EBSILON®*Professional*

Release-Notes (deutsch)

Release 17.0

iqony

Release-Notes

EBSILON®*Professional*

Release 17.0

Inhaltsverzeichnis

1. Rechenkern

1.1. Neue Bauteile

1.1.1. Bauteil 171: Schwungradspeicher

Das Bauteil 171 ermöglicht es, einen Schwungradspeicher zu modellieren. Die Energie wird dabei in mechanischer Form als Rotationsenergie eines Rotors gespeichert. Das Bauteil hat Wellen- sowie Elektroanschlüsse. Der Energieinhalt des Speichers ist proportional zur Drehzahl des Rotors. Eine Beschleunigung bewirkt somit das Beladen und eine Abnahme der Drehzahl das Entladen des Speichers. Mit Hilfe des Bauteils kann auch generell eine Beschleunigung einer rotierenden Masse, z. B. eines Turbinenläufers, bilanziert werden.
Für weitere Details wird auf die Online-Hilfe verwiesen.

1.2. Erweiterungen vorhandener Bauteile

1.2.1. Allgemeines

Für alle Bauteile, die durch Setzen des internen „Nicht-Konvergiert“ Zustands ein Ende der Berechnung ausdrücklich unterbinden, wird nun eine Warnung ausgegeben, falls die Simulation aufgrund des Erreichens der maximalen Iterationszahl abgebrochen wird. Betroffene Bauteile sind i. d. R. Iterationsregler oder Bauteile mit interner Regelung, die zusätzlich eine Regelungstotzeit verwenden. Es kann sich aber auch um benutzerdefinierte Bauteile mit EbsScript-Code oder externer Dll handeln, die den Signal-Not-Converged-Mechanismus verwenden.

1.2.2. Bauteile 12 und 39: Ergebniswerte bei Reglern

Die Ergebniswerte bei den Reglern wurden erweitert, um eine Analyse des Regelungsvorgangs zu erleichtern.

Schon bisher gab es die Ergebniswerte RSCV für den Sollwert, RACT für den erreichten Zielwert und RCORR für den Wert der Stellgröße. Diese wurden beibehalten. Zu beachten ist allerdings, dass RCORR den Wert der Stellgröße angibt, den der Regler im aktuellen Iterationsschritt berechnet hat. Dieser ist nicht identisch mit dem Leitungswert für die Stellgröße, da der Regler den Wert nicht direkt setzt, sondern lediglich einen Gradienten berechnet, der in das Gleichungssystem eingeht. Welchen Wert die Stellgröße dann wirklich annimmt, hängt deshalb nicht allein vom Regler selbst ab, sondern ergibt sich aus der Lösung des Gleichungssystems.

Deshalb wurde ein neuer Ergebniswert CORR ergänzt, der den Wert der Stellgröße angibt, der sich nach der Lösung des Gleichungssystems einstellt, sowie ein Ergebniswert CORR0, der den Wert von CORR im vorhergehenden Iterationsschritt anzeigt, und die Änderung der Korrekturgröße CHCORR (= CORR-CORR0).

Der Ergebniswert DEV (= RACT–RSCV) gibt den Unterschied zwischen Sollwert und Istwert an. CHDEV gibt an, wie stark sich DEV im Vergleich zum vorhergehenden Iterationsschritt geändert hat.

Es folgen dann einige Ergebniswerte, die vom Regler intern verwendet werden, um den Gradienten zwischen Zielwert und Stellgröße zu ermitteln, der in die vom Regler abgesetzte Gleichung eingeht:

REF ist die aus dem Iterationsverlauf von Zielwert und Sollwert ermittelte Referenzgröße, die für die interne Normierung der Abweichung von Soll- und Istwert verwendet wird, RELACT (= CHDEV/REF) die auf diese Referenzgröße bezogene Änderung des Istwertes und RELCORR (= CHCORR/CORR) die relative Änderung der Stellgröße im letzten Iterationsschritt. Das Verhältnis dieser beiden Größen ist RATREL (= RELCORR/RELACT).

Durch den Vorgabewert CHL2 bzw. CHL3 und die Kennlinie CCHL2 bzw. CCHL3 kann der Anwender die relative Änderung der vom Regler berechneten Stellgröße begrenzen. Der Wert dieser Kennlinie wird jetzt als Ergebniswert RCCHL ausgewiesen und das sich insgesamt ergebende Änderungslimit als Ergebniswert RCHL.

Wegen dieser Begrenzung sowie der Dämpfung und der Begrenzung durch die Minimum- und Maximum-Werte für die Stellgröße wird für die Berechnung des Gradienten häufig ein modifizierter Wert von RATREL verwendet, der im Ergebniswert RATRELUSED angezeigt wird.

Durch die folgenden Ergebniswerte wird ersichtlich, wie sich die einzelnen Dämpfungs- und Begrenzungsmechanismen auswirken: CHCORRORIG gibt die ursprünglich berechnete Änderung der Stellgröße an, CHCORRDAMP den Wert nach der Dämpfung und CHCORRCHL den Wert nach Anwendung des Änderungslimits.

GRADMAT weist den Gradienten aus, der in das Gleichungssystem eingeht. Dieser gibt an, wie die Stellgröße geändert werden muss, um den Istwert um eine Einheit zu verändern.

Durch Maßnahmen zur Verbesserung der Konvergenz werden an diversen Stellen allerdings auch Änderungen an den genannten Werten vorgenommen.

Der bisherige Ergebniswert RCHL3 ist identisch mit CHCORRCHL. RCHL3 wurde deshalb als Alias-Name für CHCORRCHL zugelassen, so dass Skripte und Ausdrücke, in denen RCHL3 vorkommt, wie bisher funktionieren.

1.2.3. Bauteile 15 und 16: Wärmeaus- und -einkopplung

Bei diesen Bauteilen wird der Druckverlust in Teillast analog zum Bauteil 13 (Rohrleitung) auf das 10-fache des Nominalwertes begrenzt. Grund dafür sind zu hohe Druckverluste bei zu kleinem Auslegungsmassenstrom. Anders als bei Bauteil 13 ist der Faktor 10 hier fest eingestellt und nicht durch einen Vorgabewert (DPDPNMAX) einstellbar. Wenn diese Begrenzung aktiv wird, empfiehlt es sich ohnehin, einen anderen Auslegungspunkt zu wählen, beispielsweise im aktuellen Unterprofil eine lokale Auslegung durchzuführen.

1.2.4. Bauteile 20 und 70: Druckberechnung bei Trommeln

Bei den Trommel-Bauteilen besteht jetzt die Möglichkeit, in Teillast den Trommeldruck vom Bauteil berechnen zu lassen.

Dadurch können in Ebsilon jetzt linksdrehende Prozesse mit ähnlich guten Konvergenzeigenschaften abgebildet werden wie rechtsdrehende Prozesse.
Bei rechtsdrehenden Prozessen ist es das Bauteil Kondensator, dass die Möglichkeit bietet, direkt den Druck berechnen zu lassen, bei dem die Kondensationsenthalpie genau so groß ist, die freiwerdende Kondensationswärme durch das Kühlwasser abgeführt werden kann.

Bei den Trommeln musste man bisher einen Regler verwenden, um den entsprechenden Druck einzustellen, bei dem die Verdampfungsenthalpie genau so groß ist, dass die benötigte Verdampfungswärme vom Wärmetauscher bereit gestellt werden kann. Nun können die Bauteile dies direkt berechnen.

Allerdings ist hierfür (wie auch beim Kondensator) eine interne Iteration erforderlich. Für das Feintuning dieser internen Iteration stehen folgende Parameter zur Verfügung:

- START (Startwert für die interne Iteration)
  Die Vorgabe ist optional. Wenn kein Startwert angegeben ist, wird bei Bauteil 20 ein Wert von 1 bar angenommen, bei Bauteil 70 der Nominalwert P1N
- CHL (maximale relative Änderung des Drucks von einem Iterationsschritt zum nächsten)
- ITSTEP (Häufigkeit der Neuberechnung des Druckes)
  Bei ITSTEP = 1 wird der Druck z. B. in jedem Iterationsschritt neu berechnet (Defaultwert), bei ITSTEP = 5 nur in jedem fünften Iterationsschritt.

Außerdem steht zum Konvergenz-Tuning der Parameter DERIVLEV zur Verfügung, mit dem eingestellt werden kann, wann im Modus Druckberechnung die partiellen Ableitungen mit in das Gleichungssystem genommen werden sollen. Diese werden aktiviert, wenn während der Iteration der berechnete Druck sich im Vergleich zum vorhergehenden Iterationsschritt um weniger als DERIVLEV unterscheidet. Da die Abhängigkeit sehr empfindlich ist (in der Beispielschaltung ändert sich der Druck um 100 mbar, wenn sich der Massenstrom nur um 1 g/s ändert), wurde der Defaultwert auf 0 gestellt. Es mag aber Schaltungen geben, bei denen dieser Parameter für Verbesserungen sorgt.

Für die innere Iteration nach dem Druck gibt es zwei Varianten:

- bei FSPEC = 3 (Bauteil 20) bzw. FSPECP = 3 (Bauteil 70) erfolgt die Druckvariation in der Weise, dass der Dampf-Volumenstrom konstant bleibt (d. h. der Massenstrom ändert sich mit dem Druck). Diese Variante ist zu empfehlen, wenn sich die Dampfmenge durch einen Kompressor einstellt, der einen bestimmten Volumenstrom fördern kann.
- bei FSPEC = 4 bzw. FSPECP = 4 erfolgt die Druckvariation beim einem konstanten Dampfmassenstrom.

Beim Bauteil 20 stand dieses Feature teilweise schon ab Patch 16.03 zur Verfügung.

1.2.5. Bauteil 30: Darstellung von Temperaturdifferenzen beim Differenzmesser

Bei FTYP = 2 (Temperaturdifferenz) wird das Ergebnis der Differenz jetzt als Leitungswert DELTA_T auf der Leitung ausgegeben.

Hintergrund: Intern verwendet Ebsilon ja die Variablen Druck, Massenstrom und Enthalpie. Bei FTYP = 2 wird der Zahlenwert der Differenz (in Kelvin) auf Variable Enthalpie auf die Leitung

geschrieben. Bisher wurde das Ergebnis auch als Enthalpie (in kJ/kg) angezeigt. Dies war jedoch unschön, vor allem, wenn man andere Einheiten verwenden wollte. Deshalb war zunächst naheliegend, das Ergebnis als Temperatur auf die Leitung zu schreiben. Dies führt aber bei einer Einheitenumrechnung ebenfalls zu einem Fehler. Wenn auf beiden Seiten dieselbe Temperatur herrscht, würde dann T = 0°C angezeigt werden, was bei Umschaltung auf Fahrenheit dann zu 32°F führen würde. Deshalb wurde der Wert jetzt auf DELTA_T geschrieben, was eine korrekte Einheitenumschaltung ermöglicht.

1.2.6. Bauteil 34: Salzwasser als Einspritzung beim Entspanner

Bei diesem Bauteil kann jetzt auch Salzwasser als Einspritzung (Anschluss 4) verwendet werden. Auch Salzwasser im Universalfluid wird jetzt korrekt behandelt (Anmerkung: im Universalfluid konnte auch bisher schon Salzwasser ausgewählt werden. Da das Bauteil jedoch kein Salz bei der Einspritzung behandeln konnte, wurde der Salzanteil aus der Einspritzung ignoriert).

1.2.7. Bauteil 45 (Wertanzeige)

1.2.7.1. Joule-Thomson-Koeffizient (FTYP = 99)

Mit FTYP = 99 kann jetzt auf den Joule-Thomson-Koeffizienten zurückgegriffen werden. Dieser ist definiert als

$$\mu = \partial T/\partial p|_h$$

und ist ein Maß für das Verhalten eines Gases beim Austritt aus einer Drossel:

- Ist $\mu = 0$ (grundsätzlich bei idealem Gas), ändert sich die Temperatur nicht
- Ist $\mu > 0$ (z. B. Luft), nimmt die Temperatur bei Entspannung ab
- Ist $\mu < 0$ (z. B. Wasserstoff), nimmt die Temperatur bei Entspannung zu

1.2.7.2. Interne Informationen (FTYP = 100)

FTYP = 100 dient zur Ausgabe interner Informationen, die über Logikleitungen zwischen den Bauteilen ausgetauscht werden, in die Ergebnis-Arrays RAINFO und RAAINFO. Diese dienen nur zu Diagnose-Zwecken für die Hotline und sollten vom Anwender nicht verwendet werden, da sich dieser Informationsaustausch von Release zu Release ändern kann.

1.2.7.3. Anteile bei 2-Phasenzuständen (FTYP = 101)

FTYP = 101 dient wie FTYP = 10 zur Ausgabe des Anteils einer Phase in einem 2-Phasen-Zustand. Während sich FTYP = 10 üblicherweise auf den „Dampfgehalt“ bezieht, also den Anteil der gasförmigen Phase beim Phasenübergang flüssig / gasförmig, kann FTYP = 101 auch bei anderen Phasenübergängen verwendet werden. Um welchen Phasenübergang es sich handelt, wird im Ergebniswert RPHASE angezeigt:

0: keine Phase
1: flüssig
2: gasförmig

3: gasförmig / flüssig (VLE)
4: fest
5: fest / flüssig (SLE)
6: fest / gasförmig (SVE)
7: gasförmig / flüssig / fest
8: überkritisch
9: flüssig / flüssig
10: fest / fest
11: gasförmig / flüssig / flüssig
12: fest / flüssig / flüssig
13: fest / fest / flüssig
14: flüssig / flüssig / überkritisch

Hinweis: bei den Bibliotheken Liblce und LibCO2 kann auch FTYP = 10 für Phasenübergänge fest / flüssig und fest / gasförmig verwendet werden, da diese Bibliotheken X-Werte zwischen 10 und 11 für fest / flüssig und X-Werte zwischen 100 und 101 für fest / gasförmig liefern.

1.2.7.4. Summe H + NCV

Mit FTYP = 102 wird die Summe aus Enthalpie und unterem Heizwert ausgegeben. Diese Summe ist insbesondere für Energiebilanzbetrachtungen häufig von Nutzen.

1.2.7.5. Innere Energie

Mit FTYP = 103 wird die innere Energie ausgeben. Bei Stoffwertbibliotheken, die keine Funktion für die innere Energie implementiert haben, wird diese berechnet gemäß

$$U = H - P * V$$

1.2.7.6. Diagnose Iterationsverlauf

Es gibt jetzt ein Ergebnis-Array RARESULT, in dem der Iterationsverlauf des RESULT-Wertes angezeigt wird. Die Aktivierung der Anzeige erfolgt mit dem Schalter FITLOG. Standardmäßig ist die Ausgabe deaktiviert, um nicht unnötige Datenmengen zu produzieren.

1.2.7.7. Isochore spezifische Wärmekapazität im 2-Phasen-Gebiet

Während die isobare spezifische Wärmekapazität cp bei Reinstoffen im 2-Phasen-Gebiet grundsätzlich nicht definiert ist, da sich bei konstantem Druck die Temperatur während des Phasenübergangs nicht ändert, ist die isochore spezifische Wärmekapazität cv stets definiert: wenn man nämlich das Volumen beim Phasenübergang konstant hält, ändert sich der Druck, und damit auch die Siedetemperatur, so dass die Ableitung

$$cv = \partial U / \partial T|_v$$

gebildet werden kann. Alternativ kann die Berechnung über die Beziehung

$$cv = T * \partial S / \partial T|_v$$

erfolgen.

Dieser Wert wird jetzt im 2-Phasen-Gebiet sowohl von Bauteil 45 als auch von Bauteil 168 (Größenkonverter) berechnet.

1.2.7.8. Schallgeschwindigkeit im 2-Phasen-Gebiet

Auch wenn im 2-Phasen-Gebiet die tatsächliche Ausbreitung einer Schallwelle von den Details der Geometrie abhängig ist (Blasen- bzw. Tröpfchen-Größe, Bildung eines Flüssigkeitsfilms an der Rohrwand), kann man über die thermodynamische Beziehung

$$w^2 = \partial\rho/\partial p|s$$

auch im 2-Phasen-Gebiet eine „Schallgeschwindigkeit“ berechnen. Dies wird sowohl von Bauteil 45 als auch von Bauteil 168 (Größenkonverter) so gemacht.

Tatsächlich hat diese Größe auch eine praktische Bedeutung bei Dampfturbinen:
Es ist nämlich auch im 2-Phasen-Gebiet die Geschwindigkeit, bei der der Durchfluss durch die Turbine maximal wird. Hierzu muss allerdings mit statischen Größen gerechnet werden, was mit Bauteil 168 möglich ist.

1.2.8. Bauteil 48 (Umschalter)

1.2.8.1. Ein- / Aus-Schalter FFU

Bei diesem Bauteil gibt es jetzt auch einen Schalter FFU, mit dem man die Übertragung ein- oder ausschalten kann.

1.2.8.2. FOUT = 0

Wie beim Bauteil 36 (Signalübertrager) gibt es jetzt hier auch die Einstellung FOUT = 0, mit der das Ausgangssignal auf den gleichen Typ wie das Eingangssignal gesetzt wird. Dies ist jetzt auch die Standardeinstellung.
Da in den meisten Fällen Eingangs- und Ausgangssignal vom selben Typ sind, erspart man sich dadurch die Umschaltung von FOUT, wenn man FIN umschaltet.

1.2.8.3. Temperaturübertragung (FIN = 2)

Es besteht jetzt auch die Möglichkeit, die Temperatur zu übertragen. Dies ist dann sinnvoll, wenn die beteiligten Leitungen unterschiedliche Zusammensetzungen haben, da man ansonsten ja auch die Enthalpie übertragen könnte.

Die Optionen zur Skalierung, Kennlinienverwendung und Übertragung auf eine andere Größe sind bei Bauteil 48 für die Temperaturübertragung allerdings nicht implementiert. Wegen der Besonderheiten bei der Einheitenumrechnung der Temperatur wären hierfür zusätzliche Angaben erforderlich, die allerdings beim Bauteil 36 implementiert sind.

Aus diesem Grunde wird bei Einstellung von FIN auch der Schalter FOUT und die Schalter und Vorgabewerte für die Skalierung inaktiv.

1.2.9. Bauteil 77: Rechenbaustein

Der Schalter FOPR erlaubt zwei neue Modi:

- FOPR = 14: Immer Daten von Port 1 weiterleiten
- FOPR = 15: Immer Daten von Port 2 weiterleiten

1.2.10. Bauteile 111 (Naturzugkühlturm), 112 (Ventilatorkühlturm): transiente Erweiterung

Die Bauteile 111 (Naturzugkühlturm) und 112 (Ventilatorkühlturm) wurden in Release 17 um transiente Funktionalität erweitert. In transienten Zuständen besitzen die Kühltürme große thermische Trägheit. Dies betrifft insbesondere die Kühlturmtasse, die in der Regel ein großes Volumen hat und daher wie ein großer Wasserspeicher wirkt. Aber auch die Kühleinbauten besitzen eine entsprechende Masse, die im transienten Zustand die Wärme ein oder ausspeichert. Diese Effekte können ab Release 17 in den Bauteilen 111 und 112 berücksichtigt werden. Für weitere Details wird auf die Online-Hilfe verwiesen.

1.2.11. Bauteil 118 (Massenspeicher)

1.2.11.1. Vorgabe und Berechnung der flächenspezifischen Wärmeverluste

Das Bauteil 118 ermöglicht jetzt die flächenspezifischen Wärmeverluste vorzugeben. Dazu wurde ein neuer Schalter FQLOSS eingebaut. Für

- FQLOSS = 0 wird (wie bisher) der Wärmeverlust durch den temperaturabhängigen Wert QLOSSR (kW/K) vorgegeben
- FQLOSS = 1 wird dagegen der flächen- und temperaturdifferenzspezifische Wärmeverlust (kW/m²K) verwendet

Bei den flächen- und temperaturdifferenzspezifischen Wärmeverlusten wird es von einem vertikalen Zylinder ausgegangen und zwischen den spezifischen Verlustwerten

- QLS – zur Seite des Zylinders (die verlustrelevante Fläche ist vom Füllstand des Speichers abhängig)
- QLT – nach oben
- QLB – nach unten

unterschieden.

Die Vorgabe der flächenspezifischen Wärmeverluste (FQLOSS = 1) ist nur bei einer klaren Definition der Zylindergeometrie bzw. der für den Verlust relevanten Fläche möglich (FFILL = 0 und FLEV = 0 oder FLEV = 1).

Eine sinnvolle Abschätzung für die flächenspezifischen Wärmeverluste kann aus der Annahme erfolgen, dass der höchste Widerstand bei der Wärmeübertragung zur Umgebung in der Isolierung des Speichers entsteht. Für eine bekannte Isolierungsdicke Delta (m) und eine bekannte Wärmeleitfähigkeit der Isolierung LAMBDA (W/mK) kann der flächen- und temperaturspezifischer Verlustwert wie

QLOSS = LAMBDA/DELTA

abgeschätzt werden.

1.2.11.2. Berechnung des Fluiddruckes

Das Bauteil 118 ermöglicht jetzt die Berechnung des Fluiddruckes, PSTO, PNEW sowie Druck am Anschluss 2 (Austritt), aus der geodätischen Höhe des Fluids, die wiederum vom Füllstand des Speichers abhängig ist. Zusätzlich wird ein überlagerter Druck PLOAD oberhalb der Fluidsäule berücksichtigt. Dazu wurde ein neuer Wert beim Schalter FPSTO eingeführt. Für

- FPSTO = 2

wird aus PLOAD und dem aktuellen Füllstand der neue Fluiddruck PNEW berechnet. Der Druck am Anschluss 2 ergibt sich als Mittelwert zwischen dem Druck zu Beginn des Zeitschrittes – PSTO- und dem Druck zu Ende des Zeitschrittes – PNEW.

Der beschriebe Druckberechnungsmodus (FPSTO = 2) ist nur bei einer klaren Definition der Zylindergeometrie bzw. der für die Berechnung relevanten Höhe möglich (FFILL = 0 und FLEV = 0 oder FLEV = 1)

1.2.12. Bauteil 117 (Sonne): Horizont Daten hinzugefügt

Das Bauteil 117 wurde um die Kennlinie CHORIZON und den Schalter FHORIZON erweitert. Bei

- FHORIZON = 0 ist der Horizont waagrecht (= 0°)
- FHORIZON = 1 wird der Höhenwinkel des Horizonts aus der Kennlinie CHORIZON bestimmt. Dieser ist eine Funktion des Azimuts.

Aus Gründen der Kompatibilität wird der Horizontwinkel nur zur Verfügung gestellt. Es obliegt den Solarbauteilen, diesen Wert auch zu verwenden.

1.2.13. Bauteil 119 (Indirekter Speicher): Geometrie basierte Druckverlustberechnung

Das Bauteil 119 wurde um eine Geometrie basierte Druckverlustberechnung erweitert. Dazu wurde der Schalter FDP eingeführt. Bei

- FDP = 0 wird (wie bereits vor Release 17) der Druckverlust aus dem Wert DP12N und der entsprechenden Off-Design-Skalierung berechnet
- FDP = 1 wird die Geometrie des Rohres verwendet, um den Druckverlust zu berechnen

Die Geometrie basierte Druckverlustberechnung ist nur dann sinnvoll, wenn das Bauteil 119 ein echtes Rohr modelliert. Zu Berechnung des Druckverlusts wird in diesem Fall außer dem Volumenstrom des Mediums der innere Durchmesser DIAI, die Rohrlänge LSTO sowie die Wandrauigkeit KS verwendet.

1.2.14. Bauteile 119, 145, 165, 166: neuer Vorgabewert FNUMSC

Das numerische Schema für die Berechnung der Fluidtemperaturen war in den Bauteilen 119, 166 vom Schalter FSPECM abhängig. Für FSPECM = 1 (Fluidmasse vernachlässigt) wurde immer das Schema CDS (Zentraldifferenzen) verwendet. Für FSPECM > 1 (Fluidmasse berücksichtigt) wurde bei diesen Bauteilen Upwind-Schema verwendet. In den Bauteilen 145, 165 wurde immer Upwind-Schema

verwendet. CDS-Schema hat eine höhere Genauigkeit gegenüber Upwind-Schema. Upwind-Schema ist dagegen stabiler für die Innere Konvergenz des Bauteils.

Ab Release 17 gibt es in den Bauteilen 119, 145, 165, 166 einen neuen Schalter FNUMSC. Damit hat der Benutzer die Möglichkeit, unabhängig vom FSPECM-Wert das numerische Schema zu wählen. FNUMSC = 0 entspricht dem Upwind-Schema, FNUMSC = 1 entspricht dem CDS-Schema.

Sollte in der inneren Iteration aufgrund von erkannten Konvergenz-Problemen vom CDS auf Upwind umgeschaltet werden, wird der Benutzer am Ende der Simulation davon gewarnt.

In den Schaltungen, die mit früheren EBSILON-Versionen als Release 17 gespeichert wurden, wird beim Laden der Wert von FNUMSC-Schalter in Abhängigkeit vom Schalter FSPECM gesetzt, um die alten Ergebnisse zu reproduzieren. Jedoch funktioniert dieses automatische Setzen des Schalters nicht, wenn FSPECM keinen einfachen gültigen numerischen Wert hat, sondern z. B. ein Verweis (Kernelexpression) auf weitere Bauteile enthält. In einem solchen Fall möge der Benutzer selbst den Schalter FNUMSC so setzen, dass die alten Ergebnisse reproduziert werden.

1.2.15. Bauteile 119, 138, 165, 166: alternativer Bezugswert für die Druckverlustberechnung in Off-Design, neuer Vorgabewert FDPBASE

Die Druckverlustberechnung in den Bauteilen 119, 138, 165, 166 bezog sich bis Release 17 immer auf das spezifische Volumen am Eintritt des Bauteils. Dies kann zu höheren Ungenauigkeiten führen, wenn die Temperaturänderung zwischen Ein- und Austritt hoch ist und das strömende Medium kompressibel / gasförmig ist.

Ab Release 17 gibt es einen Schalter FDPBASE, bei dem

- FDPBASE = 0 – hier wird als Bezugsgröße für die Druckverlustberechnung bei der Off-Design-Rechnung der Mittelwert des spezifischen Volumens zwischen Ein- und Austritt verwendet
- FDPBASE = 1 – hier wird als Bezugsgröße für die Druckverlustberechnung bei der Off-Design-Rechnung das spezifische Volumen am Eintritt verwendet (wie bis Release 17)

Zusätzlich wurde als Bezugsgröße ein neuer Nominalwert V12N (das mittlere spezifische Volumen zwischen Ein- und Austritt) den Bauteilen hinzugefügt.

1.2.16. Bauteile 119, 165, 166: Berücksichtigung der Wärmeleitung im Fluid, neue Vorgabewerte FHC, CLMFL

Die Wärmeleitung in Fluid wurde bis Release 17 nur im Bauteil 145 berücksichtigt und in den Bauteilen 119, 165, 166 vernachlässigt. Diese Annahme ist für den Fall des strömenden Fluids im Speicher gerechtfertigt. Bei der Berechnung eines ruhenden Fluids (z. B. Bereitschaft-Betrieb des Speichers) ist das doch erforderlich den Wärmeaustausch zwischen den Fluidelementen im Speicher zu berücksichtigen. Dazu wurde in den Bauteilen 119, 165, 166 neue Vorgabewerte eingefügt

Der Schalter FHC steuert das Berücksichtigen der Wärmeleitung im Fluid

- FHC = 0 – dabei wird (wie bis Release 17) die Wärmeleitung im Fluid komplett vernachlässigt

- FHC = 1 – hier wird die Wärmeleitung im Fluid berücksichtigt. Dabei werden die Stoffwerttafeln des Fluids für die Rechnung der Wärmeleitfähigkeit verwendet

Zusätzlich kann die Wärmeleitfähigkeit mit einem Faktor CLMFL multipliziert werden. Das kann z. B. im Fall der freien Konvektion im Speicher erforderlich sein.

1.2.17. Bauteile 119, 165: Modus FSPECM = 4 funktioniert auch mit NFLOW > 1

Bis Release 17 war der Modus FSPECM = 4 – Vorgabe der Massenströme, Berechnung des Druckes – nur im Bauteil 119 vorfügbar. Dieser Modus funktionierte nur mit einem Element in der Strömungsrichtung NFLOW = 1.

Ab Release 17 ist der Modus FSPECM = 4 sowohl im Bauteil 119 als auch im Bauteil 165 verfügbar. Weiterhin funktioniert dieser Modus in den beiden Bauteilen für eine beliebige Anzahl der Elemente in der Strömungsrichtung (NFLOW >= 1).

1.2.18. Bauteil 137: Photovoltaik

Es wurden Methoden zur Berücksichtigung von Abschattungsverlusten hinzugefügt. Mit dem Schalter FSHADING > 0 können diese Aktiviert werden.

- FSHADING = 1: die Eingabematrix MXSHADINGLOSS wird verwendet
- FSHADING = 2: der Eingabewert SHADINGLOSS wird verwendet
- FSHADING = 3: die Kennlinie CSHADINGLOSS wird verwendet
- FSHADING = 4: die Kernelexpression ESHADINGLOSS wird verwendet

Der Output wird um diese relativen Verluste verringert.

1.2.19. Bauteil 146: Getriebe

Beim Getriebe wurde die Möglichkeit geschaffen, die Drehzahl auf beiden Seiten auf der Welle vorzugeben (FSPEC = -1). Dadurch wird die Nutzung des neuen Bauteils 171 (Schwungradspeicher) erleichtert.

1.2.20. Bauteil 160: Alternative Vorgabe des Startwertes für den Füllstand statt der Fluidenthalpie für 2-Phasen-Medien

Im Bauteil 160 musste man bis Release 17 den Wert HFLSTART (Startwert für die Fluidenthalpie im Speicher) vorgeben. Daraus hat sich der Startwert für den Füllstand des Speichers bei 2-Phasen-Medien (z. B. Dampfgefällespeicher) ergeben.

Ab Release 17 gibt es alternativ die Möglichkeit den Startwert für den Füllstand des 2-Phasen-Medium im Speicher vorzugeben. Dazu gibt es einen Schalter FENTHFL und einen Vorgabewert XFLSTART.

- Bei FENTHFL = 0 wird (wie bisher) der Wert HFLSTART verwendet
- Bei FENTHFL = 1 wird direkt der Wert XFLSTART (Startwert für Volumenanteil flüssig - Füllstand) verwendet

Aus dem Wert XFLSTART und dem vorgegebenen Druck PFLSTART ergibt sich im Fall FENTHFL = 1 die entsprechende Fluidenthalpie. Die Verwendung von XFLSTART hat für reine Gasspeicher keinen Sinn.

1.2.21. Bauteil 165 (Thermischer Regenerator / Schüttgutspeicher)

1.2.21.1. Anschluss 6 zur Kopplung einer zusätzlichen externen Wärmequelle / Wärmesenke
Das Bauteil 165 wurde um einen weiteren Logikanschluss erweitert (Anschluss 6). An diesem Anschluss kann ein Wärmestrom vorgegeben werden, der in die Energiebilanz des Bauteils eingeht. Damit kann z. B. eine zusätzliche externe Heizung oder Kühlung (der Wärmestrom darf auch negativ sein) des Speichers modelliert werden.

1.2.21.2. Neuer Rechenmodus, Schalter FCALC
Das Bauteil 165 wurde um einen weiteren Rechenmodus erweitert. Bis Release 17 wurde in der Berechnung des Bauteils die innere Speichermasse und das Gehäuse immer separat bilanziert und dabei der gesamte Fluidmassenstrom entsprechend den Flächenanteilen zwischen 2 Teilen aufgeteilt. Ab Release 17 ist es möglich, beide Teile, innere Speichermasse und das Gehäuse, simultan zu betrachten. Dies ist eine realistischere Betrachtung und sie wird ab Release 17 empfohlen. Aus Kompatibilitätsgründen bleibt die bisherige separate Betrachtung der inneren Speichermasse und des Gehäuses erhalten. Um zwischen 2 Betrachtungsweisen zu wählen, wurde ein neuer Schalter FCALC eingeführt

- FCALC = 0 entspricht der separaten Lösung der inneren Speichermasse und des Gehäuses
- FCALC = 1 entspricht der simultanen Lösung der inneren Speichermasse und des Gehäuses

1.2.22. Bauteil 168 (Größenkonverter)

1.2.22.1. Geschwindigkeitsvorgabe von außen
Das Bauteil wurde mit einem Logikeingang (Anschluss 3) versehen, um die Strömungsgeschwindigkeit von außen vorgeben zu können, die für die Berechnung der statischen Größen verwenden wird. Der Schalter FIN3 dient dazu, die Nutzung von Anschluss 3 zu aktivieren. Die gewünschte Geschwindigkeit ist dann – beispielsweise durch einen Messwert mit FTYP = 33 – als Enthalpie vorzugeben.

Es wurde auch ein Ergebniswert RVEL hinzugefügt, in dem angezeigt wird, mit welcher Strömungsgeschwindigkeit die Berechnung durchgeführt wurde.

1.2.22.2. Joule-Thomson-Koeffizient
Wie bei Bauteil 45 steht jetzt auch beim Bauteil 168 der Joule-Thomson-Koeffizient zur Verfügung. Bei Bauteil 168 ist hierfür FMAPx = 415 einzustellen.

1.2.23. Bauteil 170: Sammelschienenadapter
Dieses Bauteil ist ein Adapter, der auf einer Sammelschiene zusammen mit den Bauteilen 148 – 150 verwendet werden kann. Hierzu werden die „äußeren“ Anschlüsse 1 und 2 in die Sammelschiene eingebaut. Zwischen den „inneren“ Anschlüssen 3 und 4 können beliebige Ebsilon-Bauteile verbaut werden.

Die meisten Bauteile benötigen zur korrekten Berechnung auf Leitungen, die echte Rohre darstellen, einen nicht-negativen Massenstrom.

Der Sammelschienenadapter (vgl. Erklärung von FCALCDIR) stellt sicher, dass der Massenfluss von Anschluss 3 nach 4 positiv ist. D.h. je nach Strömungsrichtung in der Sammelschiene werden entweder die Leitungsdaten von Anschluss 1→3 und 4→2 durchgeleitet (bei positiver Richtung) bzw. 2→3 und 4→1 durchgeleitet (bei negativer Richtung). Einfache Konstruktionen, die auf der Verbindung von Anschluss 3 nach 4 nicht den Massenstrom ändern, funktionieren i.d.R. mit den vorab Einstellungen (z.B. Druck/Wärmesenken bzw. -quellen).

Der Vorgabewert FCALCDIR wird für die Zuordnung der Anschlüsse 1 und 2 zu den Anschlüssen 3 und 4 verwendet:

- „Wie der größere Massenstrom: 0“: der äußere Strom mit dem größeren absoluten Massenfluss bestimmt die Strömungsrichtung
- „Wie M1: 1“: Die Strömungsrichtung an Anschluss 1 bestimmt die Strömungsrichtung
- „Wie M2: -1“: Die Strömungsrichtung an Anschluss 2 bestimmt die Strömungsrichtung
- „Vom höheren zum niedrigeren Druck: 2“: der äußere Anschluss mit dem höheren Druck bestimmt die Strömungsrichtung (z.B. für Rohre, Bauteil mit Druckabfall)
- „Vom niedrigeren zum höheren Druck: -2“: der äußere Anschluss mit dem niedrigeren Druck bestimmt die Strömungsrichtung (z.B. für Pumpen, Kompressoren, Bauteil mit Druckerhöhung)
- „Kernelexpression ECALCDIR verwenden: 3“: das Ergebnis der Kernelexpression ECALCDIR bestimmt die Strömungsrichtung:
    - `evalexpr >= 0.0`: Anschluss 1→3 und Anschluss 4→2
    - `evalexpr < 0.0`: Anschluss 2→3 und Anschluss 4→1

Bei Änderung des Massenstroms muss ggfs. mit Reglern gearbeitet werden.
Das Bauteil besitzt zusätzlich einen Logikanschluss (5) auf dem der Massenfluss die Werte 1 und -1 annimmt, je nachdem ob der aktuelle Massenstrom auf der Sammelschiene größer gleich 0 ist (Wert 1) oder negativ ist (Wert -1).
Diese Information kann z.B. an Anschluss 4 von Speicherbauteil 119 weitergeleitet werden, um es über die Durchströmungsrichtung zu informieren (vgl. Bauteil119.FDIR = „entsprechend Massenstrom Anschluss 4...“).

Die Vorgabewerte FSPECM und FSPECP legen fest, welche der Variablen M3, M4 bzw. P3, P4 als von außen definiert angesehen wird. I.d.R. funktioniert die Einstellung „von außen ableiten“. Bei Konstruktionen im inneren von Bauteil 170, die gewisse Massenstrom oder Druck Vorgaben benötigen, kann hier nachjustiert werden.

ACHTUNG: Dieses Bauteil befand sich im Laufe der Patches zu Release 16 in der Entwicklung. Daher kommt es möglicherweise zu Änderungen im Verhalten des Bauteils in Release 17, z. B. neue/andere Spezifikationswerte, verändertes Verhalten während der Simulation, etc.

1.3. Stoffwerte

1.3.1. FRAGOLTHERM-Öle

Für den Leitungstyp Öl/Schmelze wurden folgende Wärmeträger-Öle der Firma Fragol aufgenommen:

- Fragoltherm DPO
- Fragoltherm 660
- Fragoltherm HT
- Fragoltherm 620
- Fragoltherm 550
- Fragoltherm 590
- Fragoltherm 500
- Fragoltherm Q-HTF
- Fragoltherm Q-32-N
- Fragoltherm FG-35
- Fragoltherm Q-20
- Fragoltherm Q-7

Da vom Hersteller angegeben wurde, dass Fragoltherm DPO auch in der Gasphase nutzbar ist, wurden sowohl „Fragoltherm DPO liquid“ und „Fragoltherm DPO gaseous“ aufgenommen. Dabei ist allerdings zu beachten, dass für die Gasphase keine zusätzlichen Daten über die Druckabhängigkeit angegeben waren. Zur Modellierung der Wärmeübertragung ist eine Vernachlässigung der Druckabhängigkeit sicherlich genau genug. Die Dichte ist allerdings nur in der Nähe des Sattdampfpunkts korrekt, und die Näherung als inkompressibles Fluid dürfte zu einer deutlichen Unterschätzung von Pumpen- bzw. Kompressorleistungen führen.

1.3.2. UserProps-DLL

Für den Leitungstyp Universalfluid gibt es die neue Bibliothek „User-defined properties (dll)“ (kurz UserProps). Hierfür kann der Anwender eine Dll erstellen, welche die Berechnung der Stoffwerte ausführt.
Im Gegensatz zur bereits existierenden User2Phase-Dll Schnittstelle, erlaubt UserProps die Spezifikation von Zusammensetzungen (Stoffwert-Vektoren) und zusätzlichen Attributen.
Die Schnittstelle ist in der Datei „user_props.h“ in Unterordner „<Ebsilon-Installationverzeichnis>\Data\Examples\user_props_dll“ beschrieben und verwendet die Sprache C++. Dort befindet sich auch ein Beispiel-Projekt. Es wird empfohlen zum Erstellen einer eigenen Bibliothek „Microsoft VisualStudio 2022“ zu verwenden.

Jeder Eintrag „User-defined properties (dll)“ eines Universalfluids hat die zwei Standardattribute:

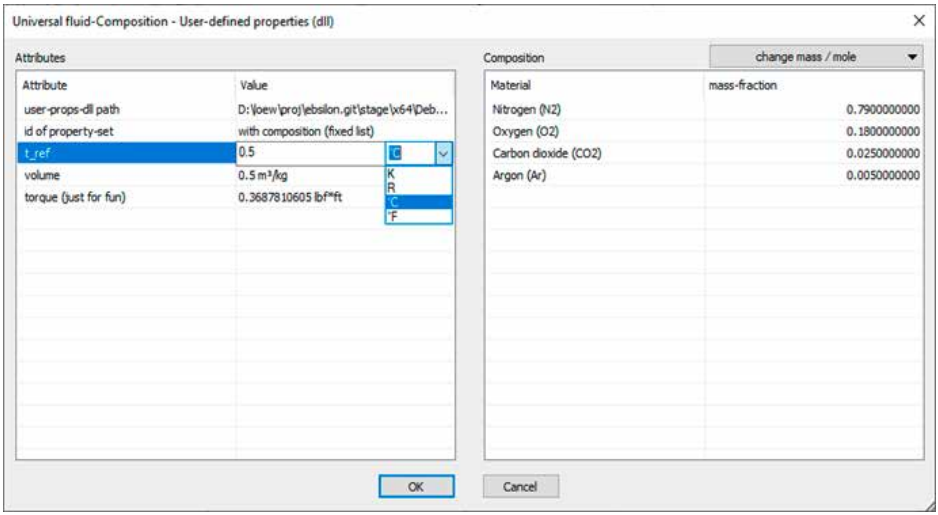
1. „user-props-dll path“: Pfad zur Dll. Ggfs. können auch mehrere unterschiedlich Dll in einem Modell, oder auch Fluid verwendet werden.
2. „id of property-set“: Id der ausgewählten property-sets (Stoffwertmodelle). Eine Dll kann beliebig viele unterschiedliche Stoffwertmodelle enthalten.

Für jedes „property-set“ können zum einen weitere Attribute von folgenden Typen definiert werden:

- „double“: floating-point Wert (double-precision)
- „value with unit”: Wert mit Einheit
- „list“: Liste (Combobox) zum Auswählen
- „string“: Text

Und eine Vorgabe, welche Substanzen vorkommen dürfen (ggfs. auch gar keine für Reinstoffmodelle).

Weitere Informationen über die Funktionen und Typen der Schnittstelle können in der Datei „user_props.h“ gefunden werden. Im gleichen Verzeichnis befindet sich auch Code für eine Beispiel-Dll.



1.3.3. VLMIX

Die VLMIX-Bibliothek erweitert die in FDBR verfügbaren Komponenten um die Fähigkeit, Dampf-Flüssigkeit-Gleichgewichte zu berechnen. Die eingesetzten Methoden konzentrieren sich auf robuste Berechnungen mit angemessener Geschwindigkeit unter Inkaufnahme einer verminderten Genauigkeit.

Des Weiteren wird empfohlen, die Grenzen der im Folgenden beschriebenen eingesetzten Methoden zu beachten.

1.3.3.1. Thermische Eigenschaften

Die thermischen Eigenschaften werden entweder mit einem idealen Ansatz (druckunabhängig) oder mit einer kubischen Zustandsgleichung (Peng-Robinson, Redlich-Kwong, Soave-Redlich-Kwong) berechnet.

Beim idealen Ansatz zur Ableitung der Enthalpie und Entropie der flüssigen Phase wurde eine Regressionsfunktion für die Verdampfungswärme benutzt.

$$H_{liq} = H_{gas} - dH_{vap}$$

$$S_{liq} = S_{gas} - \frac{dH_{vap}}{T}$$

Die Mischungsregeln nutzen den klassischen linearen Ansatz.

1.3.3.2. Dampf-Flüssigkeit-Gleichgewicht (VLE)

Für VLE gilt die folgende Gleichung:

$$f_{liq} = f_{gas}$$

wobei f_{phase} die Fugazität der Komponente *i* in der entsprechenden Phase ist.

Die Fugazität der flüssigen Phase wird nach dem Raoultschen Gesetz berechnet:

$$f_{liq} = x_i * pvap_i$$

wobei x_i die molare Fraktion der Komponente *i* in der flüssigen Phase und $pvap_i$ der Gasdruck der reinen Komponente *i* bei aktueller Temperatur ist.

Es wird angenommen, dass die Fugazität der Gasphase dem im Dalton-Gesetz definierten Partialdruck entspricht:

$$f_{vap} = pres_i = y_i * pres$$

$$\sum_i y_i = 1$$

$$\sum_i y_i * pres_i = pres$$

wobei y_i die molare Fraktion, $pres_i$ der Partialdruck der Komponente *i* in der Gasphase und *pres* der aktuelle Druck des Mediums ist.

Der Gasdruck jeder Komponente wird zur Bestimmung des VLE benötigt. Für einige Komponenten waren Daten für eine erweiterte Antoine-Gleichung verfügbar. Für die restlichen wurden diese wie unten beschrieben geschätzt.

1.3.3.3. Schätzung des Dampfdrucks

Die Antoine-Gleichung für den Dampfdruck hat die folgende Form:

$$ln(pvap_i) = A_i + \frac{B_i}{C_i + T}$$

Daher brauchen wir drei Datenpunkte, um die Parameter A_i, B_i und C_i zu bestimmen.

Kandidaten sind der kritische Punkt, der Tripelpunkt und der natürliche Siedepunkt, wodurch sichergestellt wird, dass diese Punkte genau aufeinander abgestimmt sind.

Wenn wir nur zwei Datenpunkte haben, wird die folgende Gleichung verwendet

$$ln(pvap_i) = A_i + \frac{B_i}{T}$$

1.3.3.4. Einschränkungen

Der beschriebene Ansatz kann nur einfache VLE mit nur einer flüssigen Phase bestimmen. Auch azeotropes Verhalten (z. B. Ethanol-Wasser) kann nicht beschrieben werden und Mischungen nahe dem azeotropen Punkt werden abweichen.

Da die VLE-Berechnungen die Antoine-Gleichung für die Dampfberechnung nutzen, welche selbst nur bis zum kritischen Punkt gültig ist, beschreibt dieser Ansatz die Löslichkeit überkritischer Komponenten in einer flüssigen Phase nur schlecht. In diesem Fall wäre ein Ansatz mit dem Henry-Gesetz notwendig, was für eine zukünftige Release geplant ist.

Außerdem ist die Verdampfungswärme für überkritische Komponenten bei Anwendung der Idealgas-Formulierung nicht definiert und die Wärmeeigenschaften der flüssigen Phase einer solchen Komponente weichen daher ab. Glücklicherweise ist die Konzentration solcher Komponenten in der flüssigen Phase niedrig.

Bei hohem Druck, wenn der Kompressibilitätsfaktor Z_i entfernt von 1 ist, führt die zur Berechnung der Flüchtigkeiten der Gasphase verwendete Methode zu einem größeren Fehler für die Verteilung der Zusammensetzung auf die einzelnen Phasen.

2. Benutzeroberfläche

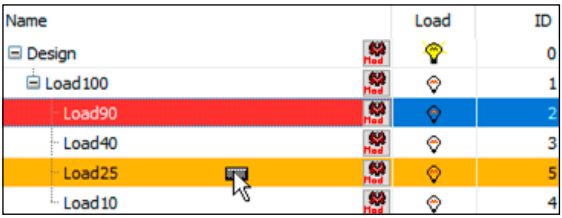
2.1. QT Carnot-Faktor Diagramm: Anzeige der Exergieverluste

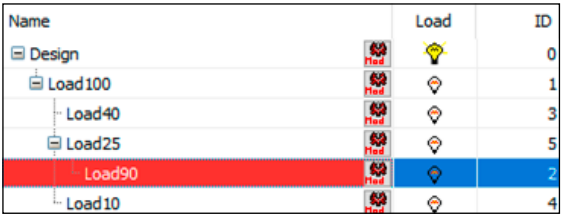
Im QT Carnot-Faktor Diagramm ist es ab Release 17 möglich, die Exergieverluste (entspricht der SI-Einheit von kW) anzuzeigen. Diese Verluste werden als integrale Fläche zwischen den 2 Kurven („kalte“ und „heiße“ Seite) eines Wärmetauschers ermittelt. Die Anzeige wird aktiviert / deaktiviert über den Menüpunkt „Ansicht“ – „Exergieverluste anzeigen“ im Diagrammen-Dialog. Das Aktivieren der Anzeige ist ausschließlich für den QT Carnot-Faktor Diagramm-Typ möglich.

2.2. Profildialog überarbeitet

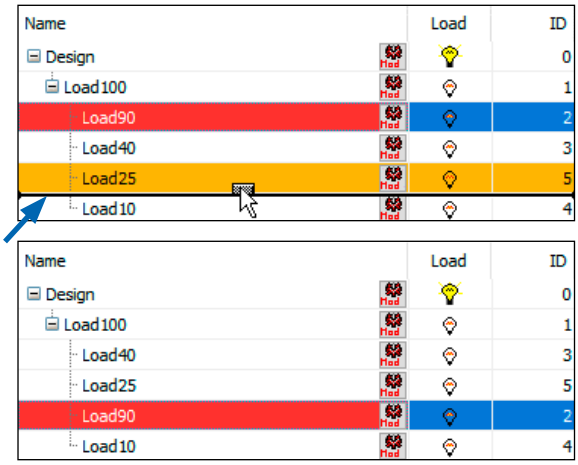
Der Profildialog verwendet nun ein verbessertes Kontrollelement zur Darstellung der Baumstruktur. Diese Kontrollelement erlaubt Multi-Selektion mit der Maus. Weiterhin können Profile mit der Maus per Drag&Drop verschoben, bzw. durch zusätzliches Halten von STRG kopiert und mit SHIFT + STRG rekursiv kopiert werden. Bitte beachten Sie, dass das Drop-Target (das Element, über dem die bewegten Elemente losgelassen werden) entweder

- falls keine Einfügemarken über oder unter dem Element angezeigt werden, das neue Elternelement ist

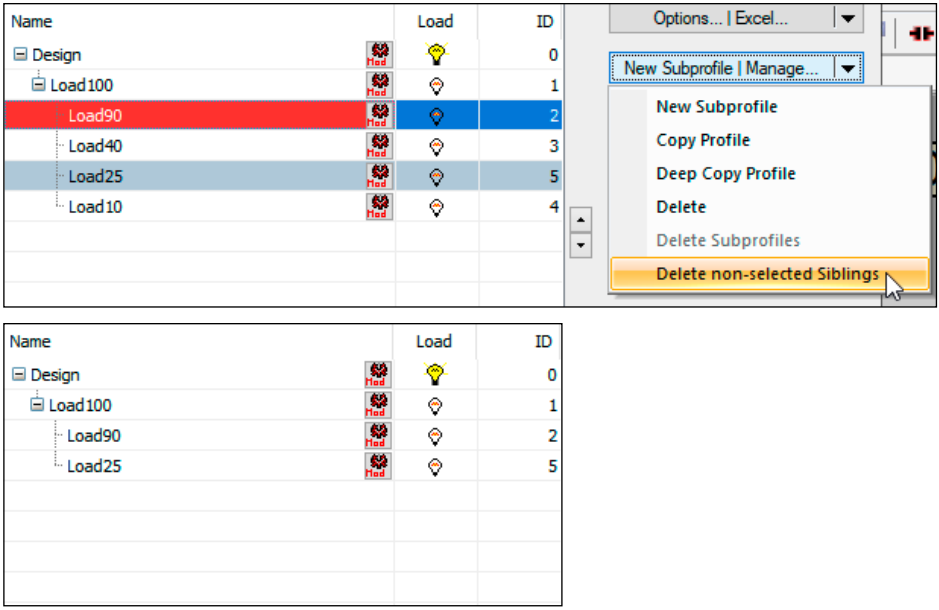




- andernfalls (d.h. Einfügemarken über oder unter dem Element angezeigt werden) das Vorgänger- bzw. Nachfolgerelement ist



Es gibt nun auch eine Funktion, welche alle nicht-selektierten Geschwisterprofile löscht:

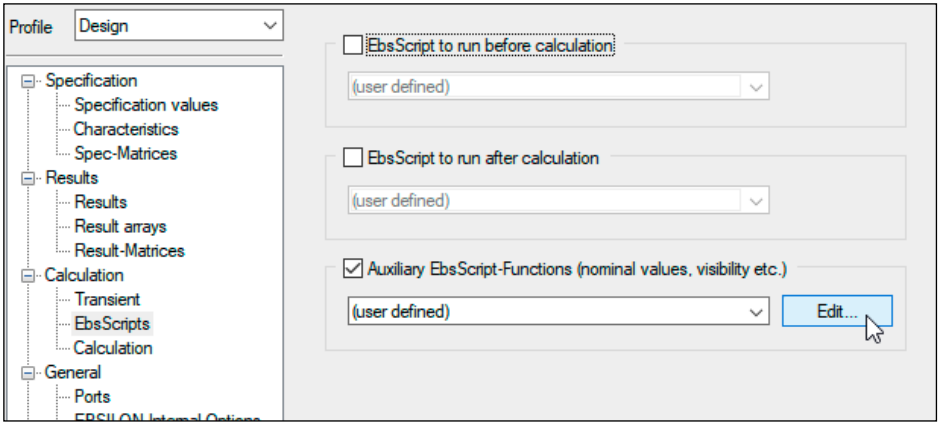


Die zusätzliche Sicherheitsabfrage vor dem Verschieben oder Löschen von Profilen kann mittels einer Checkbox abgeschaltet werden.

2.3. Macro-Objekte: dynamische Darstellung von Spez-/Ergebniswerten

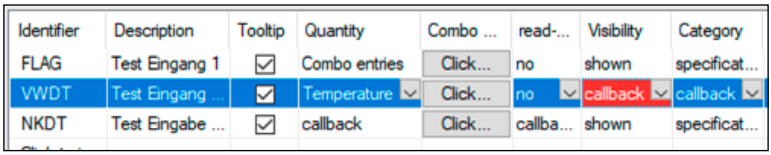
Bisher konnte die Darstellung von Spezifikations- und Ergebniswerten nur mittels des XUI-Dll Interfaces dynamisch aktualisiert werden.

Ab Release 17 ist dies nun auch mittels EbsScript-Code, der mit im Modell gespeichert wird, möglich. Öffnen Sie hierfür im Macro-Interface die Hilfs-EbsScripte



und fügen Sie dort den gewünschten Code ein.

Bitte beachten Sie, dass zusätzlich bei der Definition des Vorgabe- bzw. Ergebniswerts die entsprechende Eigenschaft auf „callback“ gesetzt werden muss:



Die folgenden Funktionen sind Beispiele und können als Vorlagen verwendet werden, müssen jedoch natürlich an das jeweilige Modell angepasst werden:

```
// dynamically change visibility
// will be called for every spec-/result-value with visibility set to „callback“
// return EbsValueVisibilityCallback to use default (see @System unit)
function getEbsValueVisibility(e:ebsvar):EbsValueVisibilityEnum;

begin
        // default behaviour for other values
        getEbsValueVisibility:=EbsValueVisibilityCallBack;

                if getEbsVarIdentifier(e) = „VWDT“ then
                begin
                        if $.NKDT > 10 then
                        begin
                                getEbsValueVisibility:=EbsValueVisibilityGrayed;
                        end
                else
                        begin
                                getEbsValueVisibility:=EbsValueVisibilityShown;
                        end
                end;
end;

// dynamically change read-only
// will be called for every spec-/result-value with read-only set to „callback“
// return EbsValueReadonlyEnum to use default (see @System unit)
function getEbsValueReadOnly(e:ebsvar):EbsValueReadonlyEnum;

begin
        // default behaviour for other values
        getEbsValueReadOnly:=EbsValueReadonlyCallBack;

        // example code: changes spec-value „NKDT“ to read-only, when FLAG is equal to 1
                if getEbsVarIdentifier(e) = „NKDT“ then
                begin
                        if $.FLAG = 1 then
                        begin
                                getEbsValueReadOnly:=EbsValueReadonlyYes;
                        end
                else
                        begin
                                getEbsValueReadOnly:=EbsValueReadonlyNo;
                        end
                end;
end;
```

```
// dynamically change category
// will be called for every spec-/result-value with category set to „callback“
// return EbsValueCategoryEnum to use default (see @System unit)
function getEbsValueCategory(e:ebsvar):EbsValueCategoryEnum;

begin
        // default behaviour for other values
        getEbsValueCategory:=EbsValueCategoryCallBack;

        // example code: changes spec-value „VWDT“ to nominal, when FLAG is equal to 1
                if getEbsVarIdentifier(e) = „VWDT“ then
                begin
                        if $.FLAG = 1 then
                        begin
                                getEbsValueCategory:=EbsValueCategoryNominalValue;
                        end
                else
                        begin
                                getEbsValueCategory:=EbsValueCategorySpecificationValue;
                        end
                end;
end;

// dynamically change dimension
// will be called for every spec-/result-value with category set to „callback“
// return DIM_ERROR to use default (see @System unit)
function getEbsValueDimension(e:ebsvar):DIM_Enum;

begin
        // default behaviour for other values
        getEbsValueDimension:=DIM_ERROR;

        // example code: changes spec-value „NKDT“ to bar, when FLAG is equal to 4
                if getEbsVarIdentifier(e) = „NKDT“ then
                begin
                        if $.FLAG = 4 then
                        begin
                                getEbsValueDimension:=DIM_bar;
                        end
                else
                        begin
                                getEbsValueDimension:=DIM_K;
                        end
                end;
end;
```

3. Zusatzmodule

3.1. Allgemeines

3.1.1. Liste aller Dimensionen und Einheiten

Im Installationsverzeichnis befindet sich im Unterordner „Docu“ eine HTML-Datei namens „dimensions_and_units.html“. Diese enthält eine Liste aller in Ebsilon verwendeten Dimension und der zugehörigen Einheiten. Dimensionen und zugehörigen Einheiten sind intern verlinkt, und die verwendeten Umrechnungsfaktoren (bzw. Offsets) sind aufgelistet.

3.2. EbsScript

3.2.1. Editor

Bereits im Laufe von Release 16 erhielt der Editor ein neues Farbschema. Mit Release 17 kommen nun folgende Änderungen und Erweiterungen hinzu:

- Der Hintergrund von selektiertem Text wird hellblau dargestellt, und die Vordergrundfarbe bleibt unverändert.
- Bei den Syntax-Optionen können Vorder- und Hintergrundfarbe, und ob der Text fett dargestellt wird, für die jeweiligen Elemente eingestellt werden
- Im Editor-Menü gibt es den Eintrag „Bearbeiten -> Leerraum anzeigen“. Wenn diese Option aktiviert ist werden Leerzeichen und Tabulatoren als sichtbare Zeichen dargestellt.

3.2.2. Neue EbsVar Funktionalität / Funktionen

Zur Verwendung mit EbsScript Variablen / Objekten vom Typ *ebsvar* gibt es folgende neue Funktionen:

- *Function setExpression(value:const ebsvar; expression:string):Boolean;*
  Setzt bei dem *ebsvar* value, die angegebene *expression*.
  Anmerkung: Diese Funktion ermöglicht es, bei Spez- / Ergebniswerten mit Dimension „Text“, „String“ o. ä. von EbsScript einen Ausdruck zu setzen.

- *Procedure copyEbsVar(source:const ebsvar; dest:const ebsvar; sourceProfileId:integer=-1; destProfileId:integer=-1);*
  Kopiert den Inhalt von *source* nach *dest*. Wenn *sourceProfileId* bzw. *destProfileId* nicht angegeben oder gleich -1 wird das aktuelle Profil für den entsprechenden Wert verwendet. Ein Aufruf mit *source* gleich *dest* ist ausdrücklich erlaubt, um z. B. einen Spez- / Ergebniswert in ein anderes Profil zu kopieren.

Weiterhin ist nun eine direkte Zuweisung zwischen den Typen *ebsvar* und *Variant* möglich. Dies erlaubt z. B. den Wert eines *ebsvars* zu sichern und wieder zurückzuschreiben (Achtung: Es wird immer der evaluierte Wert des *ebsvars* gespeichert. Etwaige Ausdrücke werden nicht gespeichert.)

3.2.3. Hyperbolische and Area Funktionen

Die mathematischen Funktionen von EbsScript wurden um die hyperbolischen Funktionen

- *Function sinh(value:Real):Real;*
- *Function cosh(value:Real):Real;*
- *Function tanh(value:Real):Real;*

sowie deren Umkehrfunktionen (Areafunktionen)

- *Function asinh(value:Real):Real;*
- *Function acosh(value:Real):Real;*
- *Function atanh(value:Real):Real;*

erweitert.

3.2.4. „for-in“ Schleifen

Zum Iterieren über Arrays kann nun auch der „*for-in*“ Syntax verwendet werden:

for x in arr do loop-instruction;

Wobei *arr* ein Ausdruck vom Typ eines Arrays ist (d. h. Array mit fester Größe, Dynamisches Array oder Open-Array), und die Schleifenvariable *x* eine Variable ist, die den gleichen Typ hat, wie die Elemente des Arrays. Die Schleife wird dann für jedes Element im Array einmal durchlaufen (in aufsteigender Reihenfolge des Element-Index) und *x* ist jeweils eine **Kopie** des Array-Elements mit dem aktuellen Index.

Informell entspricht die „*for-in*“-Schleife folgendem Konstrukt:

```
for index:=low(arr) to high(arr) do
begin
        x:=arr[index];
        loop-instruction;
end;
```

Der Ausdruck *arr* wird **genau einmal** ausgewertet (auch wenn die Schleife mehrfach durchlaufen wird. Zur Erklärung: *arr* könnte z. B. ein Funktionsaufruf sein, der ein Array zurückgibt).

Die Schlüsselworte *continue* und *break* können genauso, wie in Index-basierten for-Schleifen verwendet werden.

Alternativ kann folgender Referenz-bindender Syntax verwendet werden:

*for **var** x in arr do loop-instruction;*

bzw.

*for **const** x in arr do loop-instruction;*

Die Schleifenvariable *x* muss in diesen Fällen ein neuer Variablenname im lokalen Block sein. Sie verhält sich wie ein *var* bzw. *const* Parameter einer Funktion und stellt eine Referenz auf das aktuellen Array-Element dar.
Im Falle von *var* kann das Array-Element durch Zuweisung an *x* geändert werden.

Bitte beachten Sie:
Falls *arr* innerhalb der Schleife manipuliert wird (mit Ausnahme der Änderung der Werte der Elemente des Arrays), dann ist die Ausführung eines **weiteren** Iterationsschritts oder das implizite Überprüfen der Schleifenabbruchbedingung **undefiniertes Verhalten**. (Z. B. ist ein Abspringen mit break zulässig, um undefiniertes Verhalten zu verhindern.)
Im Falle von *„for **var**“* und *„for **const**“* ist bereits der Zugriff auf die Schleifenvariable nach der Manipulation undefiniertes Verhalten.

Beispiele:

Programm	**Ausgabe**
var ar:array[1..3] of integer; *x:integer;* *begin* *ar:=[11,12,13];* *for x in ar do* *begin* *println(x);* *end;* *end;*	11 12 13
var x:integer; *begin* *for x in [11,12,13] do* *begin* *println(x);* *end;* *end;*	11 12 13
var ar:array[1..3] of integer; *begin* *ar:=[11,12,13];* *println(„numbers“);* *for var x in ar do* *begin* *println(x);* *x:=x*x;* *end;* *println(„squared in place“);* *for const x in ar do* *begin* *println(x);* *end;* *end;*	numbers 11 12 13 squared in place 121 144 169

3.2.5. Exceptions

EbsScript unterstützt nun das Auslösen und Abfangen von ***Exceptions***. Hierbei handelt es sich um eine Möglichkeit alternative Code-Pfade darzustellen.

Eine Exception wird mit dem Befehl ***raise*** ausgelöst und die Programmkontrolle geht danach zu dem ***except*** bzw. ***finally*** Handler des zuletzt betretenen ***try***-Blocks.

Um eine Exception abzufangen, muss die auslösende raise-Operation innerhalb eines ***try***-Blocks oder einer von dort aus aufgerufenen Funktion/Prozedur liegen (d.h. try-Blöcke schützen sämtlichen Code, der zwischen des Betretens und Verlassens ausgeführt wird.)
Z.B. erzeugt folgender Code

```
begin
        println(„before try block“);
        try
                println(„before raise“);
                raise Exception.Create(„Hello exceptional world!“);
                println(„after raise“);

        except on e:Exception do
                println(e.message);
        end;
        println(„after try block“);
end;
```

die Ausgabe:

```
before try block
before raise
Hello exceptional world!
after try block
```

Mit *raise* können ausschließlich Objekte geworfen werden, die zudem vom Typ *Exception* (in @System) oder davon abgeleitet sind.

Ein try-except Block hat folgende Syntax:

```
try
        // geschützter Code-Block
        Instructions; ...
except
        // Ausnahmebehandlungs Code-Block
        Instructions; ...
end;
```

Hierbei wird beim Auslösen einer Ausnahme diese vom *except*-Block abgefangen und dessen Instruktionen ausgeführt, danach läuft die Programmausführung normal weiter.

Alternativ kann auch folgender Syntax verwendet werden:

```
try
        // geschützter Code-Block
        Instructions; ...
except
        on e1:<Exception-Type 1> do
                Instruction;
        on e2:<Exception-Type 2>do
                Instruction;
        on e3:<Exception-Type ...>do
                Instruction;
else
        // Ausnahmebehandlungs Code-Block
        Instructions; ...
end;
```

Hierbei wird beim Auslösen einer Ausnahme von oben nach unten die angegebenen Exception-Typen mit der ausgelösten verglichen. Der **erste Typ** (und nur der erste), der die an die ausgelöste Exception gebunden werden kann (d. h. die Exception hat den gleichen oder einen abgeleiteten Typ), bindet die Exception an die angegebene Variable und führt die Instruktion aus (die darf natürlich auch ein *begin*-Instruktionen-*end*-Block sein).
Falls keine der angegeben Typen passt wird der *else*-Block ausgeführt.
Die angegebene Exception-Variablen können auch (inkl. des Doppelpunkts) jeweils weggelassen werden. In diesem Fall ist kein Zugriff auf weitere Informationen der Exception möglich.
Auch der *else*-Block ist optional.
Falls keine der angegeben Exception-Typen passt und kein *else*-Block vorhanden ist, wird die Exception an den nächsten übergeordneten *try-except/try-finally* Block weitergeleitet.

Ist kein solcher Block vorhanden wird ein ggfs. gesetzter ***unhandledExceptionFilter*** aufgerufen bzw. das Programm terminiert (eine sog. ***uncaught Exception***).
Während der Behandlung einer Exception-Behandlung kann die Prozedur *raise*; (ohne Argument!) aufgerufen werden, womit die gerade abgefangene Exception wieder aktiviert wird und zum nächsten übergeordneten *try-except/try-finally* Block (zur *raise*; Instruktion) weitergeleitet wird (Achtung: nachfolgende Handler des aktuellen *try*-Blocks werden NICHT mehr betrachtet! Exception-/*finally*-Handler gelten nicht als Teil des zugehörigen *try*-Blocks.)

Eine weitere Möglichkeit, auf Exceptions zu reagieren besteht mit dem *try-finally* Block:

```
try
        // geschützter Code-Block
        Instructions; ...
finally
        // Code-Block der IMMER ausgeführt wird
        Instructions; ...
end;
```

Hierbei wird der Code des *finally* Blocks immer ausgeführt. Falls zu Beginn des *finally* Blocks eine Exception aktiv war, dann wird diese deaktiviert und am Ende des Blocks automatisch wieder aktiviert. *try-finally* Blöcke eignen sich z. B., um sicherzustellen, dass externe Ressourcen wie Dateien korrekt geschlossen werden.

Ein *try*-Block mit *except* und *finally* Handler ist nicht möglich. Ein solches Verhalten kann aber leicht mittels verschachtelter *try*-Blöcke erreicht werden. Z. B.

```
try
        try
                // geschützter Code-Block
                Instructions; ...
        finally
                // Code-Block der IMMER ausgeführt wird
                Instructions; ...
        end;
except
        // Exception-Handler
end;
```

Das Verwenden von *Try-Except*- bzw. *Try-Finally*-Blöcken verursacht NUR beim Auslösen von Exceptions Laufzeitkosten. Wenn keine Exception in einem solchen Block ausgelöst wird, wird zur Laufzeit auch kein zusätzlicher Code ausgeführt.

Warnung: Exceptions sollten ausschließlich zur Ausnahmebehandlung in Fehlerfällen oder nicht erwarteten Zuständen verwendet werdet. Es wird davon abgeraten Exceptions zu verwenden, um z. B. aus tief verschachtelten Aufrufen gültige Ergebnisse zurückzugeben oder um Abkürzungen zu nehmen.

Achtung, ab hier wird es SEHR TECHNISCH:
Eine durch *raise* ausgelöste Exception gilt als ***aktiv***. Während der Behandlung in einem Exception-Handler bzw. *finally*-Block wird die (ggfs.) aktive Exception deaktiviert. Diese kann mittels *raise*; (ohne Argument) aktiviert werden oder sie wird automatisch am Ende des *finally*-Blocks wieder aktiv (falls es eine aktive Exception beim Eintritt gab).
Verlässt eine Exception bei der Suche nach einem Handler den Bereich einer Funktion / Prozedur, dann werden deren lokale Variablen freigegeben (man spricht von ***Stack-Unwinding***).

UND JETZT KOMMT DIE EINSCHRÄNKUNG: Das Programm wird augenblicklich **terminiert** falls

- bei der Ausführung eines Destruktors aufgrund eines Stack-Unwindings eine weitere Exception ausgelöst wird und diese den Destruktor verlässt
- oder bei der Ausführung eines *finally*-Handlers, bei dessen Beginn eine aktive Exception vorlag, eine weitere Exception ausgelöst wird und diese den Destruktor verlässt

Hierbei orientiert sich das Exception-Modell von EbsScript an dem von C++.

Anmerkung: Beim Ausführung von Destruktoren und *finally*-Handlern dürfen Exceptions ausgelöst werden, i. d. R. sollten diese aber innerhalb dieses Blocks abgefangen werden. Die in *@System* definierte Funktion

```
function uncaughtException:integer;
```

liefert die Anzahl der aktiven und in *finally*-Blöcken automatisch deaktivierten Exception, was erlaubt zu detektieren, ob eine neu ausgelöste Exception, die einen Destruktor oder *finally*-Block verlässt, zu einem Programmabbruch führt oder nicht.

Zusätzlich existiert noch die Funktion

```
function currentException:Exception;
```

welche die aktuell behandelte Exception zurückgibt oder *nil* falls keine solche existiert.

Hier ein Beispiel, welches mehrere aktive Exceptions gleichzeitig produziert:

```
procedure test(do_raise:integer);
begin
        if do_raise > 0 then
        begin
                raise Exception.Create(„exception raised in test“);
        end;
end;

var i:integer;
begin
        for i := 0 to 1 do
        begin
                // outer try-except
                try
                        // inner try-finally
                        try
                                println(„i=“, i);
                                test(i);
                        finally
                                println(„running finally“);
                                raise Exception.Create(„exception raised in finally“);
                        end;
                except
                        println(„exception caught“);
                end;
        end;
end.
```

Ausgabe:

i=0
running finally
exception caught
i=1
running finally

A runtime-error occured! Program execution stopped! Time: 2/6/2024 10:24:38 AM

Exception: Runtime ERROR, more than one active exception! Terminated due to raising of an exception during stack-unwinding (in „finally“ or during class destruction)

3.2.6. Debugging

Der Start-Debug-Button ermöglicht nun das zu startende EbsScript auszuwählen. Hierfür drücken Sie bitten auf den kleinen Dropdown-Pfeil am rechten Rand des Buttons und wählen das gewünschte Script aus. Wenn Sie nun den Start-Debug-Button drücken wird das Script gegebenenfalls geöffnet und gestartet (und nicht das im Editor aktive Script). Um wie bisher immer das im Editor aktive Script zu starten, wählen Sie den Eintrag „Aktives Script“ aus. Der Start-ohne-Debugger-Button startet das gleiche Script, wie der Start-Debug-Button. Die Einstellung, welches das Start-Script ist, wird mit dem Modell gespeichert.

Die Liste der Breakpunkte wird nun mit im Modell gespeichert.
Weiterhin gibt es eine neue Breakpunkt-Leiste. In dieser werden alle Breakpunkte des Modells angezeigt. Sie können dort aktiviert, deaktiviert und gelöscht werden. Bei einem Doppelklick auf einen Eintrag, wird das entsprechende EbsScript aktiviert (ggfs. geladen) und in die zugehörige Zeile gesprungen.

3.2.7. EbsOpen Interface-Unit

Es wurde die Interface-Unit @EbsOpen hinzugefügt. Diese ermöglicht den Zugriff auf die EbsOpen-Schnittstelle aus EbsScript heraus. Hierdurch können ggfs. Funktionen und Eigenschaften von Ebsilon bzw. des Models verwendet werden, welche nicht direkt aus EbsScript zugänglich sind.

3.3. UserProps-Dll

Siehe UserProps-Dll

4. Änderungen in den Ergebnissen

4.1. Bauteil 124: Ergebniswerte DP34ECO, DP34EVA, DP34SUP entfernt

Im Bauteil 124 wurden die Ergebniswerte DP34ECO, DP34EVA, DP34SUP nicht auf Basis der Flächenanteile im Strömung-Pfad 3->4 (heiße Seite) sondern auf Basis der Flächen im Strömung-Pfad 1->2 (kalte Seite) berechnet. Wegen mangelnder Relevanz dieser Ergebniswerte wurden sie im Release 17 entfernt.

5. Bekannte Fehler

5.1. Dokumentation

Leider konnten die neuen Features der Release 17 noch nicht vollständig in die Hilfe aufgenommen werden. Bis ein Patch für die Hilfe verfügbar ist, greifen Sie bitte auf diese Release-Notes zurück.

Es besteht auch die Möglichkeit, auf den im Internet verfügbaren aktuellen Stand der Online-Hilfe zuzugreifen, in dem man die Hilfeeinstellungen auf „im Browser starten“ umstellt.

Iqony Solutions GmbH
Wetzbach 35
64673 Zwingenberg
T +49 6251 1059-0
info@ebsilon.com
www.ebsilon.com