Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eidgenössisches Departement für
Umwelt, Verkehr, Energie und Kommunikation UVEK

Bundesamt für Energie BFE
Sektion Energieforschung und Cleantech

Schlussbericht vom 6. März 2026

HiTemHP

Effizienter Einsatz von Hochtemperatur- Wärmepumpen in Altbauten und bei Sanierungen

Quelle: CO_2 Wärmepumpe in NEST © Empa 2021

Subventionsgeberin:
Bundesamt für Energie BFE
Sektion Energieforschung und Cleantech
CH-3003 Bern
www.energieforschung.ch

Subventionsempfänger/innen:
Empa
Überlandstrasse 129
8600 Dübendorf
www.empa.ch

Autor:
Robert Weber, Empa, robert.weber@empa.ch

BFE-Projektbegleitung:
Nadège Vetterli, nadege.vetterli@anex.ch (bis Ende 2024)
Martin Ménard, menard@lowtechlab.ch
Andreas Eckmanns, andreas.eckmanns@bfe.admin.ch

BFE-Vertragsnummer: SI/502168-01

Für den Inhalt und die Schlussfolgerungen sind ausschliesslich die Autoren dieses Berichts verantwortlich.

Zusammenfassung

Hochtemperatur-CO_2-Wärmepumpen mit Vorlauftemperaturen bis zu 90 °C gelten als vielversprechende Option, um Öl- der Gasheizungen in (geschützten) Altbauten zu ersetzen, da sie einen grossen Temperaturhub ermöglichen. Allerdings fällt bei diesen Systemen nicht nur Wärme auf hohem, sondern auch auf niedrigem Temperaturniveau an. Für eine hohe Energieeffizienz muss die gesamte produzierte Wärme genutzt werden. Voraussetzung dafür ist eine möglichst niedrige Rücklauftemperatur des Gebäudes.

In diesem Projekt werden für TRNSYS nicht standardmässig vorhandene Module – insbesondere für transkritische einstufige CO_2-Wärmepumpen – neu entwickelt und anhand von Messdaten aus der NEST-Datenbank kalibriert. Zudem werden Messdaten der CO_2-Wärmepumpe in der SFW-Unit ausgewertet, um physikalische Zusammenhänge besser zu verstehen und korrekt zu modellieren. Ein idealisiertes Matlab-Modell wurde erstellt und mit den Messwerten verglichen, um zu verstehen, wieso der bestehende Prototyp in gewissen Betriebspunkten noch nicht wie erwartet arbeitet.

Der Fortran Code von Type8505, die Dokumentation der Files, die Schnittstellenfiles zu TRNSYS und die dazugehörenden Datenfiles wurden unter der Lizenz LGPLv3 freigegeben und können heruntergeladen werden von:
https://github.com/hues-platform/trnsys-type8505

Summary

High-temperature CO_2 heat pumps with flow temperatures of up to 90°C are considered a promising option for replacing oil or gas heating systems in (protected) older buildings, as they enable a large temperature lift. However, these systems generate heat not only at high temperatures, but also at low temperatures. To achieve high energy efficiency, all of the heat produced must be utilized. This requires the building's return temperature to be as low as possible.

In this project, modules that are not available as part of the standard TRNSYS package – in particular for transcritical single-stage CO_2 heat pumps – are being newly developed and calibrated using measurement data from the NEST database. In addition, measurement data from the CO_2 heat pump in the SFW unit is being evaluated in order to better understand physical relationships and model them correctly. An idealized Matlab model was created and compared with the measured values in order to understand why the existing prototype does not yet work as expected at certain operating points.

The Type8505 Fortran code, the file documentation, the interface files for TRNSYS, and the associated data files have been released under the LGPLv3 license and can be downloaded from:
https://github.com/hues-platform/trnsys-type8505

Kernbotschaften («Take-Home Messages»)

- **Hochtemperatur-CO_2-Wärmepumpen sind eine zentrale Zukunftstechnologie für Altbauten mit hohen Systemtemperaturen.** Sie ermöglichen den Ersatz fossiler Heizsysteme auch dort, wo klassische Wärmepumpen aus technischen Gründen scheitern – etwa in denkmalgeschützten, schlecht isolierten oder nur schwer sanierbaren Gebäuden.
- **Die Systemeffizienz steht und fällt mit der Auslegung des Wärmeabgabesystems – insbesondere den Rücklauftemperaturen.** Hohe Effizienz (hohe COP-Werte) kann nur erreicht werden, wenn möglichst tiefe Rücklauftemperaturen realisiert werden. Das erfordert geeignete Heizflächen und gegebenenfalls Anpassungen im Bestand.

- **Mischsysteme aus bestehenden Radiatoren, Flächenheizungen und Konvektoren bieten technisch realisierbare Lösungen.** Solche Hybridkonzepte ermöglichen energetisch sinnvolle Temperaturabsenkungen, ohne Komfort und Gebäudestruktur zu beeinträchtigen – ein wichtiger Faktor für Umrüstungen in Bestandsbauten.
- **Das Projekt liefert ein frei nutzbares, detailliertes Simulationsmodell (TRNSYS Type 8505) für einstufige transkritische CO_2-Wärmepumpen.** Dieses Werkzeug schafft einen unmittelbaren praktischen Nutzen für Planung, Ingenieurbüros und Forschung, indem es Variantenvergleiche und Optimierungsstudien realitätsnah unterstützt.
- **Trotz technischer Hürden entstand ein substanzieller Mehrwert für zukünftige Entwicklungen.** Auch wenn ein Prototypenausfall die Experimentphase verhinderte, ist mit dem Simulationswerkzeug ein wichtiger Baustein für weitere Industrialisierungsschritte und zukünftige Förder- oder Entwicklungsprogramme gelegt.

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

COP	Coefficient of performance
CO_2	Kohlendioxid
DeCarbCH	Acronym des Projektes "Decarbonisation of Cooling and Heating in Switzerland"
EN ISO	Europäische und Internationale Norm
GC	Gas Cooler
HFO	Hydrofluorolefin, Hochtemperaturkältemittel
IHX	Interner Wärmetauscher (Rekuperator)
LMTD	Log Mean Temperature Difference
NEST	Demonstrationsgebäude an der Empa
NTU	Number of Transfer Unit
PMV	Predicted Mean Vote, Index für thermischen (Dis-)Komfort
PPD	Predicted Percentage of Dissatisfied person in a room
SFW	Solar, Fitness and Wellness unit in NEST
SWEET	"SWiss Energy research for the Energy Transition", BFE Call
T	Temperatur
WP	Work Package
WT	Wärmetauscher

h	Enthalpie
p	Druck
s	Entropie
ε	Effektivität

1 Einleitung

1.1 Ausgangslage und Hintergrund

Um bis 2050 die Ziele des CO_2 Reduktion in Gebäuden zu erreichen, werden früher oder später die fossilen Heizungen ersetzt werden müssen. In Altbauten, wo eine Isolierung finanziell oder aus Gründen des Heimat- bzw. Ortsbildschutzes nicht oder nur teilweise realisiert werden kann, sind häufig noch sehr hohe Vorlauftemperaturen für die Radiatorenheizungen üblich und nötig [1]. In solchen Gebäuden ist ein Ersatz von fossilen Heizungen durch Standardwärmepumpen in der Regel nicht einfach möglich. Eine Lösung des Problems könnte der Einsatz von Hochtemperaturwärmepumpen darstellen. Da schwach oder nicht isolierte Gebäude typischerweise einen hohen Wärmebedarf haben, ist ein hoher Wirkungsgrad Voraussetzung für einen erfolgreichen Ersatz. Es wäre fatal, wenn in Peak Load Situationen die Heizenergie praktisch rein elektrisch erbracht werden müsste.

1.2 Motivation des Projektes

Einstufige Hochtemperaturwärmepumpen mit dem Kältemittel CO_2 erreichen theoretisch für hohe Temperaturhübe noch gute Wirkungsgrade. So wird z.B. in der solaren Fitness und Wellness Unit im Demonstrationsgebäude NEST [2] an der Empa bei einer Quellentemperatur von 10°C in den Verdampfer und einer Auslasstemperatur von 115°C im Hochtemperatur-Gaskühler ein Gesamt-COP von ca. 2.9 erreicht. Allerdings gelingen solch hohe COPs nur, wenn tiefe Rücklauftemperaturen aus dem Gaskühler 3 möglich sind.

Der COP warmseitig berechnet sich aus der Summe der abgegebenen Wärme der Wärmetauscher a (Punkte 3 → 4), b (Punkte 4 → 5) und c (Punkte 5 → 6) (grüne Beschriftung in Abbildung 2 & 3) im Vergleich zur aufgewendeten elektrischen Energie.

$$COP = \frac{\sum_a^c Q_{Warmwasser}}{Q_{Kompressor}} \quad (1)$$

Dabei ist die aufgewendete elektrische Energie des Kompressors proportional zur Druckerhöhung ((Punkte 2 → 3 in Abbildung 2 & 3) und hängt ab von der Eingangstemperatur und -druck des CO_2 und der Drehzahl des Motors (Anzahl Hübe pro Minute).

Um einen hohen COP zu erreichen, muss auf der Warmseite des Gaskreislaufes möglichst die gesamte Wärme nutzbringend "eingesetzt" werden können. Das ist eine Herausforderung, denn damit in den Wärmetauschern b und c die gesamte Wärme aus dem CO_2 ins Wasser übertragen werden kann, muss die Eintrittstemperatur des Wassers in den Wärmetauscher 3 (WT3 in Abbildung 4) entsprechend tief sein (möglichst <35°C) (Punkt 6, Abbildung 2 & 3). Je tiefer die Rücklauftemperatur des Verbrauchers ist, desto grösser ist die auskoppelbare Wärme aus dem Gas. Das bedeutet, dass das Wärmeabgabesystem im Gebäude entsprechend modifiziert werden muss, damit mehrere und/oder verschiedene Verbraucher mit unterschiedlichen Temperaturanforderungen verwendet werden können.

1.3 Ursprüngliche Projektziele

Das im Antrag formulierte Ziel des Projektes ist es, für (schlecht sanierbare) Altbauten Wärmepumpenalternativen zu fossil befeuerten Heizungen aufzuzeigen, welche vor allem während der Wintermonate einen möglichst hohen COP erreichen, damit der Anteil nichterneuerbarer Energien reduziert werden kann und die elektrischen Netze nicht unnötig belastet werden. Darunter soll aber das Raumklima, vor allem die Strahlungsverteilung, nicht leiden müssen. Die Idee war, Heizsysteme mit transkritischer Hochtemperatur-Wärmepumpen für spezielle Altbauten mit Hilfe von Simulationen mit einer Ölheizung und einer zweistufigen Wärmepumpe zu vergleichen. Vor allem Energieeffizienz und Raumkomfort sollten die wesentlichen Bewertungskriterien darstellen.

Leider konnten diese Ziele aus verschiedenen Gründen nicht erreicht werden:

Technische Gründe: Der Prototyp der CO_2 Wärmepumpe im NEST stellte sich als defekt heraus. Somit konnten die vorhandenen Messreihen nicht verwendet werden, um ein Grey-Box Modell zu erstellen.

Softwaregründe: Das erste entwickelte TRNSYS Modell arbeitete viel zu langsam. Die geplanten Simulationen konnten damit nicht durchgeführt werden.

Personelle Gründe: Die Personalfluktuationen in unserem Lab waren unerwartet hoch. Dadurch konnte das Projekt nur verzögert starten da der Projektverantwortliche in Notfällen bei anderen Projekten einspringen musste.

Das alles führte dazu, dass im Projektverlauf die Ziele angepasst wurden.

1.4 Angepasste Projektziele

Das Ziel des Projektes wurde in dem Sinne reduziert, dass nur noch die Fertigstellung des TRNSYS Types im Vordergrund stand. Der ursprünglich geplante Grey-box Type konnte nicht realisiert werden, da die Messreihen von NEST nicht zur Verfügung standen. Dadurch musste ein komplexerer Type programmiert werden, der weitgehend die gesamten physikalischen Hintergründe abbildete. Dies führte zu einer erheblich aufwändigeren Modellbildung. Dafür ist das Modell nun auch auf weitere einstufige, transkritische CO_2 Wärmepumpen anwendbar. Dieser Type wurde ausführlich beschrieben und wird der TRNSYS Community zur Verfügung gestellt.

2 Vorgehen, Methode, Ergebnisse und Diskussion

2.1 Anlagenbeschrieb

Die Untersuchungen wurden hauptsächlich mit Matlab [3] und dem Simulationstool TRNSYS 18 [4] durchgeführt. TRNSYS arbeitet mit einem internen Solver und einer Sammlung von vorbereiteten Standardmodellen (Types) für Gebäude und Komponenten, welche in internationalen Vergleichen immer wieder validiert wurden und werden. Dies garantiert eine hohe Aussagekraft für energetische Simulationen. Haustechnikkomponenten, die nicht als Standard-Types vorhanden sind, werden selbst entwickelt und in FORTRAN [5] programmiert. In diesem Projekt wird ein neuer Type für die CO_2 Wärmepumpe entwickelt.

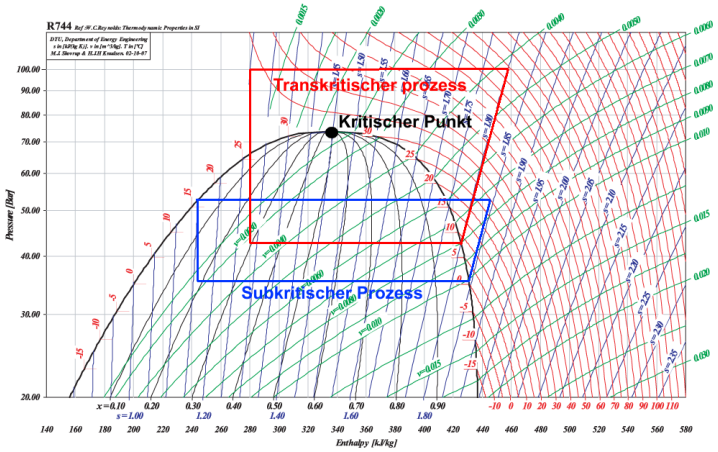

Abbildung 1: Schematische Gegenüberstellung eines transkritischen Prozesses (rot) mit einem subkritischen Prozess (blau).

Als Vorlage für den TRNSYS Type wird die im NEST installierte CO_2 Wärmepumpe verwendet. Die in diesem Bericht beschriebenen CO_2 Hochtemperatur- Wärmepumpen werden grundsätzlich in einem transkritischen Prozess betrieben. Die Erläuterung des Unterschiedes zwischen einem subkritischen

und transkritischen Prozess ist in Abbildung 1 dargestellt. Der transkritische Prozess bedeutet, dass die Wärmeabgabe über der kritischen Temperatur erfolgt und das Kältemittel daher auf der Warmseite keinen Phasenwechsel durchläuft. Das Kältemittel ist in diesem Temperaturbereich während der gesamten Wärmeübertragung gasförmig.

Die im NEST-SFW installierte CO_2 Wärmepumpe wurde für die Speisung von Saunen ausgelegt (sehr hohe Heisswassertemperaturen). Die dort installierte Wärmepumpe muss Heisswasser mit bis zu 115°C erzeugen. Es sind fünf Wärmetauscher im Einsatz (WT1-5 in Abbildung 4), wobei der vierte (WT4) als interner Rekuperator dient (auch Intercooler (IHX) genannt). Mit diesem Rekuperator wird auf der Hochdruckseite das Heissgas weiter abgekühlt (Punkte 6 → 7 in Abbildung 2 & 3) und auf der Niederdruckseite wird das Gas nach dem Verdampfer vorgewärmt (Punkte 1 → 2 in Abbildung 2 & 3). So wird sichergestellt, dass keine Flüssigkeitströpfchen in den Kompressor gelangen.

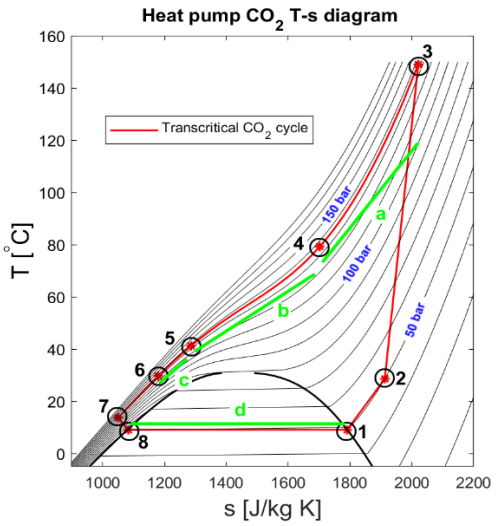

Abbildung 2: Temperatur – Entropiediagramm eines Hochtemperatur CO_2 Kreislaufes

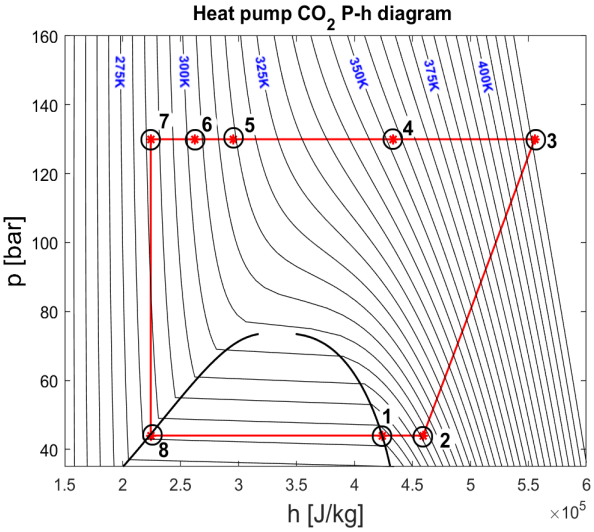

Abbildung 3: Druck – Enthalpiediagramm des gleichen Hochtemperatur CO_2 Kreislaufes

Relevante Komponenten im CO_2 Kreislauf:

1 – 2: Gaserwärmung mit innerem Wärmetauscher (Rekuperator), Wärme kommt von 6 – 7, (WT4, IHX)

2 – 3: Kompressor (adiabate Komprimierung)

3 – 4: Heissgaskühler (a), (WT1, GC1)

4 – 5: Gaskühler mittlere Temperaturen (b), (WT2, GC2)

5 – 6: Gaskühler tiefe Temperaturen (c), (WT3, GC3)

6 – 7: Rekuperator, Wärme geht nach 1 – 2, (WT4, IHX)

7 – 8: Drosselventil (isenthalpe Entspannung)

8 – 1: Verdampfer (isobare Verdampfung) (d), (WT5, Verdampfer)

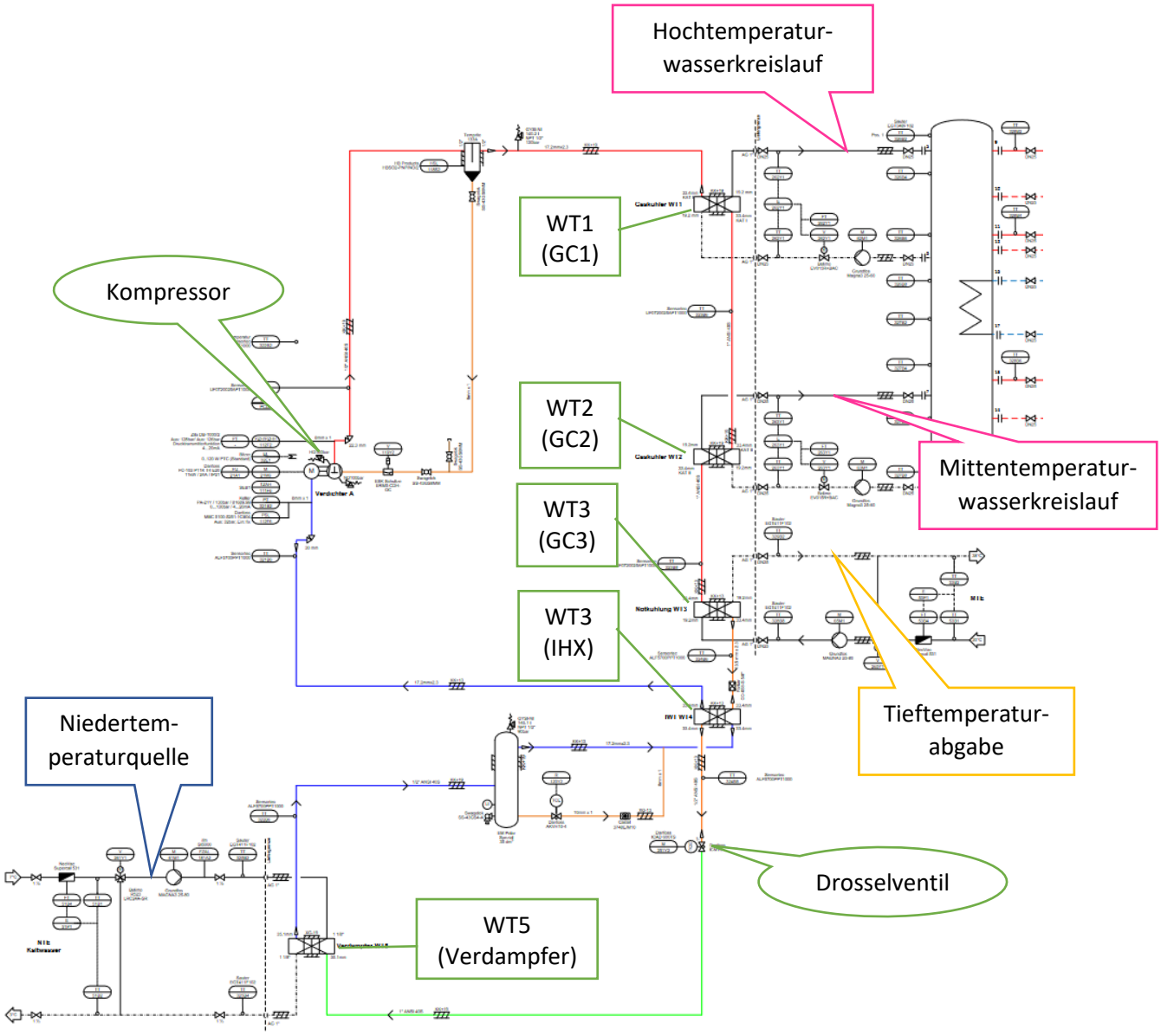

Abbildung 4: Hydraulisches Schema der CO_2 Wärmepumpe im NEST

Die Wärmekapazität von gasförmigem CO_2 im verwendeten Temperatur/Druckbereich ist nicht linear. Dies kann in den Abbildungen 2 & 3 erkannt werden (rote Kurven). Im T-s Diagramm (Abbildung 2) wird die Abkühlung des Gases isobar dargestellt (T3 → T7). Deutlich ist zu erkennen, dass diese Kurve kein Gerade ist. Die Abkühlung von T3→T4 beträgt 60°C und die von T4→T5 40°C. Im p-h Diagramm (Abbildung 3) ist ergänzend sichtbar, dass die Enthalpiedifferenz zwischen den Punkten T3→T4 trotz grösserer Temperaturdifferenz kleiner ist als die Enthalpiedifferenz zwischen den Punkten T4→T5. Es ist wichtig zu erkennen, dass die Linearität der Wärmekapazität auf den Isobaren mit tieferem Druck noch weiter abnimmt. Dieser Effekt kommt von der Temperaturabhängigkeit der Wärmekapazität des CO_2 und tritt in der Nähe des kritischen Punktes verstärkt auf. Da die Wärmekapazität von Wasser in diesem Temperaturbereich hingegen nur schwach von der Temperatur abhängt, kann dessen Temperaturänderung im T - s Diagramm in erster Näherung linear eingezeichnet werden (grüne Geraden in Abbildung 2). Die Steigung dieser Geraden hängt vom Wassermassendurchfluss ab. Die Konsequenz dieses unterschiedlichen Verhaltens der Wärmekapazitäten hat einen grossen Einfluss auf die Auslegung der verwendeten Gaskühler (Wärmetauscher). Ein einzelner Wärmetauscher im System kann nicht gleichzeitig hohe Temperaturen und hohen Wärmefluss kombinieren. Durch die Verwendung von drei Wärmetauschern anstelle von einem kann der Massenstrom pro Wärmetauscher und die Tauscherfläche so eingestellt werden, dass eine optimale Wärmeübertragung vom Gas aufs Wasser stattfinden kann (Kurve vom Gas und vom Wasser im T – s Diagramm möglichst parallel und nahe beieinander).

Beim bestehenden System im NEST wird das Wasser des Schichtspeichers für die Beheizung der drei Saunen (Finnisch, Bio und Dampf) sowie für Duschwasser verwendet. Der Speicher hilft die unterschiedlichen Massenströme der Wärmetauscher auszugleichen und erlaubt dank der Schichtung, der Wärmepumpe möglichst tiefe Vorlauftemperaturen im Punkt T6 zuzuführen. Wie in Abbildung 4 sichtbar, durchströmt das vom Kompressor komprimierte Gas der Reihe nach den Hochtemperatur- (WT1), den Mittentemperatur- (WT2) und den Tieftemperatur-Wärmetauscher (WT3) sowie den Rekuperator (WT4). Das Heisswasser von WT1 und WT2 wird in einen Schichtspeicher geleitet, das Wasser von WT3 wird im NEST während der Wintermonate für Niedertemperaturheizung (Flächenheizung) eingesetzt und im Sommer für die Regenerierung von Erdsonden. Die Niedertemperaturwärme für den Verdampfer wird im Winter von der Erdsonde und im Sommer durch die Raumkühlung geliefert.

In diesem Projekt soll nun untersucht werden, wie der CO_2-Kreislauf, die Heizwasserrücklauf- und Vorlauftemperaturen, der Gasdruck (CO_2) vor und nach dem Kompressor sowie die Wärmequellen und -senken angepasst werden müssen, damit alte Gebäude mit solchen Wärmepumpen beheizt werden können. Um das zu erreichen, wird der CO_2-Kreislauf, wie er in Abbildung 4 dargestellt ist, in einem Simulationsmodell abgebildet. Mit diesem kann ermittelt werden, wie die Parameter verändert werden müssen, damit eine einstufige transkritische CO_2 Wärmepumpe optimal eingesetzt werden kann.

2.2 Durchgeführte Arbeiten und Ergebnisse

Im WP1 wurden Gebäude und Strukturen evaluiert. Je nach Schutzwürdigkeit dieser Gebäude, wird die Auswahl der erlaubten Wärmeabgabekomponenten (Radiator, Konvektor…) durch die Denkmalpflege eingeschränkt. Dabei ist zu beachten, dass in repräsentativen Räumen häufig gar keine Veränderung oder nur "unsichtbare" Veränderungen durchgeführt werden dürfen. Das bedeutet, dass bestehende Gussradiatoren sehr häufig die einzigen erlaubten Heizelemente in solchen Räumen sind. Z.T besteht aber die Möglichkeit, in Nebenräumen Flächenheizungen oder Konvektoren einzusetzen. Abgesehen von den Restriktionen bei den Wärmeabgabesystemen sind häufig auch die Möglichkeiten der Wärmeisolation stark eingeschränkt.

Zusammen mit der Denkmalpflege des Kantons Zürich wurde für die vergleichenden Simulationen ein Gebäude ausgewählt, das im Erdgeschoss Büroräume und im Obergeschoss Wohnungen aufweist. Es handelt sich um das Kutscherhaus der Villa Patumbah [6]. Da dieses Gebäude erst kürzlich renoviert wurde, waren Pläne und Angaben des physikalischen Aufbaus der Wände vorhanden. Dieses Gebäude war vorgesehen, um verschiedene Heizabgabesysteme zu evaluieren und bot den Vorteil, dass es zugleich Wohnungen und Büro abdeckte.

In WP2 wurden Haustechnikkomponenten zusammengetragen, die für einen erfolgreichen Einsatz der CO_2 Wärmepumpen in Altbauten benötigt würden.

Altbauten sind sehr oft mit Gussradiatoren ausgerüstet. Diese Radiatoren weisen sehr häufig Einlauftemperaturen von bis zu 90°C auf. Da diese Radiatoren eine kleine Gesamtoberfläche aufweisen im Vergleich zu Stahlblechradiatoren oder Konvektoren, bleibt auch der Anteil der konvektiven Wärmeabgabe gering. Das führt dazu, dass die Wärme zum grossen Teil als Strahlung abgegeben wird. Zudem ist das Wasservolumen in diesen Radiatoren hoch, was dazu führt, dass die Rücklauftemperatur aus den Radiatoren im Rahmen von hohen 65°C bis 80°C liegt. Diese Temperatur ist aber zu hoch, damit einstufige transkritische Wärmepumpen betrieben werden können. Die oberst mögliche Einlauftemperatur des Wassers in den Gaskühler 3 liegt bei ca. 55°C, sollte idealerweise aber bei < 35°C liegen.

Alte und nicht effiziente Radiatoren auszuwechseln gegen z.B. Konvektoren, die eine tiefere Vor- und Rücklauftemperatur und einen höheren Konvektiven Anteil an der Wärmeabgabe erreichen, wird aber z.T. von der Denkmalpflege nicht erlaubt und birgt zudem den Nachteil, dass das Raumklima unangenehm wird. Die Strahlungswärme (der Gussradiatoren) wird vor allem an die Oberflächen der Wände abgegeben (aufheizen der Wandoberflächen), die konvektive Abgabe der Wärme geschieht an die Luft. Alte Bauten dürfen oder können häufig nicht oder nur teilweise nachisoliert werden. Die Verwendung von konvektoren führt zu kalten Wandoberflächen im Raum. Wen vor allem die Luft aufgeheizt wird, bleiben die Oberflächen der Wände kühl. Diese Situation führt zu eine sogenannten Barackenklima.

Dies gilt es zu vermeiden. Eine mögliche Lösung dieses Problems ist die geschickte Kombination vom verschiedenen Wärmeabgabesystemen über das Haus verteilt.

Meistens müssen nicht in allen Räumen die Gussradiatoren erhalten bleiben. In Räumen, die einen kleinen Anteil an Aussenwänden aufweisen, kann unter Umständen ein Ersatz mit Konvektoren durchgeführt werden. Ausserdem ist es z.T. möglich, dass Flächenheizungen mit Kapillarrohren unter die Wandoberfläche eingebaut werden. Wenn Innenisolationen durchgeführt werden, so dass nicht mehr die gesamte Wandoberfläche kühl ist, können auch kleine wassergespiesene Umluftheizungen oder Konvektoren mit Zusatzventilatoren eingesetzt werden. Wenn es gelingt, dass die hydraulische Anordnung so geändert werden kann, dass eine Serienschaltung von Gussradiatoren mit einer Flächenheizung, Konvektor oder einer Umluftheizung möglich ist, können die nötigen tiefen Vorlauftemperaturen in den Gaskühler 3 erreicht werden.

Nicht nur die Wärmeabgabesysteme sind anzupassen, auch auf der Quellenseite müssen Überlegungen gemacht werden, wie das System optimal betrieben werden kann. Wenn die Wärmequelle Temperaturen von ca. -10°C unterschreitet, schafft es die einstufige CO_2 Wärmepumpe kaum mehr, genügend hohe Temperaturen für die Gussradiatoren bei genügend hoher Leistung zu liefern (der Temperaturhub wird zu gross). Grundsätzlich ist zu beachten, dass alte Gebäude in der Regel einen sehr hohen Wärmebedarf aufweisen. Dies bedeute auch, dass die Quellen entsprechend leistungfähig sein müssen.

Erdsonden als Quelle sind (meistens) unproblematisch. Systeme, welche nur mit Luftwärmetauscher ausgeführt werden, sind aber u.U. überfordert. Zwar kann Wärme aus dem Speicher verwendet werden für das Auftauen der Wärmetauscher. Dies hätte zudem den Vorteil, dass die Vorlauftemperatur in Gaskühler 3 abgesenkt würde. Aber es ist nicht zu übersehen, dass der Einlassdruck des CO_2 bei einem einstufigen System in den Kompressor zu tief würde. Da müssten Ansätze diskutiert werden, welche ev. Erdsonden oder Eisspeicher mit Luftwärmetauschern kombinieren. Erdsonden bzw. Eisspeicher könnten Tieftemperaturspitzen der Aussenluft kompensieren.

Für das Gesamtsystem muss noch mit einem Warmwasserspeicher gerechnet werden. Da die Wärmepumpe in Spitzenzeiten eine gewisse Leistung erbringen muss, ist in Teillastzeiten ein Speicher erforderlich, damit genügend lange Laufzeiten der Wärmepumpe erreicht werden können (eine Frage der Lebensdauer). Aber auch für den Ausgleich der Wasserströme durch die drei Gaskühler ist eine Vorrichtung notwendig. Deser Ausgleich kann nur über ein Speicher erfolgen.

Im WP3 wurden die Module programmiert, die für die Simulation der verschiedenen Szenarien benötigt würden. Das Wichtigste ist ein korrekter digitaler Zwilling der CO_2 – Wärmepumpe. Da für diese Art von Wärmepumpe auf keine bestehenden Vorarbeiten zurückgegriffen werden konnte (weder in TRNSYS noch in Matlab), musste dieses Modul (oder Type wie das in TRNSYS genannt wird) von Grund auf neu entwickelt werden. Zu diesem Zweck wurden theoretische Untersuchungen und die Messdaten der im NEST SFW bestehenden Wärmepumpe (Prototyp) ausgewertet. Ziel war es, die physikalischen Zusammenhänge zu verstehen und Korrelationen zu erhalten, welche in das mathematische Modell übernommen werden können. Um Zeit zu sparen war es angedacht, die physikalischen Korrelationen nur wo nötig einzusetzen. Zusammen mit Messdaten sollte ein Greybox Modell erstellt werden. Es wurde eine Masterarbeit ausgeschrieben und wir konnten dafür Hendrik Leliveld gewinnen, der ein kombiniertes Praktikum und Masterarbeit auf diesem Thema durchgeführt hat [7].

Die Auswertung der Messdaten und die Erstellung eines ersten Modells vom Prototyp wurde mit Hilfe von Matlab durchgeführt. Matlab ist erheblich besser als TRNSYS geeignet, grundlegende Modell- und Parameteridentifikationen durchzuführen. Die Idee war es, das erarbeitete Matlabmodell später so zu modifizieren, damit es in TRNSYS angewendet werden könnte. Dabei sind wir auf unerwartete Hindernisse gestossen. Schon die detaillierte Auswertung der Daten führte zu unerklärlichen Resultaten. Um Zufallseffekte auszuschliessen, wurden dann Betriebseinstellungen gewählt, die immer wieder auftreten. Eine dieser Einstellungen war die Bereitstellung von Heisswasser für die Heizung der finnischen Sauna (110°C), die andere gewählte Betriebseinstellung war für die Produktion von Brauchwarmwasser (Dusche, 56°C). Bei diesen Auswertungen wurden rund 190 Betriebspassagen à mindestens zwanzig Minuten Betrieb für die Duschwasseraufbereitung (Abbildung 5) und rund 80 Passagen für die Saunaheizung ausgewertet. Bei diesen Passagen wurden rund 10 Minuten Einlaufzeit und Auslaufzeit abgeschnitten, um Einschwingvorgänge und Abschaltvorgänge mit variierenden Drehzahlen des

Kompressors zu eliminieren. Im Weiteren durften sich die Betriebsparameter über die angeschaute Zeit nur geringfügig ändern. Damit konnten störende Abweichungen durch dynamische Effekte stark reduziert werden.

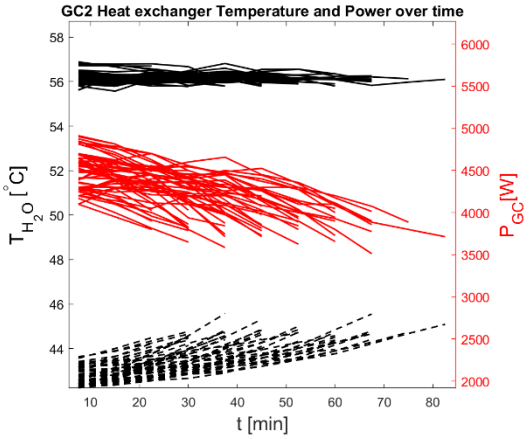

Abbildung 5: Verdampferleistung (T8 – T1) (rot) für eine Heisswassertemperatur von 56°C (schwarz)

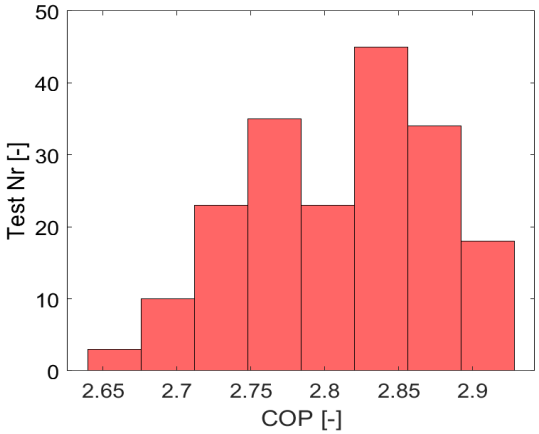

Abbildung 6: Resultierende COP's bei 56°C

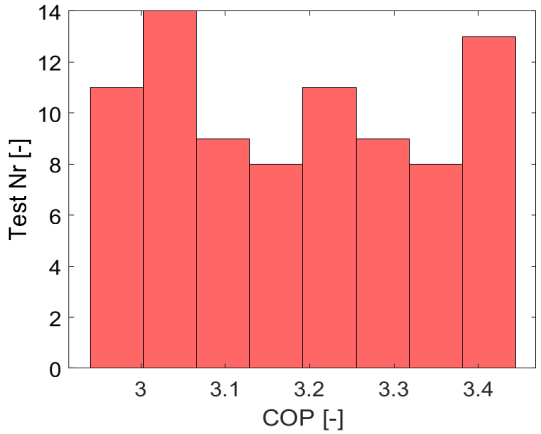

Abbildung 7: Resultierende COP's bei 110°C

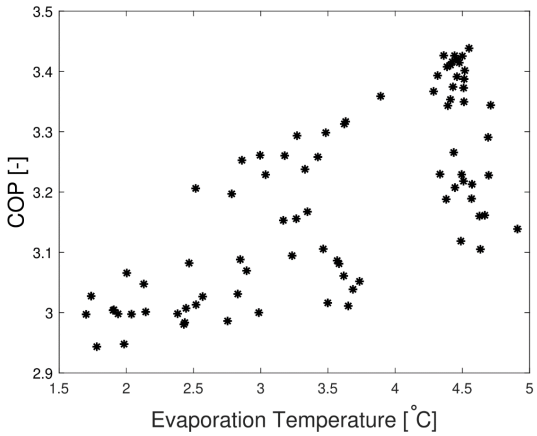

Abbildung 8: COP für Saunaheizung in Abhängigkeit der Verdampfertemperatur

Wir stellten fest, dass bei der Produktion von Wasser mit einer Temperatur von 110°C (Saunaheizung)(bei einer Quellentemperatur von ca. 5°C ein COP von durchschnittlich 3.18 erreicht wird (Abbildung 7). Überraschenderweise zeigte die Auswertung aber auch, dass bei der Produktion von tieferen Warmwassertemperaturen (Duschwasser mit 56°C) der COP tiefer ausfiel (2.81, Abbildung 6), was für uns aufgrund theoretischer Überlegungen nicht nachvollziehbar war. Im Rahmen der Masterthesis wurde deshalb ein Matlab Modell erstellt, das in der Lage ist, stationäre Betriebszustände der Wärmepumpe zu berechnen. Dabei wurde der ganze CO_2 Kreislauf ideal simuliert, d.h. Wärmeverluste von Gasleitungen und Wärmetauschern wurden (noch) nicht berücksichtigt. Mit diesem Modell wurden Simulationen in den beiden Temperaturbereichen 56°C und 110°C durchgeführt und mit den Messwerten verglichen, um zu ermitteln, von wo die Diskrepanz beim COP herkommen könnte. Da im NEST ist die Quelltemperatur relativ stabil auf ca. 5-8°C geregelt wird, wurde für das Matlab Modell ebenfalls eine konstante Quelltemperatur verwendet. Die Suche nach den Ursachen dieser Diskrepanz ging in verschiedene Richtungen. Untersucht wurden Abhängigkeiten vom Kompressionsdruck des CO_2, Verdampfungstemperaturen (Abbildung 8), Linearität der Wärmekapazität von CO_2 und der damit verbundenen Einschränkungen in den Wärmetauschern, Regelstrategien, Frequenzabhängigkeit des Kompressors (Abbildung 9), Zusammenspiel der 3 Gaskühler, Einfluss der Austrittstemperatur aus dem

Gaskühler 3 (Abbildung 10), etc. Überall zeigte es sich, dass bei einer niederen Temperatur eigentlich ein höherer COP zu erwarten gewesen wäre, eine klare Erklärung konnten wir nicht finden. Auch die Kennzahlen, die der Hersteller verwendet hatte, um die Funktionsfähigkeit der Anlage zu testen, zeigten keine Anomalie, bzw. konnten das Phänomen nicht erklären.

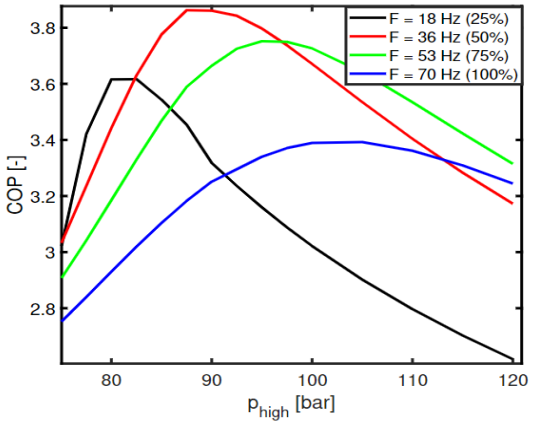

Abbildung 9 Theoretischer COP in Abhängigkeit der Frequenz und des Druckes

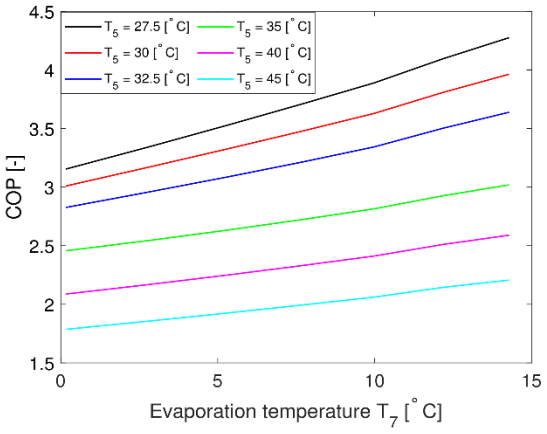

Abbildung 10: Theoretischer COP in Abhängigkeit der Verdampfertemperatur und der CO_2 Austrittstemperatur aus dem letzten Gaskühler

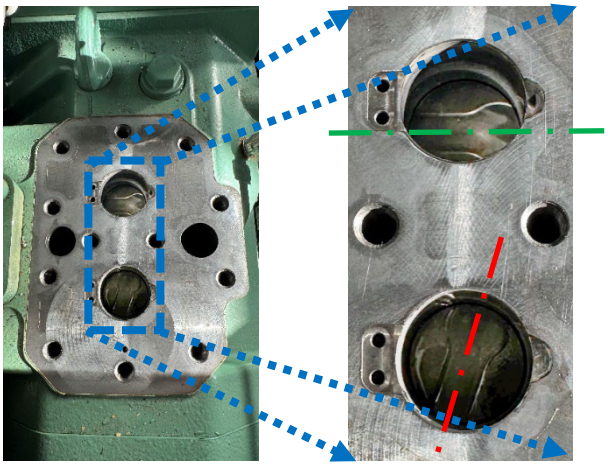

Abbildung 11: Schädigung im Kompressor. Der obere Kolben arbeitet korrekt (grüne Achse), der untere ist abgedreht (d.h. abgebrochen) und im Zylinder eingeklemmt (rote Achse)

Abbildung 12: Zerbrochene Pleuelstange des unteren Kolbens

Erst als wir Zugriff auf das Dimensionierungstool von Bitzer [8] bekamen (Hersteller des Kompressors), fiel auf, dass die absoluten Leistungen (elektrisch und thermisch) nur etwa halb so gross waren wie ursprünglich bei der Planung der Anlage dimensionierten Leistungen (Abbildung 13). Dies war versteckt geblieben, da die Regelung den fehlende Volumenstrom des defekten Kolbens zum Teil mit höheren Frequenzen ausgeglichen hatte (in Abbildung 13 vor allem bei der 56°C Variante sichtbar). Zudem wurde realisiert, dass der CO_2 Massenstrom, der mit Hilfe des Matlab Modells berechnet wurde, nur etwa halb so gross war, wie der vom Dimensionierungstool berechneten Massenstrom. Die Resultate des CO_2 Massenstromes liessen vermuten, dass ein Ventil nicht korrekt arbeitete. Nach dem Öffnen der Maschine zeigte sich aber, dass sich ein Kolben in der falschen Position befand (Abbildung 11). Beim Demontieren zeigte sich, dass der Pleuel gebrochen war und dass sich der Kolben im Zylinder festgesetzt hatte (Abbildung 12).

Ein solcher Defekt darf als sehr selten betrachtet werden. Im Normalfall würde ein defekter Pleuel die Seitenwand des Gehäuses perforieren. Zudem würde der Kolben ins Gehäuseinnere fallen und der Kompressor könnte überhaupt keine hohen Drücke mehr erzeugen. So aber arbeitete nur noch ein Kolben und ein Teil des Druckgases ging über den defekten Kolben verloren. So kann die Diskrepanz im COP erklärt werden. Bei Teillast (tiefere Frequenz als Regelvorgabe um die tiefere Temperatur zu erreichen) ging ein höherer Anteil des Gasstromes über den defekten Kolben verloren als beim Arbeiten unter Volllast.

Die Maschine konnte erst rund drei Monate nach Feststellung des Fehlers repariert werden, da die Ersatzteile nicht verfügbar waren. Die Untersuchungen, die gemacht wurden, um die Diskrepanz des COP's zu finden, sind in der Masterarbeit von H.C. Leliveld [7] dokumentiert.

Als nächster Schritt wurde untersucht, wann dieser Defekt aufgetreten war. Aufgrund der Messwerte in der NEST-Datenbank konnte der genaue Zeitpunkt ermittelt werden. Es war der 27.11.2018. Die Konsequenz dieser Entdeckung war für dieses Projekt gross. Zu diesem Zeitpunkt waren die Saunen noch nicht im Regelbetrieb, d.h. es gab keine brauchbaren Zeitreihen des Systems, mit denen wir unsere Simulationen kalibrieren konnten. Alle Messreihen, die in der Datenbank abgespeichert waren, sind mit der defekten Wärmepumpe durchgeführt worden. Somit sind alle Auswertungen der Messungen im NEST, die in [7] beschrieben sind, mit Vorsicht zu betrachten.

Die Entwicklung des Wärmepumpenmodells in Matlab verlief ebenfalls nicht besonders geradlinig. Die Komponenten wurden zuerst stark vereinfacht, später aber immer detaillierter modelliert. Die verfeinerten Modelle wurden jeweils wieder mit den Messungen verglichen und man stellte fest, dass die Abweichungen immer noch vorhanden waren. Zu dem Zeitpunkt ging man noch davon aus, dass das Modell Fehler aufwies und dachte nicht an einen Defekt in der Wärmepumpe. Nachdem der Fehler bekannt war, ist klar geworden, dass nicht die Modelle ein Problem hatten. Damit konnten die vorhanden Modelle für die weiteren Arbeiten verwendet werden.

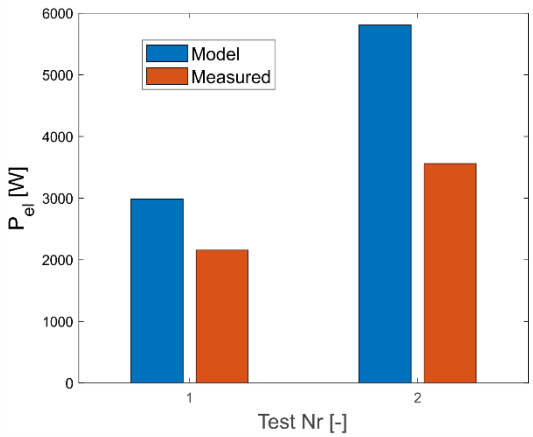

Abbildung 13: Modellierte bzw. gemessene Leisungsaufnahme bei 56°C und bei 110°C

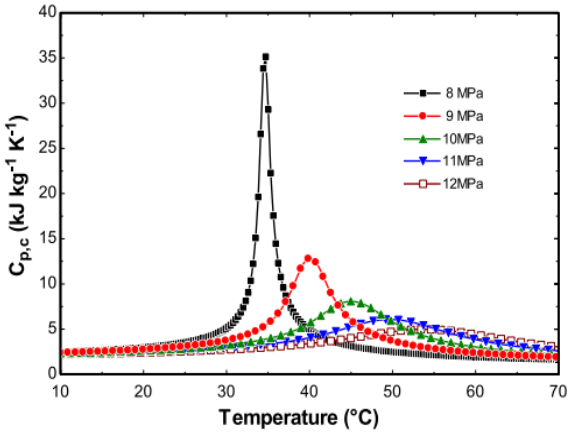

Abbildung 14: Abhängigkeit der Wärmekapazität von CO_2 von Druck und Temperatur

Die Entwicklung der einzelnen Komponenten für das Matlabmodell der Wärmepumpe werden nachfolgend kurz erläutert. Für die Modellierung der Stoffwerte von CO_2 wurde auf die CoolProp [9] Datenbank zugegriffen und für die Stoffwerte von Wasser wurden Werte des VDI-Wärmeatlas [10] verwendet.

Kompressormodell: Zuerst wurde der Kompressor aufgrund der gemessenen Werte als ein Greybox Modell simuliert. Dabei wurde eine grobe dreidimensionale Matrix aus Messwerten erstellt. Input ins Modell waren elektrischen Leistung, Auslassdruck und Einlasstemperatur, Output waren Massenstrom und Auslasstemperatur (linearisierte Werte). Zu diesem Zeitpunkt wurde der Einlassdruck in den Kompressor noch als konstant angenommen (40.0 bar). Später wurde dieses doch eher rudimentäre Modell durch polynomiale Ansätze des Kompressorherstellers ersetzt. Diese Polynome wurden durch die Bitzer Software [8] für verschiedene Betriebszustände erstellt und entsprechen dem Standard EN12900 / ARI

540. Allerdings stellte sich heraus, dass die elektrische Effizienz des Motors mit 100% zugrunde gelegt wird. Dies wurde von uns angepasst.

Wärmetauscher: Die Gaskühler und der Intercooler wurden zuerst nach der LMTD-Methode berechnet. Im Vergleich zu den Messwerten zeigte es sich aber bald, dass diese Methode sehr ungenau war. Vor allem Gaskühler 2 und 3 zeigten falsche Resultate, da die LMTD-Methode über den Temperaturbereich (beinahe) konstante Wärmekapazitätswerte voraussetzt. Für CO_2 ist das aber nicht der Fall (siehe Abbildung 14). Dagegen darf für Wasser eine mittlere Wärmekapazität verwendet werden, da diese Wärmekapazität sich über den Temperaturverlauf nur geringfügig ändert. Als Ersatz für die LMTD-Methode wurde für die Gaskühler (WT1-3 in Abbildung 4) und den Intercooler (WT4 in Abbildung 4) die ε-NTU Methode eingeführt. Mit dieser Methode kann der einzelne Wärmetauscher in "Zellen" aufgeteilt werden und jede dieser Zellen kann einzeln berechnet werden. Je nach Wärmetauscher wurden zwischen 50 und 250 Zellen eingeführt. Allerdings erhöhte sich dadurch die benötigte Zeit zur Berechnung des CO_2 Kreislaufs drastisch. Ein einzelner Run (Zustand) benötigte mit Matlab ca. 10 Minuten. Dabei wurde der Verdampfer noch nicht dynamisch berechnet. Er lieferte konstant 40 bar bei 8°C Austrittstemperatur.

Die Wärmeverluste des Ölabscheiders und der Gaskühler wurden im Matlab Modell nicht berücksichtigt.

Bevor die Wärmepumpe an der Empa eingebaut wurde, wurde sie in einem Testprüfstand auf ihre Leistungsfähigkeit überprüft. Diese Resultate wurden mit unserem Matlab Modell verglichen. Das Modell wies eine rund 6-13% zu hohe Wärmeleistungsfähigkeit aus, je nachdem ob Teillast oder Volllast betreiben wurde. Diese Abweichung kann erklärt werden mit den nicht berücksichtigten thermischen Verlusten im Modell. Aufgrund dieser Resultate wurde beschlossen, das Matlab Modell nach TRNSYS (interne Bezeichnung: Type8501) zu portieren, denn die Jahressimulationen mit Gebäude und Haustechnikkomponenten kombiniert sind nur in TRNSYS möglich. Nach TRNSYS portieren bedeutet, dass eine neue FORTRAN Subroutine geschrieben werden musste [5].

Im Type 8501 wurden dann Wärmeverluste des Ölabscheiders berücksichtigt und der Verdampfer wurde dynamisch nach der LMTD-Methode berechnet. Dieses Modell wurde zusammen mit einem 2000 Liter Wasserspeicher und einem "Schuhboxgebäudemodell" getestet. Das Schuhboxgebäudemodell war ein Quader mit 5 x 5 x 2.5 Meter Abmessungen,15 cm Betonwand auf allen Seiten und dem Wetter von der SMA Zürich ausgesetzt. Die Heizung bestand aus einem Radiator. Der Zeitschritt der Simulation war mit einer Stunde angesetzt. Pro Zeitschritt (1 Stunde) benötigte das Modell rund 40 Minuten! Das ist für Ganzjahressimulation absolut ungeeignet und auch unbrauchbar, um Variantenstudien zu erstellen. Im Nachhinein hat sich die Entscheidung, das Matlab Modell auf TRNSYS zu migrieren als ungünstig erwiesen. Das TRNSYS Modell musste neu konzipiert und von Grund auf neu erstellt werden, wobei schon gewisse Lehren aus der Portierung des ersten Modells gezogen werden konnten (Type 8505). So wurden im zweiten Modell überall Timer eingebaut, die genau Auskunft geben konnten, wie lange welche Funktion oder Subroutine zur Abarbeitung benötigte.

Der grösste Zeitverzögerung kam daher, dass die Stoffwerte von CO_2 bei jeder Iteration von jedem Knoten der Wärmetauscher aus der CoolProp Datenbank abgefragt wurden. Um Zeit zu sparen, wurden daher vorberechnete Matrizen mit Stützpunkten der Wärmekapazitäten und Enthalpien in Abhängigkeit von Druck und Temperatur vorberechnet und in Dateien abgespeichert. Diese Dateien werden vor der Simulation eingelesen und im RAM "zwischengelagert". Die Stoffwerte wurden berechnet, indem die Stützpunkte ausgelesen werden und Zwischenwerte linearisiert berechnet werden[10] .

Eine weitere Zeitersparnis konnte erreicht werden, indem die Anzahl Zellen in der Wärmetauscherberechnung dynamisch angepasst wurden. Zellen wurden nur noch aufgeteilt, wenn die Wärmekapazitätsdifferenzen und die Temperaturdifferenzen der Ein- und Auslasstemperaturen der Zellen 10% bzw. 5% überschritten haben. Neu wurde aber eine maximale Anzahl von rund 1030 Hellen zugelassen. Zudem wurde der ganze Ablauf der Berechnung des CO_2 Kreislaufes neu erstellt und angepasst. Insgesamt führte das zu einer gewaltigen Reduktion des Zeitbedarfs im neuen Type8505. Pro Zustand werden nur noch ca. 0,8 bis 1,2 Sekunden gebraucht, womit jetzt Simulation von verschiedenen Gebäudevarianten auch mit Jahressimulationen möglich sein sollten. Der Zeitbedarf für eine Simulation wurde erheblich reduziert, allerdings ist die Genauigkeit ebenfalls etwas reduziert. Unter der Annahme, dass der Type 8501 genau rechnet (Abbild des Matlab Modells mit identischen Resultaten bei gleichen Randbedingungen) zeigt der Type 8505 Abweichungen von max. 0.2%.

Dieser Type wurde mit verschiedenen Anfangs- und Randbedingungen auf Stabilität getestet. Leider konnte der Type nicht mit realen Messwerten kalibriert werden. Im Herbst 2022 kam für die Schweiz die Energiemangellage und der Aufruf zum Sparen von Elektrizität. Die Direktion der Empa verfügte, dass die Saunalandschaft im NEST vorläufig nicht betrieben werden darf. Nach der Reparatur der Wärmepumpe im Sommer 2023 wurde die Wärmepumpe nicht mehr im Hochtemperaturmodus eingesetzt. Als man im September 2025 endlich eine Messreihe mit Hochtemperatur durchführen wollte, stellte man fest, dass das Energiemessgerät des Gaskühler 1 ausgefallen ist. Obwohl eine Reparatur in Auftrag gegeben wurde, ist das neue Gerät aktuell (Ende November 2025) immer noch nicht einsatzbereit. Der Type8505 wurde so aufbereitet, dass er grundsätzlich für alle einstufigen, transkritischen CO_2 Kompressoren verwendet werden kann, die Polynome nach dem Standard EN12900 / ARI 540 zur Verfügung stellen (was bei der Firma Bitzer der Fall ist). Die ausgelesenen Resultatfiles mit den Polynomen müssen allerdings für jede Pumpe angepasst werden und in 4 Files übertragen werden. Das Vorgehen wird in der Dokumentation des Types 8505 erläutert [12].

WP4 wäre dafür gedacht gewesen, die Simulationen der verschiedenen Kombinationen von WP2 vergleichend zu beurteilen. Da hätten einerseits die CO2 Emissionen aber auch die Folgen für das Raumklima untersucht werden müssen. Aus den oben genannten Gründen war aber die Erstellung des TRNSYS Types im vorgesehenen zeitlichen Rahmen nicht möglich, Aus diesem Grund wurde mit dem BFE eine Änderung der Anforderungen an das Projekt und eine teilweise Rückzahlung der Unterstützung vereinbart.

3 Schlussfolgerungen und Ausblick

Für den Prototypen der CO_2 Wärmepumpe wurde in Matlab ein vereinfachtes mathematisches Modell erstellt, welches den idealen CO_2 Kreislauf simulieren kann. Dieses Modell hat geholfen, unrealistische Messwerte zu identifizieren und war der Ausgangspunkt zur Portierung des ersten TRNSYS Modells (Type8501). Mit der Hilfe des Matlab Models konnte auch nachgewiesen werden, dass die Wärmepumpe im NEST seit Jahren defekt war und nur die halbe Leistungsfähigkeit aufwies. Aufgrund der Analyse des Defektes konnte dann schlüssig erklärt werden, wieso bei den Auswertungen der Wärmepumpe die COPs nicht unseren Erwartungen entsprachen. So wurde im defekten Zustand bei den höchsten Frequenzen (70Hz) die zur Produktion von Heisswasser (bis 115°C) nötig waren ein COP von ca. 3.2 ausgewiesen. Bei den tiefen Frequenzen für die Produktion von Warmwasser (ca. 56°C) lagen die COPs zu tief bei ca. 2.8 statt der erwarteten 3.4. Bei beiden Temperaturen war die Leistung um bis zu 50% reduziert. Dies ist allerdings nicht früher aufgefallen, da der Defekt zu einem Zeitpunkt auftrat, bevor vollständige Messreihen durchgeführt werden konnten und damit korrekte Referenzleistungen nicht verfügbar waren. Das Matlab Modell wurde nach TRNSYS migriert (Type8501) und so ausgebaut, damit Wärmeverluste nach dem Kompressor berücksichtigt werden können und damit der Verdampfer mit variablen Quelltemperaturen umgehen kann. Allerdings zeigte es sich, dass dieser Type ein so hoher Zeitbedarf für eine Simulation benötigt, dass er nicht für die vorgesehenen Aufgaben in diesem Projekt verwendet werden konnte. Type8501 wurde analysiert und es wurde ein von Grund auf neues Modell erarbeitet, das für TRNSYS programmiert und schrittweise optimiert wurde (Type8505).

Der fertig dokumentierte Type wird der TRNSYS Community zur Verfügung gestellt.

4 Nationale und internationale Zusammenarbeit

Mit der Firma Scheco Winterthur bestand eine enge Zusammenarbeit. Als Lieferant des Prototypen kennt Scheco die thermodynamischen und technischen Herausforderungen dieser CO_2 Wärmepumpen sehr gut. Mit Rolf Löhrer (ehemaliger Geschäftsführer von Scheco) bestand ein intensiver Austausch, nicht nur wegen des Defektes an der Wärmepumpe im NEST, sondern auch um mögliche sinnvolle Konfigurationen mit transkritischen Wärmepumpen in Gebäuden zu finden.

Vom Institut IEFE der ZHAW (Frank Tillenkamp), haben wir Unterstützung erhalten, um die Wärmepumpe korrekt zu modellieren. Vor allem, als völlig unklar war, wieso unsere Modelle nicht mit den Messresultaten übereinstimmen, haben wir vom Institut IEFE Hilfe für die Überprüfung unserer Modelle bekommen in Form von Anregungen, Tipps und auch kritischen Fragen.

Für die Auswahl eines geeigneten Gebäudes in den Simulationen standen wir in Kontakt mit der Kantonalen Denkmalpflege des Kantons Zürich (Lukas Knörr). Er hat uns das Nebengebäude der Villa Patumbah als mögliches Test- Objekt vorgeschlagen. Mit ihm hatten wir auch Diskussionen, welche Massnahmen in einem Gebäude möglich sein könnten.

Durch die Teilnahme am SWEET DeCarbCH WP5 [13] wurde eine nationale Zusammenarbeit im Bereich der Hochtemperaturwärmepumpen gewährleistet.

5 Publikationen und andere Kommunikation

Innerhalb diese Projektes wurde ein Masterarbeit durchgeführt [7].

Das Projekt wurde an der Wärmepumpentagung vom 14.6.2023 in Burgdorf unter dem Titel "Effizienter Einsatz von Hochtemperatur- Wärmepumpen in Altbauten und bei Sanierungen" mit den damals zur Verfügung stehenden Resultaten vorgestellt.

6 Literaturverzeichnis

[1] Wohlleben M., Moeri S. et al: Energie und Baudenkmal, Teil III: Haustechnik, Kantonale Denkmalpflege Bern und Kantonale Denkmalpflege Zürich, V1 – 2014

[2] Solare Fitness & Wellness SFW, NEST, Empa Dübendorf, https://www.empa.ch/web/nest/sfw

[3] The MathWorks Inc. (2021). MATLAB version: 9.11.0 (R2021b), Natick, Massachusetts: The MathWorks Inc. https://www.mathworks.com

[4] TRNSYS: TRansient SYstem Simulation Tool, Vers. 18.01, Thermal Energy System Specialists, Madison, Wisconsin 53703

[5] FORTRAN: Intel® Fortran Compiler 19.0 for Windows, Intel corporation

[6] K. Artho, G. Weber, Villa Patumbah, eds., Die Villa Patumbah in Zürich: Geschichte und Restaurierung, Kanton Zürich, Baudirektion, Amt für Raumentwicklung, Kantonale Denkmalpflege, Zürich, 2014.

[7] H.C. Leliveld: Efficient use of a high temperature transcritical CO_2 heat pump system, Master thesis, Dübendorf, 03.03.2023

[8] BITZER Software Ver. 6.17.9, Build 2773. url: https://www.bitzer.de/websoftware/. Download vom 30.01.2023. Später ersetzt durch Ver. 6.18.0, Build 2811. (21.02.2023)

[9] Bell IH, Wronski J, Quoilin S, Lemort V. Pure and Pseudo-pure Fluid Thermophysical Property Evaluation and the Open-Source Thermophysical Property Library CoolProp. Ind Eng Chem Res 2014;53:2498–508. https://doi.org/10.1021/ie4033999.

[10] Verein Deutscher Ingenieure, Gesellschaft Verfahrenstechnik und Chemieingenieurwesen, editors. VDI-Wärmeatlas: Berechnungsblätter für den Wärmeübergang. 8., überarb. und erw. Aufl. Berlin: Springer; 1997.

[11] S.A. et al Klein, TRNSYS, Ver. 18.01.0000, A Transient System Simulation Program, (2017). http://sel.me.wisc.edu/trnsys. Programmer's Guide, 7.4.4.2. CoolProp_Fluid_Properties( )

[12] R. Weber, Transcritical water-water CO_2 heat pump, TRNSYS Type8505, Ver 1.0, Documentation, Dübendorf, 02/08/2026 (im Anhang beigefügt)

[13] SWEET DeCarbCH WP5: BFE SWEET, Decarbonisation of Cooling and Heating in Switzerland, WP5: Combination of renewables, heat transformation and storage for medium and high temperature heating as well as cooling, 2021-2025

7 Anhang

Wie schon oben erwähnt werden für die Simulation mit dem Type8505 diverse Datenfiles benötigt. Diese Files werden zusammen mit dem Code, der *.dll und dem Proforma File veröffentlicht. Die Veröffentlichung mit der *.dll ist darum sinnvoll, weil nicht jedermann im Besitze eines FORTRAN Compilers ist. Im Weiteren wird ein Proforma File beigelegt, mit dem die Schnittstellen im Simulation Studio von TRNSYS definiert werden.

In diesem Anhang wird die Dokumentation und der FORTRAN Code des Type8505 eingefügt. Stand der Arbeiten ist 08.02.2026.

7.1 Documentation TYPE 8505

Documentation TYPE 8505
Transcritical water-water CO_2 heat pump
TRNSYS Type8505

CO_2 Heat pump © Empa 2021

Dübendorf, 02/08/2026

Ver 1.0

Robert Weber

This work was supported by Empa and the Swiss Federal Office of Energy SFOE in the frame of the project HiTemHP, contract no. SI/502168-01.
The authors of this documentation are solely responsible for its content.

Summary

This TRNSYS Type 8505 calculates the energy flows in transcritical CO_2 heat pumps that may reach high temperatures (up to approx. 140°C). The compressor performance is calculated using regressions according to the EN12900 / ARI 540 standard. The frequency of the motor can be varied. The frequency and the pressure of the CO_2 on the high-pressure side are specified as set parameters for controlling. Heat transfer fluid (HTF) mass flows and their inlet temperatures are specified for the simulation. Three heat exchangers (gas coolers) are situated on the high-pressure side and dissipate heat, the heat exchanger on the low-pressure side (evaporator) absorbs low-temperature heat. An additional internal heat exchanger serves as a recuperator and prevents CO_2 in droplet form from entering the compressor. The results of the simulation are the electrical demand of the compressor, the heat converted in the gas coolers and evaporator, as well as the outlet temperatures from the heat exchangers.

Nomenclature

A	Area	i	continuous counter of cells
ARI	Air Cond. and Refrigeration Institute	IHX	Internal heat exchanger
CO_2	Carbon dioxide	INP	Input to the Type
COP	Coefficient of performance	K	Kelvin
Cp	Specific heat capacity	LMTD	Logarithmic mean temperature difference
csv	Comma-separated values, file format	ln	logarithmus naturalis
dat	data file, space-separated values	$\dot{m}$	Mass flow
Edot	Electrical power	Max	Maximum
EER	Energy efficiency ratio	Min	Minimum
Eff	Efficiency	n	total number of cells
EN	European standard	NTU	Heat transfer coefficient
Eq.	Equation	OS	Oil separator
Evap	Evaporator	OUT	Outlet of the Type
GC	Gas cooler	p	pressure
h	Enthalpy	PAR	Parameter input to the Type
H_2O	Water	Q	Heat
HTC	Heat transfer coefficient	$\dot{Q}$	heat flux
HTF	Heat transfer fluid	R	Thermal resistance
HX	Heat exchanger	U	Heat transfer coefficient
Hz	Hertz	VDI	Verein Deutscher Ingenieure

Symbols

α	Convective heat transfer coefficient	λ	Specific heat transfer coefficient
Δ	Temperature difference per node	ϑ	Temperature
ε	Effectiveness		

7.1.1. PARAMETER/INPUTS/OUTPUTS Reference

The following tables explain the Parameters, Inputs and Outputs to and from Type 8505.

PARAMETER

1	Logical unit for data file	[-]	Logical unit for the data file with the matrix of parameter sets to calculate the CO_2 mass flow in the heat pump. (LU_mflow_data)
2	Logical unit for data file	[-]	Logical unit for the data file with the matrix of parameter sets to calculate the electrical power of the compressor. (LU_E_Pow_data)
3	Logical unit for data file	[-]	Logical unit for the data file with the matrix of parameter sets to calculate the pressure limits in which the polynomial relations are defined. (LU_P_Limit)
4	Logical unit for data file	[-]	Logical unit for the data file with the matrix of enthalpy and heat capacity of CO_2 in the range of 60° to 180°C and 74.0 to 140.0 bar. (LU_180_60_data)
5	Logical unit for data file	[-]	Logical unit for the data file with the matrix of enthalpy and heat capacity of CO_2 in the range of 20° to 60°C and 74.0 to 140.0 bar. (LU_60_20_data)
6	Logical unit for data file	[-]	Logical unit for the data file with the matrix of enthalpy and heat capacity of CO_2 in the range of -33.2° to 62.8°C and 12.0 to 40.0 bar. (LU_12_40_dat)
7	Logical unit for data file	[-]	Logical unit for the data file with the matrix of enthalpy and heat capacity of CO_2 in the range of -33.2° to 62.8°C and 40.0 to 70.0 bar. (LU_40_70_dat)
8	Logical unit for data file	[-]	Logical unit for the data file with the matrix of temperature, enthalpy and pressure of CO_2 close to the phase change in the range of -33.2° to 62.8°C. (LU_PC_Limit)
9	A_HX1	[m^2]	Heat exchanger area of the high temperature gas cooler (HX1)
10	HTC_HX1	[kJ/hr.m^2.K]	Heat transfer coefficient of the high temperature gas cooler (HX1) between CO_2 (gas) and water (liquid).
11	A_HX2	[m^2]	Heat exchanger area of the medium temperature gas cooler (HX2)
12	HTC_HX2	[kJ/hr.m^2.K]	Heat transfer coefficient of the medium temperature gas cooler (HX2) between CO_2 (gas) and water (liquid).
13	A_HX3	[m^2]	Heat exchanger area of the low temperature gas cooler (HX3)
14	HTC_HX3	[kJ/hr.m^2.K]	Heat transfer coefficient of the low temperature gas cooler (HX3) between CO_2 (gas) and water (liquid).
15	A_HX4	[m^2]	Heat exchanger area of the inter cooler (HX4)
16	HTC_HX4	[kJ/hr.m^2.K]	Heat transfer coefficient of inter cooler (HX4) between CO_2 (gas) and CO_2 (gas).
17	A_HX5	[m^2]	Heat exchanger area of the evaporator (HX5)
18	HTC_HX5	[kJ/hr.m^2.K]	Heat transfer coefficient of the evaporator (HX5) between CO_2 (gas) and water (liquid).
19	HTC_boil_HX5	[kJ/hr.m^2.K]	Heat transfer coefficient of the evaporator (HX5) between CO_2 (boiling/phase change) and water (liquid).
20	Highest Frequency	[1/s]	Maximal frequency allowed for the compressor

21	Lowest Frequency	[1/s]	Minimal frequency allowed for the compressor
22	Cp_cold_HTF	[kJ/kg.K]	Average heat capacity of the heat transfer fluid into the evaporator. If cp = 100, then the heat capacity of water is calculated internally (for T_H2O_in_HX5 > 0°C)
23	Freeze_Temp_HTF	[°C]	Temperature, where the HTF freezes
24	Tol_Q_Deviat	[-]	If the difference of the total heat flow through the membrane compared with the enthalpy difference on the ho side of the gas cooler or IHX are smaller than Tol_Q_Deviat, the gas cooler iteration might be stopped as the precision is reached
25	Tol_cp_ratio	[-]	Safety criteria if cp difference of CO_2 between inlet and outlet of the cell becomes to high
26	Tol_dT_ratio	[-]	Temperature criteria to reduce size of nodes -->Cell to divide
27	Tol_Phi_iter	[-]	If the difference of the heat flows on the hot and cold side of the gas cooler or IHX are smaller than Tol_Phi_iter, eps-NTU iteration in the cells will be stopped.
28	Tol_Q_Def_Tot	[-]	Stop criteria of the difference of heat flows in iterations of the whole refrigerant cycle. The ration of the cycle before and actual cycle are compared.
29	Tol_area_split	[-]	Criteria, how precise the area split in the evaporator between phase change part and overheating part must be calculated to meet the overheating temperature given in INP 11
30	Lambda	[kJ/hr.m.K]	Insulation conductivity of the tube and oil separator insulation.
31	Ins_thick_tube	[m]	Thickness of the insulation layer for the tube
32	Ins_thick_OS	[m]	Thickness of the insulation layer for the oil separator
33	h_out_tube	[kJ/hr.m².K]	Convective heat transfer coefficient outside of the pipe and the oil separator
34	h_in_tube	[kJ/hr.m².K]	Convective heat transfer coefficient inside of the pipe and the oil separator
35	Diam_Tube	[m]	Outside diameter of the pipe (without insulation)
36	Tub_1_Leng	[m]	Tube length between compressor and oil separator
37	Tub_2_Leng	[m]	Tube length between oil separator and first heat exchanger
38	Diam_OS	[m]	Outside diameter of the oil separator (without insulation)
39	Hight_OS	[m]	Hight of the oil separator wall

INPUTS

1	P_High_Set	[bar]	Desired CO_2 pressure behind the compressor on the high pressure side. This changes the behavior of the CO2 circle.
2	Frequency Set	[1/s]	Desired frequency of the compressor
3	T_H2O_in_HX1	[°C]	Inlet temperature of water into the high temperature gas cooler (HX1)
4	T_H2O_in_HX2	[°C]	Inlet temperature of water into the medium temperature gas cooler (HX2)
5	T_H2O_in_HX3	[°C]	Inlet temperature of water into the low temperature gas cooler (HX3)
6	T_H2O_in_HX5	[°C]	Inlet temperature of HTF (water or water glycol mix) into the evaporator (HX5)

7	m_H2O_in_GC1	[kg/hr]	Inlet mass flow of water into the high temperature gas cooler (HX1). If m_H2O_in_GC1 = 0.0, HX1 is not calculated.
8	m_H2O_in_GC2	[kg/hr]	Inlet mass flow of water into the medium temperature gas cooler (HX2). If m_H2O_in_GC2 = 0.0, HX2 is not calculated.
9	m_H2O_in_GC3	[kg/hr]	Inlet mass flow of water into the medium temperature gas cooler (HX3). If m_H2O_in_GC3 = 0.0, HX3 is not calculated.
10	m_HTF_in_Evap	[kg/hr]	Inlet mass flow of HTF into the evaporator (HX5). If m_HTF_in_Evap = 0.0, the compressor is switched off (Mode 2).
11	T_Over_heat	[K]	This is the expected overheating temperature of CO_2 at the end of the evaporator
12	T_Freeze_save	[K]	The HTF outlet temperature of the evaporator must be higher than Freeze_Temp_HTF + T_Freeze_save. This typically in the range of 3K.
13	Mech_Eff	[0 .. 1]	Efficiency of the combination of electrical engine and compressor
14	On_Off	[0, 1]	0 = heat pump is off; 1 heat pump is on.
15	Temp_Air	[°C]	Air temperature inside heat pump housing

OUTPUTS

1	Electric power	[kJ/hr]	Electric power needed by the compressor
2	Total heat	[kJ/hr]	Total heat delivered with HX1, HX2, and HX3
3	CO_2 mass flow	[kJ/hr]	CO_2 mass flow in CO_2 cycle
4	High pressure CO_2	[bar]	High pressure of CO_2 reached. If there is any reason to reduce the set pressure, it will be done automatically. The maximum allowed pressure is indicated here.
5	Frequency	[1/s]	Reached frequency of the compressor after reducing speed in case of limitations
6	Heat from HX1	[kJ/hr]	Heat delivered by CO_2 in HX1 (difference of the inlet and outlet enthalpy times the mass flow), (high temperature gas cooler GC1).
7	T_{H2O}, inlet HX1	[°C]	H_2O inlet temperature into HX1, (high temperature gas cooler GC1).
8	T_{H2O}, outlet HX1	[°C]	H_2O outlet temperature from HX1, (high temperature gas cooler GC1).
9	m_{H2O}, HX1	[kg/hr]	H_2O mass flow in HX1, (high temperature gas cooler GC1).
10	Heat from HX2	[kJ/hr]	Heat delivered by CO_2 in HX2 (difference of the inlet and outlet enthalpy times the mass flow), (medium temperature gas cooler GC2).
11	T_{H2O}, inlet HX2	[°C]	H_2O inlet temperature into HX2, (medium temperature gas cooler GC2).
12	T_{H2O}, outlet HX2	[°C]	H_2O outlet temperature from HX2, medium temperature gas cooler GC2).
13	m_{H2O}, HX2	[kg/hr]	H_2O mass flow in HX2, (medium temperature gas cooler GC2).
14	Heat from HX3	[kJ/hr]	Heat delivered by CO_2 in HX3 (difference of the inlet and outlet enthalpy times the mass flow), (low temperature gas cooler GC3).
15	T_{H2O}, inlet HX3	[°C]	H_2O inlet temperature into HX3, (low temperature gas cooler GC3).

16	T_{H2O}, outlet HX3	[°C]	H_2O outlet temperature from HX3, low temperature gas cooler GC3).
17	m_{H2O}, HX3	[kg/hr]	H_2O mass flow in HX3, (low temperature gas cooler GC3).
18	Heat to Evaporator	[kJ/hr]	Heat uptake by CO_2 in the evaporator (difference of the inlet and outlet enthalpy times the mass flow). Both enthalpy differences from the phase change part and overheating part of the heat exchanger are considered.
19	T_{H2O}, inlet Evaporator	[°C]	H_2O inlet temperature into the evaporator.
20	T_{H2O}, outlet Evaporator	[°C]	H_2O outlet temperature from the evaporator.
21	m_{H2O}, Evaporator	[kg/hr]	H_2O mass flow in the evaporator.
22	T_{CO2}, compressor	[°C]	CO_2 outlet temperature behind compressor.
23	T_{CO2}, hot tube	[°C]	CO_2 outlet temperature behind the oil separator and the hot gas tube. Identically with CO_2 inlet temperature to HX1
24	T_{CO2}, behind HX1	[°C]	CO_2 outlet temperature behind HX1. Identically with CO_2 inlet temperature to HX2.
25	T_{CO2}, behind HX2	[°C]	CO_2 outlet temperature behind HX2. Identically with CO_2 inlet temperature to HX3.
26	T_{CO2}, behind HX3	[°C]	CO_2 outlet temperature behind HX3. Identically with CO_2 inlet temperature to IHX, high pressure side (HP).
27	T_{CO2}, , behind IHX (high pressure)	[°C]	CO_2 outlet temperature behind the IHX (internal heat recuperator), high pressure side.
28	T_{CO2}, PC in evaporator	[°C]	CO_2 phase change temperature inside the evaporator
29	T_{CO2}, behind evaporator	[°C]	CO_2 outlet temperature behind the evaporator. Identically with CO_2 inlet temperature to IHX, low pressure side (LP).
30	T_{CO2}, behind IHX (low pressure)	[°C]	CO_2 outlet temperature behind the IHX (internal heat recuperator), low pressure side (LP).
31	Heat transfer IHX	[kJ/hr]	Heat recuperated in the IHX.
32	Low pressure CO_2	[bar]	Low pressure of CO_2 (defined by phase change temperature).
33	Emergency stop	[0, 1]	1 => emergency stop
34	COP	[-]	Warm side efficiency factor of the heat pump. Consider the heat gained from GC1, GC2, and GC3. As there are internal heat losses calculated, $COP - EER \neq 1.0$
35	EER	[-]	Cold side efficiency factor of heat pump

7.1.2. Explanation of the refrigerant cycle

The refrigerant, in this case CO_2, passes through a closed cycle driven by a mechanical compressor. Pressure differences are generated, which heat or cool the gas. The high-temperature heat is removed from the cycle via the heat exchangers (GC1 - GC3). The low-temperature heat is added to the circuit via the evaporator (Figure 1). While conventional mechanical heat pumps with conventional refrigerants operate in the subcritical range, high-temperature CO_2 heat pumps are operated in the transcritical range (Figure 2).

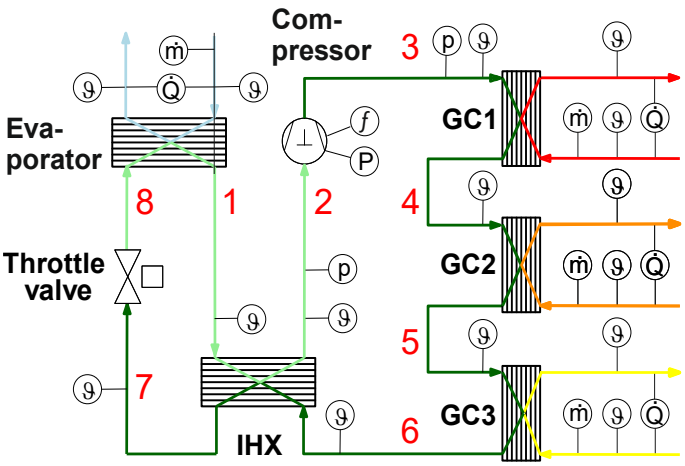

Figure 1: Schematic of the CO_2 circle used in the Type.

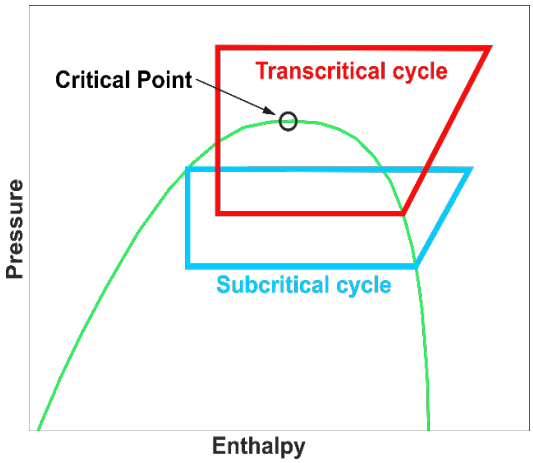

Figure 2: Schematics of a transcritical and subcritical circle in the P-H diagram.

The most significant difference between transcritical and subcritical heat pump types concerns heat dissipation. In contrast to conventional heat pumps, transcritical heat pumps do not operate at a constant temperature during phase change (gas condensation), but instead the hot gas is cooled over a wide temperature range (Figure 3 & Figure 4). Depending on the size of the gas coolers (GC1 – GC3) and the inlet conditions of the HTF (mass flow and inlet temperature), high outlet temperatures of the HTF in gas cooler 1 can be achieved.

The closer the cooling curve of the gas passes the critical point (the cooling curve depends on the gas pressure on the high-pressure side), the higher the change in the specific heat capacity of the gas over the temperature range from point 3 to 6 (Figure 4 & 5). As a consequence, the gas coolers cannot be calculated using an approximation equation (LMTD), as this assumes a (nearly) constant specific heat capacity over the entire temperature range. In addition, a decision must be made as to how many gas coolers to use. With only one gas cooler, it is only possible to achieve either a high outlet temperature with low heat transfer or a low outlet temperature with high heat transfer (or something in between). However, it is not possible to achieve high outlet temperatures and high heat transfer with only one

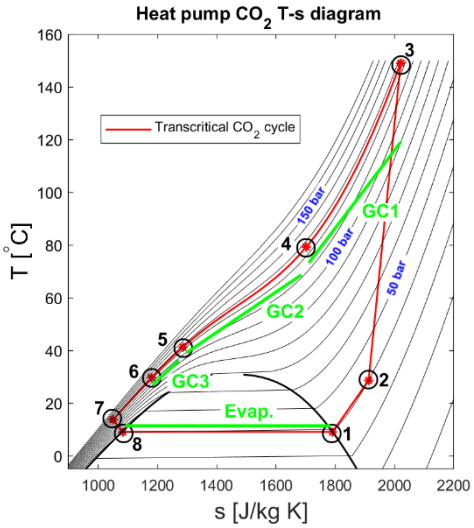

Figure 3: Temperature – entropy diagram of a circle with R744 (CO_2). Green: Temperature of the HTF.

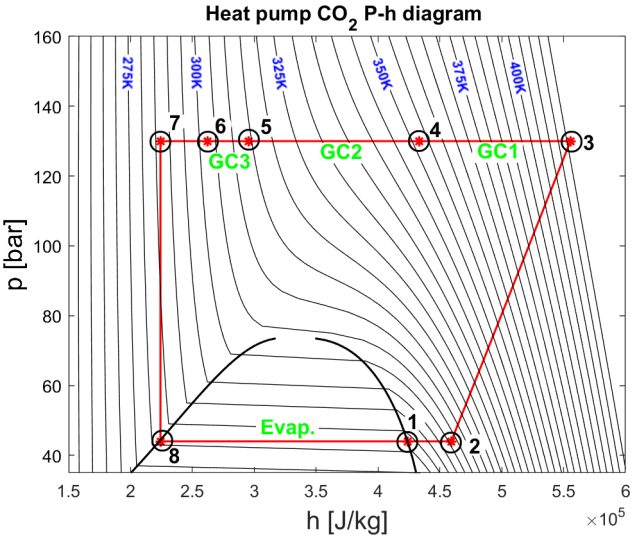

Figure 4: Same circle but in the pressure – enthalpy diagram.

heat exchanger (Figure 6). This can only be achieved with several heat exchangers. The Type 8505 therefore supports up to three gas coolers.

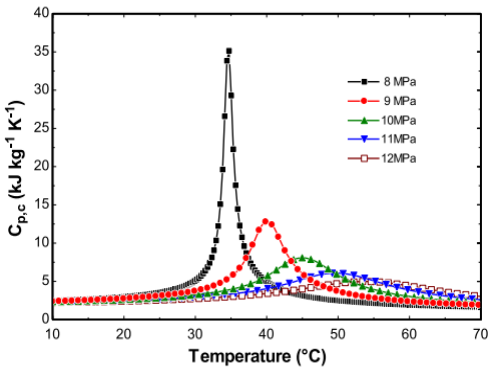

Figure 5: The heat capacity changes with distance to the critical point

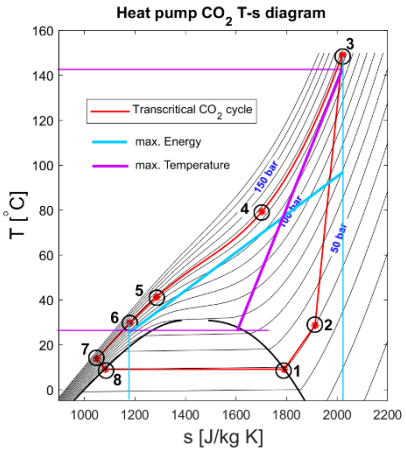

Figure 6: Extract maximum energy (blue line) or maximum heat (magenta) with only one heat exchanger.

INP 1 can be used to set the CO_2 pressure on the high-pressure side of the cycle. The pressure determines the maximum gas temperature as well as the behavior of the cooling curve. This can be seen in Figure 6. Higher pressure also results in a less curved cooling curve, but more electrical energy is required to reach the high pressure (point 3). This pressure must be optimized depending on the application. In a real machine, the high pressure on the outlet side of the compressor is adjusted using the throttle valve. Type 8505 therefore supports up to three gas coolers.

The mass flow in the cycle is determined by the speed (INP 2) and size of the compressor, as well as by the inlet pressure and inlet temperature of the CO_2 in the compressor. The low inlet pressure results

from the phase change temperature in the evaporator and is therefore dependent on the HTF inlet temperature and its mass flow.

7.1.3. The components in the refrigerant cycle and their functions

The various components in the coolant cycle are connected with steel pipes. These pipes are simulated as ideal conductors, i.e., any pressure drop in the pipes is not taken into account. In addition, heat losses through the pipe wall (except for pipe 3) are not calculated. The points mentioned below correspond in reality to the pipe connections in the heat pump (Figure 1). Since these pipes are assumed to be ideal, they can be reduced to points in Figure 3 & Figure 4 (i.e., no change in the physical values inside the pipe). The important components are situated between the points:

2 – 3: Compressor (adiabatic compression)

3 – 4: Hot gas cooler, (GC1)

4 – 5: Medium temperature gas cooler, (GC2)

5 – 6: Low temperature gas cooler, (GC3)

6 – 7: Internal heat exchanger (IHX), heat goes to 1 – 2

7 – 8: Throttle valve (isenthalpic expansion)

8 – 1: Evaporator (isobaric evaporation), (Evap.)

1 – 2: Internal heat exchanger IHX, heat comes from 6 – 7, evaporates any droplets

Compressor

Compressors designed for CO_2 as a refrigerant and for high gas pressures and temperatures, whose compressor performance data is available with polynomials in accordance with EN12900 / ARI 540, can be used for simulation. Gas compression is assumed to be an adiabatic process in the simulation. Any thermal and electrical losses are taken into account via INP 13.

The compressor is controlled by specifying the rotational frequency of the electric motor. However, it is important to understand that the electrical power consumed depends not only on the frequency of the motor, but also on the high and low gas pressure of the refrigerant and the intake temperature of the gas into the compressor. This determines how much CO_2 flows through the cycle and how much power is required to increase the pressure. On the other hand, the outlet temperature from the compressor is determined by the pressure on the high-pressure side, which is controlled by the throttle valve.

Gas cooler

Gas coolers are air-water heat exchangers that can withstand high pressures and high temperatures. It is assumed that counterflow heat exchangers are used. In these heat exchangers, the heat from the hot gas coming from the compressor is transferred to water. As mentioned above, in this Type it is possible to extract the heat with 1, 2, or 3 gas coolers.

If only one gas cooler is used, it is physically impossible to achieve high outlet temperatures and a high heat transfer. This is because water has a largely constant heat capacity over the expected

temperature range, whereas CO_2 does not. Especially in the high temperature range (>70°C), the heat capacity of CO_2 is lower than in the medium temperature range (~ 30 - 45°C). If a larger water flow is set, the entire energy of the gas can be transferred to the water in the medium temperature range, and a temperature gradient between the gas and water remains. In the high temperature range, there is a much higher temperature gradient between the gas and the water, but the gas does not contain enough energy to heat the water to a higher temperature. In contrast, with a smaller water flow, not all of the energy from the low temperature range can be transferred because there is no longer a temperature gradient between the CO_2 and the water. However, in the higher temperature range, the energy of the gas is sufficient to achieve high temperatures because less water needs to be heated.

If several gas coolers are used, different water flows can be employed. When using two gas coolers, a higher water flow can be used in the lower temperature range and a lower water flow in the higher temperature range, thus achieving high outlet temperatures at the hot gas cooler and a high heat transfer across both gas coolers taken together.

In order to achieve a high COP for the heat pump, it is very important that the low-temperature heat of the hot gas can also be used. For this reason, it makes sense to use a third gas cooler, which is only used for low-temperature heat extraction. This allows for more flexible adjustments of the heat demand to the respective temperature spread of the gas coolers.

Internal heat exchanger (IHX)

The internal heat exchanger is an air-to-air heat exchanger that must withstand high pressures and is also designed as a counterflow heat exchanger. It is also referred to as a recuperator. Its main task is to ensure that any droplets from the evaporator are vaporized. Otherwise, damage could occur if liquid coolant enters the compressor. At the same time, this also cools the hot gas a little further before it enters the throttle valve.

The throttle valve

The throttle valve is electrically driven and is regulated to the specified high gas pressure. The high pressure after the compressor is a control variable of the heat pump and can therefore be used to change the behavior of the heat pump. The expansion of the refrigerant is modeled as an isenthalpic process. Ideally, the entire refrigerant would be liquefied. In reality, however, only part of it is converted into liquid, meaning that a mixture of gas and liquid enters the evaporator.

The Evaporator

Evaporators are counterflow heat exchangers. In these heat exchangers, the liquefied refrigerant is evaporated using low-temperature ambient heat. Geothermal heat exchangers, air heat exchangers, or waste heat can be used as heat sources. For sources that can drop below 0°C, a heat transfer fluid with antifreeze must be used to prevent damage to the evaporator.

In principle, the evaporator can also be used directly as an air heat exchanger for ambient air, but this Type does not have a defrost feature.

In real machines, a liquid separator is inserted into the cooling circuit after the evaporator and before the IHX. This separator stores liquid coolant and thus compensates for different phase change

temperatures. However, as this is not relevant for pure energy considerations over longer periods of time, the liquid separator is not simulated in this Type.

The oil separator

Between the compressor and the hot gas cooler (GC1) there is a component that, ideally, is not directly involved in energy exchange. The oil separator serves only to separate the lubricating oil that is carried along by the refrigerant as it exits the compressor and return it to the compressor. If a large amount of this lubricating oil were to enter the gas cooler, the heat transfer there could be impaired. However, this component is very hot (up to 180°C, directly after the compressor) and has a large surface area. It therefore loses a considerable amount of heat to the environment, especially if it is insufficiently insulated. Care must be taken to ensure that the insulation can withstand high temperatures over a longer period of time. For this reason, the heat loss of the oil separator and the pipes coming from the compressor and continuing to the hot gas cooler is taken into account in the simulation.

7.1.4. The programming of the individual components

Compressor

The compressor is modeled using standardized regressions according to EN12900 / ARI 540 for calculating electrical power and mass flow. The polynomial used is shown in Eq. 1:

$$y = c_1 + c_2 \cdot \vartheta_0 + c_3 \cdot p_{high} + c_4 \cdot \vartheta_0^2 + c_5 \cdot \vartheta_0 \cdot p_{high} + c_6 \cdot p_{high}^2 + c_7 \cdot \vartheta_0^3 + c_8 \cdot \vartheta_0^2 \cdot p_{high} + \cdots + c_9 \cdot \vartheta_0 \cdot p_{high}^2 + c_{10} \cdot p_{high}^3 \quad \text{(Eq. 1)}$$

The parameters c_1 to c_{10} are calculated in a pre-processing step using the program provided by the manufacturer of the compressor [1] (see “Creation of data files” below). The following variables are varied step by step:

- Motor speed (frequency in [s^{-1}])
- Gas temperature after the last gas cooler [°C]
- and the temperature range between phase change and inlet temperature into the compressor in [K]

The resulting parameters and the limits of the validity range are summarized in input files. At the beginning of the simulation, parameters c_1 to c_{10} are read from the input files into RAM and linearized during the simulation using the TRNSYS routine “InterpolateData”.

Equation 1 is used to calculate the electrical power of the compressor and the mass flow of the refrigerant. The phase change temperature ϑ_0 required for this is calculated in the evaporator, and the high pressure p_{high} (INP 1) is a specification for controlling the heat pump. To calculate the mechanical power of the compressor, an efficiency factor is used that takes into account the internal losses (friction and electrical losses) in the compressor (INP 13).
If it becomes apparent during the iteration process that ice is forming in the evaporator, the preset frequency of the compressor is reduced until no more ice can form (OUT 5). If the frequency cannot be reduced sufficiently, the heat pump is switched off for this time step (OUT 33 =>1).
The gas pressure p_{low} upstream of the compressor is determined by the phase change temperature in the evaporator. The gas temperature upstream of the compressor ϑ_2 is the outlet temperature from

the IHX low pressure side. Using these values, CoolProp [2] can be used to calculate the enthalpy h_2 of the CO_2 upstream of the compressor (Eq. 2).

$$h_2 = f(p_{low}, \vartheta_2) \quad \text{(Eq. 2)}$$

$$h_3 = h_2 + \frac{P_{el} \cdot \varepsilon_{mech}}{\dot{m}_{CO_2}} \quad \text{(Eq. 3)}$$

$$\vartheta_3 = f(p_{high}, h_3) \quad \text{(Eq. 4)}$$

The enthalpy h_3 can be calculated using the mechanical power and the mass flow of the refrigerant (Eq. 3). The pressure p_{high} is a simulation parameter and, together with the enthalpy h_3 CoolProp can be used to determine the CO_2 temperature ϑ_2 after the compressor (Eq. 4).

The oil separator

To determine the heat losses in the oil separator and the associated pipes, the specific thermal resistance R_{tube} and R_{lid} (cover of the oil separator) are calculated first (Eq. 5) and (Eq. 6). The thermal resistance of the metal pipe is neglected because the thermal conductivity of the metal is high and the wall thickness is low.

$$R_{tube} = \frac{1}{radius_{in} \cdot \alpha_{in} \cdot \pi} + \frac{Ln\left(\frac{radius_{in}}{radius_{out}}\right)}{2 \cdot \lambda_{insulation} \cdot \pi} + \frac{1}{radius_{out} \cdot \alpha_{out} \cdot \pi} \quad \text{(Eq. 5)}$$

$$R_{lid} = \frac{1}{\alpha_{in}} + \frac{thick_{insulation}}{\lambda_{insulation}} + \frac{1}{\alpha_{out}} \quad \text{(Eq. 6)}$$

$$Q_{loss,tube} = \frac{Length \cdot (\vartheta_{CO2} - \vartheta_{Air})}{R_{tube}} \quad \text{(Eq. 7)}$$

$$Q_{loss,OS} = \frac{Hight \cdot (\vartheta_{CO2} - \vartheta_{Air})}{R_{tube}} + 2 \cdot \frac{radius_{out}^2 \cdot \pi \cdot (\vartheta_{CO2} - \vartheta_{Air})}{R_{lid} \cdot 2} \quad \text{(Eq. 8)}$$

$$\vartheta_{loss} = \vartheta_3 - f\left(p_{high}, h_3 - \frac{Q_{loss,OS}}{\dot{m}_{CO_2}}\right) \quad \text{(Eq. 9)}$$

The heat loss of the pipes and the side wall of the oil separator as well as the cover of the oil separator are calculated using (Eq. 7) and (Eq. 8). The temperature loss ϑ_{loss} is determined for each component (pipes, oil separator) based on the respective enthalpy loss (Eq. 9).

Gas cooler and IHX

The procedure for the three gas coolers and the IHX is identical. The simulations are based on a piecewise ε-NTU method according to [3]. The aim is to calculate the outlet temperatures of the cells. The underlying methodology is recorded in Eqs. 11–16.

To calculate the heat capacity rates ($c_p \cdot \dot{m}$), the average heat capacities of the warm or cold fluid at the inlet and outlet are always used for each cell (see cell i in Figure 7) before being multiplied by the mass flow of the corresponding fluid (Eq. 11 and Eq. 12).

$$c_{p,CO2} = f(p, \vartheta_{CO2}); \quad c_{p,H2O} = f(\vartheta_{H2O}); \quad \text{(Eq. 10)}$$

$$C_{hot} = \frac{c_{p,hot,i} + c_{p,hot,i+1}}{2} \cdot \dot{m}_{hot} = c_{p,average,hot} \cdot \dot{m}_{hot} \quad \text{(Eq. 11)}$$

$$C_{cold} = \frac{c_{p,cold,i} + c_{p,cold,i+1}}{2} \cdot \dot{m}_{cold} = c_{p,average,cold} \cdot \dot{m}_{cold} \quad \text{(Eq. 12)}$$

$$C_{Min} = Min(C_{hot}, C_{cold}); \quad C_{Max} = Max(C_{hot}, C_{cold}) \quad \text{(Eq. 13)}$$

$$NTU = \frac{UA}{C_{Min}} \quad \text{(Eq. 14)}$$

$$\varepsilon = \frac{1 - e^{\left(-NTU \cdot \frac{C_{Min}}{C_{Max}}\right)}}{1 - \frac{C_{Min}}{C_{Max}} \cdot e^{\left(-NTU \cdot \frac{C_{Min}}{C_{Max}}\right)}} \quad \text{(Eq. 15)}$$

The two heat capacity rates C_{hot} and C_{cold} of the cold and warm sides of the cell are compared with each other. The smaller value becomes C_{Min}, the larger value becomes C_{Max} (Eq. 13). The NTU is calculated using the UA value (Eq. 14). The calculation of the effectiveness ε depends on the type of heat exchanger (counterflow heat exchanger) and is shown in (Eq. 15). This and the difference in inlet temperatures can be used to calculate the heat transferred per cell (Eq. 16). The new outlet temperatures from the cell are calculated using (Eq. 17) and (Eq. 18).

$$\theta = \varepsilon \cdot C_{Min} \cdot \left(\vartheta_{hot,i+1} - \vartheta_{cold,i}\right) \quad \text{(Eq. 16)}$$

$$\vartheta_{hot,i} = \vartheta_{hot,i+1} - \frac{\theta}{c_{p,average,hot} \cdot \dot{m}_{hot}} \quad \text{(Eq. 17)}$$

$$\vartheta_{cold,i+1} = \vartheta_{cold,1} + \frac{\theta}{c_{p,average,cold} \cdot \dot{m}_{cold}} \quad \text{(Eq. 18)}$$

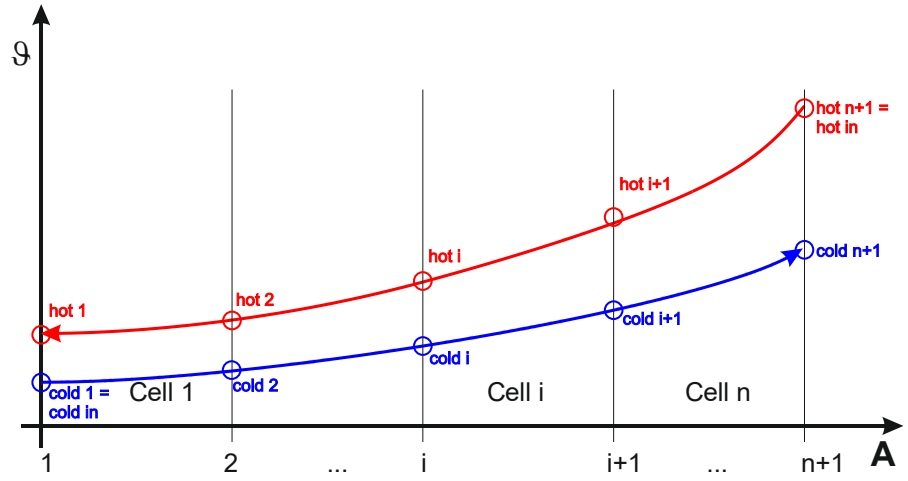

Figure 7: Schematics of heat exchanger cells and node numbering

In the very first iteration, the entire heat exchanger is considered as a single cell (n=1) and (Eq. 10 - 18) are applied to nodes 1 and n+1 (node 2). Since the temperature of hot 1 is not yet available in the first iteration, the temperature of cold 1 is used as the starting value for the first calculation of $cp_{hot,1}$. Similarly, the temperature of hot n+1 is used for cold n+1.

Since the specific heat capacity of CO_2 does not change linearly with temperature, the heat exchanger cannot be treated as a single cell, but must be divided into logical cells. Cells that do not meet the termination criteria for iteration are further halved. Figure 8 shows how cells i and n were halved. Two additional nodes have been added in the middle of the cell. Hot i+1 becomes hot i+2, and the new node in the middle of the cell becomes hot i+1. Similarly, cold i+1 becomes cold i+2, and the new node becomes cold i+1. The new nodes are assigned to the average temperatures of the neighboring nodes as starting values (Eq. 19). Furthermore, the transfer area A of the cell is halved in each case, so the value UA is also halved per cell.

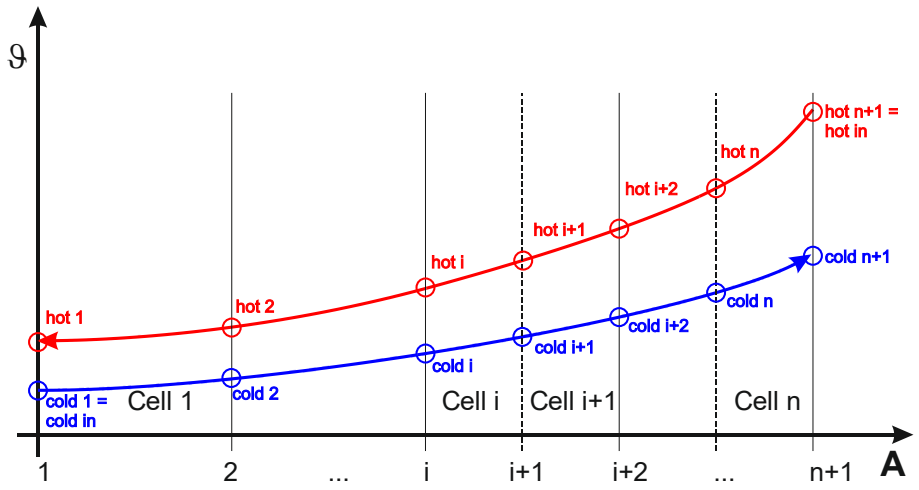

Figure 8: Schematics with splitted heat exchanger cells

$$\vartheta_{hot,i+1} = \frac{\vartheta_{hot,i} + \vartheta_{hot,i+2}}{2}; \quad \vartheta_{cold,i+1} = \frac{\vartheta_{cold,i} + \vartheta_{cold,i+2}}{2} \qquad \text{(Eq. 19)}$$

The forward iteration starts with cell 1 and runs to cell n. The backward iteration starts with cell n and runs back to cell 1. It should be noted that during the forward iteration, only the temperature of cold in is known, and during the backward iteration, only that of hot in is known. The other temperatures are either starting values or come from previous iterations. After the second backward iteration, the heat transfers (cold side and warm side) are compared with the termination criterion Tol_Phi_iter (PAR 27) (Eq. 20). As long as this termination criterion is not met, the ε-NTU iteration continues.

$$\sum_1^n Max\left(\frac{\theta_{i,forward}}{\theta_{i,backward}}, \frac{\theta_{i,backward}}{\theta_{i,forward}}\right) - n \leq \text{Tol\_Phi\_iter} \qquad \text{(Eq. 20)}$$

$$\frac{Abs\left(\dot{m}_{hot} \cdot Abs\left(h(p_{high}, \vartheta_{hot,i}) - h(p_{high}, \vartheta_{hot,i+1})\right) - Abs(\theta_i)\right)}{Abs(\theta_i)} \leq \text{Tol\_Q\_Deviat} \qquad \text{(Eq. 21)}$$

$$Max\left(Max\left(\frac{c_{p,i,hot\ side}}{c_{p,i+1,hot\ side}}, \frac{c_{p,i+1,hot\ side}}{c_{p,i,hot\ side}}\right), Max\left(\frac{c_{p,i,cold\ side}}{c_{p,i+1,cold\ side}}, \frac{c_{p,i+1,cold\ side}}{c_{p,i,cold\ side}}\right)\right) \leq \text{Tol\_cp\_ratio} \qquad \text{(Eq. 22)}$$

$$Max\left(\frac{\vartheta_{hot,i} - \vartheta_{cold,i}}{\vartheta_{hot,i+1} - \vartheta_{cold,i+1}}, \frac{\vartheta_{hot,i+1} - \vartheta_{cold,i+1}}{\vartheta_{hot,i} - \vartheta_{cold,i}}\right) \leq \text{Tol\_dT\_ratio} \qquad \text{(Eq. 23)}$$

If Tol_Phi_iter is satisfied (Eq.20), this only means that the calculation has arrived at a solution using the current number of cells, but it does not mean that all heat capacities, temperature differences, and the heat transfer calculated from them are correct across the entire heat exchanger. Three additional criteria must be met per cell, otherwise it will be divided. The first criterion is the difference between the calculated heat flow through the heat exchanger surface and the difference in enthalpy values at the inlet and outlet of the warm-side fluid into and out of the cell (PAR 24)(Eq. 21). The second criterion concerns the variability of the heat capacity of CO_2 (PAR 25)(Eq. 22). If the c_p values of the fluid on the cold or warm side differ greatly, the cell is split. The third criterion is the ratio of the

temperatures on both sides of the cell, which must remain below the maximum value Tol_dT_ratio (PAR 26) (Eq. 23).

For all these calculations, the initial and final enthalpy as well as the average heat capacity of the refrigerant and water are very often required. The heat capacities of water are calculated using regression. The data used to calculate the regression comes from the VDI Heat Atlas [4]. For refrigerants, TRNSYS provides access to the CoolProp refrigerant database [5]. However, this access is very slow. For this reason, the enthalpy and specific heat capacity in a wide range typically relevant for heat exchangers were read from CoolProp in a pre-processing step and stored in data files (see below). These data files are also read in and linearized during the simulation. Using this data files leads to a small total error in the order of max. 0.5%. This error was accepted because the speed of the simulation becomes approx. 3-4 orders of magnitude faster. However, values close to the critical point are excluded from this simplification, as otherwise the accuracy would suffer noticeably. The description for creating the files is provided in the chapter "Files".

In the case of a water mass flow of 0, the respective exchanger performance is not calculated. If all gas coolers together have no mass flow, the compressor is switched off.

The throttle valve

The throttle valve is simulated isenthalpically. This means that the enthalpy is the same before and after the valve. The function of the throttle valve is to reduce the pressure in the refrigerant circuit. This also lowers the temperature of the refrigerant. However, since the pressure depends on the phase change temperature of the evaporator, the resulting temperature is only calculated in the evaporator. After the throttle valve, the refrigerant is largely liquid.

Evaporator

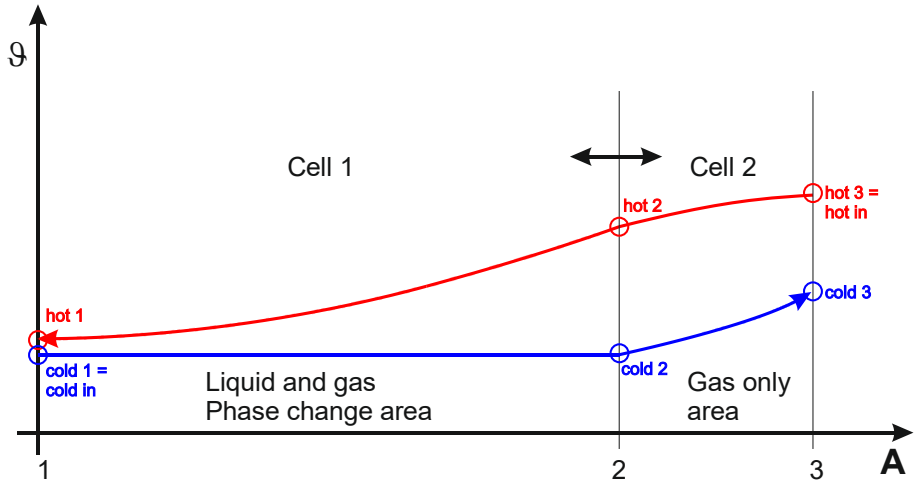

Figure 9: Two-cell approach to calculate the evaporator

In the evaporator, the liquid refrigerant coming from the throttle valve is re-evaporated using low-temperature ambient heat, and the gas is then heated to approx. 1K above the phase change temperature. The degree of heating of the gas is a design parameter when selecting the heat exchanger (typically between approx. 0.5 and 2K) and is specified in INP11. Due to the division of the process into two parts in the heat exchanger, the evaporator is simulated with two cells (Figure 9).

The phase change is calculated in the first cell and the increase in gas temperature in the second cell. Both cells are simulated ideally, i.e., pressure loss due to flow and thermal equilibrium with the environment are ignored. The LMTD calculation method is used for both cells, but with different thermal conductivity coefficients, one for boiling of the refrigerant (Eq. 24) and the other for heat transfer with gas only (Eq. 25).

$$\dot{Q}_{Phase\ change} = U \cdot A \cdot LMTD = U \cdot A \cdot \frac{\Delta\vartheta_2 - \Delta\vartheta_1}{ln\left(\frac{\Delta\vartheta_2}{\Delta\vartheta_1}\right)} = HTC_{Boil} \cdot A \cdot (1 - Split) \cdot \frac{\Delta\vartheta_2 - \Delta\vartheta_1}{ln\left(\frac{\Delta\vartheta_2}{\Delta\vartheta_1}\right)} \quad \text{(Eq. 24)}$$

$$\dot{Q}_{gas,LMTD} = U \cdot A \cdot LMTD = HTC_{gas} \cdot A \cdot Split \cdot \frac{Max(\Delta\vartheta_2, \Delta\vartheta_3) - Min(\Delta\vartheta_2, \Delta\vartheta_3)}{ln\left(\frac{Max(\Delta\vartheta_2, \Delta\vartheta_3)}{Min(\Delta\vartheta_2, \Delta\vartheta_3)}\right)} \quad \text{(Eq. 25)}$$

$$\dot{Q}_{gas,Enthalpy} = \dot{m}_{cold} \cdot Abs\left(h(p_{low}, \vartheta_{cold,3}) - h(p_{low}, \vartheta_{cold,2})\right) \quad \text{(Eq. 26)}$$

$$\frac{\dot{Q}_{gas,LMTD}}{\dot{Q}_{gas,Enthalpy}} - 1 \leq \text{Tol\_area\_split} \quad \text{(Eq. 27)}$$

The overall size of the evaporator is static, but the split of the transfer surface between cell 1 and cell 2 is recalculated each time. The heat absorbed by the CO_2 in cell 2 can be calculated using the enthalpies of temperatures ϑ_{cold3} and ϑ_{cold2} (Eq. 26). On the other hand, the heat absorbed can also be calculated using the LMTD approach, in which case it is directly dependent on the size of the transfer area. The split of the areas is adapted until the tolerance (Tol_area_split, PAR 29) and the defined increase in gas temperature are met (Eq. 27). Even in reality, this split is dynamic and depends on the inlet temperatures and mass flows of the two fluids.

Checks are also carried out to ensure that the fluid does not freeze. For this purpose, a safety temperature is determined by adding a reserve of 1-3K (INP 12) to the freezing temperature of the heat transfer fluid (PAR 24). If the heat transfer fluid falls below this safety temperature when leaving the evaporator, the frequency of the compressor is automatically adjusted downwards. If the frequency falls below the defined minimum limit (PAR 21), the compressor is switched off to prevent frost in the evaporator. The circulation pumps of the gas coolers and the evaporator continue to operate, but no more heat is transferred.

Simulation of the entire coolant cycle

Before the cycle is simulated, start values for the phase change temperature in the evaporator, the proportion of CO_2 vapor entering the evaporator, and the CO_2 outlet temperature from the gas cooler and IHX are calculated statically based on the water inlet temperatures to the heat exchangers. The following start values are set and are pre-programmed:

$$\vartheta_{phase\ change(in\ evaporator)} = \vartheta_{HTF,in,evaporator} - 3.0K \quad \text{(Eq. 28)}$$

$$Quality_{CO2,start} = 80\%\ liquid, 20\%\ gas \quad \text{(Eq. 29)}$$

$$pressure_{CO2,low,start} = f\left(\vartheta_{phase\ change(in\ evaporator)}, Quality_{CO2,start}\right) \quad \text{(Eq. 30)}$$

$$\vartheta_{CO2,behind\ GC3} = \vartheta_{Water,in,GC3} + 2K \quad \text{(Eq. 31)}$$

$$\vartheta_{CO2,before\ compressor} = \vartheta_{CO2,behind\ GC3} + 1K \quad \text{(Eq. 32)}$$

Then iteration of the coolant cycle starts with the calculation of the compressor. The sequence of calculations is:

Compressor -> Oil separator -> Gas cooler 1 -> Gas cooler 2 -> Gas cooler 3 -> IHX -> Throttle valve -> Evaporator -> IHX -> Test of the termination criterion (Eq.33).

$$For\ all\ heat\ exchangers\ GC1, GC2, GC3, Evap: \frac{\dot{Q}_i}{\dot{Q}_{i-1}} \leq \text{Tol\_Q\_Def\_Tot} \qquad \text{(Eq. 33)}$$

The cycle is calculated at least twice. The termination criterion is defined with Tol_Q_Def_Tot (PAR 28) and compares the quotient of the heat flows per heat exchanger with the results of the previous cycle calculation.

7.1.5. Additional data files

Various additional files are used for Type 8505, on the one hand for calculating the electrical power and mass flow of CO_2 from the compressor and on the other hand for reading in the pressure- and temperature-dependent heat capacities and enthalpies of CO_2.

Creation of data files for the compressor

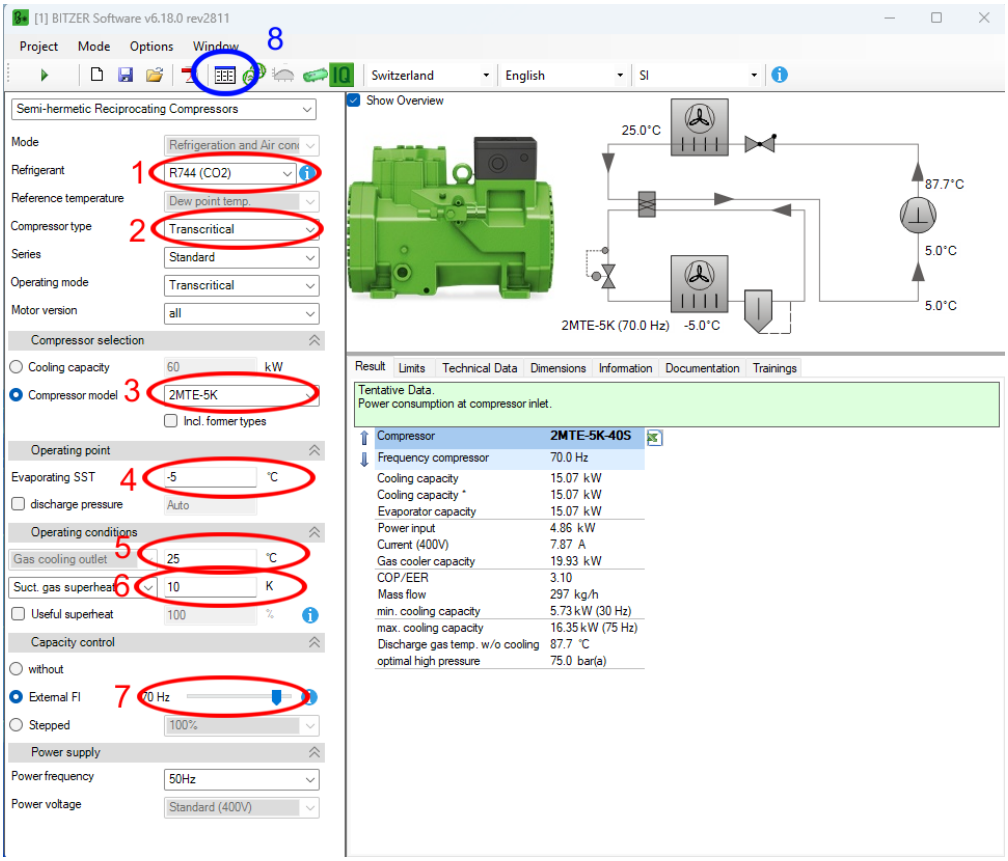

Figure 10: The window in the Bitzer software where the specifications for the compressor calculation can be entered.

As mentioned before, the electrical power and mass flow of CO_2 from the compressor are calculated using regression (standardized polynomial according to EN12900 / ARI 540). The coefficients of this polynomial are calculated as an example using Bitzer software [1] and stored in data files. In

principle, all transcritical CO_2 heat pumps for which standardized polynomials according to EN12900 / ARI 540 are available can be calculated. During the simulation, these data files are then read in and linearized using the TRNSYS routine “InterpolateData”.

After starting the Bitzer software, a window appears with a selection of calculation tools for Bitzer products. For this TRNSYS type, the “Semi-hermetic Reciprocating Compressors” are selected, as only these compressor types are suitable for transcritical operation, but even here, not all compressors are suitable for high temperatures! The window (Figure 10) opens and the parameters for calculating the compressor must be entered. The red circles numbered in Figure 10 are specifications for calculating the coefficients:

1. The refrigerant is CO_2
2. The compressor is operated in the transcritical range

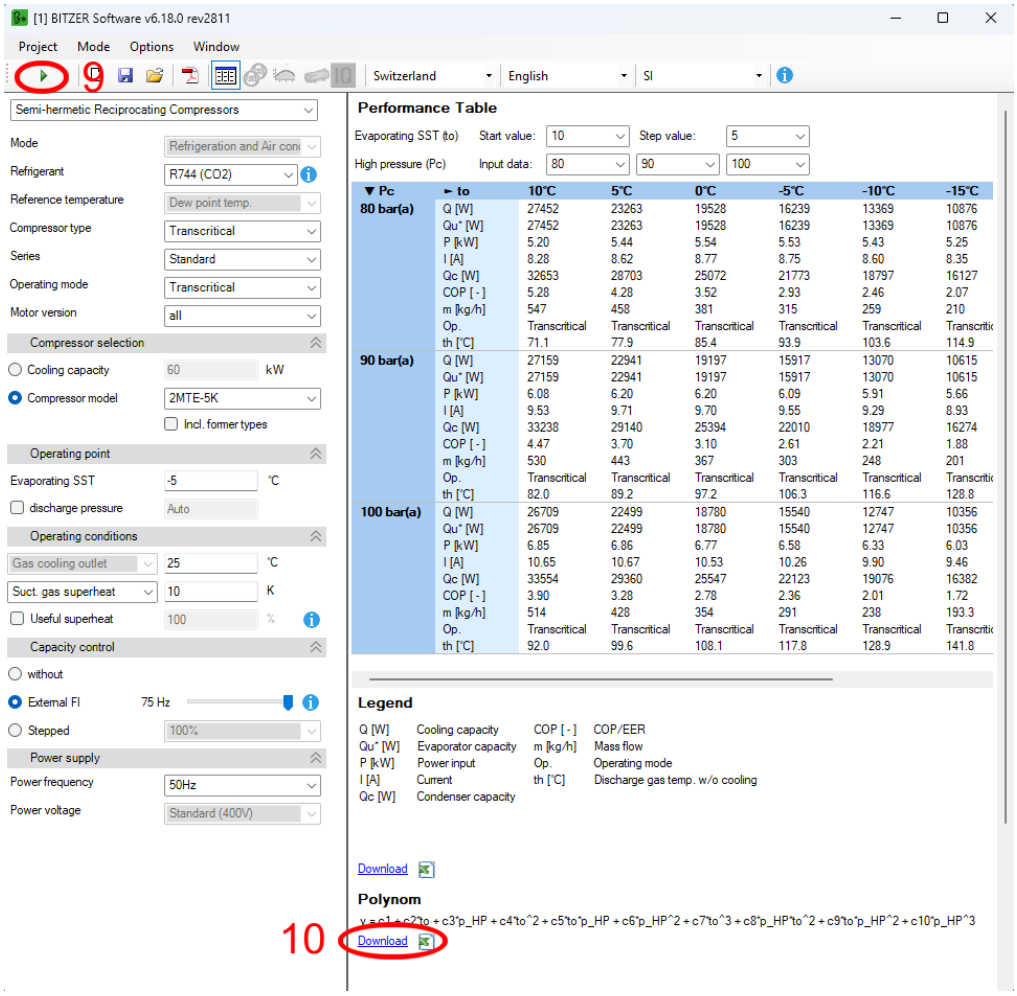

Figure 11: Table view of performance with the selected parameters.

3. Selection of a suitable compressor. Not all specified compressors are suitable for high temperatures! The “2MTE-5K” model was selected for the example calculated in full in this Type
4. This is the phase change temperature in the evaporator. It must be specified between -20°C and 20°C. This temperature has no influence on the calculation of the coefficients and can be freely selected within the limits. However, if the limits are not observed (temperature between -20°C and 20°C), the coefficients will not be calculated.
5. The temperatures after the gas cooler must be between 21°C and 60°C. These temperatures must be varied as they influence the coefficients. The data files use the temperatures 21°, 25°, 30°, ...+5°C..., 55°, 60°C.
6. The temperature difference between the phase change temperature in the evaporator and the inlet to the compressor must also be varied. Valid values are between 5K and 50K. The data files use the temperatures 5K, 10K, 15K, ...+5K..., 45K, 50K.
7. The frequencies must also be varied. Typically, the motors are operated between 30 Hz and 75 Hz. The data file takes into account the frequencies 30 Hz, 35 Hz, ...+5Hz..., 70 Hz, 75 Hz.
8. To calculate the coefficients, switch to the table view (Figure 11).
9. Perform calculation
10. Download the *.csv file
11. The selected parameters are listed again in the *.csv file (Figure 12) for verification purposes.
12. The polynomial used for the regression.

	1	2	3	4	5	6	7	8	9	10	11
7	Presentation of compressor performance data with polynomials to EN 12900 / ARI 540										
8	---										
9	Input Values:										
10											
11	Compressor model			2MTE-5K							
12	Mode			Refrigeration and Air conditioning							
13	Refrigerant			R744							
14	Reference temperature			Dew point temp.							
15	Gas cooling outlet			25.0 °C							
16	Suct. gas superheat			10.00 K							
17	Operating mode			Transcritical							
18	Power supply			400V-3-50Hz							
19	Frequency compressor			70.0 Hz							
20	Useful superheat			100%							
21											
22	---										
23	Polynomial:										
24	y = c1 + c2*to + c3*p_HP + c4*to^2 + c5*to*p_HP + c6*p_HP^2 + c7*to^3 + c8*p_HP*to^2 + c9*to*p_HP^2 + c10*p_HP^3										
25											
26	Coefficients:										
27		c1	c2	c3	c4	c5	c6	c7	c8	c9	c10
28	Q [W]	0	0	0	0	0	0	0	0	0	0
29	P [W]	-4840.77727	-250.079139	205.574723	-3.96175244	4.061738	-1.26728209	-0.02525995	0.02282071	-0.01290494	0.00324674
30	m [kg/h]	481.015112	14.0818751	-1.9013507	0.19119462	0.00325032	0.0035401	0.00165439	0.00010598	-0.00018757	1.6838E-09
31	I [A]	-4.29519505	-0.30855021	0.24685133	-0.00501509	0.00478488	-0.00139895	-3.4857E-05	2.6331E-05	-1.3188E-05	4.4802E-06
32											
33	---										
34	Validity range of Polynomials										
35	Evaporating SST:			-20°C	...	20°C					
36	High pressure:			73.8bar(a)	...	123.7bar(a)					
37	Attention: Consider also application range of compressor!										
38											

Figure 12: Contents of the *.csv file with the coefficients for the parameters selected in points 6–8

13. Coefficients c1 … c10 for calculating the electrical power of the compressor. This data is transferred to a separate data file (in our example: Elec_Pow_2MTE-5K.dat).
14. Coefficients c1 … c10 for calculating the mass flow in the refrigerant circuit. This data is transferred to a separate data file (in our example: MassFlow_2MTE-5K.dat).
15. The polynomial is only valid in the specified pressure range (high-pressure side) and in the specified phase change temperature range in the evaporator. This data is also transferred to a data file (in our example: Limits_2MTE-5K.dat).

For the creation of the data files, the variations specified in 6. to 8. result in 900 permutations per compressor. The coefficients must be copied from the *.csv files and transferred to the data files. For the "2MTE-5K" compressor, the set of data files is attached. (Elec_Pow_2MTE-5K.dat, MassFlow_2MTE-5K.dat, and Limits_2MTE-5K.dat).

Data files for the specific heat capacity and enthalpy of CO_2

TRNSYS provides access to CoolProp's material properties. This allows the material properties to be calculated with high precision. However, this access is very slow, and if, as in this case, CoolProp has to be accessed very frequently, the simulation becomes very slow. As a workaround, the specific heat capacity and enthalpy of CO_2 as a function of temperature and pressure were exported from CoolProp to a support point matrix (data files). These data files are also read in and linearized by the TRNSYS routine "InterpolateData." The advantage of this is that these matrices are stored in RAM and are therefore available very quickly.

Since both the specific heat capacity and the enthalpy are not particularly linear, the distance between the support points had to be kept small, otherwise the accuracy would have been reduced. This meant that the data files for InterPolate Data became too large in various respects. As a workaround, the temperature vector had to be split, and the matrix also had to be distributed across 4 files:

Filename:	**Temperature Range**	**Pressure Range**
CO2_cp&h_-33-62C_12-40bar.dat	-33.2 – 62.8°C; Step: 0.2°C	12.0 – 40.0 bar; Step: 1.0bar
CO2_cp&h_3-62C_40-70bar.dat.	2.8 – 62.8°C; Step: 0.1°C	12.0 – 40.0 bar; Step: 1.0bar
CO2_cp&h_20-60C_74-140bar.dat	20.0 – 60.0°C; Step: 0.1°C	74.0 – 140.0 bar; Step: 2.0bar
CO2_cp&h_60-180C_74-140bar.dat	60.0 – 180.0°C; Step: 0.2°C	74.0 – 140.0 bar; Step: 2.0bar

Table 1: Range of precalculated specific heat capacity and enthalpy

For most values used in the simulation (>99%), these data files can be accessed. However, near the critical temperature range, the data are so dynamic that CoolProp is used for reasons of precision. This also applies to values close to the limit where the gas begins to liquefy. This limit is defined in a further data file:

CO2_PC_Limit.dat	-33.2 – 62.8°C; Step: 0.1°C

7.1.6. Literature

[1] BITZER Software, ver.: 6.18.02811, (n.d.). https://www.bitzer.de/websoftware/ (accessed February 21, 2023).

[2] I.H. Bell, J. Wronski, S. Quoilin, V. Lemort, Pure and Pseudo-pure Fluid Thermophysical Property Evaluation and the Open-Source Thermophysical Property Library CoolProp, Ind. Eng. Chem. Res. 53 (2014) 2498–2508. https://doi.org/10.1021/ie4033999.

[3] W. Wagner, Wärmeaustauscher: Grundlagen, Aufbau und Funktion thermischer Apparate, 3., überarb. und erw. Aufl, Vogel, Würzburg, 2005.

[4] Verein Deutscher Ingenieure, Gesellschaft Verfahrenstechnik und Chemieingenieurwesen, eds., VDI-Wärmeatlas: Berechnungsblätter für den Wärmeübergang, 8., überarb. und erw. Aufl, Springer, Berlin, 1997.

[5] S.A. et al Klein, TRNSYS, Ver. 18.01.0000, A Transient System Simulation Program, (2017). http://sel.me.wisc.edu/trnsys. Programmer's Guide, 7.4.4.2. CoolProp_Fluid_Properties( )

7.2 FORTRAN Code TYPE 8505

The Type8505 Fortran code, the file documentation, the interface files for TRNSYS, and the associated data files have been released under the LGPLv3 license and can be downloaded from: https://github.com/hues-platform/trnsys-type8505

Subroutine Type8505

```
! Object: Transcritical CO2 heat pump
! Simulation Studio Model: Type8505
!

! Author: Robert Weber
! Start date:    March 15, 2023
! Last modified: August 08, 2025



! (Comments and routine interface generated by TRNSYS Simulation Studio)
!************************************************************************

!-----------------------------------------------------------------------------------------------------------------
! This TRNSYS component skeleton was generated from the TRNSYS studio based on the user-supplied parameters, inputs,
! outputs, and derivatives.  The user should check the component formulation carefully and add the content to transform
! the parameters, inputs and derivatives into outputs.  Remember, outputs should be the average value over the timestep
! and not the value at the end of the timestep; although in many models these are exactly the same values.  Refer to
! existing types for examples of using advanced features inside the model (Formats, Labels etc.)
!-----------------------------------------------------------------------------------------------------------------

! Acknowledgements:
! This work was financially supported by EMPA and the SFOE in the HiTemHP project; contract number SI/502168-01.
! Parts of the code are based on work carried out by Hendrik Leliveld during his master's thesis.


!   Remarks:
!***************
!   +   Heat loss in the refrigerant circle are only assumed for the oil separator and tubing between compressor and first gas cooler.
!       All other heat losses of tubes and HX are neclected.
!   +   Compressor calculation (with data interpolation), are based on the data files derived by a software from Bitzer (BITZER Software v6.18.0 rev2811),
!       The data processed is from the compressor 2MTE-5K (transcritical CO2 compressor). See the manual for further explanations.


!   02/08/2026  v1.0    Cleaned and corrected typo in Evaporator



    Use CPINTERFACE
    Use TrnsysConstants
    Use TrnsysFunctions


!-----------------------------------------------------------------------------------------------------------------
```

```
!DEC$Attributes DLLexport :: Type8505

!------------------------------------------------------------------------------------------------------------

! Trnsys Declarations
   Implicit None

! Program controlling
   DOUBLE PRECISION Timestep, Time
   INTEGER CurrentUnit, CurrentType
   INTEGER ii, jj                                ! counters in do loops
   LOGICAL Modes(3), unfinished                  ! Some flags to signalize operation modes

! Boundary conditions
   INTEGER On_Off                                ! [0, 1] Switch to start / stop heat pump
   LOGICAL Mass_flow_available                   ! Check mass flow criteria
   DOUBLE PRECISION Lowest_Frequency, Highest_Frequency       ! Frequency range allowed for this engine

! Start parameter (only used to start the iteration of the refrigerant circle)
   DOUBLE PRECISION, PARAMETER :: T_dif_PC_start = 3.0D0         ! Phase change temperature reduction
against water inlet temp (evap)
   DOUBLE PRECISION, PARAMETER :: Quality_start = 0.8D0       ! Start: value of CO2 quality [mol vapor / mol
(vapor + liquid)]
   DOUBLE PRECISION, PARAMETER :: T_dif_GC3 = 2.0D0          ! Temperature difference between CO2 outlet
and water inlet at GC3
   DOUBLE PRECISION, PARAMETER :: T_dif_IHX = 1.0D0          ! Temperature difference between inlet CO2
high pressure(HP) and outlet CO2 low pressure(LP) at IHX

! Parameters to control iterations
   DOUBLE PRECISION Tol_Q_Deviat                 ! Tolerance factor for the power criteria to define the end
of eps-NTU calculation in the HX
   DOUBLE PRECISION Tol_cp_ratio                 ! safety criteria if cp difference becomes to high -->
Warning
   DOUBLE PRECISION Tol_dT_ratio                 ! Criteria to reduce size of nodes.
   DOUBLE PRECISION Tol_Phi_iter                 ! Criteria of the different heat flows in the cell modell
   DOUBLE PRECISION Tol_Q_Def_Tot                ! Stop criteria of the difference of heat flows in iterations
   DOUBLE PRECISION Tol_area_split               ! Criteria of the area split in evaporator
   INTEGER, PARAMETER :: max_iter = 200          ! Maximal nomber of iterations per call of cell evaluation
   DOUBLE PRECISION Tolerance_Par(10)            ! Vector with tolerances
   DOUBLE PRECISION emergency                    ! [-] 0 => Stop, 1 => On in case of ice in the evoprator
   DOUBLE PRECISION Heat_Flow(4), Heat_Flow_Old(4)          ! Needed to calculate the overall precision

! Function calls
   DOUBLE PRECISION Pressure_CO2                 ! Calculate CO2 pressure from temperature and quality
   DOUBLE PRECISION Enthalpy_CO2                 ! Calculate CO2 enthalpy from temperature and pressure
   DOUBLE PRECISION Temperature_CO2                    ! Calculate CO2 temperature from pressure and
enthalpy

! UNITS conversion
   DOUBLE PRECISION, PARAMETER :: kJ2J       = 1.0D3        ! Factor to convert [kJ] to [J]
   DOUBLE PRECISION, PARAMETER :: kW2kJ_per_h = 3.6D3       ! Factor to convert [kW] to [kJ/h]
   DOUBLE PRECISION, PARAMETER :: W2kJ_per_h  = 3.6D0       ! Factor to convert [W] to [kJ/h]
   DOUBLE PRECISION, PARAMETER :: bar2Pa      = 1.0D5       ! Factor to convert [bar] to [Pa]

! File handle number from data files (1 - 3 from compressor model and 4 - 8 from CO2 data)
   INTEGER LU_data_file(8)

! Device Properties
   DOUBLE PRECISION DeviceProp(8, 13)            ! DeviceProp(device_numbering, device_property) -->
see list 3&4 below
   INTEGER IHX, Comp, Tube, GC1, GC2, GC3, Valve, Evap, Device ! Devices
   INTEGER T_prim_in, T_sec_in, T_prim_out, T_sec_out, m_prim, &
```

```
        m_sec, A_Exc, H_Flux, TC_HX, TCb_HX, &
        Div_1, Div_2, Div_3                         ! List of device properties
   DOUBLE PRECISION Tube_Parameter(13)                    ! Parameter list to calculate tube and oil separator
losses

! CO2 Properties & Cycle states
   DOUBLE PRECISION CO2prop(10,5)                   ! CO2prop(position, CO2property) --> see list 1&2 below
   INTEGER St_a_Evap, St_a_IHX_LP, St_a_Comp, St_a_Tube, &
        St_a_GC1, St_a_GC2, St_a_GC3, St_a_IHX_HP, &
        St_a_Valve, St_i_Evap                       ! List of states in the CO2 circle
   INTEGER Temp, Press, Entha, mflow, H_Cap             ! List of CO2 properties

! Heat transfer fluid (HTF) Properties
   DOUBLE PRECISION Freeze_Temp_HTF                     ! [°C] Temperature, where the HTF freezes

! Compressor
   DOUBLE PRECISION Pressure(2)                     ! [Pa] Pressure(1) = High; Pressure(2) = Low
   LOGICAL Defined                                  ! FALSE if the polynoms are out of the allowed Range

! IHX, Evaporator
   DOUBLE PRECISION Superheating                        ! Temperature lift in evaporator and from evaporation
temperature to inlet compressor
   LOGICAL Call_CoolProp                            ! Used in case of under temperature behind evaporator

! Performance Evaluation
   DOUBLE PRECISION COP, EER
   DOUBLE PRECISION Q_tot_hot

! Execution Time Evaluation
   REAL(4) sec(10)                                  ! Execution time per device


!-----------------------------------------------------------------------------------------------------------------
! *** Explanations of the state variables ***
!-----------------------------------------------------------------------------------------------------------------
!
! +++ List1: State position in the circle of CO2 property --> numbering
!-------------------------------------------------------------------
   St_a_Evap   = 1 ! State of CO2 behind Evaporator and before the internal heat exchanger IHX (low pressure)
   St_a_IHX_LP = 2 ! State of CO2 behind the IHX (low pressure) and before the compressor
   St_a_Comp   = 3 ! State of CO2 behind the compressor and before hot gas tube and oil separator
   St_a_Tube   = 4 ! State of CO2 behind hot gas tube and oil separator and before the 1st gas cooler
   St_a_GC1    = 5 ! State of CO2 behind the 1st gas cooler and before the 2nd gas cooler
   St_a_GC2    = 6 ! State of CO2 behind the 2nd gas cooler and  before the 3rd gas cooler
   St_a_GC3    = 7 ! State of CO2 behind the 3rd gas cooler and before the IHX (high pressure)
   St_a_IHX_HP = 8 ! State of CO2 behind the IHX (high pressure) and before the throttle valve
   St_a_Valve  = 9 ! State of CO2 behind the throttle valve and before the evaporator
   St_i_Evap   = 10! State of CO2 IN the evaporator (phase change)

! +++ List2: Characterisitc of CO2 property numbering
!-----------------------------------------------------
   Temp    = 1 ! CO2 Temperature [°C]
   Press   = 2 ! CO2 Pressure [Pa]
   Entha   = 3 ! CO2 Enthalpy [kJ/kg]
   mflow   = 4 ! CO2 Mass flow [kg/h]
   H_Cap   = 5 ! CO2 Heat capacity [kJ/kg K]

! +++ List3: Device numbering (Components are called devices ) (DeviceProp(Device 1-8, yyy))
!-------------------------------------------------------------------------------
   IHX     = 1 ! IHX, internal heat exchanger
   Comp    = 2 ! Compressor
   Tube    = 3 ! Hot gas tube and oil separator
   GC1     = 4 ! 1st gas cooler
```

```
   GC2     = 5 ! 2nd gas cooler
   GC3     = 6 ! 3rd gas cooler
   Valve   = 7 ! Throttle valve
   Evap    = 8 ! Evaporator

! +++ List4: Meaning of devices property numbering (DeviceProp(xxx, property 1-13))
!------------------------------------------------------------------------------
   T_prim_in   = 1 ! [°C] Heat exchanger inlet temperature hot side
   T_sec_in    = 2 ! [°C] Heat exchanger inlet temperature cold side
   T_prim_out  = 3 ! [°C] Heat exchanger outlet temperature hot side
   T_sec_out   = 4 ! [°C] Heat exchanger outlet temperature cold side
   m_prim      = 5 ! [kg/h] Heat exchanger mass flow hot side
   m_sec       = 6 ! [kg/h] Heat exchanger mass flow cold side                Compressor: [Pa] Set Pressure
   A_Exc       = 7 ! [m2] Heat transfer area of whole heat exchanger           Compressor: [-] Mechanical efficiency
   H_Flux      = 8 ! [kJ/h] Heat flux                                    Oil separator: heat loss
   TC_HX       = 9 ! [kJ/(h*K*m2)] Specific heat transfer coefficient         Compressor: [1/s] Set frequency
   TCb_HX      = 10! [kJ/(h*K*m2)] Specific heat transfer coefficient boil     Compressor: [-] Frequency reduction
   Div_1       = 11! Evaporator: [°C] Overheating CO2                    Compressor: [kW] Electrical power
   Div_2       = 12! Evaporator: [°C] Freeze save Temp
   Div_3       = 13! Evaporator: [kJ/kg K] HTF heat capacity
!------------------------------------------------------------------------------------------------------------------------


!------------------------------------------------------------------------------------------------------------------------
! Get the Global Trnsys Simulation Variables
   Time=getSimulationTime()
   Timestep=getSimulationTimeStep()
   CurrentUnit = getCurrentUnit()
   CurrentType = getCurrentType()
!------------------------------------------------------------------------------------------------------------------------

!------------------------------------------------------------------------------------------------------------------------
! Set the Version Number for This Type
   If(getIsVersionSigningTime()) Then
      Call SetTypeVersion(17)
      Return
   EndIf
!------------------------------------------------------------------------------------------------------------------------

!------------------------------------------------------------------------------------------------------------------------
! Do Any Last Call Manipulations Here
   If(getIsLastCallofSimulation()) Then
      Return
   EndIf
!------------------------------------------------------------------------------------------------------------------------

!------------------------------------------------------------------------------------------------------------------------
! Perform Any "After Convergence" Manipulations That May Be Required at the End of Each Timestep
   If(getIsEndOfTimestep()) Then
      Return
   EndIf
!------------------------------------------------------------------------------------------------------------------------

!------------------------------------------------------------------------------------------------------------------------
! Do All of the "Very First Call of the Simulation Manipulations" Here
   If(getIsFirstCallofSimulation()) Then

      !Tell the TRNSYS Engine How This Type Works
      Call SetNumberofParameters(39)          !The number of parameters that the the model wants
      Call SetNumberofInputs(15)            !The number of inputs that the the model wants
      Call SetNumberofDerivatives(0)          !The number of derivatives that the the model wants
      Call SetNumberofOutputs(43)             !The number of outputs that the the model produces
```

```
      Call SetIterationMode(1)                    !An indicator for the iteration mode (default=1).  Refer to section 8.4.3.5
of the documentation for more details.
      Call SetNumberStoredVariables(0,0)          !The number of static variables that the model wants stored in the
global storage array and the number of dynamic variables that the model wants stored in the global storage array
      Call SetNumberofDiscreteControls(0)         !The number of discrete control functions set by this model (a
value greater than zero requires the user to use Solver 1: Powell's method)


   ! Set the Correct Input and Output Variable Types
   ! Call SetInputUnits(1,'DM1')
   ! Call SetInputUnits(2,'TE1')


!-----------------------------------------------------------
! +++ Initialize some values to prevent undefined states
!-----------------------------------------------------------

      CO2prop(1:10, 1:5) = 0.0D0
      DeviceProp(1:8, 1:13) = 0.0D0
      Heat_Flow(1:4) = 0.0D0

      Return

   EndIf
!-------------------------------------------------------------------------------------------------------------------------

!-------------------------------------------------------------------------------------------------------------------------
! Do All of the First Timestep Manipulations Here - There Are No Iterations at the Intial Time
   If (getIsStartTime()) Then

! Read Parameters
      LU_data_file(1)          = jfix(getParameterValue(1)+0.1)    ! [-] Logical Unit number from Mass Flow data file
(compressor)
      LU_data_file(2)          = jfix(getParameterValue(2)+0.1)    ! [-] Logical Unit number from Electric Power data
file (compressor)
      LU_data_file(3)             = jfix(getParameterValue(3)+0.1)     ! [-] Logical Unit number from Limit data file
(compressor)
      LU_data_file(4)          = jfix(getParameterValue(4)+0.1)    ! [-] Logical Unit number from CO2 data file (60°C
- 180°C; 74 - 140bar)
      LU_data_file(5)          = jfix(getParameterValue(5)+0.1)    ! [-] Logical Unit number from CO2 data file (20°C
- 60°C; 74 - 140bar)
      LU_data_file(6)         = jfix(getParameterValue(6)+0.1)   ! [-] Logical Unit number from CO2 data file (-33.2°C
- 62.8°C; 12 - 40bar)
      LU_data_file(7)          = jfix(getParameterValue(7)+0.1)    ! [-] Logical Unit number from CO2 data file (2.8°C
- 62.8°C; 40 - 70bar)
      LU_data_file(8)         = jfix(getParameterValue(8)+0.1)   ! [-] Logical Unit number from CO2 data file (-33.1°C
- 30.9°C), Phase change test


      DeviceProp(GC1, A_Exc)     = getParameterValue(9)          ! [m2] Heat transfer area of GC1 (hot temperatur
HX)
      DeviceProp(GC2, A_Exc)      = getParameterValue(11)             ! [m2] Heat transfer area of GC2 (medium
temperatur HX)
      DeviceProp(GC3, A_Exc)     = getParameterValue(13)         ! [m2] Heat transfer area of GC3 (low temperatur
HX)
      DeviceProp(IHX, A_Exc)     = getParameterValue(15)          ! [m2] Heat transfer area of IHX (internal HX)
      DeviceProp(Evap, A_Exc)    = getParameterValue(17)          ! [m2] Heat transfer area of evap (evaporator)
      DeviceProp(GC1, TC_HX)      = getParameterValue(10) * W2kJ_per_h! [kJ/(h m2 K)] Heat transfer of GC1
(hot temperatur HX)
      DeviceProp(GC2, TC_HX)      = getParameterValue(12) * W2kJ_per_h! [kJ/(h m2 K)] Heat transfer of GC2
(medium temperatur HX)
      DeviceProp(GC3, TC_HX)      = getParameterValue(14) * W2kJ_per_h! [kJ/(h m2 K)] Heat transfer of GC3
(low temperatur HX)
```

```
      DeviceProp(IHX, TC_HX)       = getParameterValue(16) * W2kJ_per_h! [kJ/(h m2 K)] Heat transfer of IHX
(internal HX)
      DeviceProp(Evap, TC_HX)      = getParameterValue(18) * W2kJ_per_h! [kJ/(h m2 K)] Heat transfer of evap
(evaporator) overheating
      DeviceProp(Evap, TCb_HX)     = getParameterValue(19) * W2kJ_per_h! [kJ/(h m2 K)] Heat transfer of evap
(evaporator) boiling / phase change

      Highest_Frequency          = getParameterValue(20)          ! [s^-1] Highest Frequency allowed for this engine
      Lowest_Frequency           = getParameterValue(21)          ! [s^-1] Lowest Frequency allowed for this engine
      DeviceProp(Evap, Div_3)    = getParameterValue(22)          ! [kJ/kg.K] Heat capacity of the evaporator HTF.
If cp = 100, then water cp is calculated internally
      Freeze_Temp_HTF            = getParameterValue(23)          ! [°C] Temperature, where the HTF freezes
      Tol_Q_Deviat               = getParameterValue(24)          ! [-] Tolerance factor for the power criteria to define
the end of eps-NTU calculation in the HX
      Tol_cp_ratio               = getParameterValue(25)          ! [-] safety criteria if cp difference becomes to high --
> Warning
      Tol_dT_ratio               = getParameterValue(26)          ! [-] Criteria to reduce size of nodes.
      Tol_Phi_iter               = getParameterValue(27)          ! [-] Criteria of the different heat flows in the cell modell
      Tol_Q_Def_Tot              = getParameterValue(28)          ! [-] Stop criteria of the difference of heat flows in
iterations
      Tol_area_split             = getParameterValue(29)          ! [-] Criteria of the area split in evaporator

      Tube_Parameter(1)          = getParameterValue(30)          ! [kJ/(hr.m.K)] Insulation conductivity
      Tube_Parameter(2)          = getParameterValue(31)          ! [m] Insulation thickness tube
      Tube_Parameter(3)          = getParameterValue(32)          ! [m] Insulation thickness oil separator
      Tube_Parameter(4)          = getParameterValue(33)          ! [kJ/(hr.m2.K)] Heat transfer coefficient outside
tube / oil separator
      Tube_Parameter(5)          = getParameterValue(34)          ! [kJ/(hr.m2.K)] Heat transfer coefficient inside
tube / oil separator
      Tube_Parameter(6)          = getParameterValue(35)          ! [m] Diameter of pipe (without insulation)
      Tube_Parameter(7)          = getParameterValue(36)          ! [m] Tube length between compressor and oil
separator
      Tube_Parameter(8)          = getParameterValue(37)          ! [m] Tube length between oil separator and first
heat exchanger
      Tube_Parameter(9)          = getParameterValue(38)          ! [m] Diameter of oil separator  (without insulation)
      Tube_Parameter(10)         = getParameterValue(39)          ! [m] Hight of oil separator wall

! Read initial value of the Inputs
      DeviceProp(Comp, m_sec)     = GetInputValue(1) * bar2Pa      ! [bar] --> [Pa] Set pressure
      DeviceProp(Comp, TC_HX)     = GetInputValue(2)               ! [s^-1] Set frequency
      DeviceProp(GC1, T_sec_in)   = GetInputValue(3)               ! [°C] Temperature water in GC1
      DeviceProp(GC2, T_sec_in)   = GetInputValue(4)               ! [°C] Temperature water in GC2
      DeviceProp(GC3, T_sec_in)   = GetInputValue(5)               ! [°C] Temperature water in GC3
      DeviceProp(Evap, T_prim_in) = GetInputValue(6)               ! [°C] Temperature heat transfer fluid in Evaporator
      DeviceProp(GC1, m_sec)      = GetInputValue(7)               ! [kg/h] Mass flow of water in GC1
      DeviceProp(GC2, m_sec)      = GetInputValue(8)               ! [kg/h] Mass flow of water in GC2
      DeviceProp(GC3, m_sec)      = GetInputValue(9)               ! [kg/h] Mass flow of water in GC3
      DeviceProp(Evap, m_prim)    = GetInputValue(10)              ! [kg/h] Mass flow of heat transfer fluid in
Evaporator
      DeviceProp(Evap, Div_1)     = GetInputValue(11)              ! [°C] Overheating Temperature CO2 in Evaporator
      DeviceProp(Evap, Div_2)     = GetInputValue(12)              ! [°C] Freeze save temperature of HTF in Evaporator
      DeviceProp(Comp, A_Exc)     = GetInputValue(13)              ! [-] Mechanical efficiency of the compressor
      On_Off                      = jfix(GetInputValue(14)+0.1)    ! [0, 1] Switch to start/stop heat pump
      Tube_Parameter(11)          = GetInputValue(15)              ! [°C] Ambient temperature inside heat pump housing


! Check the Parameters for Problems (#,ErrorType,Text)
! Sample Code: If( PAR1 <= 0.) Call FoundBadParameter(1,'Fatal','The first parameter provided to this model is not
acceptable.')

      DO ii = 1 , 8
        If (LU_data_file(ii) <= 20.) Call FoundBadParameter(ii,'Fatal','Logical unit number of data file must be higher
than 20.')
```

```
    End Do
    If (DeviceProp(GC1, A_Exc) .LE.0.0) Call FoundBadParameter(9,'Fatal','Heat transfer area must be higher than 0.0')
    If (DeviceProp(GC2, A_Exc) .LE.0.0) Call FoundBadParameter(11,'Fatal','Heat transfer area must be higher than 0.0')
    If (DeviceProp(GC3, A_Exc) .LE.0.0) Call FoundBadParameter(13,'Fatal','Heat transfer area must be higher than 0.0')
    If (DeviceProp(IHX, A_Exc) .LE.0.0) Call FoundBadParameter(15,'Fatal','Heat transfer area must be higher than 0.0')
    If (DeviceProp(Evap, A_Exc) .LE.0.0) Call FoundBadParameter(17,'Fatal','Heat transfer area must be higher than 0.0')
    If (DeviceProp(GC1, TC_HX) .LE.0.0) Call FoundBadParameter(10,'Fatal','Specific heat transfer coefficient must be higher than 0.0')
    If (DeviceProp(GC2, TC_HX) .LE.0.0) Call FoundBadParameter(12,'Fatal','Specific heat transfer coefficient must be higher than 0.0')
    If (DeviceProp(GC3, TC_HX) .LE.0.0) Call FoundBadParameter(14,'Fatal','Specific heat transfer coefficient must be higher than 0.0')
    If (DeviceProp(IHX, TC_HX) .LE.0.0) Call FoundBadParameter(16,'Fatal','Specific heat transfer coefficient must be higher than 0.0')
    If (DeviceProp(Evap, TC_HX) .LE.0.0) Call FoundBadParameter(18,'Fatal','Specific heat transfer coefficient must be higher than 0.0')
    If (DeviceProp(Evap, TCb_HX) .LE.0.0) Call FoundBadParameter(19,'Fatal','Specific heat transfer coefficient must be higher than 0.0')
    If (DeviceProp(Evap, Div_3) .LE.0.0) Call FoundBadParameter(22,'Fatal','Specific heat capacity of the evaporator HTF must be higher than 0.0')
    If (Tol_Q_Deviat .LE.0.0) Call FoundBadParameter(24,'Fatal','Tolerance factor for the criteria to define the end of eps-NTU iteration must be higher than 0.0')
    If (Tol_cp_ratio .LE.1.0) Call FoundBadParameter(25,'Fatal','Cell dividing criteria for cp ratio must be higher than 1.0')
    If (Tol_dT_ratio .LE.1.0) Call FoundBadParameter(26,'Fatal','Cell dividing criteria for temperature difference ratio must be higher than 1.0')
    If (Tol_Phi_iter .LE.0.0) Call FoundBadParameter(27,'Fatal','Tolerance factor for the minimal heat flow to end the eps-NTU iteration in the GC must be higher than 0.0')
    If (Tol_Q_Def_Tot .LE.0.0) Call FoundBadParameter(28,'Fatal','Tolerance factor for the minimal heat flow to end the cooling cycle iterationmust be higher than 0.0')
    If (Tol_area_split .LE.0.0) Call FoundBadParameter(29,'Fatal','Tolerance factor for area split in the evaporator must be higher than 0.0')
    If (Tube_Parameter(1) .LE.0.0) Call FoundBadParameter(30,'Fatal','Insulation conductivity Lambda must be higher than 0.0')
    If (Tube_Parameter(2) .LE.0.0) Call FoundBadParameter(31,'Fatal','Insulation thickness of the tubes must be higher than 0.0')
    If (Tube_Parameter(3) .LE.0.0) Call FoundBadParameter(32,'Fatal','Insulation thickness of the oil separatoro must be higher than 0.0')
    If (Tube_Parameter(4) .LE.0.0) Call FoundBadParameter(33,'Fatal','Surface heat transfer coefficient outside the tube and oil separator must be higher than 0.0')
    If (Tube_Parameter(5) .LE.0.0) Call FoundBadParameter(34,'Fatal','Surface heat transfer coefficient inside the tube and oil separator must be higher than 0.0')
    If (Tube_Parameter(6) .LE.0.0) Call FoundBadParameter(35,'Fatal','Diameter of pipe must be higher than 0.0')
    If (Tube_Parameter(7) .LE.0.0) Call FoundBadParameter(36,'Fatal','Tube length between compressor and oil separator must be higher than 0.0')
    If (Tube_Parameter(8) .LE.0.0) Call FoundBadParameter(37,'Fatal','Tube length between oil separator and first heat exchanger must be higher than 0.0')
    If (Tube_Parameter(9) .LE.0.0) Call FoundBadParameter(38,'Fatal','Diameter of oil separator must be higher than 0.0')
    If (Tube_Parameter(10) .LE.0.0) Call FoundBadParameter(39,'Fatal','Hight of oil separator must be higher than 0.0')

! Set the Initial Values of the Outputs (#,Value)
    DO ii = 1 , 42
       Call SetOutputValue(ii, 0.d0)
    End Do

    Return
```

```
   EndIf
!-----------------------------------------------------------------------------------------------------------------------

!-----------------------------------------------------------------------------------------------------------------------
! ReRead the Parameters if Another Unit of This Type Has Been Called Last
   If(getIsReReadParameters()) Then
      !Read in the Values of the Parameters from the Input File

      LU_data_file(1)          = jfix(getParameterValue(1)+0.1)    ! [-] Logical Unit number from Mass Flow data file
(compressor)
      LU_data_file(2)          = jfix(getParameterValue(2)+0.1)    ! [-] Logical Unit number from Electric Power data
file (compressor)
      LU_data_file(3)             = jfix(getParameterValue(3)+0.1)     ! [-] Logical Unit number from Limit data file
(compressor)
      LU_data_file(4)          = jfix(getParameterValue(4)+0.1)    ! [-] Logical Unit number from CO2 data file (60°C
- 180°C; 74 - 140bar)
      LU_data_file(5)          = jfix(getParameterValue(5)+0.1)    ! [-] Logical Unit number from CO2 data file (20°C
- 60°C; 74 - 140bar)
      LU_data_file(6)         = jfix(getParameterValue(6)+0.1)   ! [-] Logical Unit number from CO2 data file (-33.2°C
- 62.8°C; 12 - 40bar)
      LU_data_file(7)          = jfix(getParameterValue(7)+0.1)    ! [-] Logical Unit number from CO2 data file (2.8°C
- 62.8°C; 40 - 70bar)
      LU_data_file(8)         = jfix(getParameterValue(8)+0.1)   ! [-] Logical Unit number from CO2 data file (-33.1°C
- 30.9°C), Phase change test

      DeviceProp(GC1, A_Exc)     = getParameterValue(9)          ! [m2] Heat transfer area of GC1 (hot temperatur
HX)
      DeviceProp(GC2, A_Exc)      = getParameterValue(11)             ! [m2] Heat transfer area of GC2 (medium
temperatur HX)
      DeviceProp(GC3, A_Exc)     = getParameterValue(13)         ! [m2] Heat transfer area of GC3 (low temperatur
HX)
      DeviceProp(IHX, A_Exc)     = getParameterValue(15)          ! [m2] Heat transfer area of IHX (internal HX)
      DeviceProp(Evap, A_Exc)    = getParameterValue(17)          ! [m2] Heat transfer area of evap (evaporator)
      DeviceProp(GC1, TC_HX)      = getParameterValue(10) * W2kJ_per_h! [kJ/(h m2 K)] Heat transfer of GC1
(hot temperatur HX)
      DeviceProp(GC2, TC_HX)      = getParameterValue(12) * W2kJ_per_h! [kJ/(h m2 K)] Heat transfer of GC2
(medium temperatur HX)
      DeviceProp(GC3, TC_HX)      = getParameterValue(14) * W2kJ_per_h! [kJ/(h m2 K)] Heat transfer of GC3
(low temperatur HX)
      DeviceProp(IHX, TC_HX)      = getParameterValue(16) * W2kJ_per_h! [kJ/(h m2 K)] Heat transfer of IHX
(internal HX)
      DeviceProp(Evap, TC_HX)     = getParameterValue(18) * W2kJ_per_h! [kJ/(h m2 K)] Heat transfer of evap
(evaporator) overheating
      DeviceProp(Evap, TCb_HX)    = getParameterValue(19) * W2kJ_per_h! [kJ/(h m2 K)] Heat transfer of evap
(evaporator) boiling / phase change

      Highest_Frequency          = getParameterValue(20)          ! [s^-1] Highest Frequency allowed for this engine
      Lowest_Frequency           = getParameterValue(21)          ! [s^-1] Lowest Frequency allowed for this engine
      DeviceProp(Evap, Div_3)    = getParameterValue(22)          ! [kJ/kg.K] Heat capacity of the evaporator HTF.
If cp = 100, then water cp is calculated internally
      Freeze_Temp_HTF            = getParameterValue(23)          ! [°C] Temperature, where the HTF freezes
      Tol_Q_Deviat               = getParameterValue(24)          ! [-] Tolerance factor for the power criteria to define
the end of eps-NTU calculation in the HX
      Tol_cp_ratio               = getParameterValue(25)          ! [-] safety criteria if cp difference becomes to high --
> Warning
      Tol_dT_ratio               = getParameterValue(26)          ! [-] Criteria to reduce size of nodes.
      Tol_Phi_iter               = getParameterValue(27)          ! [-] Criteria of the different heat flows in the cell modell
      Tol_Q_Def_Tot              = getParameterValue(28)          ! [-] Stop criteria of the difference of heat flows in
iterations
      Tol_area_split             = getParameterValue(29)          ! [-] Criteria of the area split in evaporator

      Tube_Parameter(1)          = getParameterValue(30)          ! [kJ/(hr.m.K)] Insulation conductivity
```

```
      Tube_Parameter(2)        = getParameterValue(31)        ! [m] Insulation thickness tube
      Tube_Parameter(3)        = getParameterValue(32)        ! [m] Insulation thickness oil separator
      Tube_Parameter(4)        = getParameterValue(33)        ! [kJ/(hr.m2.K)] Heat transfer coefficient outside
tube / oil separator
      Tube_Parameter(5)        = getParameterValue(34)        ! [kJ/(hr.m2.K)] Heat transfer coefficient inside
tube / oil separator
      Tube_Parameter(6)        = getParameterValue(35)        ! [m] Diameter of pipe (without insulation)
      Tube_Parameter(7)        = getParameterValue(36)        ! [m] Tube length between compressor and oil
separator
      Tube_Parameter(8)        = getParameterValue(37)        ! [m] Tube length between oil separator and first
heat exchanger
      Tube_Parameter(9)        = getParameterValue(38)        ! [m] Diameter of oil separator  (without insulation)
      Tube_Parameter(10)       = getParameterValue(39)        ! [m] Hight of oil separator wall


   EndIf
!-----------------------------------------------------------------------------------------------------------------

! Read the Inputs

   DeviceProp(Comp, m_sec)     = GetInputValue(1) * bar2Pa ! [bar] --> [Pa] Set pressure
   DeviceProp(Comp, TC_HX)     = GetInputValue(2)          ! [s^-1] Set frequency
   DeviceProp(GC1, T_sec_in)   = GetInputValue(3)          ! [°C] Temperature water in GC1
   DeviceProp(GC2, T_sec_in)   = GetInputValue(4)          ! [°C] Temperature water in GC2
   DeviceProp(GC3, T_sec_in)   = GetInputValue(5)          ! [°C] Temperature water in GC3
   DeviceProp(Evap, T_prim_in) = GetInputValue(6)          ! [°C] Temperature water in Evaporator
   DeviceProp(GC1, m_sec)      = GetInputValue(7)          ! [kg/h] Mass flow of water in GC1
   DeviceProp(GC2, m_sec)      = GetInputValue(8)          ! [kg/h] Mass flow of water in GC2
   DeviceProp(GC3, m_sec)      = GetInputValue(9)          ! [kg/h] Mass flow of water in GC3
   DeviceProp(Evap, m_prim)    = GetInputValue(10)         ! [kg/h] Mass flow of water in Evaporator
   DeviceProp(Evap, Div_1)     = GetInputValue(11)         ! [°C] Overheating Temperature CO2 in Evaporator
   DeviceProp(Evap, Div_2)     = GetInputValue(12)         ! [°C] Freeze save temperature of HTF in Evaporator
   DeviceProp(Comp, A_Exc)     = GetInputValue(13)         ! [-] Mechanical efficiency of the compressor
   On_Off                      =jfix(GetInputValue(14)+0.1)! [0, 1] Switch to start/stop heat pump
   Tube_Parameter(11)          = GetInputValue(15)         ! [°C] Ambient temperature inside heat pump housing

!
! Check the Inputs for Problems (#,ErrorType,Text)
! Sample Code: If( IN1 <= 0.) Call FoundBadInput(1,'Fatal','The first input provided to this model is not acceptable.')

   If(DeviceProp(Comp, m_sec) .LT. 74.0  * bar2Pa) Call FoundBadInput(1,'Fatal','Set CO2 pressure too low.')
   If(DeviceProp(Comp, m_sec) .GT. 140.0  * bar2Pa) Call FoundBadInput(1,'Fatal','Set CO2 pressure too high.')
   If(DeviceProp(Comp, TC_HX) .LT. 0.0)  Call FoundBadInput(2,'Fatal','Compressor frequency below zero!')
   If(DeviceProp(GC1, T_sec_in) .LT. 0.0)  Call FoundBadInput(3,'Fatal','Water Temeprature in GC1 below zero!')
   If(DeviceProp(GC2, T_sec_in) .LT. 0.0)  Call FoundBadInput(4,'Fatal','Water Temeprature in GC2 below zero!')
   If(DeviceProp(GC3, T_sec_in) .LT. 0.0)  Call FoundBadInput(5,'Fatal','Water Temeprature in GC3 below zero!')
   If(DeviceProp(GC1, m_sec) .LT. 0.0)  Call FoundBadInput(7,'Fatal','Mass flow of water in GC1 below zero!')
   If(DeviceProp(GC2, m_sec) .LT. 0.0)  Call FoundBadInput(8,'Fatal','Mass flow of water in GC2 below zero!')
   If(DeviceProp(GC3, m_sec) .LT. 0.0)  Call FoundBadInput(9,'Fatal','Mass flow of water in GC3 below zero!')
   If(DeviceProp(Evap, m_prim) .LT. 0.0)  Call FoundBadInput(10,'Fatal','Mass flow HTF in Evaporator below zero!')
   If(DeviceProp(Comp, A_Exc) .LT. 0.0)  Call FoundBadInput(13,'Fatal','Mechanical efficiency of the compressor
below zero!')
   If(DeviceProp(Comp, A_Exc) .GT. 1.0)  Call FoundBadInput(13,'Fatal','Mechanical efficiency of the compressor
above 100%!')

   If(ErrorFound()) Return

   If(DeviceProp(Comp, TC_HX) .GT. Highest_Frequency) Then
      DeviceProp(Comp, TC_HX) = Highest_Frequency
      Call Messages(-1,'Set frequency of compressor too high. Reset to ', Highest_Frequency, ' Hz', 'Warning', Cur-
rentUnit,CurrentType)
   EndIf
```

```
   If(DeviceProp(Comp, TC_HX) .LT. Lowest_Frequency) Then
      Call Messages(-1,'Set frequency of compressor too low. Compressor stopped', 'Warning', CurrentUnit,Cur-
rentType)
   EndIf


!
!-------------------------------------------------------------------------------------------------------------------
!-------------------------------------------------------------------------------------------------------------------
! *** PERFORM ALL THE CALCULATION HERE FOR THIS MODEL. ***
!-------------------------------------------------------------------------------------------------------------------
!-------------------------------------------------------------------------------------------------------------------

!-------------------------------------------------------------------------------------------------------------------
!    +++ Initialize some variables at start time of iteration
!-------------------------------------------------------------------------------------------------------------------

   Modes(1:3) = .TRUE.                          ! 1 => First iteration, 2 => Heat pump has started, 3 => Heat pump
set off
   Tolerance_Par(1) = Tol_Q_Deviat                 ! Tolerance power criteria eps-NTU calculation
   Tolerance_Par(2) = Tol_cp_ratio                ! safety criteria if cp difference becomes to high --> Warning
   Tolerance_Par(3) = Tol_dT_ratio                ! Criteria to reduce size of nodes.
   Tolerance_Par(4) = Tol_Phi_iter                ! Criteria of the different heat flows in the cell modell
   Tolerance_Par(5) = Tol_Q_Def_Tot                ! Stop criteria of the difference of heat flows in iterations
   Tolerance_Par(6) = Tol_area_split              ! Criteria of the area split in evaporator
   Tolerance_Par(7) = Lowest_Frequency               ! Lowest frequency allowed for this engine
   Tolerance_Par(8) = DBLE(200)                   ! Maximal nomber of iterations per call of cell evaluation
   DeviceProp(Comp, TCb_HX)    = 1.0D0               ! Frequency reduction [0..1]
   emergency = 0.0D0                          ! Ice in Evaporator => 1.0
   sec(1:10) = 0.0                          ! Execution times set to 0

   If (NOT(DeviceProp(Evap, Div_3) .EQ. 100.0)) THEN      ! HTF with Glycol is used
      DeviceProp(Evap, Div_2) = DeviceProp(Evap, Div_2) + Freeze_Temp_HTF
   ENDIF


!-------------------------------------------------------------------------------------------------------------------
! +++ Check Boundary condiditons
!-------------------------------------------------------------------------------------------------------------------

   Mass_flow_available = .TRUE.                     ! If there is no massflow in the evaporator or in at least one
GC, stop heat pump
   unfinished = .TRUE.                        ! Set the iteration flag for the circle iteration

   IF (DeviceProp(Evap, m_prim) .LE. 0.0) Mass_flow_available = .FALSE.                              ! No
massflow in the evaporator, stop heat pump
   IF ((DeviceProp(GC1, m_sec) + DeviceProp(GC2, m_sec) + DeviceProp(GC3, m_sec)) .LE. 0.0)
Mass_flow_available = .FALSE.  ! No massflow in the gas coolers, stop heat pump

   IF ((On_Off .EQ. 0) .OR. (NOT(Mass_flow_available))) THEN   ! Heat pump set off
      Modes(3) = .FALSE.
      unfinished = .FALSE.                     ! Continue without circle iteration
   ENDIF



!-------------------------------------------------------------------------------------------------------------------
! +++ Start assumptions
!-------------------------------------------------------------------------------------------------------------------

! Start assumption 1: The phase change temperature of CO2 is T_dif_PC_start lower than the water inlet tempera-
ture into the evaporator.
!                Needed to calculate the first low pressure state of CO2
```

```
! Start assumption 2: The gascooler 3 CO2 outlet temperature is 2° higher than the water inlet temperature in gascooler 3.
!                Needed to calculate the first compressor state.
! Start assumption 3: The IHX outlet temperature on the low pressure side is 1°C lower than the gascooler 3 CO2 outlet temperature
!                Needed to calculate the gas superheating and the compressor state.
! The start assumption are limitted to a reasonable range. Just to omit out of range calculation in time 0

   CO2prop(St_i_Evap, Temp) = MIN(22.0D0, MAX(-22.0D0, DeviceProp(Evap, T_prim_in) - T_dif_PC_start))       ![°C] Guess of the CO2 phase change temperature in the evaporator
   CO2prop(St_i_Evap, Press) = Pressure_CO2(CO2prop(St_i_Evap, Temp), Quality_start, CurrentUnit, CurrentType) ! [Pa] low pressure guess depending on phase change temperatur
   CO2prop(St_a_GC3, Temp)  = MIN(65.0D0, MAX(18.0D0, DeviceProp(GC3, T_sec_in) + T_dif_GC3))             ![°C] CO2 temperature guess behind 3rd gas cooler
   CO2prop(St_a_IHX_LP, Temp) = CO2prop(St_a_GC3, Temp) - T_dif_IHX                                        ! [°C] CO2 temperature guess behind IHX low pressure side

! Calculate a first guesses of the CO2 the CO2 properties
   CALL Compressor(LU_data_file, CO2prop, DeviceProp, Tolerance_Par, Modes, CurrentUnit, CurrentType, Defined)
   CO2prop(St_a_IHX_LP, Entha) = Enthalpy_CO2(CO2prop(St_a_IHX_LP, Temp), CO2prop(St_i_Evap, Press), CurrentUnit, CurrentType) ! [kJ/kg], ([°C], [Pa], [-], [-])
   CO2prop(St_a_Evap, Temp) = CO2prop(St_i_Evap, Temp) + DeviceProp(Evap, Div_1)

!-------------------------------------------------------------------------------------------------------------------------
! +++ Start the calculation loop for the HP circle
!-------------------------------------------------------------------------------------------------------------------------


   DO WHILE (unfinished)


      Heat_Flow_Old = Heat_Flow                                                    ! Save the heat flows of the HX

! +++ CO2 behind Compressor model
!-------------------------------
      sec(St_a_Comp) = SECNDS(sec(St_a_Comp))

      CO2prop(St_a_Comp, Entha) = CO2prop(St_a_IHX_LP, Entha) + DeviceProp(Comp, H_Flux) / CO2prop(St_a_Comp, mflow)! [kJ/kg] Enthalpy behind compressor (high pressure side)
      CO2prop(St_a_Comp, Temp) = Temperature_CO2(CO2prop(St_a_Comp,Press), CO2prop(St_a_Comp,Entha), &
      CurrentUnit, CurrentType)                                                    ! [°C] Temperature calculation with pressure and enthalpy

      sec(St_a_Comp) = SECNDS(sec(St_a_Comp))


! +++ Hot gas tube and oil separator calculation
!-----------------------------------------------
! Some heat is lost, which cannot be neglected due to the high surface temperatures.

      sec(St_a_Tube) = SECNDS(sec(St_a_Tube))

      DeviceProp(Tube, T_prim_in) = CO2prop(St_a_Comp, Temp)
      Call HeatLoss(CO2prop, DeviceProp, Tube_Parameter, LU_data_file, CurrentUnit, CurrentType)

      sec(St_a_Tube) = SECNDS(sec(St_a_Tube))


! +++ CO2 behind the oil separator
!---------------------------------
```

```
        CO2prop(St_a_Tube, Temp) = DeviceProp(Tube, T_prim_out)


! +++ Gas cooler GC1 (high temperature)
!-------------------------------------

        sec(St_a_GC1) = SECNDS(sec(St_a_GC1))

        DeviceProp(GC1, T_prim_in) = CO2prop(St_a_Tube, Temp)
        Call HX_calculation(GC1, DeviceProp, CO2prop, Tolerance_Par, LU_data_file, CurrentUnit, CurrentType)
        Heat_Flow(1) = DeviceProp(GC1, H_Flux)

        sec(St_a_GC1) = SECNDS(sec(St_a_GC1))


! +++ CO2 behind the gas cooler GC1
!---------------------------------

        CO2prop(St_a_GC1, Temp) = DeviceProp(GC1, T_prim_out)


! +++ Gas cooler GC2 (medium temperature)
!---------------------------------------

        sec(St_a_GC2) = SECNDS(sec(St_a_GC2))

        DeviceProp(GC2, T_prim_in) = CO2prop(St_a_GC1, Temp)
        Call HX_calculation(GC2, DeviceProp, CO2prop, Tolerance_Par, LU_data_file, CurrentUnit, CurrentType)
        Heat_Flow(2) = DeviceProp(GC2, H_Flux)

        sec(St_a_GC2) = SECNDS(sec(St_a_GC2))


! +++ CO2 behind the gas cooler GC2
!---------------------------------

        CO2prop(St_a_GC2, Temp) = DeviceProp(GC2, T_prim_out)


! +++ Gas cooler GC3 (low temperature, safety cooler)
!---------------------------------------------------

        sec(St_a_GC3) = SECNDS(sec(St_a_GC3))

        DeviceProp(GC3, T_prim_in) = CO2prop(St_a_GC2, Temp)
        Call HX_calculation(GC3, DeviceProp, CO2prop, Tolerance_Par, LU_data_file, CurrentUnit, CurrentType)
        Heat_Flow(3) = DeviceProp(GC3, H_Flux)

        sec(St_a_GC3) = SECNDS(sec(St_a_GC3))


! +++ CO2 behind the gas cooler GC3
!---------------------------------

        CO2prop(St_a_GC3, Temp) = DeviceProp(GC3, T_prim_out)


! +++ Internal gas heat exchanger IHX, high pressure side (HP)
!-------------------------------------------------------------

        sec(St_a_IHX_HP) = SECNDS(sec(St_a_IHX_HP))

        DeviceProp(IHX, T_prim_in) = CO2prop(St_a_GC3, Temp)
```

```
      DeviceProp(IHX, T_sec_in) = CO2prop(St_a_Evap, Temp)
      DeviceProp(IHX, m_sec) = DeviceProp(IHX, m_prim)
      Call HX_calculation(IHX, DeviceProp, CO2prop, Tolerance_Par, LU_data_file, CurrentUnit, CurrentType)

      sec(St_a_IHX_HP) = SECNDS(sec(St_a_IHX_HP))


! +++ CO2 behind the Internal gas heat exchanger IHX, high pressure side (HP)
!-------------------------------------------------------------------------

      CO2prop(St_a_IHX_HP, Temp) = DeviceProp(IHX, T_prim_out)
      Call CO2_h_cp_interpol(LU_data_file, CO2prop(St_a_IHX_HP, Temp), CO2prop(St_a_IHX_HP, Press), &
         CO2prop(St_a_IHX_HP, Entha), CO2prop(St_a_IHX_HP, H_Cap), Call_CoolProp, CurrentUnit, Cur-
rentType)            ! [-], [°C], [Pa], [kJ/kg], [kJ/(kgK)], [-], [-]


! +++ Throttle Valve (isenthalpic behavior)
!----------------------------------------

      DeviceProp(Valve, T_prim_in) = CO2prop(St_a_IHX_HP, Temp)


! +++ CO2 behind the Throttle Valve, change to low pressure
!---------------------------------------------------------

      CO2prop(St_a_Valve, Press) = CO2prop(St_i_Evap, Press)
      CO2prop(St_a_Valve, Entha) = CO2prop(St_a_IHX_HP, Entha)


! +++ Evaporator
!--------------

      sec(St_a_Evap) = SECNDS(sec(St_a_Evap))

      IF (DeviceProp(Evap, m_prim) .EQ. 0.0D0) THEN                                    ! No inlet water
to the evaporator
         Modes(2) = .FALSE.                                                  ! Stop compressor operation
      EndIf
      Call HX_calculation(Evap, DeviceProp, CO2prop, Tolerance_Par, LU_data_file, CurrentUnit, CurrentType)
      IF ((DeviceProp(Comp, TCb_HX) .LT. 1.0D0) .AND. (DeviceProp(Comp, TC_HX) .LE. 3.0D1)) THEN
! Frequency goes below limit
         Modes(2) = .FALSE.                                                  ! Stop compressor operation
         Call Messages(-1,'Not enough low temperature heat, freezing in Evaporator', 'Warning', CurrentUnit,Cur-
rentType)
      EndIf
      Heat_Flow(4) = DeviceProp(Evap, H_Flux)

      sec(St_a_Evap) = SECNDS(sec(St_a_Evap))


! +++ CO2 behind the Evaporator (overhating included)
!---------------------------------------------------

      CO2prop(St_a_Evap, Temp) = DeviceProp(Evap, T_sec_out)
      CO2prop(St_a_Evap : St_a_IHX_LP, Press) = CO2prop(St_i_Evap, Press)


! +++ Internal gas heat exchanger IHX, low pressure side (LP)
!-----------------------------------------------------------

      sec(St_a_IHX_LP) = SECNDS(sec(St_a_IHX_LP))

      DeviceProp(IHX, T_prim_in) = CO2prop(St_a_GC3, Temp)
```

```
      DeviceProp(IHX, T_sec_in) = CO2prop(St_a_Evap, Temp)
      Call HX_calculation(IHX, DeviceProp, CO2prop, Tolerance_Par, LU_data_file, CurrentUnit, CurrentType)

      sec(St_a_IHX_LP) = SECNDS(sec(St_a_IHX_LP))


! +++ CO2 behind the Internal gas heat exchanger IHX, low pressure side (LP)
!-------------------------------------------------------------------------

      CO2prop(St_a_IHX_LP, Temp) = DeviceProp(IHX, T_sec_out)


! +++ Compressor calculation
!---------------------------

      sec(St_a_Comp) = SECNDS(sec(St_a_Comp))

      CALL Compressor(LU_data_file, CO2prop, DeviceProp, Tolerance_Par, Modes, CurrentUnit, CurrentType,
Defined)

      sec(St_a_Comp) = SECNDS(sec(St_a_Comp))


! +++ Iteration control
!----------------------
      jj = 0

      IF (Modes(1)) THEN                                ! First iteration in this call
         Modes(1) = .FALSE.

      ELSE IF (NOT(Modes(2))) THEN                              ! Safety stop of compressor. Not enough heat from
Evap
         DeviceProp(IHX:Evap, H_Flux) = 0.0D0
         DeviceProp(IHX:Evap, T_prim_out) = DeviceProp(IHX:Evap, T_prim_in)
         DeviceProp(IHX:Evap, T_sec_out) = DeviceProp(IHX:Evap, T_sec_in)
         unfinished = .FALSE.
         emergency = 1.0D0
      ELSE

         DO ii = 1, 4
            IF (Heat_Flow_Old(ii) .EQ. 0.0) THEN
               IF (Heat_Flow(ii) .EQ. 0.0) THEN             ! Heat exchanger not used
                  jj = jj + 1
               ELSE                                  ! Ommit division by zero
                  jj = jj + 0
               ENDIF
            ELSE IF (ABS((Heat_Flow(ii)/Heat_Flow_Old(ii)) - 1.0D0) &
               .LT. Tolerance_Par(5)) THEN                  !
               jj = jj + 1
            ENDIF
         ENDDO
      ENDIF
      IF (jj .EQ. 4) unfinished = .FALSE.

   ENDDO                                            ! calculation loop

   IF (NOT(Modes(3))) THEN                                  ! Heat pump ist set off
      DeviceProp(IHX:Evap, m_prim:H_Flux) = 0.0D0
      DeviceProp(IHX:Evap, T_prim_out) = DeviceProp(IHX:Evap, T_prim_in)
      DeviceProp(IHX:Evap, T_sec_out) = DeviceProp(IHX:Evap, T_sec_in)
   ENDIF
```

```
! Performance Evaluation
!----------------------
    Q_tot_hot = DeviceProp(GC1, H_Flux) + DeviceProp(GC2, H_Flux) + DeviceProp(GC3, H_Flux)
    COP = Q_tot_hot / DeviceProp(Comp, Div_1)                    ! COP (warm side effiency of heat pump)
    EER = DeviceProp(Evap, H_Flux) / DeviceProp(Comp, Div_1)     ! EER (cold side effiency of heat pump)


!Set the Outputs from this Model (#,Value)

    Call SetOutputValue(1, DeviceProp(Comp, Div_1))              ! [kJ/hr] Electric power of the compressor
    Call SetOutputValue(2, Q_tot_hot)                            ! [kJ/hr] Total heat delivered
    Call SetOutputValue(3, CO2prop(St_a_Comp, mflow))            ! [kg/hr] CO2 mass flow
    Call SetOutputValue(4, CO2prop(St_a_Comp, press) / bar2Pa)   ! [bar] High pressure of CO2 reached
    Call SetOutputValue(5, DeviceProp(Comp, TC_HX))              ! [s^-1] realized frequency
    Call SetOutputValue(6, DeviceProp(GC1, H_Flux))              ! [kJ/hr] Heat delivered GC1
    Call SetOutputValue(7, DeviceProp(GC1, T_sec_in))            ! [°C] H2O inlet temperature into GC1
    Call SetOutputValue(8, DeviceProp(GC1, T_sec_out))           ! [°C] H2O outlet temperature from GC1
    Call SetOutputValue(9, DeviceProp(GC1, m_sec))               ! [kg/hr] H2O mass flow in GC1
    Call SetOutputValue(10,DeviceProp(GC2, H_Flux))              ! [kJ/hr] Heat delivered GC2
    Call SetOutputValue(11,DeviceProp(GC2, T_sec_in))            ! [°C] H2O inlet temperature into GC2
    Call SetOutputValue(12,DeviceProp(GC2, T_sec_out))           ! [°C] H2O outlet temperature from GC2
    Call SetOutputValue(13,DeviceProp(GC2, m_sec))               ! [kg/hr] H2O mass flow in GC2
    Call SetOutputValue(14,DeviceProp(GC3, H_Flux))              ! [kJ/hr] Heat delivered GC3
    Call SetOutputValue(15,DeviceProp(GC3, T_sec_in))            ! [°C] H2O inlet temperature into GC3
    Call SetOutputValue(16,DeviceProp(GC3, T_sec_out))           ! [°C] H2O outlet temperature from GC3
    Call SetOutputValue(17,DeviceProp(GC3, m_sec))               ! [kg/hr] H2O mass flow in GC3
    Call SetOutputValue(18,DeviceProp(Evap, H_Flux))             ! [kJ/hr] Heat uptake Evaporator
    Call SetOutputValue(19,DeviceProp(Evap, T_prim_in))          ! [°C] H2O inlet temperature into Evaporator
    Call SetOutputValue(20,DeviceProp(Evap, T_prim_out))         ! [°C] H2O outlet temperature from Evaporator
    Call SetOutputValue(21,DeviceProp(Evap, m_prim))             ! [kg/hr] H2O mass flow in Evaporator
    Call SetOutputValue(22,CO2prop(St_a_Comp,Temp))              ! [°C] CO2 outlet temperature behind
compressor
    Call SetOutputValue(23,CO2prop(St_a_Tube,Temp))              ! [°C] CO2 outlet temperature behind hot gas
tube and oil separator
    Call SetOutputValue(24,CO2prop(St_a_GC1,Temp))               ! [°C] CO2 outlet temperature behind 1st gas
cooler
    Call SetOutputValue(25,CO2prop(St_a_GC2,Temp))               ! [°C] CO2 outlet temperature behind 2nd gas
cooler
    Call SetOutputValue(26,CO2prop(St_a_GC3,Temp))               ! [°C] CO2 outlet temperature behind 3rd gas
cooler
    Call SetOutputValue(27,CO2prop(St_a_IHX_HP,Temp))            ! [°C] CO2 outlet temperature behind internal
heat recuperator, high pressure side
    Call SetOutputValue(28,CO2prop(St_i_Evap,Temp))              ! [°C] CO2 phase change temperature inside
evaporator
    Call SetOutputValue(29,CO2prop(St_a_Evap,Temp))              ! [°C] CO2 outlet temperature behind evaporator
    Call SetOutputValue(30,CO2prop(St_a_IHX_LP,Temp))            ! [°C] CO2 outlet temperature behind internal
heat recuperator (IHX), low pressure side
    Call SetOutputValue(31,DeviceProp(IHX, H_Flux))              ! [kJ/hr] Heat recuparated in IHX
    Call SetOutputValue(32,CO2prop(St_i_Evap,Press) / bar2Pa)    ! [bar] Low pressure of CO2 (defined by
phase change temperature)
    Call SetOutputValue(33,emergency)                            ! [0, 1] 0 => emergency stop
    Call SetOutputValue(34,COP)                                  ! [-] Warm side efficiency factor of heat pump
    Call SetOutputValue(35,EER)                                  ! [-] Cold side efficiency factor of heat pump
    Call SetOutputValue(36,DBLE(sec(St_a_Comp)))                 ! [s] Execution time compressor calculation
    Call SetOutputValue(37,DBLE(sec(St_a_Tube)))                 ! [s] Execution time oil separator calculation
    Call SetOutputValue(38,DBLE(sec(St_a_GC1)))                  ! [s] Execution time gas cooler 1 calculation
    Call SetOutputValue(39,DBLE(sec(St_a_GC2)))                  ! [s] Execution time gas cooler 2 calculation
    Call SetOutputValue(40,DBLE(sec(St_a_GC3)))                  ! [s] Execution time gas cooler 3 calculation
    Call SetOutputValue(41,DBLE(sec(St_a_IHX_HP)))               ! [s] Execution time IHX high pressure calculation
    Call SetOutputValue(42,DBLE(sec(St_a_Evap)))                 ! [s] Execution time evaporator calculation
    Call SetOutputValue(43,DBLE(sec(St_a_IHX_LP)))               ! [s] Execution time IHX low pressure calculation
```

```
    Return
End
!-----------------------------------------------------------------------------------------------------------------------


!-----------------------------------------------------------------------------------------------------------------------
! +++ Subroutines to calculate CO2 properties
!-----------------------------------------------------------------------------------------------------------------------


SUBROUTINE CO2_h_cp_interpol(LU_data_file, Temperature, Pressure, Enthalpy, Heat_Capacity, &
                             Call_CoolProp, CurrentUnit, CurrentType)                    ! [-], [°C], [Pa], [kJ/kg], [kJ/(kgK)], [Bin],
[-], [-]

! Enthalpie and heat capacity interpolatation for CO2

    Implicit None

    INTEGER LU_data_file(8)                         ! File handle number from data files (1 - 3 from compressor model
and 4 - 8 from CO2 data)
    DOUBLE PRECISION, PARAMETER :: bar2Pa = 1.0D5
    DOUBLE PRECISION Temperature                        ! Input CO2 temperature
    DOUBLE PRECISION Pressure, P_bar                    ! [Pa], [bar] Input CO2 pressure
    DOUBLE PRECISION Enthalpy                       ! Output CO2 enthalpy
    DOUBLE PRECISION Heat_Capacity                      ! Output CO2 heat capacity
    INTEGER CurrentUnit,CurrentType

! Function calls
    DOUBLE PRECISION Enthalpy_CO2
    DOUBLE PRECISION Heat_Cap_CO2

! Interpolation data file
    INTEGER ii                                  ! Counter
    INTEGER  n_Inp_Var(3)                           ! Vektor with number of input data points
    INTEGER, PARAMETER :: Dim_Inp_Vektor = 3            ! Dimension of Input Vektor (1 < n < 6)
    INTEGER, PARAMETER :: Dim_Out_Vektor = 3            ! Dimension of Output Vektor (1 < n < 10)
    INTEGER, PARAMETER :: Dim_Inp_Test_V = 2            ! Dimension of Input Vektor (1 < n < 6)
    INTEGER, PARAMETER :: n_Inp_Var_HP_3 = 34           ! 1st line in data file, Number of pressure points
    INTEGER, PARAMETER :: n_Inp_Var_12_3 = 29           ! 1st line in data file, Number of pressure points
    INTEGER, PARAMETER :: n_Inp_Var_40_3 = 31           ! 1st line in data file, Number of pressure points
    INTEGER, PARAMETER :: n_Inp_Var_60_2 = 15               ! 2nd line in data file, Number of temperature
multiplicators (*8.0) points
    INTEGER, PARAMETER :: n_Inp_Var_20_2 = 10               ! 2nd line in data file, Number of temperature
multiplicators (*4.0) points
    INTEGER, PARAMETER :: n_Inp_Var_12_2 = 12               ! 2nd line in data file, Number of temperature
multiplicators (*8.0) points
    INTEGER, PARAMETER :: n_Inp_Var_40_2 = 15               ! 2nd line in data file, Number of temperature
multiplicators (*4.0) points
    INTEGER, PARAMETER :: n_Inp_Test_2 = 16         ! 2nd line in data file, Number of temperature multiplicators
(*4.0) points
    INTEGER, PARAMETER :: n_Inp_Var_1 = 41          ! 3rd line in data file, Number of temperature multiplicators
(*0.2 resp. *0.1) points
    DOUBLE PRECISION Input_Vektor(3)                ! 1 = Temp_Mult (*0.2 / *0.1), 2 = Temp_Mult (*8.0 / *4.0 ),
3 = Pressure
    DOUBLE PRECISION Input_Test_V(2)                ! 1 = Temp_Mult (*0.1), 2 = Temp_Mult (*4.0)
    DOUBLE PRECISION Output_Vektor(3)                   ! 1 = Temperature, 2 = Enthalpy, 3 = Heat Capacity
    DOUBLE PRECISION Output_Test_V(3)                   ! 1 = Temperature, 2 = Enthalpy, 3 = Pressure

! Range calculation for linearization
    DOUBLE PRECISION, PARAMETER :: Press_Forb = 82.5D0      ! [bar] Between this pressure and the critical
pressure, there is a temperature range where access to matrix is not allowed
```

```
   DOUBLE PRECISION Peak_Temp_Forb                    ! Peak temperature in the forbidden pressure range,
where access to matrix is not allowed
   DOUBLE PRECISION, PARAMETER :: Temp_Spread_Forb = 3.1D0 ! Temperature range around the peak tem-
perature
   DOUBLE PRECISION, PARAMETER :: Critical_Press = 7.38D1  ! Pressure of the critical point
   DOUBLE PRECISION, PARAMETER :: Low_Press_HP = 7.4D1     ! Low pressure boundary in the high pressure
CO2 matrices
   DOUBLE PRECISION, PARAMETER :: High_Press_HP = 1.4D2    ! High pressure boundary in the high pressure
CO2 matrices

! Define some flags
   LOGICAL Close_2_Critic_Point                    ! TRUE if the range is not precise enough with the linearized
method. Use direct access of Coolprop
   LOGICAL Valid_Press_Range                     ! TRUE if pressure is given in the high pressure data matrix
   LOGICAL Call_CoolProp                       ! TRUE if Coolprop must be used

! Initialize some data
!---------------------
   Close_2_Critic_Point = .FALSE.
   Valid_Press_Range = .FALSE.
   Call_CoolProp = .FALSE.
   DO ii = 1, 3
      Output_Vektor(ii) = 0.0D0
   EndDo
   P_bar = Pressure / bar2Pa


   IF (P_bar .GT. Critical_Press) THEN                                ! High Pressure calculations

! Define range, where linearization is not precise enough
!--------------------------------------------------------------
      Peak_Temp_Forb = 0.0062D0 * P_bar * P_bar - 0.4132D0 * P_bar + 27.631D0          ! [°C] Regression of
the nonlinear peaks of enthalpy and heat capacity. Do not use linearization close to critical point.

      IF ((P_bar .LT. Press_Forb) .AND. &                          ! In the forbidden range, call the CoolProp
database (slow!)
         ((Temperature .LT. (Peak_Temp_Forb + Temp_Spread_Forb)) .AND. &
         (Temperature .GT. (Peak_Temp_Forb - Temp_Spread_Forb)))) THEN
         Close_2_Critic_Point = .TRUE.
      ENDIF

      IF ((P_bar .GE. Low_Press_HP) .AND. (P_bar .LE. High_Press_HP)) THEN          ! If pressure range is
in the matrices defined, call the matrix calculation, else CoolProp database (slow!)
         Valid_Press_Range = .TRUE.
      ENDIF

      IF ((Temperature .GE. 60.0D0) .AND. (Temperature .LT. 180.0D0) &
         .AND. (Valid_Press_Range)) THEN                          ! Read from Matrix 60-180. (higher
distance of the base points)
         n_Inp_Var(1) = n_Inp_Var_1                            ! Define number of indexes in 3rd line of
data file
         n_Inp_Var(2) = n_Inp_Var_60_2                          ! Define number of indexes in 2nd line
of data file
         n_Inp_Var(3) = n_Inp_Var_HP_3                          ! Define number of indexes in 1st line
of data file
         Input_Vektor(3) = P_bar
         Input_Vektor(2) = DBLE(INT((Temperature - 4.0D0)/8.0D0))
         Input_Vektor(1) = DMOD((Temperature - 4.0D0), 8.0D0) * 5.0D0
         Call InterpolateData(LU_data_file(4), Dim_Inp_Vektor, n_Inp_Var, Dim_Out_Vektor, Input_Vektor, Out-
put_Vektor)

      ELSE IF ((Temperature .GE. 20.0D0) .AND. (Temperature .LT. 60.0D0) &
```

```
        .AND. (Valid_Press_Range) .AND. NOT(Close_2_Critic_Point))THEN                ! Read from Matrix 20-
60. (lower distance of the base points)
          n_Inp_Var(1) = n_Inp_Var_1                                                  ! Define number of indexes in 3rd line of
data file
          n_Inp_Var(2) = n_Inp_Var_20_2                                               ! Define number of indexes in 2nd line
of data file
          n_Inp_Var(3) = n_Inp_Var_HP_3                                               ! Define number of indexes in 1st line
of data file
          Input_Vektor(3) = P_bar
          Input_Vektor(2) = DBLE(INT((Temperature / 4.0D0)))
          Input_Vektor(1) = DMOD(Temperature, 4.0D0) * 10.0D0
          Call InterpolateData(LU_data_file(5), Dim_Inp_Vektor, n_Inp_Var, Dim_Out_Vektor, Input_Vektor, Out-
put_Vektor)

        ELSE
          Call_CoolProp = .TRUE.
        ENDIF                                                                         ! End calculation for high pressure CO2

      ELSE                                                                            ! Low Pressure calculations

        IF ((P_bar .GE. 12.0D0) .AND. (P_bar .LT. 40.0D0) .AND. &
          (Temperature .LT. 62.8D0) .AND. (Temperature .GT. -33.2D0)) THEN            ! Read from Matrix P12-
P40. (higher distance of the base points)
          n_Inp_Var(1) = n_Inp_Var_1                                                  ! Define number of indexes in 3rd line of
data file
          n_Inp_Var(2) = n_Inp_Var_12_2                                               ! Define number of indexes in 2nd line
of data file
          n_Inp_Var(3) = n_Inp_Var_12_3                                               ! Define number of indexes in 1st line of
data file
          Input_Vektor(3) = P_bar
          Input_Vektor(2) = DBLE(INT((Temperature + 33.2D0)/8.0D0))
          Input_Vektor(1) = DMOD((Temperature + 33.2D0), 8.0D0) * 5.0D0
          Call InterpolateData(LU_data_file(6), Dim_Inp_Vektor, n_Inp_Var, Dim_Out_Vektor, Input_Vektor, Out-
put_Vektor)

        ELSE IF ((P_bar .GE. 40.0D0) .AND. (P_bar .LT. 70.0D0) .AND. &
          (Temperature .LT. 62.8D0) .AND. (Temperature .GT. 2.8D0)) THEN              ! Read from Matrix P40-
P70. (lower distance of the base points)
          n_Inp_Var(1) = n_Inp_Var_1                                                  ! Define number of indexes in 3rd line of
data file
          n_Inp_Var(2) = n_Inp_Var_40_2                                               ! Define number of indexes in 2nd line
of data file
          n_Inp_Var(3) = n_Inp_Var_40_3                                               ! Define number of indexes in 1st line of
data file
          Input_Vektor(3) = P_bar
          Input_Vektor(2) = DBLE(INT((Temperature - 2.8D0)/ 4.0D0))
          Input_Vektor(1) = DMOD((Temperature - 2.8D0), 4.0D0) * 10.0D0
          Call InterpolateData(LU_data_file(7), Dim_Inp_Vektor, n_Inp_Var, Dim_Out_Vektor, Input_Vektor, Out-
put_Vektor)

        ELSE
          Call_CoolProp = .TRUE.
        ENDIF

        IF (Temperature .LE. 30.8D0) THEN                                             ! Test if CO2 is not in phase change
but in pure liquid aera
          n_Inp_Var(1) = n_Inp_Var_1                                                  ! Define number of indexes in 2nd line of
data file
          n_Inp_Var(2) = n_Inp_Test_2                                                 ! Define number of indexes in 1st line of
data file
          Input_Test_V(2) = DBLE(INT(((Temperature + 33.1D0 - 0.1D0) / 4.0D0)))
          Input_Test_V(1) = DMOD((Temperature + 33.1D0 - 0.1D0), 4.0D0) * 10.0D0
```

```
          Call InterpolateData(LU_data_file(8), Dim_Inp_Test_V, n_Inp_Var, Dim_Out_Vektor, Input_Test_V, Out-
put_Test_V)
          IF (Output_Test_V(3) .LT. P_bar) THEN
             Call_CoolProp = .TRUE.
          ENDIF
       ENDIF
    ENDIF

    IF (Call_CoolProp) THEN
       Enthalpy     = Enthalpy_CO2(Temperature, Pressure, CurrentUnit, CurrentType)          ! [kJ/kg], ([°C], [Pa], [-
], [-])
       Heat_Capacity = Heat_Cap_CO2(Temperature, Pressure, CurrentUnit, CurrentType)          ! [kJ/(kgK)], ([°C],
[Pa], [-], [-])
    ELSE
       Enthalpy     = Output_Vektor(2)                                                     ! [kJ/kg]
       Heat_Capacity = Output_Vektor(3)                                                      ! [kJ/(kgK)]
    ENDIF


END SUBROUTINE


!------------------------------------------------------------------------------------------------------------------------------
! +++ Functions that call coolprop
!------------------------------------------------------------------------------------------------------------------------------

DOUBLE PRECISION FUNCTION Enthalpy_CO2(Temp, pressure, CurrentUnit, CurrentType)     ! [kJ/kg], ([°C],
[Pa], [-], [-])

! Calculate Enthalpy [kJ/kg] of CO2 calling Coolprop

    Use CPINTERFACE
    Use TrnsysFunctions

    Implicit None

    Integer CurrentUnit, CurrentType
    Double Precision Temp, pressure

    if (Temp < - 273.15) then
       Call Messages(-1,'Physically impossible coolprop input, temperature lower than 0 K', &
                     'Fatal', CurrentUnit,CurrentType)
       Return
    end if

    if (pressure > 1.0D8) then
       Call Messages(-1,'Physically impossible coolprop input, pressure higher than 1000 bar', &
                    'Fatal', CurrentUnit,CurrentType)
       Return
    end if

! CoolProp_PropsSI calculates Enthalpy [kJ/kg] of CO2 by calling CoolProp with temperature [K] and pressure of
CO2 [Pa]
       Enthalpy_CO2 = CoolProp_PropsSI("H"//CHAR(0), "T"//CHAR(0), Temp + 273.15, "P"//CHAR(0), pressure, &
                                  "R744"//CHAR(0)) / 1000.0

    Return

END FUNCTION Enthalpy_CO2


!******************************************************************************
```

```
DOUBLE PRECISION FUNCTION Enthalpy_PC_CO2(Temp, Quality, CurrentUnit, CurrentType)  ! [kJ/kg], ([°C], [Pa], [-], [-])

! Calculate Enthalpy [kJ/kg] of CO2

    Use CPINTERFACE
    Use TrnsysFunctions

    Implicit None

    Integer CurrentUnit, CurrentType
    Double Precision Temp, Quality

    if (Temp < - 273.15) then
        Call Messages(-1,'Physically impossible coolprop input, temperature lower than 0 K', &
                      'Fatal', CurrentUnit,CurrentType)
        Return
    end if

    if ((Quality > 1.0D0) .OR. (Quality < 0.0D0)) then
        Call Messages(-1,'Physically impossible coolprop input, Quality not between 0 & 1', &
                    'Fatal', CurrentUnit,CurrentType)
        Return
    end if

! CoolProp_PropsSI calculates Enthalpy [kJ/kg] of CO2 by calling CoolProp with temperature [K] and pressure of CO2 [Pa]
        Enthalpy_PC_CO2 = CoolProp_PropsSI("H"//CHAR(0), "T"//CHAR(0), Temp + 273.15, "Q"//CHAR(0), Quality, &
                                    "R744"//CHAR(0)) / 1000.0

    Return

END FUNCTION Enthalpy_PC_CO2


!*******************************************************************************


DOUBLE PRECISION FUNCTION Heat_Cap_CO2(Temp, Pressure, CurrentUnit, CurrentType)   ! [kJ/(kgK)], ([°C], [Pa], [-], [-])

! Calculate heat capacity of CO2 [kJ/(kgK)] by calling CoolProp with temperature and pressure

    Use CPINTERFACE
    Use TrnsysFunctions

    Implicit None

    Integer CurrentUnit, CurrentType
    Double Precision Temp, Pressure
    Double Precision Enthalpy_high, Enthalpy_low
    DOUBLE PRECISION, PARAMETER :: Delta_temp = 2.5D-2
    DOUBLE PRECISION, PARAMETER :: kJ = 1.0D3


    if (Temp < (-273.15)) then
        Call Messages(-1,'Physically impossible coolprop input, temperature lower than 0 K', &
                    'Fatal', CurrentUnit, CurrentType)
        Return
    end if

    if (pressure > 1.0D8) then
```

```
      Call Messages(-1,'Physically impossible coolprop input, pressure higher than 1000 bar', &
                    'Fatal', CurrentUnit, CurrentType)
      Return
   end if

! CoolProp_PropsSI calculates twice the internal energy of CO2 [Pa] by calling CoolProp with
! a temperature [K] spread around the temperature given and the pressure given [Pa]
   Enthalpy_high = CoolProp_PropsSI("H"//CHAR(0), "T"//CHAR(0), Temp + 273.15 + Delta_temp, &
                                    "P"//CHAR(0), Pressure, "R744"//CHAR(0))
   Enthalpy_low = CoolProp_PropsSI("H"//CHAR(0), "T"//CHAR(0), Temp + 273.15 - Delta_temp, &
                                    "P"//CHAR(0), Pressure, "R744"//CHAR(0))

! Heat capacity: Delta U / Delta T
   Heat_Cap_CO2 = (Enthalpy_high - Enthalpy_low) / (2.0 * Delta_temp) / kJ

   Return

END FUNCTION Heat_Cap_CO2


!*******************************************************************************


DOUBLE PRECISION FUNCTION Temperature_CO2(pressure, Enthalpy, CurrentUnit, CurrentType) ! [°C], ([Pa],
[kJ/kg], [-], [-])

! Calculate Temperature of CO2 [°C]

   Use CPINTERFACE
   Use TrnsysFunctions

   Implicit None

   Integer CurrentUnit, CurrentType
   Double Precision pressure, Enthalpy
   DOUBLE PRECISION, PARAMETER ::  kJ2J = 1.0D3

   if (pressure > 1.0D8) then
      Call Messages(-1,'Physically impossible coolprop input, pressure higher than 1000 bar', &
                    'Fatal', CurrentUnit,CurrentType)
      Return
   end if

! CoolProp_PropsSI calculates Temperature of CO2 [K] by calling CoolProp with pressure [Pa] and enthalpy [J/kg]
   Temperature_CO2 = CoolProp_PropsSI("T"//CHAR(0), "P"//CHAR(0), pressure, "H"//CHAR(0), &
                                    Enthalpy *  kJ2J, "R744"//CHAR(0)) - 273.15

   Return

END FUNCTION Temperature_CO2


!*******************************************************************************


DOUBLE PRECISION FUNCTION Pressure_CO2(Temp, Quality, CurrentUnit, CurrentType)      ! [Pa], ([°C],
[mol/mol], [-], [-])

! Calculate Pressure of CO2 [Pa] by calling CoolProp with temperature and Quality

   Use CPINTERFACE
   Use TrnsysFunctions
```

```
    Implicit None

    Integer CurrentUnit, CurrentType
    Double Precision Temp, Quality

    if (Temp < (-273.15)) then
       Call Messages(-1,'Physically impossible coolprop input, temperature lower than 0 K', &
                     'Fatal', CurrentUnit,CurrentType)
       Return
    end if

! CoolProp_PropsSI calculates Pressure of CO2 [Pa] by calling CoolProp with temperature [K] and Quality [mol vapor / mol mixture (liquid&vapor) CO2]
    Pressure_CO2 = CoolProp_PropsSI("P"//CHAR(0), "T"//CHAR(0), Temp + 273.15, "Q"//CHAR(0), Quality, &
                                    "R744"//CHAR(0))

    Return

END FUNCTION Pressure_CO2


DOUBLE PRECISION FUNCTION Heat_Cap_H2O(Temp, CurrentUnit, CurrentType)  ! [kJ/(kgK)], ([°C], [-], [-])

! Calculate heat capacity of H2O [kJ/(kgK)] by a regression derived from Data of the VDI Wärmeatlas
! To increase precision, the regression is split into 2 parts with an linear overlapping between the two parts.

    Use TrnsysFunctions

    Implicit None

    Integer CurrentUnit, CurrentType
    Double Precision Temp, Temp2, Temp3, Frac

    if (Temp .LE. (0.0D0)) then
       Call Messages(-1,'Heat capacitiy calculation, water temperature lower than 0 °C', 'Fatal', CurrentUnit, CurrentType)
       Return
    end if

    if (Temp .GE. (2.0D2)) then
       Call Messages(-1,'Water temperature out of defined range, higher than 200 °C', 'Fatal', CurrentUnit, CurrentType)
       Return
    end if

    Temp2 = Temp * Temp
    Temp3 = Temp * Temp2

    IF (Temp .LE. 7.5D1) THEN                                ! Heat capacity for temperatures 0.01°C to 75°C
       Heat_Cap_H2O = -3.184799114768650D-01 + 6.172890329388810D-02 * Temp -5.048559975396990D-05 * Temp2 -1.197996907583730D-05 * Temp3 &
                      +2.224146575534990D-07 * Temp2 * Temp2 -1.801036285383610D-09 * Temp2 * Temp3 + 5.643976662308390D-12 * Temp3 * Temp3 &
                      +4.547490089164010D+00 * DEXP(-1.524659270682380D-02 * Temp)
    ELSE IF (Temp .GE. 8.5D1) THEN                           ! Heat capacity for temperatures 75°C to 200°C
       Heat_Cap_H2O = 9.73928331460531D-01 + 2.32676990702029D-02 * Temp +1.38606151448530D-04 * Temp2 -2.92080260527313D-06 * Temp3 &
                      +1.88607573040776D-08 * Temp2 * Temp2 -5.69581145617517D-11 * Temp2 * Temp3 + 6.90700195644651D-14 * Temp3 * Temp3 &
                      +3.63968151163685D+00 * DEXP(-1.22800007770101D-02 * Temp)
    ELSE                                                     ! Heat capacity for temperatures 75°C to 85°C (overlapping regression to ommit a step)
```

```
      Frac = MAX(0.0, MIN(1.0D0, (8.5D1 - Temp) * 1.0D-1))       ! 85°C: Frac = 0.0, 75°C: Frac = 1.0
      Heat_Cap_H2O =      Frac  *  (-3.184799114768650D-01  +  6.172890329388810D-02  *  Temp  -
5.048559975396990D-05 * Temp2 -1.197996907583730D-05 * Temp3 &
                +2.224146575534990D-07  *  Temp2  *  Temp2  -1.801036285383610D-09  *  Temp2  *  Temp3  +
5.643976662308390D-12 * Temp3 * Temp3 &
                +4.547490089164010D+00 * DEXP(-1.524659270682380D-02 * Temp)) &
                +  (1.0D0  -  Frac)  *  (9.73928331460531D-01  +  2.32676990702029D-02  *  Temp
+1.38606151448530D-04 * Temp2 -2.92080260527313D-06 * Temp3 &
                +1.88607573040776D-08  *  Temp2  *  Temp2  -5.69581145617517D-11  *  Temp2  *  Temp3  +
6.90700195644651D-14 * Temp * Temp3 &
                +3.63968151163685D+00 * DEXP(-1.22800007770101D-02 * Temp))
   ENDIF

   Return

END FUNCTION Heat_Cap_H2O



!-----------------------------------------------------------------------------------------------------------------------
! +++ Heat loss of high temperature parts
!-----------------------------------------------------------------------------------------------------------------------

SUBROUTINE HeatLoss(CO2prop, DeviceProp, Tube_Parameter, LU_data_file, CurrentUnit, CurrentType)

! Subroutine to calculate heat loss [kJ/hr] between compressor and first gas cooler.
! All other heat losses (also for first gas cooler) are neglected

   IMPLICIT NONE
   INTEGER CurrentUnit, CurrentType
   INTEGER LU_data_file(8)                              ! File handle number from data files (1 - 3 from compressor
model and 4 - 8 from CO2 data)
   DOUBLE PRECISION DeviceProp(8, 13)                   ! DeviceProp(device_numbering, device_property)
--> see list 3&4 below
   INTEGER IHX, Comp, Tube, GC1, GC2, GC3, Valve, Evap, Device        ! Devices
   INTEGER T_prim_in, T_sec_in, T_prim_out, T_sec_out, m_prim, m_sec, &
         A_Exc, H_Flux, TC_HX, TCb_HX, Div_1, Div_2, Div_3          ! List of device properties
   DOUBLE PRECISION Tube_Parameter(13)                  ! Parameter list to calculate tube and oil separator
losses
   DOUBLE PRECISION CO2prop(10,5)                       ! CO2prop(position, CO2property) --> see list 1&2
below
   INTEGER St_a_Evap, St_a_IHX_LP, St_a_Comp, St_a_Tube, St_a_GC1, &
         St_a_GC2, St_a_GC3, St_a_IHX_HP, St_a_Valve, St_i_Evap     ! List of states in the CO2 circle
   INTEGER Temp, Press, Entha, mflow, H_Cap             ! List of CO2 properties
   LOGICAL Call_CoolProp                                ! Used in case of under temperature behind evaporator

   DOUBLE PRECISION R_Therm_Tube                        ! Function call to calculate heat resistance of tube
insulation
   DOUBLE PRECISION Pressure, Enthalpy, cp
   DOUBLE PRECISION R_Lid_Oil_sep
   DOUBLE PRECISION Heat_loss_1st_tube, Heat_loss_oil_sep_wall, Heat_loss_oil_sep_lid,
Heat_loss_2nd_tube
   DOUBLE PRECISION dTLoss, Thigh, T_Oil_Sep, T_Tube2

! Parameters
!-----------
   DOUBLE PRECISION, PARAMETER ::  pi         = 3.1415926535897932D0
   DOUBLE PRECISION Lambda                ! [kJ/(hr.m.K)] Insulation conductivity
   DOUBLE PRECISION Ins_thick_tube        ! [m] Insulation thickness tube
   DOUBLE PRECISION Ins_thick_OS          ! [m] Insulation thickness oil separator
   DOUBLE PRECISION Temp_Air              ! [°C] Air temperature inside heat pump housing
   DOUBLE PRECISION alpha_out             ! [kJ/(hr.m2K)] Heat transfer coefficient outside tube / oil separator
   DOUBLE PRECISION alpha_in              ! [kJ/(hr.m2K)] Heat transfer coefficient inside tube / oil separator
```

```
! Parameters for tube
!--------------------
   DOUBLE PRECISION Rad_in                ! [m] Radius of pipe (without insulation)
   DOUBLE PRECISION Rad_out               ! [m] [m] Radius of pipe (with insulation)
   DOUBLE PRECISION Length_1              ! [m] Tube length between compressor and oil separator
   DOUBLE PRECISION Length_2              ! [m] Tube length between oil separator and first heat exchanger

! Parameters for oil separator
!-----------------------------

   DOUBLE PRECISION Rad_in_OS             ! [m] Inner radius of oil separator
   DOUBLE PRECISION Rad_out_OS            ! [m] Outer radius of oil separator
   DOUBLE PRECISION Hight                 ! [m] Hight of oil separator wall

! Define the CO2 and device properties
!-------------------------------------
   St_a_Comp   = 3 ! State of CO2 behind the compressor and before hot gas tube and oil separator
   St_a_Tube   = 4 ! State of CO2 behind hot gas tube and oil separator and before the 1st gas cooler

   Temp        = 1 ! [°C] CO2 Temperature
   Press       = 2 ! [Pa] CO2 Pressure

   Tube        = 3 ! Hot gas tube and oil separator

   T_prim_in   = 1 ! [°C] Heat exchanger inlet temperature hot side
   T_prim_out  = 3 ! [°C] Heat exchanger outlet temperature hot side
   m_prim      = 5 ! [kg/h] Heat exchanger mass flow hot side
   H_Flux      = 8 ! [kJ/h] Heat flux


   Lambda =           Tube_Parameter(1)      ! [kJ/(hr.m.K)] Insulation conductivity
   Ins_thick_tube =   Tube_Parameter(2)      ! [m] Insulation thickness tube
   Ins_thick_OS =     Tube_Parameter(3)      ! [m] Insulation thickness oil separator
   alpha_out =        Tube_Parameter(4)      ! [kJ/(hr.m2.K)] Heat transfer coefficient outside tube / oil separator
   alpha_in =         Tube_Parameter(5)      ! [kJ/(hr.m2.K)] Heat transfer coefficient inside tube / oil separator
   Rad_in =           Tube_Parameter(6) / 2.0 ! [m] Radius of pipe (without insulation)
   Rad_out = Rad_in + Ins_thick_tube         ! [m] Radius of pipe (with insulation)
   Length_1 =         Tube_Parameter(7)      ! [m] Tube length between compressor and oil separator
   Length_2 =         Tube_Parameter(8)      ! [m] Tube length between oil separator and first heat exchanger
   Rad_in_OS =        Tube_Parameter(9) / 2.0 ! [m] Radius of oil separator (without insulation)
   Rad_out_OS = Rad_in_OS + Ins_thick_OS     ! [m] Radius of oil separator (with insulation)
   Hight =            Tube_Parameter(10)     ! [m] Hight of oil separator wall
   Temp_Air =         Tube_Parameter(11)     ! [°C] Air temperature inside heat pump housing


! Adapt some variables to short names
!------------------------------------

   Pressure = CO2prop(St_a_Comp, Press)


! Heat loss first tube between compressor and oil separator
   Heat_loss_1st_tube = Length_1 * (CO2prop(St_a_Comp, Temp) - Temp_Air) / &
                        R_Therm_Tube(Rad_in, Rad_out, Alpha_in, Alpha_out, Lambda)          ! [kJ/h]
   Call CO2_h_cp_interpol(LU_data_file, CO2prop(St_a_Comp, Temp), Pressure, Enthalpy, cp, &
         Call_CoolProp, CurrentUnit, CurrentType)                                  ! [-], [°C], [Pa], [kJ/kg], [kJ/(kgK)], [bin],
[-], [-]
   dTLoss = Heat_loss_1st_tube / (DeviceProp(Tube, m_prim) * cp)                     ! [°C]
   T_Oil_Sep = CO2prop(St_a_Comp, Temp) - dTLoss                                     ! [°C]

! Heat loss oil separator
   Heat_loss_oil_sep_wall = Hight * (T_Oil_Sep - Temp_Air) / &
```

```
                    R_Therm_Tube(Rad_in_OS, Rad_out_OS, Alpha_in, Alpha_out, Lambda)        ! [kJ/h]
    R_Lid_Oil_sep = (1.0D0 / alpha_in) + (Ins_thick_OS / Lambda) + (1.0D0 / alpha_out)          ! [(hr.m2.K)/kJ]
    Heat_loss_oil_sep_lid = Rad_out_OS * Rad_out_OS * pi * (T_Oil_Sep - Temp_Air) / &
                    R_Lid_Oil_sep * 2.0D0                                                      ! [kJ/h]
    Call CO2_h_cp_interpol(LU_data_file, T_Oil_Sep, Pressure, Enthalpy, cp, &
          Call_CoolProp, CurrentUnit, CurrentType)                                          ! [-], [°C], [Pa], [kJ/kg], [kJ/(kgK)], [bin],
[-], [-]
    dTLoss = (Heat_loss_oil_sep_wall + Heat_loss_oil_sep_lid) / (DeviceProp(Tube, m_prim) * cp) ! [°C]
    T_Tube2 = T_Oil_Sep - dTLoss

! Heat loss 2nd tube between oil separator and first gas cooler
    Heat_loss_2nd_tube = Length_2 * (T_Tube2 - Temp_Air) / &
                    R_Therm_Tube(Rad_in, Rad_out, Alpha_in, Alpha_out, Lambda)              ! [kJ/h]
    Call CO2_h_cp_interpol(LU_data_file, T_Tube2, Pressure, Enthalpy, cp, &
          Call_CoolProp, CurrentUnit, CurrentType)                                          ! [-], [°C], [Pa], [kJ/kg], [kJ/(kgK)], [Bin],
[-], [-]
    dTLoss = Heat_loss_2nd_tube / (DeviceProp(Tube, m_prim) * cp)                           ! [°C]
    DeviceProp(Tube, T_prim_out) = T_Tube2 - dTLoss                                     ! [°C]

! Total heat losses
    DeviceProp(Tube, H_Flux) = Heat_loss_1st_tube + Heat_loss_oil_sep_wall + &
              Heat_loss_oil_sep_lid + Heat_loss_2nd_tube                                    ![kJ/h]


END SUBROUTINE

!*****************************************************************

DOUBLE PRECISION FUNCTION R_Therm_Tube(Rad_in, Rad_out, Alpha_in, Alpha_out, Lambda)    ! [m], [m],
[kJ/(hm2K)], [kJ/(hm2K)],  [kJ/(hmK)]

! Calculate heat resistance per meter [(hr.m.K)/kJ] through the tube insulation. Metal of tube neclected.

    IMPLICIT NONE
    DOUBLE PRECISION Rad_in, Rad_out, Alpha_in, Alpha_out, Lambda
    DOUBLE PRECISION R_Heat_In, R_Insulation, R_Heat_Out
    DOUBLE PRECISION, PARAMETER :: pi = 3.1415926535897932D0

    R_Heat_In       = 1.0D0 / (pi * Rad_in * Alpha_in)            ![(hmK)/kJ] Heat transfer resistant inside
    R_Insulation    = DLOG(Rad_out/Rad_in) / (2.0D0 * pi * Lambda)  ![(hmK)/kJ] Heat transfer resistant through
insulation
    R_Heat_Out      = 1.0D0 / (pi * Rad_out * Alpha_out)          ![(hmK)/kJ] Heat transfer resistant outside

    R_Therm_Tube    = R_Heat_In + R_Insulation + R_Heat_Out

    Return

END FUNCTION R_Therm_Tube



SUBROUTINE interp1(x, y, xq, yq, nq,S)
    INTEGER S
    DOUBLE PRECISION x(S), y(S), xq(S) !IN
    DOUBLE PRECISION yq(S)           !OUT
    INTEGER nq                    !IN

    INTEGER n, i, j
    DOUBLE PRECISION x1, x2, y1, y2, xq1, xq2, yq1, yq2

! Interpolate the data
    j = 1
```

```
    DO i = 1, nq
      DO n = j, SIZE(x) - 1
        x1 = x(n)
        x2 = x(n + 1)
        y1 = y(n)
        y2 = y(n + 1)

        IF (xq(i) >= x1 .AND. xq(i) <= x2) THEN
          xq1 = xq(i) - x1
          xq2 = x2 - xq(i)
          yq1 = y1 * xq2 / (x2 - x1)
          yq2 = y2 * xq1 / (x2 - x1)
          yq(i) = yq1 + yq2
          j = n
          EXIT
        END IF
      END DO
    END DO

END SUBROUTINE


SUBROUTINE Compressor(LU_data_file, CO2prop, DeviceProp, Tolerance_Par, Modes, CurrentUnit, CurrentType, Defined)

! +++ Compressor model using interpolation of Bitzer polynomials
!-----------------------------------------------------------------------------------------------------------------------

    Use TrnsysFunctions

    IMPLICIT NONE
    INTEGER CurrentUnit, CurrentType

    DOUBLE PRECISION, PARAMETER :: bar2Pa = 1.0D5
    DOUBLE PRECISION, PARAMETER :: W2kJ_per_h = 3.6D0

    INTEGER LU_data_file(8)                    ! File handle number from data files (1 - 3 from compressor model and 3 - 8 from CO2 data)
    DOUBLE PRECISION CO2prop(10,5)             ! CO2prop(CO2state,CO2property) --> see list 1&2 in main routine
    INTEGER St_a_Evap, St_a_IHX_LP, St_a_Comp, &
          St_a_GC3, St_a_IHX_HP, St_i_Evap     ! State position in the circle of CO2 property
    INTEGER Temp, Press, mflow                 ! Property of CO2
    DOUBLE PRECISION DeviceProp(8,13)          ! DeviceProp(device_numbering, device_property) --> see list 3&4 in main routine
    INTEGER IHX, Comp, Valve, Evap             ! Devices
    INTEGER m_prim, m_sec, A_Exc, H_Flux, &
          TC_HX, TCb_HX, Div_1                 ! Property of the devices
    DOUBLE PRECISION Tolerance_Par(10)         ! Tolerance information
    LOGICAL Modes(3), Defined                  ! Some flags to signalize operation modes

    DOUBLE PRECISION Frequency                 ! Frequency of the frequency converter
    DOUBLE PRECISION T_Superheating            ! CO2 temp. diff. between phase change in Evap and IHX outlet LP
    DOUBLE PRECISION p_high_set, p_high        ! [Pa] Set pressure from Input, reached pressure
    DOUBLE PRECISION CO2_flow                  ! [kg/h] CO2 mass flow
    DOUBLE PRECISION T_Phase_Change            ! [°C] CO2 phase change temperature in Evaporator
    INTEGER ii                                 ! counter in DO loop

! Interpolation temporary variables for HP files gained by calculation with the Bitzer Software
    INTEGER, PARAMETER :: Num_Of_Indep_Var = 3 ! In routine "InterpolateData", a maximium of 6 independent inlet variables to the interpolation routine are allowed
```

```
   INTEGER, PARAMETER :: Num_Of_Dep_Var_c = 10          ! In routine "InterpolateData", a maximium of 10 dependent outlet variables from the interpolation routine are allowed
   INTEGER, PARAMETER :: Num_Of_Dep_Var_L = 2           ! In routine "InterpolateData", a maximium of 10 dependent outlet variables from the interpolation routine are allowed
   INTEGER, PARAMETER :: Num_Of_Samp_Freq = 10          ! In all HP data files, 10 base points of frequency were used (from 30Hz to 75 Hz by 5Hz)
   INTEGER, PARAMETER :: Num_Of_Samp_Superheating = 10 ! In all HP data files, 10 base points of super-heating were used (from 5K to 50K by 5k)
   INTEGER, PARAMETER :: Num_Of_Samp_GC_Out_Temp = 9   ! In all HP data files, 9 base points of super-heating were used (from 21°C to 60°C by 5°C, except the first point (4°C))

! Inlet variables into the interpolation routine
   INTEGER Num_Of_Sample_Points(Num_Of_Indep_Var)     ! number of state points in data matrix
   DOUBLE PRECISION Indep_Var(Num_Of_Indep_Var)       ! number of boundary conditions in data matrix

! Outlet variables from the interpolation routine
   DOUBLE PRECISION cmf(Num_Of_Dep_Var_c)             ! coefficient for polynom mass flow CO2
   DOUBLE PRECISION cep(Num_Of_Dep_Var_c)             ! coefficient for polynom electric power
   DOUBLE PRECISION P_Limits(Num_Of_Dep_Var_L)        ! Pressure limits in which the Polynom from Bitzer is safe

! Define the CO2 and device properties
!-------------------------------------

   St_a_Evap   = 1 ! State of CO2 behind Evaporator and before the internal heat exchanger IHX (low pressure)
   St_a_IHX_LP = 2 ! State of CO2 behind the IHX (low pressure) and before the compressor
   St_a_Comp   = 3 ! State of CO2 behind the compressor and before hot gas tube and oil separator
   St_a_GC3    = 7 ! State of CO2 behind the 3rd gas cooler and before the IHX (high pressure)
   St_a_IHX_HP = 8 ! State of CO2 behind the IHX (high pressure) and before the throttle valve
   St_i_Evap   = 10! State of CO2 in the evaporator (phase change)

   Temp        = 1 ! CO2 Temperature [°C]
   Press       = 2 ! CO2 Pressure [Pa]
   mflow       = 4 ! CO2 mass flow [kg/K]

   IHX         = 1 ! IHX, internal heat exchanger
   Comp        = 2 ! Compressor
   Valve       = 7 ! Throttle valve
   Evap        = 8 ! Evaporator

   m_prim      = 5 ! [kg/h] CO2 mass flow
   m_sec       = 6 ! Compressor: [Pa] Set Pressure
   A_Exc       = 7 ! Compressor: [-] Mechanical efficiency
   H_Flux      = 8 ! [kJ/h] Heat flux
   TC_HX       = 9 ! Compressor: [1/s] Set frequency
   TCb_HX      = 10! Compressor: [-] Frequency reduction
   Div_1       = 11! Compressor: [kJ/h] Electrical power

! Initialize some variables
!--------------------------
   p_high = 0.0D0

! Calculate only if compressor is on
!-----------------------------------

   IF (Modes(2)) Then                                   ! Compressor is running

! Adapt some variables to short names
!------------------------------------
      Frequency = DeviceProp(Comp, TC_HX) * DeviceProp(Comp, TCb_HX)          ! [1^-s] Frequency of the frequency converter
      Frequency = Max(Frequency, Tolerance_Par(7))                            ! [1^-s] Limit value to lowest frequency
```

```
      T_Superheating = CO2prop(St_a_IHX_LP, Temp) - CO2prop(St_i_Evap, Temp)  ! [K] Superheating of CO2 in
IHX before entering the compressor
      p_high_set = DeviceProp(Comp, m_sec)                              ! [Pa] Set Pressure
      T_Phase_Change = CO2prop(St_i_Evap, Temp)                          ! [°C] CO2 phase change temperature in
Evaporator

! Fill in the arrays for the interpolation routine
!-------------------------------------------------
      Num_Of_Sample_Points(1) = Num_Of_Samp_GC_Out_Temp   ! 3rd line in file --> Index invers to line num-
ber!!!!!!
      Num_Of_Sample_Points(2) = Num_Of_Samp_Superheating  ! 2nd line in file --> Index invers to line num-
ber!!!!!!
      Num_Of_Sample_Points(3) = Num_Of_Samp_Freq         ! 1st line in file --> Index invers to line number!!!!!!

      Indep_Var(1) = CO2prop(St_a_GC3, Temp)            ! [°C] Gas outlet temperature of the 3rd gascooler
      Indep_Var(2) = T_Superheating                      ! [K] Superheating of CO2 before entering the compressor
      Indep_Var(3) = Frequency                          ! [1^-s] Frequency of the frequency converter

      IF (Indep_Var(1) .GT. 6.0D1) then
        Call Messages(-1,'CO2 outlet temperature GC3 too high for compressor model calculation', &
                      'Warning', CurrentUnit,CurrentType)
      END IF

      IF (T_Superheating .GT. 5.0D1) then
        Call Messages(-1,'CO2 superheating temperature too high for compressor model calculation', &
                      'Warning', CurrentUnit,CurrentType)
      ELSE IF (T_Superheating .LT. 5.0D0) then
        Call Messages(-1,'CO2 superheating temperature too low for compressor model calculation', &
                      'Warning', CurrentUnit,CurrentType)
      END IF

! Read and calculate the coefficients for the CO2 mass flow regression
      Call InterpolateData(LU_data_file(1), Num_Of_Indep_Var, Num_Of_Sample_Points, Num_Of_Dep_Var_c,
Indep_Var, cmf)

! Read and calculate the coefficients for the electrical power regression
      Call InterpolateData(LU_data_file(2), Num_Of_Indep_Var, Num_Of_Sample_Points, Num_Of_Dep_Var_c,
Indep_Var, cep)

! Read and calculate the pressure limits for the above regressions
      Call InterpolateData(LU_data_file(3), Num_Of_Indep_Var, Num_Of_Sample_Points, Num_Of_Dep_Var_L,
Indep_Var, P_Limits)

! Make sure, pressure used is in the allowed range
      p_high_set = p_high_set / bar2Pa              ! [Pa] --> [bar]
      IF (p_high_set .LT. P_Limits(1)) THEN
        p_high = P_Limits(1)
        Call Messages(-1,'Set CO2 pressure lower (HP) than the allowed limits', 'Warning', CurrentUnit,Cur-
rentType)
      ELSE IF (p_high_set .GT. P_Limits(2)) THEN
        p_high = P_Limits(2)
        Call Messages(-1,'Set CO2 pressure higher (HP) than the allowed limits', 'Warning', CurrentUnit,Cur-
rentType)
      ELSE
        p_high = p_high_set
      ENDIF

! Calculate mass flow, power and electrical current with polynoms from Bitzer software with the help of interpolated
coefficients from data files
      CO2_flow = cmf(1) + cmf(2) * T_Phase_Change + cmf(3) * p_high + cmf(4) * T_Phase_Change**2 + cmf(5) *
&
      T_Phase_Change * p_high + cmf(6) * p_high**2 + cmf(7) * T_Phase_Change**3 + cmf(8) * p_high *  &
```

```
      T_Phase_Change**2 + cmf(9) * T_Phase_Change * p_high**2 + cmf(10) * p_high**3                    !
[kg/h]

      DeviceProp(Comp, Div_1) = cep(1) + cep(2) * T_Phase_Change + cep(3) * p_high + cep(4) *
T_Phase_Change**2 + cep(5) * &
      T_Phase_Change * p_high + cep(6) * p_high**2 + cep(7) * T_Phase_Change**3 + cep(8) * &
      p_high * T_Phase_Change**2 + cep(9) * T_Phase_Change * p_high**2 + cep(10) * p_high**3           !
[W]

      DeviceProp(Comp, Div_1) = DeviceProp(Comp, Div_1) * W2kJ_per_h

   ELSE                                    ! Compressor is stopped
      CO2_flow = 0.0D0
      DeviceProp(Comp, Div_1) = 0.0D0

   ENDIF                                   ! Compressor operation

! Fill in the CO2prop matrix
   DO ii = St_a_Evap, St_i_Evap
      CO2prop(ii, mflow) = CO2_flow
   ENDDO
   DO ii = St_a_Comp, St_a_IHX_HP
      CO2prop(ii, press) = p_high * bar2Pa
   ENDDO

! Fill in the DeviceProp matrix
   DO ii = IHX, Valve
      DeviceProp(ii, m_prim) = CO2_flow
   ENDDO
   DeviceProp(Evap, m_sec) = CO2_flow
   DeviceProp(Comp, H_Flux) = DeviceProp(Comp, Div_1) * DeviceProp(Comp, A_Exc)

! Check for not defined states of the polynoms
   Defined = .TRUE.
   IF ((Indep_Var(1) .GT. 55.0) .AND. (Frequency .GT. 70.0) .AND. (T_Superheating .LT. 15.0)) THEN
      Defined = .FALSE.
   ELSE IF ((Indep_Var(1) .GT. 50.0) .AND. (Frequency .GT. 70.0) .AND. (T_Superheating .LT. 10.0)) THEN
      Defined = .FALSE.
   ELSE IF ((Indep_Var(1) .GT. 55.0) .AND. (Frequency .GT. 65.0) .AND. (T_Superheating .LT. 10.0)) THEN
      Defined = .FALSE.
   ENDIF

   IF (NOT(Defined) .AND. Modes(2)) then
      Call Messages(-1,'CO2 compressor calculation out of defined range for the polynomial regressions', &
                  'Warning', CurrentUnit,CurrentType)
   END IF

END SUBROUTINE


Subroutine HX_calculation(Device, DeviceProp, CO2prop, Tolerance_Par, LU_data_file, CurrentUnit, Cur-
rentType)

! The HX_calculation is split into two subroutines. This first one controls which type of HX is called and if the each
node is small enough to guarantee
! a linear calculation. The heat transfer for each node is calculated in three further subroutines, two use a cellular
eps-NTU method, the third uses a LMTD method.

! Initalization
   Use TrnsysConstants
   Use TrnsysFunctions

! Trnsys Declarations
```

```
    Implicit None
    Integer CurrentUnit,CurrentType

! INPUTS/OUTPUTS
    INTEGER LU_data_file(8)                          ! File handle number from data files (1 - 3 from compressor
model and 4 - 8 from CO2 data)
    DOUBLE PRECISION DeviceProp(8, 13)               ! DeviceProp(device_numbering, device_property) -
-> see list 3&4 below
    INTEGER GC1, GC2, GC3, IHX, Evap, Device, Comp, Tube, Valve     ! List of devices
    INTEGER T_prim_in, T_sec_in, T_prim_out, T_sec_out, m_prim, &
          m_sec, A_Exc, H_Flux, TC_HX, TCb_HX, Div_1, Div_2, Div_3! List of device properties
    DOUBLE PRECISION CO2prop(10,5)                   ! CO2prop(position, CO2property) --> see list 1&2
below
    INTEGER St_a_Evap, St_a_Comp, St_i_Evap          ! List of states
    INTEGER Temp, Press, mflow                       ! List of device properties

! Tolerance parameters
    DOUBLE PRECISION Tol_Q_Deviat                    ! Tolerance power criteria eps-NTU calculation
    DOUBLE PRECISION Tol_cp_ratio                    ! safety criteria if cp difference becomes to high -->
Warning
    DOUBLE PRECISION Tol_dT_ratio                    ! Criteria to reduce size of nodes.
    DOUBLE PRECISION Tol_Phi_iter                    ! Criteria of the different heat flows in the cell modell
    DOUBLE PRECISION Tol_Q_Def_Tot                   ! Criteria of the difference of heat flows (Phi against
Delta Enthalpy)
    DOUBLE PRECISION Tol_area_split                  ! Criteria of the area split in evaporator
    INTEGER max_iter                                 ! Maximal nomber of iterations per call of cell evaluation
    DOUBLE PRECISION Tolerance_Par(10)               ! Vector with tolerances

! Calculation
    INTEGER ii, jj                                   ! Counters
    INTEGER nodes, nodes_old, test_counter           ! Actual Node, nodes of the iteration before
    INTEGER, PARAMETER :: max_nodes = 1030           ! Max number of nodes per HX
    DOUBLE PRECISION Pressure(2)                     ! [Pa] Pressure(1) = High; Pressure(2) = Low
    DOUBLE PRECISION T_hot(max_nodes +1), T_cold(max_nodes +1)      ! Temperatur vectors hot and cold
side
    DOUBLE PRECISION cp_hot(max_nodes +1), cp_cold(max_nodes +1)    ! Heat capacity vectors hot and
cold side
    DOUBLE PRECISION Enth_hot(max_nodes +1), Enth_cold(max_nodes +1)  ! Enthalpy vectors hot and cold
side
    DOUBLE PRECISION T_hot_in, T_cold_in, T_OH, T_hot_out, T_cold_out  ! Fluid data temperatures
    DOUBLE PRECISION m_hot, m_cold                   ! Fluid data mass flow
    DOUBLE PRECISION HTA, HTC, HTC_Boil, UA, UA_divider(max_nodes), NC  ! HX data
    DOUBLE PRECISION Phi(max_nodes), Phi_old(max_nodes), Phi_Tot, Q_Trans  ! [kJ/hr] Heat rate (eps * delta
T * cp_av * mflow), Q_Trans: heat rates from heat transfer (LMTD)
    DOUBLE PRECISION Q_dev, Q_dev_tot                ! [kJ/hr] Qdev: heat rates from enthalpy
difference * mflow
    DOUBLE PRECISION T_spread_ratio(max_nodes +1), cp_ratio(max_nodes)  ! [-] Temperature spread ratio
inlet/outlet of node, ratio of CO2 heat capacity (in/out per node)
    DOUBLE PRECISION freq_red                        ! [-] Frequency reduction factor for compressor
    DOUBLE PRECISION Cp_cold_HTF                     ! [kJ/kg.K] Heat capacity of the evaporator HTF.
If cp = 100, then water cp is calculated internally
    LOGICAL Node_split_needed(max_nodes)             ! [-] If criterias not fullfilled, split node

    Logical Done, node_to_big                        ! Flags
    Logical Bilance_node, div_cp_small, div_T_spread_small  ! Flags Boundary conditions
    LOGICAL CoolProp_Used(max_nodes)                 ! Store Flag of CoolProp
    LOGICAL Call_CoolProp                            ! Used in case of under temperature behind evaporator

!-----------------------------------------------------------------------------------------------------------
! *** Explanations to the state variables ***
!-----------------------------------------------------------------------------------------------------------
!
! +++ List1: State position in the circle of CO2 property --> numbering
```

```
!-----------------------------------------------------------------------
   St_a_Evap   = 1 ! State of CO2 behind Evaporator and before the internal heat exchanger IHX (low pressure)
   St_a_Comp   = 3 ! State of CO2 behind the compressor and before hot gas tube and oil separator
   St_i_Evap   = 10! State of CO2 in the evaporator (phase change)

! +++ List2: Characterisitc of CO2 property numbering
!-----------------------------------------------
   Temp        = 1 ! CO2 Temperature [°C]
   Press       = 2 ! CO2 Pressure [Pa]
   mflow       = 4 ! CO2 Mass flow [kg/h]

! +++ List3: Device numbering (Components are called devices to ommit danger of confusion with compressor)
(DeviceProp(Device 1-8,yyy))
!-----------------------------------------------------------------------------------------------------------------------------
   IHX         = 1 ! IHX, internal heat exchanger
   Comp        = 2 ! Compressor
   GC1         = 4 ! 1st gas cooler
   GC2         = 5 ! 2nd gas cooler
   GC3         = 6 ! 3rd gas cooler
   Evap        = 8 ! Evaporator

! +++ List4: Meaning of device property numbering (DeviceProp(xxx,property 1-10))
!-------------------------------------------------------------------------
   T_prim_in   = 1 ! [°C] Heat exchanger inlet temperature hot side
   T_sec_in    = 2 ! [°C] Heat exchanger inlet temperature cold side
   T_prim_out  = 3 ! [°C] Heat exchanger outlet temperature hot side
   T_sec_out   = 4 ! [°C] Heat exchanger outlet temperature cold side
   m_prim      = 5 ! [kg/h] Heat exchanger mass flow hot side
   m_sec       = 6 ! [kg/h] Heat exchanger mass flow cold side                Compressor: [Pa] Set Pressure
   A_Exc       = 7 ! [m2] Heat transfer area of whole heat exchanger          Compressor: [-] Mechanical efficiency
   H_Flux      = 8 ! [kJ/h] Heat flux                                         Oil separator: heat loss
   TC_HX       = 9 ! [kJ/(h*K*m2)] Specific heat transfer coefficient         Compressor: [1/s] Set frequency
   TCb_HX      = 10! [kJ/(h*K*m2)] Specific heat transfer coefficient         Compressor: [-] Frequency reduction
   Div_1       = 11! Evaporator: [°C] Overheating CO2                         Compressor: [kW] Electrical power
   Div_2       = 12! Evaporator: [°C] Freeze save Temp
   Div_3       = 13! Evaporator: [kJ/kg K] HTF heat capacity

! Transfer some variables to short names
!---------------------------------------
   T_hot_in    = DeviceProp(Device, T_prim_in)
   T_cold_in   = DeviceProp(Device, T_sec_in)
   m_hot       = DeviceProp(Device, m_prim)
   m_cold      = DeviceProp(Device, m_sec)
   HTA         = DeviceProp(Device, A_Exc)
   HTC         = DeviceProp(Device, TC_HX)
   HTC_Boil    = DeviceProp(Device, TCb_HX)
   Pressure(1) = CO2prop(St_a_Comp, Press)

! Tolerance parameters
!---------------------
   Tol_Q_Deviat = Tolerance_Par(1)              ! Tolerance power criteria eps-NTU calculation
   Tol_cp_ratio = Tolerance_Par(2)              ! safety criteria if cp difference becomes to high --> Warning
   Tol_dT_ratio = Tolerance_Par(3)              ! Criteria to reduce size of nodes.
   Tol_Phi_iter = Tolerance_Par(4)              ! Criteria of the different heat flows in the cell modell
   Tol_Q_Def_Tot = Tolerance_Par(5)                    ! Criteria of the difference of heat flows (Phi against Delta
Enthalpy)
   Tol_area_split = Tolerance_Par(6)            ! Criteria of the area split in evaporator
   max_iter = INT(Tolerance_Par(7))             ! Maximal nomber of iterations per call of cell evaluation

! Without mass flow no heat exchange
!-----------------------------------
   IF (m_hot * m_cold .EQ. 0.0) THEN            ! Either or both massflow does not exist
      DeviceProp(Device, T_prim_out) = T_hot_in
```

```
      DeviceProp(Device, T_sec_out) = T_cold_in
      DeviceProp(Device, H_Flux)    = 0.0
   ELSE                                    ! GoTo calculation

! +++ 1st: Calculate a one node HX
!---------------------------------

! Prepare calculation of first node
!----------------------------------
      Bilance_node = .FALSE.                ! Flag if heat balance is reached
      div_cp_small = .FALSE.                ! Flag if heat capacity is small enough
      div_T_spread_small = .FALSE.              ! Flag if temperature difference on both side of the node is small
enough
      node_to_big = .TRUE.                  ! Flag that indicates if node is not small enough
      nodes = 1                             ! First guess of HX is done with one node
      Node_split_needed = .FALSE.               ! Which node must be split?
      UA_divider(1) = 1.0D0
      T_hot(1) = T_cold_in                  ! Outlet temperature hot side, can't be lower
      T_hot(2) = T_hot_in                   ! Inlet temperature hot side (given)
      T_cold(1) = T_cold_in                 ! Inlet temperature cold side (given)
      T_cold(2) = T_hot_in                  ! Outlet temperature cold side, can't be higher

! +++ Branch to the according nodes calculations
!-----------------------------------------------

      IF ((Device .EQ. GC1) .OR. (Device .EQ. GC2) .OR. (Device .EQ. GC3)) THEN             ! This is a gas
cooler

         UA = HTA * HTC
         CALL Calc_GC_nodes(UA, nodes, Pressure, T_hot, T_cold, m_hot, m_cold, max_iter, &      ! Input: [kJ/hr
K], [-], [Pa], [°C], [°C], [kg/hr], [kg/hr], [-]
                            LU_data_file, CurrentUnit, CurrentType, Tol_Phi_iter, UA_divider, & ! Input: [-], [-], [-], [-], [-]
                            Enth_hot, cp_ratio, T_spread_ratio, Phi, jj)                    ! Output: [kJ/kg], [-], [-], [kJ/hr], [-]

! +++ 2nd: Check if heat exchanger calculation is already ok
!-----------------------------------------------------------
         Q_dev = m_hot * ABS(Enth_hot(2) - Enth_hot(1)) - ABS(Phi(1))                  ! Difference of heat with
enthalpy difference of CO2 and heat flux with averaged cp's
         Bilance_node = ABS(Q_dev) .LT. (Tol_Q_Deviat * ABS(Phi(1)))                   ! Precison per node
reached? (Ideally: Q_dev = 0.0)
         div_cp_small = cp_ratio(1) .LT. Tol_cp_ratio                          ! cp ratio (start / end) of node small
enough?
         div_T_spread_small = T_spread_ratio(1) .LT. Tol_dT_ratio                      ! Temperature spread ratio
inlet/outlet of node small enough?
         IF (Bilance_node .AND. div_cp_small .AND. div_T_spread_small) THEN            ! Boundary conditions
and precison reached?
            DeviceProp(Device, T_prim_out) = T_hot(1)
            DeviceProp(Device, T_sec_out) = T_cold(2)
            DeviceProp(Device, H_Flux) = m_hot * ABS(Enth_hot(2) - Enth_hot(1))        !

! +++ 3rd: Calculate the HX with more nodes
!------------------------------------------
         ELSE                                                  ! assume number of nodes needed
            Node_split_needed(1) = .TRUE.                          ! Node 1 must be splitted up
            DO WHILE (node_to_big)                                 ! Phi of at least one node not stable
enough
               ii = 1
               DO WHILE (ii .LE. nodes)
                  IF (Node_split_needed(ii)) THEN                  ! Add a node
                     DO jj = nodes + 1, (ii+1), -1
                        T_hot(jj + 1) = T_hot(jj)                  ! Shift all hot temperatures to one higher
node
```

```
                    T_cold(jj + 1) = T_cold(jj)                                  ! Shift all cold temperatures to one higher
node
                    Node_split_needed(jj + 1) = Node_split_needed(jj)            ! Shift "Node_split_needed"
request to the next  node
                    UA_divider(jj + 1) = UA_divider(jj)                          ! Shift the UA divider
                  EndDo
                  T_hot(ii+1) = (T_hot(ii) + T_hot(ii+2)) / 2.0D0                ! Add average hot temperature
                  T_cold(ii+1) = (T_cold(ii) + T_cold(ii+2)) / 2.0D0             ! Add average cold temperature
                  UA_divider(ii) = UA_divider(ii) * 2.0D0                        ! New UA divider for the old node
                  UA_divider(ii+1) = UA_divider(ii)                              ! New UA divider for the new node
                  nodes = nodes + 1                                              ! There is an additional node
                  ii = ii+1                                                      ! There is an additional node
                EndIf
                ii = ii+1                                                        ! got to the next node
              EndDo

              Phi_Tot = 0.0D0
              CALL Calc_GC_nodes(UA, nodes, Pressure, T_hot, T_cold, m_hot, m_cold, max_iter, &      ! Input:
[kJ/hr K], [-], [Pa], [°C], [°C], [kg/hr], [kg/hr], [-]
                                 LU_data_file, CurrentUnit, CurrentType, Tol_Phi_iter, UA_divider, & ! Input: [-], [-], [-], [-],
[-]
                                 Enth_hot, cp_ratio, T_spread_ratio, Phi, jj)                       ! Output: [kJ/kg], [-], [-], [kJ/hr],
[-]
              test_counter = 0
              DO ii = 1, nodes
                Phi_Tot = Phi_Tot + Phi(ii)                                      ! Sum of the heat transfer per node
                Q_dev = m_hot * ABS(Enth_hot(ii + 1) - Enth_hot(ii)) - ABS(Phi(ii))        ! Difference of heat with
enthalpy difference of CO2 and flux with average cp
                Bilance_node = ABS(Q_dev) .LT. (Tol_Q_Deviat * ABS(Phi(ii)))               ! Precison per node
reached? (Ideally: Q_dev = 0.0)
                div_cp_small = cp_ratio(ii) .LT. Tol_cp_ratio                    ! Check cp ratio (ideal would be 1.0)
                div_T_spread_small = T_spread_ratio(ii) .LT. Tol_dT_ratio        ! Temperature spread ratio
inlet/outlet of node small enough?
                IF (Bilance_node .AND. div_cp_small .AND. div_T_spread_small) THEN          ! Calculation has
converged
                  test_counter = test_counter + 1                                ! Nodes are small enough
                  Node_split_needed(ii) = .FALSE.
                ELSE                                                             ! Nodes are not small enough
                  Node_split_needed(ii) = .TRUE.
                EndIf
              EndDo
              Q_dev_tot = m_hot * ABS(Enth_hot(nodes + 1) - Enth_hot(1)) - ABS(Phi_Tot)      ! Difference of heat
with enthalpy difference of CO2 and flux with average cp
              node_to_big = (nodes .GT. test_counter) .AND. (nodes .LT. INT(max_nodes/2)) &
              .AND. (ABS(Q_dev_tot / Phi_Tot) .GT. Tol_Q_Def_Tot)
              IF (NOT(node_to_big)) THEN                                         ! Calculation has converged
                DeviceProp(Device, T_prim_out) = T_hot(1)
                DeviceProp(Device, T_sec_out) = T_cold(nodes+1)
                DeviceProp(Device, H_Flux)    = m_hot * ABS(Enth_hot(nodes + 1) - Enth_hot(1))
              ENDIF
            EndDo   !node_to_big
          EndIf      !Multi nodes

        ELSE IF (Device .EQ. IHX) THEN                                           ! This is an inter cooler
          Pressure(2) = CO2prop(St_i_Evap, Press)                                ! Low pressure for secondary CO2
property calculation
          UA = HTA * HTC
          CALL Calc_IHX_nodes(UA, nodes, Pressure, T_hot, T_cold, m_hot, m_cold, max_iter, &     ! Input: [kJ/hr
K], [-], [Pa], [°C], [°C], [kg/hr], [kg/hr], [-]
                              LU_data_file, CurrentUnit, CurrentType, Tol_Phi_iter, UA_divider, & ! Input: [-], [-], [-], [-], [-]
                              Enth_hot, cp_ratio, T_spread_ratio, Phi, jj, CoolProp_Used)       ! Output: [kJ/kg], [-], [-], [kJ/hr],
[-] , [-]
```

```
! +++ 2nd: Check if heat exchanger calculation is already ok
!-------------------------------------------------------------
        Q_dev = m_hot * ABS(Enth_hot(2) - Enth_hot(1)) - ABS(Phi(1))                    ! Difference of heat with
enthalpy difference of CO2 and heat flux with averaged cp's
        Bilance_node = ABS(Q_dev) .LT. (Tol_Q_Deviat * ABS(Phi(1)))                     ! Precison per node
reached? (Ideally: Q_dev = 0.0)
        div_cp_small = cp_ratio(1) .LT. Tol_cp_ratio                                ! cp ratio (start / end) of node small
enough?
        div_T_spread_small = T_spread_ratio(1) .LT. Tol_dT_ratio                        ! Temperature spread ratio
inlet/outlet of node small enough?
        IF (Bilance_node .AND. div_cp_small .AND. div_T_spread_small) THEN              ! Boundary conditions
and precison reached?
          DeviceProp(Device, T_prim_out) = T_hot(1)
          DeviceProp(Device, T_sec_out) = T_cold(2)
          DeviceProp(Device, H_Flux) = m_hot * ABS(Enth_hot(2) - Enth_hot(1))           !

! +++ 3rd: Calculate the HX with more nodes
!-----------------------------------------
        ELSE                                                                ! Smaller nodes needed
          Node_split_needed(1) = .TRUE.                                     ! Node 1 must be splitted up
          DO WHILE (node_to_big)                                            ! Phi of at least one node not stable
enough
            ii = 1
            DO WHILE (ii .LE. nodes)
              IF (Node_split_needed(ii)) THEN                               ! Add a node
                DO jj = nodes + 1, (ii+1), -1
                  T_hot(jj + 1) = T_hot(jj)                                 ! Shift all hot temperatures to one higher
node
                  T_cold(jj + 1) = T_cold(jj)                               ! Shift all cold temperatures to one higher
node
                  Node_split_needed(jj + 1) = Node_split_needed(jj)         ! Shift "Node_split_needed"
request to the next node
                  UA_divider(jj + 1) = UA_divider(jj)                       ! Shift the UA divider
                EndDo
                T_hot(ii+1) = (T_hot(ii) + T_hot(ii+2)) / 2.0D0             ! Add average hot temperature
                T_cold(ii+1) = (T_cold(ii) + T_cold(ii+2)) / 2.0D0          ! Add average cold temperature
                UA_divider(ii) = UA_divider(ii) * 2.0D0                     ! New UA divider for the old node
                UA_divider(ii+1) = UA_divider(ii)                           ! New UA divider for the new node
                nodes = nodes + 1                                           ! There is an additional node
                ii = ii+1                                                   ! There is an additional node
              EndIf
              ii = ii+1                                                     ! got to the next node
            EndDo

            Phi_Tot = 0.0D0
            CALL Calc_IHX_nodes(UA, nodes, Pressure, T_hot, T_cold, m_hot, m_cold, max_iter, &      ! Input:
[kJ/hr K], [-], [Pa], [°C], [°C], [kg/hr], [kg/hr], [-]
                                LU_data_file, CurrentUnit, CurrentType, Tol_Phi_iter, UA_divider, & ! Input: [-], [-], [-], [-],
[-]
                                Enth_hot, cp_ratio, T_spread_ratio, Phi, jj, CoolProp_Used)       ! Output: [kJ/kg], [-], [-],
[kJ/hr], [-] , [-]
            test_counter = 0
            DO ii = 1, nodes
              Phi_Tot = Phi_Tot + Phi(ii)                                   ! Sum of the heat transfer per node
              IF (CoolProp_Used(ii)) THEN                                   ! Enthalpy check might be useless
                Bilance_node = .TRUE.
              ELSE
                Q_dev = m_hot * ABS(Enth_hot(ii + 1) - Enth_hot(ii)) - ABS(Phi(ii))     ! Difference of heat with
enthalpy difference of CO2 and flux with average cp
                Bilance_node = ABS(Q_dev) .LT. (Tol_Q_Deviat * ABS(Phi(ii)))            ! Precison per node
reached? (Ideally: Q_dev = 0.0)
              ENDIF
              div_cp_small = cp_ratio(ii) .LT. Tol_cp_ratio                 ! Check cp ratio (ideal would be 1.0)
```

```
                div_T_spread_small = T_spread_ratio(ii) .LT. Tol_dT_ratio                ! Temperature spread ratio
inlet/outlet of node small enough?
                IF (Bilance_node .AND. div_cp_small .AND. div_T_spread_small) THEN          ! Calculation has
converged
                  test_counter = test_counter + 1                                  ! Nodes are small enough
                  Node_split_needed(ii) = .FALSE.
                ELSE                                                          ! Nodes are not small enough
                  Node_split_needed(ii) = .TRUE.
                EndIf
              EndDo
              Q_dev_tot = m_hot * ABS(Enth_hot(nodes + 1) - Enth_hot(1)) - ABS(Phi_Tot)      ! Difference of heat
with enthalpy difference of CO2 and flux with average cp
              node_to_big = (nodes .GT. test_counter) .AND. (nodes .LT. INT(max_nodes/2)) &
              .AND. (ABS(Q_dev_tot / Phi_Tot) .GT. Tol_Q_Def_Tot)
              IF (NOT(node_to_big)) THEN                                          ! Calculation has converged
                DeviceProp(Device, T_prim_out) = T_hot(1)
                DeviceProp(Device, T_sec_out) = T_cold(nodes+1)
                DeviceProp(Device, H_Flux) = m_hot * ABS(Enth_hot(nodes + 1) - Enth_hot(1))
              ENDIF
            EndDo   !node_to_big
          EndIf      !Multi nodes

        ELSE IF (Device .EQ. Evap) THEN                                       ! Calculate Evaporator
          T_hot(2:3) = T_hot_in                                          ! Inlet temperature hot side (given)
          T_cold(max_nodes) = DeviceProp(Evap, Div_1)                                  ! [°C] Overheating of CO2 to
prevent droplets
          T_hot(max_nodes) = DeviceProp(Evap, Div_2)                                   ! [°C] Lowest temperature to
prevent freezing in Evaporator
          Cp_cold_HTF = DeviceProp(Evap, Div_3)                                ! Heat capacity of the evaporator
HTF. If cp = 100, then water cp is calculated

          CALL CO2_h_cp_interpol(LU_data_file, DeviceProp(IHX, T_prim_out), Pressure(1), &
               Enth_cold(1), cp_hot(1), Call_CoolProp, CurrentUnit, CurrentType)          ! Read enthalpy of CO2
behind IHX (before throttle valve)

          CALL Calc_Evap(HTA, HTC, HTC_Boil, Pressure, T_hot, T_cold, m_hot, m_cold, &          ! Input: [m2],
[kJ/(h*K*m2)], [kJ/(h*K*m2)], [Pa], [°C], [°C], [kg/hr], [kg/hr]
               CurrentUnit, CurrentType, Tol_area_split, max_iter, Cp_cold_HTF, &          ! Input: [-], [-], [-], [-],
[kJ/kg.K]
               Enth_cold, freq_red, jj, Q_Trans)                              ! In/Output: [kJ/kg], [-], [-], [kJ/hr]

          DeviceProp(Device, T_prim_out) = T_hot(1)                                ! [°C] Water outlet temperature
          DeviceProp(Device, T_sec_out) = T_cold(3)                                ! [°C] CO2 outlet temperature
          DeviceProp(Device, H_Flux) = Q_Trans                                    ! [kJ/h] Heat Flux
          DeviceProp(Comp, TCb_HX) = freq_red                                      ! [-] frequency reduction if smaller
than 1.0
          CO2prop(St_i_Evap, Temp) = T_cold(2)                                 ! [°C] CO2 Phase change temperatur
          CO2prop(St_i_Evap, Press) = Pressure(2)                                  ! [Pa] CO2 Pressure

        ELSE
          Call Messages(-1,'Wrong Device to calculate a heat exchange', 'Fatal', CurrentUnit,CurrentType)
          Return
        EndIf         !Branch to a HX

      EndIf           !HX Calculation

END SUBROUTINE


SUBROUTINE Calc_GC_nodes(UA, node, Pressure, T_hot, T_cold, m_hot, m_cold, max_iter, &          ! Input:
[kJ/hr K], [-], [Pa], [°C], [°C], [kg/hr], [kg/hr], [-]
               LU_data_file, CurrentUnit, CurrentType, Tol_Phi_iter, UA_divider, &       ! Input: [-], [-], [-], [-], [-]
               Enth_hot, cp_ratio, T_spread_ratio, Phi, jj)                    ! Output: [kJ/kg], [-], [-], [kJ/hr], [-]
```

```
! Calculate the heat transfer of the gas cooler; Heat flux from hot CO2 gas to colder H2O
!
! Heat transfer calculation with eps-NTU
!-----------------------------------------------------------------------------------

   Implicit None

   INTEGER CurrentUnit, CurrentType
   INTEGER LU_data_file(8)                                    ! File handle number from data files (1 - 3 from
compressor model and 4 - 8 from CO2 data)
   INTEGER, PARAMETER :: max_nodes = 1030                     ! Max number of nodes per HX
   INTEGER node                                               ! Actual Node
   INTEGER ii, jj, max_iter                                   ! Counter
   DOUBLE PRECISION Heat_Cap_H2O                              ! Heat capacity H2O
   DOUBLE PRECISION m_hot, m_cold, Pressure(2)                ! Fluid data
   DOUBLE PRECISION T_hot(max_nodes +1), T_cold(max_nodes +1)          ! Temperatur vectors hot and
cold side
   DOUBLE PRECISION cp_hot(max_nodes +1), cp_cold(max_nodes +1)        ! Heat capacity vectors hot and
cold side
   DOUBLE PRECISION cp_h_av(max_nodes), cp_c_av(max_nodes)             ! Average cp per node
   DOUBLE PRECISION C_hot(max_nodes), C_cold(max_nodes)               ! [kJ/(hr K)] Heat capacity rates
   DOUBLE PRECISION C_min(max_nodes), C_max(max_nodes)                ! [kJ/(hr K)] Heat capacity rates
   DOUBLE PRECISION Cp_r_hot(max_nodes), Cp_r_cold(max_nodes)         ! [-] Heat capacity ratios
   DOUBLE PRECISION C_ratio(max_nodes), Cp_ratio(max_nodes)           ! [-] Ratios of heat capacity rates
and heat capacities
   DOUBLE PRECISION Enth_hot(max_nodes +1)                    ! Enthalpy vector hot side
   DOUBLE PRECISION UA, UA_divider(max_nodes), NTU, ExpNTU, eps       ! HX parameters
   DOUBLE PRECISION Phi(max_nodes), Phi_cold(max_nodes), Phi_hot(max_nodes)    ! [kJ/hr] Heat flux
   DOUBLE PRECISION Test_Phi, Tol_Phi_iter                    ! Precision testing
   DOUBLE PRECISION T_spread_node(max_nodes +1), T_spread_ratio(max_nodes +1)  ! Temperature spread
& ratio inlet/outlet of node
   DOUBLE PRECISION, PARAMETER :: Min_Delta_T_InOut = 1.0D-2          ! Small limit of delta temperature
between hot and cold side (inlet & outlet node)
   Logical done                                               ! Precision of iteration is reached
   Logical First_iter                                         ! First iteration per call, T_hot(1) already set
   LOGICAL Call_CoolProp                                      ! Used in case of under temperature behind evaporator

! Initialize some values to prevent undefined states
!---------------------------------------------------
   done = .FALSE.
   Phi_cold(1:node) = 0.1D-5
   Phi_hot(1:node) = 0.1D-5
   jj = 1                                                     ! iteration counter

! Calculate cp of given flows with starting temperatures
!-------------------------------------------------------
   DO ii = 1, node + 1
      CALL CO2_h_cp_interpol(LU_data_file, T_hot(ii), Pressure(1), Enth_hot(ii), cp_hot(ii), &
           Call_CoolProp, CurrentUnit, CurrentType)              ! Read cp of CO2
      cp_cold(ii) = Heat_Cap_H2O(T_cold(ii), CurrentUnit, CurrentType)          ! cp of H2O
   EndDo

! Iterate the calculation until difference of heat flow phi hot and cold is small
!--------------------------------------------------------------------------------

   DO While (NOT(done))

! Start with hot side and last node
!----------------------------------

! Calculates the heat capacity rates
```

```
!-----------------------------------
      DO ii = node, 1, -1
         cp_h_av(ii) = (cp_hot(ii) + cp_hot(ii+1)) * 5.0D-1                    ! Average cp of CO2 in node
         C_hot(ii) = cp_h_av(ii) * m_hot                                      ! Heat capacity rate CO2
         cp_cold(ii+1) = Heat_Cap_H2O(T_cold(ii+1), CurrentUnit, CurrentType)          ! cp of H2O outlet node
         cp_cold(ii) = Heat_Cap_H2O(T_cold(ii), CurrentUnit, CurrentType)              ! cp of H2O inlet node
         cp_c_av(ii) = (cp_cold(ii) + cp_cold(ii+1)) * 5.0D-1                 ! Average cp of H20 in node
         C_cold(ii) = cp_c_av(ii) * m_cold                                    ! Heat capacity rate H2O
         C_min(ii) = MIN(C_hot(ii), C_cold(ii))                               ! Minimal heat capacity rate
         C_max(ii) = MAX(C_hot(ii), C_cold(ii))                               ! Maximal heat capacity rate
         C_ratio(ii) = C_min(ii) / C_max(ii)                                  ! Ratio of heat capacity rates

! Heat transfer calculation with eps-NTU
!---------------------------------------
         NTU = UA / UA_divider(ii) / C_min(ii)                                ! Number of transfer units
         ExpNTU = EXP(-NTU * (1.0D0 - C_ratio(ii)))
         eps = (1.0D0 - ExpNTU) / (1.0D0 - C_ratio(ii) * ExpNTU)              ! Calculated effectiveness
         Phi(ii) = eps * C_min(ii) * (T_hot(ii + 1) - T_cold(ii))             ! Resulting heat transfer
         Phi_hot(ii) = Phi(ii)
         T_hot(ii) = MAX(T_cold(ii), T_hot(ii+1) - Phi(ii) / (cp_h_av(ii) * m_hot))   ! Outlet temperature node hot
side (CO2)
         CALL CO2_h_cp_interpol(LU_data_file, T_hot(ii), Pressure(1), Enth_hot(ii), &
              cp_hot(ii), Call_CoolProp, CurrentUnit, CurrentType)            ! Read cp of CO2 outlet node
      EndDo

! Continue with cold side and first node
!---------------------------------------

! Calculates the heat capacity rates
!-----------------------------------
      DO ii = 1, node
         cp_h_av(ii) = (cp_hot(ii) + cp_hot(ii+1)) * 5.0D-1                    ! Average cp of CO2 in node
         C_hot(ii) = cp_h_av(ii) * m_hot                                      ! Heat capacity rate CO2
         cp_c_av(ii) = (cp_cold(ii) + cp_cold(ii+1)) * 5.0D-1                 ! Average cp of H20 in node
         C_cold(ii) = cp_c_av(ii) * m_cold                                    ! Heat capacity rate H2O
         C_min(ii) = MIN(C_hot(ii), C_cold(ii))                               ! Minimal heat capacity rate
         C_max(ii) = MAX(C_hot(ii), C_cold(ii))                               ! Maximal heat capacity rate
         C_ratio(ii) = C_min(ii) / C_max(ii)                                  ! Ratio of heat capacity rates

! Heat transfer calculation with eps-NTU
!---------------------------------------
         NTU = UA / UA_divider(ii) / C_min(ii)                                ! Number of transfer units
         ExpNTU = EXP(-NTU * (1.0D0 - C_ratio(ii)))
         eps = (1.0D0 - ExpNTU) / (1.0D0 - C_ratio(ii) * ExpNTU)              ! Calculated effectiveness
         Phi(ii) = eps * C_min(ii) * (T_hot(ii + 1) - T_cold(ii))             ! Resulting heat transfer
         Phi_cold(ii) = Phi(ii)
         T_cold(ii+1) = T_cold(ii) + Phi(ii) / (cp_c_av(ii) * m_cold)         ! Outlet temperature node cold side
(CO2)
         cp_cold(ii+1) = Heat_Cap_H2O(T_cold(ii+1), CurrentUnit, CurrentType)          ! cp of H2O outlet node
      EndDo

! Precision of iteration reached? Test if phi cold and phi hot have a small differences
!---------------------------------------------------------------------------------------
      jj = jj + 1
      Test_Phi = 0.0D0                                                        ! Test_Phi adds correlation of phi hot against
phi cold per node
      DO ii = 1, node
         IF ((Phi_hot(ii) .EQ. 0.0) .OR. (Phi_cold(ii) .EQ. 0.0)) THEN        ! Prevent Divison by zero
            Test_Phi = Test_Phi + 1.0
         ELSE
            Test_Phi = Test_Phi + MAX(Phi_cold(ii) / Phi_hot(ii), Phi_hot(ii) / Phi_cold(ii))
         ENDIF
      EndDo
```

```
      IF((Test_Phi - DFLOAT(node) * 1.0D0) .LT. Tol_Phi_iter) THEN                    ! Check against set value
         done = .TRUE.
      EndIf
      IF (jj .GT. max_iter) THEN                                                      ! It might happen that there is not yet a result
possible in the given tolerance
         done = .TRUE.
      EndIf
   EndDo

! Calculate key values to test linearity conditions (tested in the calling subroutine)
!-------------------------------------------------------------------------------
   T_spread_node(1) = MAX(T_hot(1) - T_cold(1), Min_Delta_T_InOut)                    ! Temperature difference
between cold_in and hot_out of first node
   DO ii = 1, node
      T_spread_node(ii+1) = MAX(T_hot(ii+1) - T_cold(ii+1), Min_Delta_T_InOut)        ! Temperature difference
between cold_in and hot_out of next node
      T_spread_ratio(ii) = MAX((T_spread_node(ii) / T_spread_node(ii+1)), &
                  (T_spread_node(ii+1) / T_spread_node(ii)))                          ! Ratio of temperature difference from
left and right side of node (to test temperature linearity)
      Cp_r_hot(ii) = MAX((cp_hot(ii) / cp_hot(ii+1)), (cp_hot(ii+1) / cp_hot(ii)))    ! Heat capacity ratio hot side
      Cp_r_cold(ii) = MAX((cp_cold(ii) / cp_cold(ii+1)), (cp_cold(ii+1) / cp_cold(ii)))  ! Heat capacity ratio cold side
      Cp_ratio(ii) = MAX(Cp_r_hot(ii), Cp_r_cold(ii))                                 ! Max heat capacity ratio (to test cp
linearity)
   EndDo

END SUBROUTINE


SUBROUTINE Calc_IHX_nodes(UA, node, Pressure, T_hot, T_cold, m_hot, m_cold, max_iter, &     ! Input: [kJ/hr
K], [-], [Pa], [°C], [°C], [kg/hr], [kg/hr], [-]
                  LU_data_file, CurrentUnit, CurrentType, Tol_Phi_iter, UA_divider, &   ! Input: [-], [-], [-], [-], [-]
                  Enth_hot, cp_ratio, T_spread_ratio, Phi, jj, CoolProp_Used)          ! Output: [kJ/kg], [-], [-], [kJ/hr],
[-] , [-]

! Calculate the heat transfer of the IHX; Heat flux from hot high pressure CO2 gas to colder low pressure CO2 gas
!
! Heat transfer calculation with eps-NTU
!-----------------------------------------------------------------------------------------------------------

   Implicit None

   INTEGER CurrentUnit, CurrentType
   INTEGER LU_data_file(8)                                                            ! File handle number from data files (1 - 3 from
compressor model and 4 - 8 from CO2 data)
   INTEGER, PARAMETER :: max_nodes = 1030                                             ! Max number of nodes per HX
   INTEGER node                                                                       ! Actual Node
   INTEGER ii, jj, max_iter                                                           ! Counter
   DOUBLE PRECISION Heat_Cap_CO2                                                      ! Heat capacity CO2
   DOUBLE PRECISION m_hot, m_cold, Pressure(2)                                        ! Fluid data
   DOUBLE PRECISION T_hot(max_nodes +1), T_cold(max_nodes +1)                         ! Temperatur vectors hot and
cold side
   DOUBLE PRECISION cp_hot(max_nodes +1), cp_cold(max_nodes +1)                       ! Heat capacity vectors hot and
cold side
   DOUBLE PRECISION cp_h_av(max_nodes), cp_c_av(max_nodes)                            ! Average cp per node
   DOUBLE PRECISION C_hot(max_nodes), C_cold(max_nodes)                               ! [kJ/(hr K)] Heat capacity rates
   DOUBLE PRECISION C_min(max_nodes), C_max(max_nodes)                                ! [kJ/(hr K)] Heat capacity rates
   DOUBLE PRECISION Cp_r_hot(max_nodes), Cp_r_cold(max_nodes)                         ! [-] Heat capacity ratios
   DOUBLE PRECISION C_ratio(max_nodes), Cp_ratio(max_nodes)                           ! [-] Ratios of heat capacity rates
and heat capacities
   DOUBLE PRECISION Enth_hot(max_nodes +1), Enth_cold(max_nodes +1)                   ! Enthalpy vectors CO2 (hot
and cold)
   DOUBLE PRECISION UA, UA_divider(max_nodes), NTU, ExpNTU, eps                       ! HX parameters
   DOUBLE PRECISION Phi(max_nodes), Phi_cold(max_nodes), Phi_hot(max_nodes)    ! [kJ/hr] Heat flux
```

```
    DOUBLE PRECISION Test_Phi, Tol_Phi_iter                              ! Precision testing
    DOUBLE PRECISION T_spread_node(max_nodes +1), T_spread_ratio(max_nodes +1)  ! Temperature spread
& ratio inlet/outlet of node
    DOUBLE PRECISION, PARAMETER :: Min_Delta_T_InOut = 1.0D-2            ! Small limit of delta temperature
between hot and cold side (inlet & outlet node)
    DOUBLE PRECISION Enth_div_hot, Enth_div_cold                         ! Enthalpy differences first node calculation
    Logical done                                                 ! Precision of iteration is reached
    Logical First_iter                                           ! First iteration per call, T_hot(1) already set
    LOGICAL Call_CoolProp                                            ! Used in case of under temperature behind evaporator
    LOGICAL CoolProp_Used(max_nodes)                                     ! Store Flag of CoolProp

! Initialize some values to prevent undefined states
!-------------------------------------------------

    done = .FALSE.
    Phi_cold(1:node) = 0.1D-5
    Phi_hot(1:node) = 0.1D-5
    jj = 1                                                   ! iteration counter

!  Calculate cp and enthalpy of given flows with starting temperatures
!-------------------------------------------------------------------
    DO ii = 1, node + 1
       CALL CO2_h_cp_interpol(LU_data_file, T_hot(ii), Pressure(1), Enth_hot(ii), &
            cp_hot(ii), Call_CoolProp, CurrentUnit, CurrentType)          ! Read cp of CO2 hot side
       CALL CO2_h_cp_interpol(LU_data_file, T_cold(ii), Pressure(2), Enth_cold(ii), &
            cp_cold(ii), Call_CoolProp, CurrentUnit, CurrentType)         ! Read cp of CO2 cold side
    EndDo


! Iterate the calculation until difference of heat flow phi hot and cold is small
!--------------------------------------------------------------------------------

    DO While (NOT(done))

! Start with hot side and last node
!----------------------------------
!
! Calculates the heat capacity rates
!----------------------------------
       DO ii = node, 1, -1
          cp_h_av(ii) = (cp_hot(ii) + cp_hot(ii+1)) * 5.0D-1             ! Average cp of high pressure CO2 in
node
          C_hot(ii) = cp_h_av(ii) * m_hot                                ! Heat capacity rate of high pressure CO2
          CALL CO2_h_cp_interpol(LU_data_file, T_cold(ii+1), Pressure(2), Enth_cold(ii+1), &
               cp_cold(ii+1), Call_CoolProp, CurrentUnit, CurrentType)       ! Read cp of CO2 cold side, outlet
node
          CALL CO2_h_cp_interpol(LU_data_file, T_cold(ii), Pressure(2), Enth_cold(ii), &
               cp_cold(ii), Call_CoolProp, CurrentUnit, CurrentType)         ! Read cp of CO2 hot side, inlet
node
               CoolProp_Used(ii) = Call_CoolProp                         ! If TRUE, we may have a phase change
cp
          cp_c_av(ii) = (cp_cold(ii) + cp_cold(ii+1)) * 5.0D-1               ! Average cp of low pressure CO2 in
node
          C_cold(ii) = cp_c_av(ii) * m_cold                              ! Heat capacity rate of low pressure CO2
          C_min(ii) = MIN(C_hot(ii), C_cold(ii))                          ! Minimal heat capacity rate
          C_max(ii) = MAX(C_hot(ii), C_cold(ii))                          ! Maximal heat capacity rate
          C_ratio(ii) = C_min(ii) / C_max(ii)                            ! Ratio of heat capacity rates

! Heat transfer calculation with eps-NTU
!---------------------------------------
          NTU = UA / UA_divider(ii) / C_min(ii)                          ! Number of transfer units
          ExpNTU = EXP(-NTU * (1.0D0 - C_ratio(ii)))
          eps = (1.0D0 - ExpNTU) / (1.0D0 - C_ratio(ii) * ExpNTU)             ! Calculated effectiveness
```

```
          Phi(ii) = eps * C_min(ii) * (T_hot(ii + 1) - T_cold(ii))                    ! Resulting heat transfer
          Phi_hot(ii) = Phi(ii)
          T_hot(ii) = MAX(T_cold(ii), T_hot(ii+1) - Phi(ii) / (cp_h_av(ii) * m_hot))          ! Outlet temperature node hot
side (CO2)
          CALL CO2_h_cp_interpol(LU_data_file, T_hot(ii), Pressure(1), Enth_hot(ii), &
               cp_hot(ii), Call_CoolProp, CurrentUnit, CurrentType)                  ! Read cp of CO2 outlet node
       EndDo

! Continue with cold side and first node
!---------------------------------------
!
! Calculates the heat capacity rates
!-----------------------------------
       DO ii = 1, node
          cp_h_av(ii) = (cp_hot(ii) + cp_hot(ii+1)) * 5.0D-1                    ! Average cp of high pressure CO2 in
node
          C_hot(ii) = cp_h_av(ii) * m_hot                                      ! Heat capacity rate of high pressure CO2
          cp_c_av(ii) = (cp_cold(ii) + cp_cold(ii+1)) * 5.0D-1                  ! Average cp of low pressure CO2 in
node
          C_cold(ii) = cp_c_av(ii) * m_cold                                    ! Heat capacity rate of low pressure CO2
          C_min(ii) = MIN(C_hot(ii), C_cold(ii))                               ! Minimal heat capacity rate
          C_max(ii) = MAX(C_hot(ii), C_cold(ii))                               ! Maximal heat capacity rate
          C_ratio(ii) = C_min(ii) / C_max(ii)                                  ! Ratio of heat capacity rates

! Heat transfer calculation with eps-NTU
!---------------------------------------

          NTU = UA / UA_divider(ii) / C_min(ii)                                ! Number of transfer units
          ExpNTU = EXP(-NTU * (1.0D0 - C_ratio(ii)))
          eps = (1.0D0 - ExpNTU) / (1.0D0 - C_ratio(ii) * ExpNTU)              ! Calculated effectiveness
          Phi(ii) = eps * C_min(ii) * (T_hot(ii + 1) - T_cold(ii))             ! Resulting heat transfer
          Phi_cold(ii) = Phi(ii)
          T_cold(ii+1) = T_cold(ii) + Phi(ii) / (cp_c_av(ii) * m_cold)         ! Outlet temperature node cold side
(CO2)
          CALL CO2_h_cp_interpol(LU_data_file, T_cold(ii+1), Pressure(2), Enth_cold(ii+1), &
               cp_cold(ii+1), Call_CoolProp, CurrentUnit, CurrentType)         ! Read cp of CO2 cold side, outlet
node
               CoolProp_Used(ii+1) = Call_CoolProp                             ! If TRUE, we may have a phase
change cp
       EndDo

! Precision of iteration reached? Test if phi cold and phi hot have a small differences
!--------------------------------------------------------------------------------------
       jj = jj + 1
       Test_Phi = 0.0D0                                                        ! Test_Phi adds correlation of phi hot against
phi cold per node
       DO ii = 1, node
          IF ((Phi_hot(ii) .EQ. 0.0) .OR. (Phi_cold(ii) .EQ. 0.0)) THEN        ! Prevent Divison by zero
             Test_Phi = Test_Phi + 1.0
          ELSE
             Test_Phi = Test_Phi + MAX(Phi_cold(ii) / Phi_hot(ii), Phi_hot(ii) / Phi_cold(ii))
          ENDIF
       EndDo
       IF((Test_Phi - DFLOAT(node) * 1.0D0) .LT. Tol_Phi_iter) THEN            ! Check against set value
          done = .TRUE.
       EndIf
       IF (jj .GT. max_iter) THEN                                              ! It might happen that there is not yet a result
possible in the given tolerance
          done = .TRUE.
       EndIf
    EndDo

! Calculate key values to test linearity conditions (tested in the calling subroutine)
```

```
!-------------------------------------------------------------------------------

   T_spread_node(1) = MAX(T_hot(1) - T_cold(1), Min_Delta_T_InOut)                    ! Temperature difference
between cold_in and hot_out of first node
   DO ii = 1, node
      T_spread_node(ii+1) = MAX(T_hot(ii+1) - T_cold(ii+1), Min_Delta_T_InOut)        ! Temperature difference
between cold_in and hot_out of next node
      T_spread_ratio(ii) = MAX((T_spread_node(ii) / T_spread_node(ii+1)), &
                   (T_spread_node(ii+1) / T_spread_node(ii)))                        ! Ratio of temperature difference from
left and right side of node (to test temperature linearity)
      Cp_r_hot(ii) = MAX((cp_hot(ii) / cp_hot(ii+1)), (cp_hot(ii+1) / cp_hot(ii)))   ! Heat capacity ratio hot side
      Cp_r_cold(ii) = MAX((cp_cold(ii) / cp_cold(ii+1)), (cp_cold(ii+1) / cp_cold(ii)))  ! Heat capacity ratio cold side
      Cp_ratio(ii) = MAX(Cp_r_hot(ii), Cp_r_cold(ii))                                 ! Max heat capacity ratio (to test cp
linearity)
      IF (CoolProp_Used(ii)) THEN                                                     ! Cold side temperatur might be lower than
phase change temperatur! This test might be not useful
         Cp_ratio(ii) = 1.0
      ENDIF
   EndDo

END SUBROUTINE


SUBROUTINE Calc_Evap(HTA, HTC_OH, HTC_Boil, Pressure, T_hot, T_cold, m_hot, m_cold, &   ! Input: [m2],
[kJ/(h*K*m2)], [kJ/(h*K*m2)], [Pa], [°C], [°C], [kg/hr], [kg/hr]
                     CurrentUnit, CurrentType, Tol_area_split, max_iter, Cp_cold_HTF, &   ! Input: [-], [-], [-], [-], [kJ/kg.K]
                     Enth_cold, freq_red, jj, Q_Trans)                                ! Output: [kJ/kg], [-], [-], [kJ/hr]

! Calculate the heat transfer of the Evaporator; Heat flux from hot water (source) to colder low pressure CO2 gas.
! Water is assumed hot and CO2 cold!!!!
! Heat transfer calculation with LMTD
!-------------------------------------------------------------------------------------------------------------------

   Implicit None

   INTEGER CurrentUnit, CurrentType
   INTEGER, PARAMETER :: max_nodes = 1030                                             ! Max number of nodes per HX
   INTEGER ii, jj, max_iter                                                           ! Counter
   DOUBLE PRECISION Heat_Cap_H2O, Pressure_CO2, Enthalpy_PC_CO2, Enthalpy_CO2         ! Function calls
   DOUBLE PRECISION HTA, HTA_split_OH, HTC_OH, HTC_Boil, Expo                         ! HX parameters
   DOUBLE PRECISION m_hot, m_cold, Pressure(2)                                        ! Fluid data
   DOUBLE PRECISION T_hot(max_nodes +1), T_cold(max_nodes +1)                         ! Temperatur vectors hot and
cold side
   DOUBLE PRECISION cp_hot(max_nodes +1)                                              ! Heat capacity vector
   DOUBLE PRECISION cp_h_av(max_nodes)                                                ! Average cp per node
   DOUBLE PRECISION Enth_cold(max_nodes +1), Enthalpy_uptake_PC, Enthalpy_uptake_OH! Enthalpy cold
side
   DOUBLE PRECISION Q_uptake_PC, Q_uptake_OH, Q_Trans_OH, Q_Trans                     ! [kJ/hr] Heat flux
   DOUBLE PRECISION Tol_area_split, Area_corr                                         ! Area precision testing
   DOUBLE PRECISION LMTD_OH, Delta_T_small, Delta_T_large                             ! [°C] Temperature spread &
ratio inlet/outlet of node
   DOUBLE PRECISION freq_red                                                          ! [-] Factor for frequency reduction of compressor
   DOUBLE PRECISION Cp_cold_HTF                                                       ! [kJ/kg.K] Heat capacity of the evaporator
HTF. If cp = 100, then water cp is calculated internally
   Logical done                                                                       ! Precision of iteration is reached

!  Meaning of parameters
!-----------------------
!   Enth_cold(1) =      Enthalpy behind throttle valve (CO2 evap inlet)
!   Enth_cold(2) =      Enthalpy behind evaporation of CO2
!   Enth_cold(3) =      Enthalpy behind CO2 over heating (CO2 evap outlet)
!   T_hot(max_nodes) =  Minimal allowed water outlet temperature of the evaporator to prevent freezing inside
!   T_cold(max_nodes) = Overhating of CO2 in the evaporator
```

```
!   check if overhating set temperature > 0.0
    if (T_cold(max_nodes) .LE. 0.0) then
       Call Messages(-1,'Overheating set temperature in Evaporator less or equal 0.0°C', &
                     'Fatal', CurrentUnit,CurrentType)
       Return
    end if

! Initialize some values to prevent undefined states
!-----------------------------------------------------
    done = .FALSE.
    jj = 1                                                  ! iteration counter
    HTA_split_OH = 1.0D-1                                   ! Guess value of the HTA split factor for
overheating
    freq_red = 1.0D0                                        ! Frequency reduction parameter for the compressor
speed
    T_hot(1) = T_hot(max_nodes)                             ! Lowest temperature to prevent freezing in
HX
    If (Cp_cold_HTF .EQ. 100.0) THEN                        ! Water is used
       cp_hot(3) = Heat_Cap_H2O(T_hot(3), CurrentUnit, CurrentType)          ! [kJ/(kg*K)] Heat capacity
water inlet
       cp_h_av(1:2) = Heat_Cap_H2O((T_hot(1) + T_hot(2)) / 2.0D0, CurrentUnit, CurrentType)! [kJ/(kg*K)] Average
cp of water in phase change part and overheating part
    ELSE                                                    ! HTF with Glycol is used
       cp_hot(3) = Cp_cold_HTF                              ! [kJ/(kg*K)] Heat capacity HTF inlet
       cp_h_av(1:2) = Cp_cold_HTF                           ! [kJ/(kg*K)] Average cp of HTF in phase
change part and overheating part
    ENDIF



!  Iterate until the HTA area for overheating is defined
!-------------------------------------------------------

    DO While (NOT(done))

!  Calculate the phase change temperature of CO2 with water inlet temperature and lowest allowed water outlet
temperature
!-------------------------------------------------------------------------------------------------------------
       Expo = EXP((HTA * (1.0D0 - HTA_split_OH) * HTC_Boil) / (cp_h_av(1) * m_hot))    ! [-]
       T_cold(2) = (Expo * T_hot(1) - T_hot(2)) / (Expo - 1.0D0)          ! [°C] Phase change temperature
       T_cold(3) = T_cold(2) + T_cold(max_nodes)                          ! [°C] Temperature behind overheating
       Pressure(2) = Pressure_CO2(T_cold(2), 1.0D0, CurrentUnit, CurrentType)          ! [Pa] low pressure of CO2
depending on phase change temperatur

!  Calculate the needed enthalpy to evaporate and overheat CO2 ("PC" and "OH")
!-----------------------------------------------------------------------------
       Enth_cold(2) = Enthalpy_PC_CO2(T_cold(2), 1.0D0, CurrentUnit, CurrentType)      ! [kJ/kg] Enthalpy of CO2
at the end of the phase change
       Enth_cold(3) = Enthalpy_CO2(T_cold(3), Pressure(2), CurrentUnit, CurrentType)   ! [kJ/kg] Enthalpy of CO2
at the end of the evaporator with overheating
       Enthalpy_uptake_PC = ABS(Enth_cold(2) - Enth_cold(1))                ! [kJ/kg] Enthalpy uptake of CO2
through evaporation (1st part in Evaporator)
       Enthalpy_uptake_OH = ABS(Enth_cold(3) - Enth_cold(2))                ! [kJ/kg] Enthalpy uptake of CO2
through overheating (2nd part in Evaporator)
       Q_uptake_PC = Enthalpy_uptake_PC * m_cold                            ! [kJ/h] CO2 heat uptake with phase
change (1st part in Evaporator)
       Q_uptake_OH = Enthalpy_uptake_OH * m_cold                            ! [kJ/h] CO2 heat uptake with
overheating (2nd part in Evaporator)

!  Determine with the CO2 enthalpies the water temperatures 1, 2, and LMTD(2)
!----------------------------------------------------------------------------
```

```
      T_hot(2) = T_hot(3) - (Q_uptake_OH / (m_hot * cp_h_av(2)))                    ! [°C] water temperature behind
overheating (somewhere in the middle of the HX)
      T_hot(1) = MAX(0.1D0, T_hot(2) - (Q_uptake_PC / (m_hot * cp_h_av(1))))        ! [°C] water temperature
behind phase change (water outlet temperature)
      If (Cp_cold_HTF .EQ. 100.0) THEN                                              ! Water is used
        DO ii = 2, 1, -1
          cp_hot(ii) = Heat_Cap_H2O(T_hot(ii), CurrentUnit, CurrentType)        ! [kJ/(kg*K)] Heat capacity water
(between part 1 and 2 and water outlet)
          cp_h_av(ii) = (cp_hot(ii) + cp_hot(ii+1)) / 2.0D0                     ! [kJ/(kg*K)] New average heat capacity
water (in 1st and 2nd part)
        EndDo
      ELSE                                                                          ! HTF with Glycol is used   ! HTF with Glycol is used
        DO ii = 2, 1, -1
          cp_hot(ii) = Cp_cold_HTF                                              ! [kJ/(kg*K)] Heat capacity water (between part 1
and 2 and water outlet)
          cp_h_av(ii) = Cp_cold_HTF                                             ! [kJ/(kg*K)] New average heat capacity water
(in 1st and 2nd part)
        EndDo
      ENDIF
      Delta_T_small = Min(ABS(T_hot(2) - T_cold(2)), ABS(T_hot(3) - T_cold(3)))     ! [°C] Small temperature
differenz between water and CO2 (to calculate LMTD)
      Delta_T_large = Max(ABS(T_hot(2) - T_cold(2)), ABS(T_hot(3) - T_cold(3)))     ! [°C] Large temperature
differenz between water and CO2 (to calculate LMTD)
      IF ((Delta_T_large .EQ. Delta_T_small) .OR. (Delta_T_small .EQ. 0.0)) THEN
        LMTD_OH = 0.0
      Else
        LMTD_OH = (Delta_T_large - Delta_T_small) / LOG (Delta_T_large / Delta_T_small)
      EndIf
      Q_Trans_OH = LMTD_OH * (HTA * HTA_split_OH * HTC_OH)                          ! Heat transfer for overheating
with LMTD and guess value for HTA (heat transfer area)
      Area_corr = Q_Trans_OH / Q_uptake_OH                                          ! Correction factor for HTA split (between
phase change and overheating)
      HTA_split_OH = Max(1.0D-5, HTA_split_OH / Area_corr)                          ! Calculate the new HTA split factor
for overheating
      jj = jj + 1
      IF (ABS(Area_corr - 1.0D0) .LT. Tol_area_split) THEN                          ! Is the area split in the given tolerance?
        done = .TRUE.
      EndIf
      IF (jj .GT. max_iter) THEN                                                    ! It might happen that there is no result possible
in the given tolerance
        done = .TRUE.
      EndIf
    EndDo

!  Calculate some data to return
!--------------------------------
    Q_Trans = Q_uptake_PC + Q_uptake_OH                                             ! Total heat transfer
    IF (T_hot(1) .LT. T_hot(max_nodes)) THEN                                        ! Water outlet temperature of evaporator
return to low,
      freq_red = MIN(1.0D0, (((T_hot(2) - T_hot(max_nodes)) * (m_hot * cp_h_av(1)) &  ! reduce the frequency of
compressor
      + (T_hot(3) - T_hot(2)) * (m_hot * cp_h_av(2))) / Q_Trans))
    EndIf


END SUBROUTINE
```