	Report sulle performance delle librerie esistenti e quella sviluppata	No
	F02	Integrazione della libreria nell'applicativo Novaeka	No

Tabella 4.2: Tabella degli obiettivi suddivisi per tipologie

4.3 Analisi critica e conclusione

Fluidprops è nata con l'obiettivo di mimare librerie già esistenti, ma in corso d'opera si è rivelato un prodotto con la propria identità e valenza. Sicuramente le funzionalità di cui attualmente dispone sono in numero inferiore rispetto ai competitor, ma ciò non impedisce nel breve o lungo termine un suo potenziamento e miglioramento.
Il focus delle sue potenzialità sta nella creazione di un'architettura finalmente priva di codice ridondante e incongruenze di tipo. La struttura chiara e matematicamente coerente rende la libreria estremamente efficiente, oltre a favorire leggibilità e usabilità. Pur riducendo al minimo la complessità delle strutture e contenendo la gerarchizzazione dei tipi, si è comunque riusciti ad estrapolare l'essenza del calcolo e ad individuare le forme fondamentali delle equazioni di stato.
Grande rilievo deve avere l'attività di studio e apprendimento del contesto matematico-fisico in cui la libreria opera, non solo come diletto e conoscenza personale, ma soprattutto per la lucidità che queste conoscenze donano nella progettazione di un prodotto solido e ben strutturato. Con questo elaborato si prosegue ulteriormente l'opera di documentazione iniziata durante lo stage, nella speranza che l'apparato teorico qui sintetizzato potrà essere utile nella documentazione della libreria e nello studio del suo funzionamento.
La formazione non ha riguardato solo le nozioni informatiche sull'attività, ma ha rappresentato una palestra di allenamento personale per lo sviluppo del lavoro in team e delle *soft-skills*. Tutto ciò sarebbe stato impossibile senza la collaborazione e il supporto del team per l'intero svolgimento del progetto.

Fluidprops rappresenta l'audace tentativo di addentrarsi tra le librerie di calcolo per scovare una nuova via esplorativa nel mare delle proprietà termodinamiche dei fluidi.

Acronimi

API Application Program Interface. 63

ECS Extended Corresponding States. 63

EEO Entropy Enthalpy Offset. 9, 62

EOS Equation Of State. 7, 62

IDE Integrated Development Enviroment. 63

SAFT Statistical Associating Fluid Theory. 64

UML Unified Modeling Language. 64

Glossario

API In informatica con il termine *Application Programming Interface API* (interfaccia di programmazione di un'applicazione) si indica ogni insieme di procedure disponibili al programmatore, di solito raggruppate a formare un set di strumenti specifici per l'espletamento di un determinato compito all'interno di un certo programma. La finalità è ottenere un'astrazione, di solito tra l'hardware e il programmatore o tra software a basso e quello ad alto livello semplificando così il lavoro di programmazione [1]. 4, 61, 62

Backlog Il Product Backlog è un artefatto ufficiale di Scrum che consiste in un elenco di attività (Product Backlog Item) ordinato per priorità. Il Product Backlog viene costantemente rivisto e riordinato dal Product Owner in base alle necessità degli utenti o del cliente, le aspettative degli stakeholder, nuove idee, o in seguito alle esigenze di mercato, ma anche in base a suggerimenti da parte del team [27]. 3, 63

Commit In contesto informatico, "commit" implica genericamente il rendere effettive delle modifiche [5]. 6, 63

Coolprop Coolprop è una libreria scritta in linguaggio C++ che implemente equazioni di stato per 122 fluidi puri e pseudo-puri, oltre ad altre proprietà di transport [10]. 6, 63

Debugging Il debugging, in informatica, nell'ambito dello sviluppo software, indica l'attività che consiste nell'individuazione e correzione da parte del programmatore di uno o più errori (bug) rilevati nel software, direttamente in fase di programmazione oppure a seguito della fase di testing o dell'utilizzo finale del programma stesso [13]. 5, 63

ECS La teoria degli Extended Corresponding States trae spunto dai lavori di Bell, I. H., Wronski, J., Quoilin, S. e Lemort, V. Il principio si basa sulla simulazione del comportamento del fluido sulla base di un fluido di riferimento [**ecs**]. 21, 62

IDE L'Integrated Development Environment, o ambiente di sviluppo integrato, in informatica, è un software che, in fase di programmazione, supporta i programmatori nello sviluppo e debugging del codice sorgente di un programma [20]. 5, 62

Nesting In informatica viene detto annidamento (*nesting*) l'inserimento di una struttura di controllo all'interno di un'altra in un programma [29]. 46, 63

Plugin Il plug-in in campo informatico è un programma non autonomo che interagisce con un altro programma per ampliarne o estenderne le funzionalità originarie [34]. 5, 64

Densità critica Misurazione della densità di una sostanza al suo punto critico termodinamico [35]. 52, 64

SAFT Statistical associating fluid theory (SAFT) è una teoria nel campo della chimica, basata sulla *perturbation theory*, che utilizza la termodinamica statistica per spiegare come fluidi complessi e miscele formino associazioni attraverso legami ad idrogeno [32]. 13, 62

Testing Il collaudo del software (anche software testing), in informatica, indica un procedimento, che fa parte del ciclo di vita del software, utilizzato per individuare le carenze di correttezza, completezza e affidabilità delle componenti software in corso di sviluppo [40]. 3, 64

Toolchain Si intende un insieme di applicativi utilizzati nello sviluppo o nella creazione di un prodotto software complesso. Tali strumenti possono essere utilizzati in catena, in modo tale che l'output di ciascun tool rappresenti l'input per il successivo, ma il termine è utilizzato in maniera più estesa per riferirsi, più in generale, a qualunque insieme di software di sviluppo collegato con un altro [45]. 3, 64

UML In ingegneria del software *UML, Unified Modeling Language* (linguaggio di modellazione unificato) è un linguaggio di modellazione e specifica basato sul paradigma object-oriented. L'*UML* svolge un'importantissima funzione di "lingua franca" nella comunità della progettazione e programmazione a oggetti. Gran parte della letteratura di settore usa tale linguaggio per descrivere soluzioni analitiche e progettuali in modo sintetico e comprensibile a un vasto pubblico [46]. 62

Versione In informatica, ma soprattutto nello sviluppo software, la versione (di un software, programma o applicazione) corrisponde a un determinato stato nello sviluppo di un software secondo l'uso del versioning ("versionamento") [39]. 6, 64

VoIP Voice over IP, in telecomunicazioni e informatica, indica una tecnologia che rende possibile effettuare una conversazione, analoga a quella che si potrebbe ottenere con una rete telefonica, sfruttando una connessione Internet o una qualsiasi altra rete di telecomunicazioni dedicata a commutazione di pacchetto, che utilizzi il protocollo IP senza connessione per il trasporto dati [48]. 5, 64

Wrapper In informatica, e in particolare in programmazione, è un modulo software che ne "riveste" un altro, ossia che funziona da tramite fra i propri clienti e il modulo rivestito [50]. 26, 64

Bibliografia

Riferimenti bibliografici

[3] D. Buecker e W. Wagner. *A Reference Equation of State for the Thermodynamic Properties of Ethane for Temperatures from the Melting Line to 675 K and Pressures up to 900 MPa*. J. Phys. Chem. Ref. Data, 35, 205–266, 2006.

[4] T. H. Chung. *Generalized multiparameter correlation for nonpolar and polar fluid transport properties*. Industrial engineering chemistry research, 27, 671-679, 1988.

[12] K.M. De Reuck. *Fluorine: International Thermodynamic Tables of the Fluid State - 11*. Blackwell Scientific Publications, 1990.

[16] K. Gao, I. H. Wu J. AND Bell e E. W. Lemmon. *Thermodynamic Properties of Ammonia for Temperatures from the Melting Line to 725 K and Pressures to 1000 MPa*. J. Phys. Chem. Ref. Data, 2020.

[17] D. Green e M. Z. Southard. *Perry's Chemical Engineers' Handbook*. McGraw-Hill Education, Europe, 2018.

[18] H. von Helmholtz. *Physical memoirs, selected and translated from foreign sources*. Taylor Francis, 1882.

[19] M. L. Huber, A. Laesecke e R. A. Perkins. *Model for the Viscosity and Thermal Conductivity of Refrigerants, Including a New Correlation for the Viscosity of R134a*. Ind. Eng. Chem. Res., 42, 3163–3178, 2003.

[21] O. Kunz e W. Wagner. *The GERG-2008 Wide-Range Equation of State for Natural Gases and Other Mixtures: An Expansion of GERG-2004*. J. Chem. Eng. Data, 57, 3032–3091, 2012.

[22] E. Lemmon e R. T. Jacobsen. *A New Functional Form and New Fitting Techniques for Equations of State with Application to Pentafluoroethane (HFC-125)*. Journal of Physical e Chemical Reference Data, 34(1), 69-108, 2005.

[23] E. W. Lemmon, M. O. McLinden e W. Wagner. *Thermodynamic Properties of Propane. III. A Reference Equation of State for Temperatures from the Melting Line to 650 K and Pressures up to 1000 MPa*. J. Chem. Eng. Data, 54, 3141–3180, 2009.

[24] E. W. Lemmon et al. *Thermodynamic Properties of Air and Mixtures of Nitrogen, Argon, and Oxygen from 60 to 2000 K at Pressures to 2000 MPa*. J. Phys. Chem. Ref. Data, 29, 331–385, 2000.

[25] E.W. Lemmon e R. Span. *Short Fundamental Equations of State for 20 Industrial Fluids*. J. Chem. Eng. Data, 51, 785–850, 2006.

[28] M. O. McLinden, S. A. Klein e R. A. Perkins. *An extended corresponding states model for the thermal conductivity of refrigerants and refrigerant mixtures*. Int. J. Refrig., 23, 43-63, 2000.

[31] S. L. Outcalt e M. O. McLinden. *A Modified Benedict-Webb-Rubin Equation of State for the Thermodynamic Properties of R152a (1,1-difluoroethane)*. J. Phys. Chem. Ref. Data, 25, 605–636, 1996.

[32] L. Piazza e R. Span. *An equation of state for methanol including the association term of SAFT*. Fluid Phase Equilibria, 349, 12-24, 2013.

[33] B. Platzer, A. Polt e G. Maurer. *Thermophysical Properties of Refrigerants*. Springer-Verlag, 1990.

[35] Span R. *Multiparameter Equation of State*. Springer Berlin, Heidelberg, 2000.

[36] J. A. Schroeder, S. G. Penoncello e J. S. Schroeder. *A Fundamental Equation of State for Ethanol*. J. Phys. Chem. Ref. Data, 43, 2014.

[42] M. Thol et al. *Fundamental Equations of State for Hydrocarbons. Part I. n-Pentane*. Fluid Phase Equilib., 2019.

[43] R. Tillner-Roth e H. Dieter Baehr. *A International Standard Formulation for the Thermodynamic Properties of 1,1,1,2-Tetrafluoroethane (HFC-134a) for Temperatures from 170 K to 455 K and Pressures up to 70 MPa*. J. Phys. Chem. Ref. Data, 23, 657–729, 1994.

[44] R. Tillner-Roth e A. Yokozeki. *An international standard equation of state for difluoromethane (R-32) for temperatures from the triple point at 136.34 K to 435 K and pressures up to 70 MPa*. J. Phys. Chem. Ref. Data, 26, 1273–1328, 1997.

[49] W. Wagner e A. Pruß. *The IAPWS Formulation 1995 for the Thermodynamic Properties of Ordinary Water Substance for General and Scientific Use*. J. Phys. Chem. Ref. Data, 31, 387–535, 2002.

[51] Y. Zhou et al. *An Equation of State for the Thermodynamic Properties of Cyclohexane*. J. Phys. Chem. Ref. Data, 43, 2014.

Siti web consultati

[1] *Application Programming Interface*. URL: https://en.wikipedia.org/wiki/API.

[2] *Azure DevOps*. URL: https://azure.microsoft.com/en-us/products/devops.

[5] *Commit (version control)*. URL: https://en.wikipedia.org/wiki/Commit_(version_control).

[6] *Coolprop wrappers*. URL: http://www.coolprop.org/coolprop/wrappers/index.html.

[7] *Coolprop: Advanced Fluid Properties*. URL: http://www.coolprop.org/fluid_properties/more_reading.html.

[8] *Coolprop: Bibliography*. URL: http://www.coolprop.org/zbibliography.html#id118.

[9] *Coolprop: Equation Of State Terms*. URL: http://www.coolprop.org/_static/doxygen/html/class_cool_prop_1_1_base_helmholtz_term.html.

[10] *Coolprop: home*. URL: http://www.coolprop.org/.

[11] *Coolprop: Pure and Pseudo-pure fluids*. URL: http://www.coolprop.org/fluid_properties/PurePseudoPure.html.

[13] *Debugging*. URL: https://en.wikipedia.org/wiki/Debugging.

[14] *Discord*. URL: https://discord.com/.

[15] *Documentazione di Coolprop: TransportRoutines Class*. URL: http://www.coolprop.org/fluid_properties/more_reading.html.

[20] *Integrated Development Environmen*. URL: https://en.wikipedia.org/wiki/Integrated_development_environment.

[26] *Linguaggio Rust*. URL: https:/www.rust-lang.org/.

[27] *Manifesto Agile*. URL: http://agilemanifesto.org/iso/it/.

[29] *Nesting*. URL: https://it.wikipedia.org/wiki/Annidamento_(informatica).

[30] *NIST Reference Fluid Thermodynamic and Transport Properties Database (REFPROP)*. URL: https://www.nist.gov/srd/refprop.

[34] *Plug-in*. URL: https://en.wikipedia.org/wiki/Plug-in_(computing).

[37] *Scrum Framework*. URL: https://www.scrum.org/resources/.

[38] *Sito Novaeka*. URL: https://www.novaeka.com/.

[39] *Software versioning*. URL: https://en.wikipedia.org/wiki/Software_versioning.

[40] *Testing*. URL: https://en.wikipedia.org/wiki/Software_testing.

[41] *The Rust book*. URL: https://doc.rust-lang.org/book/.

[45] *Toolchain*. URL: https://en.wikipedia.org/wiki/Toolchain.

[46] *Unified Modeling Language*. URL: https://en.wikipedia.org/wiki/Unified_Modeling_Language.

[47] *Visual Studio Code*. URL: https://code.visualstudio.com/.

[48] *VoIP*. URL: https://en.wikipedia.org/wiki/Voice_over_IP.

[50] *Wrapper*. URL: https://it.wikipedia.org/wiki/Wrapper.