deeco

– Benutzerhandbuch –

Thomas Bruckner
Institut für Theoretische Physik der Universität Würzburg

19. Dezember 1997

Kapitel 1

Allgemeines

1.1 Kommandozeilenargumente

Das Programm deeco wird durch folgende Eingabe auf Kommandoebene gestartet:

deeco -s -t -d -n

Die Argumente -s -t -d -n sind optional und folgendermaßen definiert:

-s : "Silent Mode"; d.h. keine Bildschirmausgabe
-t : "Test Mode"; detaillierte Informationen zum Programmablauf
-l : "Log Mode"; erzeugt ein "Log-File" mit Informationen zum Programmablauf
-d : Projekt Pfad (*projectPath*, vollständiger Directory Name); Angabe des Namens direkt hinter "-d" ohne Leerzeichen
-n : Projekt Name; Angabe des Namens direkt hinter "-n" ohne Leerzeichen

1.2 Log-File

Informationen zum Programmablauf werden auf dem Bildschirm (Abschalten dieser Option mit "-s") und bei Angabe des Kommandozeilenarguments "-l" in ein File mit Namen "deeco.log" ausgegeben.

1.3 Fehlerbehandlung

Das Programm deeco besitzt ein File mit Namen "deeco.msg", das Fehlermeldungen (Nummer kleiner als 500), Warnmeldungen (Nummer zwischen 500 und 1000) und Informationen (Nummer größer als 1000) enthält, die das Programm bei Bedarf anzeigt.

1.4 Messung der Rechenzeit

Die benötigte Rechenzeit wird gemessen, am Bildschirm angezeigt und ins File “deeco.log“ geschrieben. “User time“ ist die CPU Zeit in Sekunden, die auf die Verarbeitung des Programms deeco entfallen ist; “Real time“ ist äquivalent zur “wall-clock-time“ Differenz zwischen Programmstart und -ende.

Kapitel 2

Dateien

Die Dateneingabe und -ausgabe erfolgt in *deeco* mit Hilfe folgender ASCII-Textdateien, die anhand ihrer Endung unterschieden werden. Dateien zur Dateneingabe müssen (sofern sie szenarienunabhängig sind und auch wenn sie leer sind) exisitieren. Mit einem "%" - Zeichen beginnende Zeilen stellen Kommentarzeilen dar. Sie werden beim Einlesen von Daten vollständig ignoriert. Treten in Ausgabefiles "?" auf, so bedeutet dies, das die entsprechenden Werte nicht berechnet wurden bzw. nicht sinnvoll sind. Sie deuten nicht auf ein fehlerhaftes Verhalten des Algorithmus hin.

2.1 Szenarien–Dateien

2.1.1 Szenarien–Definition: *projectPath*/*projectName*.sd

Alle Szenarien, die während eines Programmlaufes bearbeitet werden sollen, werden in der Datei *projectName*.sd (sd = scenario definition) mit folgendem Zeilenformat definiert:

scenId , *scenSelected* , *com*

Dabei steht *scenId* für die Szenarienkurzbezeichnung, *scenSelected* ist $\neq 0$, falls das Szenario bearbeitet werden soll (0 sonst). Am Ende jeder Zeile darf ein Kommentar *com* angegeben sein. Die Angabe aller Kommata ist verpflichtend; sie müssen von den anderen Einträgen durch mindestens ein Leerzeichen getrennt sein. (Soweit im folgenden bei der Angabe von Zeilenformaten Kommata auftreten, ist diese Bedingung stets zu beachten!)

2.1.2 Szenarien–Eingabedaten: *projectPath*/*scenId*.siv

Jedes Szenario, das bearbeitet werden soll, wird durch ein Szenarien-Eingabedatenfile[1] *scenId*.siv (siv = scenario input value) näher beschrieben, das sich im Directory *projectPath* befindet: In diesem File wird angegeben (programminterne Vorgabewerte in eckigen Klammern):

resPath = Directory, in dem die szenarienspezifischen Ergebnisfiles abgelegt werden [*projectPath*].
scenPath = Directory, in dem sich die szenarienspezifischen Kontrollfiles befinden (diese überschreiben[2] die Default-Kontrollfiles im Directory *projectPath*) [*projectPath*].
Die Umweltdatenzeitreihen sind im File *scenInTsFileName* [*projectPath*/*projectName*.sit] und die prozeßspezifischen Datenzeitreihen in *procInTsFileName* [*projectPath*/*projectName*.pit] abgelegt (Pfade sind vollständig anzugeben!).

Pro Szenario kann ein Parameterwert mit der Bezeichnung *scanParaName* des Prozesses *scanParaProcName* innerhalb des Bereiches *scanParaBegin* [0] bis *scanParaEnd* [0] in Schritten von *scanParaStep* [1] durchgefahren ("gescannt") werden.[3]
Soll eine dynamische Optimierung[4] durchgeführt werden, so ist *dynFlag* = 1 zu setzen. Bei quasidynamischer Optimierung ist *dynFlag* = 0 zu verwenden [Vorgabe: 0].
Um ein extensives Ergebnisfile, das die verallgemeinerten Kosten zeitaufgelöst darstellt, zu erhalten, wählt man *tsOutFlag* =1 [0]. Um eine Speicherplatzüberbeanspruchung zu vermeiden, schließt ein Durchscannen von Parameterwerten die Erzeugung von extensiven Ergebnisfiles aus. *allResFlag* =1 [0] bedeutet, daß neben den Mittelwerten der Optimierungsvariablen (Energieströme) auch andere Ergebnisdaten (Mittelwerte der Schlupfvariablen, der Schattenpreise, der Koeffizienten der Nebenbedingungen bzw. der Zielfunktion, der Inhomogenitäten der Nebenbedingungen und der verallgemeinerten Kosten sowie der Zustandsgrößen) angezeigt werden sollen. *intLength* gibt die Zeitintervallänge in s [3600s] und *intNumber* die Zahl der Zeitintervalle an [8760].[5] Die Faktoren (Gewichtung x Normierung) mit denen die einzelnen generalisierten Kosten in die Zielfunktion eingehen, stellen die Werte eines Vektors *goalWeight* dar, dessen Komponenten[6]

durch Angabe von *generalCostComp* unterschieden werden [Vorgabe: goalWeight[E]=1;

[1] Die Begriffe Eingabe und Ausgabe beziehen sich in diesem Benutzerhandbuch ausschließlich auf das Einlesen von Daten bzw. die Ausgabe von Daten durch das Programm. Sie dürfen nicht mit den gleichlautenden Begriffen der Modellbeschreibung von deeco verwechselt werden!

[2] Dabei bedeutet "überschreiben" jetzt und im folgenden, daß die in den szenarienspezifischen Kontrollfiles angegebenen Zeilen gelesen und somit szenarienunabhängige Daten, die bereits eingelesen worden sind, modifiziert werden. Es bedeutet nicht, daß die gesamten szenarienunabhängigen Files überschrieben werden!

[3] Eingabeparameter werden i.allg. von deeco auf Plausibilität geprüft. Diese Prüfung entfällt beim Durchscannen.

[4] Die gegenwärtige Version von deeco erlaubt noch keine dynamische Optimierung, d.h. es ist stets *dynFlag* = 0 zu setzen.

[5] Als Startzeitpunkt sollte immer Montag, 0 Uhr gewählt werden.

[6] Die Einheit der Zielfunktionskomponente Primärenergie ist [J/s], die der Emissionen [kg/s] und die der monetären Kosten [DM/s]. Bei linearer Zielgewichtung ist es deshalb notwendig, den Faktoren die entsprechenden Einheiten zuzuordnen (Normierung)!

Primärenergieoptimierung].
Am Ende jeder Zeile ist ein Kommentar *com* erlaubt.

Allgemeines Format eines .siv - Files :

PR , *resPath* ,	*com*
PS , *scenPath* ,	*com*
TS , *scenInTsFileName* ,	*com*
TP , *procInTsFileName* ,	*com*
S , *scanParaProcName* , *scanParaName* , *scanParaBegin* , *scanParaEnd* , *scanParaStep* ,	*com*
D , *dynFlag* ,	*com*
E , *tsOutFlag* ,	*com*
A , *allResFlag* ,	*com*
I , *intLength* , *intNumber* ,	*com*
G , *generalCostComp1* , *goalWeight[generalCostComp1]* ,	*com*
G , *generalCostComp2* , *goalWeight[generalCostComp2]* ,	*com*
G , *generalCostComp3* , *goalWeight[generalCostComp3]* ,	*com*
G , ...	

Beispiel: BSP.siv

PR , /users/thomas/deeco/projectname/BSP	
PS , /users/thomas/deeco/projectname/BSP	
TS , /users/thomas/deeco/projectname/BSP/BSP.sit	
TP , /users/thomas/deeco/projectname/BSP/BSP.pit	
S , solar , A_c , 0 , 100 , 1	
D , 0	quasi-dynamische Optimierung
E , 0	keine extensive Datenausgabe
I , 3600 , 8760 ,	Zeitintervallinformationen
G , M , 1 ,	monetäre Kosten
G , SO2 , 0.25 ,	SO2-Emissionen
G , NOX , 0.25 ,	NOX-Emissionen
G , CO2 , 0.25 ,	CO2-Emissionen
G , E , 0.25 ,	Primärenergieeinsatz

Vorgabewerte werden übernommen indem entsprechende Zeilen (inklusive der Kennbuchstaben R,P,S, ...) nicht angegeben werden.

2.1.3 Szenarien–Zeitreihen–Eingabedaten: *scenInTsFileName*.sit

In diesem File sind alle Umweltdatenzeitreihen und die Zeitreihen der Anlagenauslastungsfaktoren abgelegt (sit = scenario input time series) .

Allgemeines Zeilenformat:

intCount , *inputId* , *inputVal* , *com*

Dabei ist *intCount* die Zeitintervallnummer, die in nicht abfallender Reihenfolge angegeben werden muß. *inputId* ist die Abkürzung der Eingabedatenbezeichnung und *inputVal* stellt den entsprechenden Eingabewert dar.

Definierte Bezeichnungen:

Kurzbezeichnung	Bedeutung
T	Außentemperatur in [K]
I	Solare Einstrahlung (auf 45 Grad geneigte und nach Süden ausgerichtete Fläche) in [W/qm]
W	Windgeschwindigkeit in [m/s]
G1	Auslastungsfaktor Nr. 1 in [1]
Gi	Auslastungsfaktor Nr. i in [1]
h	Tageszeit, h$\in [0, 24]$
	Beispiel: h =2 ist der Zeitraum von t = 2 bis t = 2 + intLength
d	Wochenzeit, d$\in [0, 6]$
	Beispiel: d = 0 ist Montag.
s	Jahreszeit, s$\in [0, 3]$
	Beispiel: s = 0 ist Sommer, s = 3 ist Frühling.

Beispiel: BSP.sit

1	,	T	,	273.15
1	,	I	,	300
1	,	W	,	3
1	,	G1	,	0.75
1	,	G2	,	0.44
2	,	T	,	293.15
2	,	I	,	320
2	,	W	,	3.7
2	,	G1	,	0.9
2	,	G2	,	0.5
...	,	...	,	...

2.1.4 Szenarien–Ausgabedaten: *resPath/scenId*.sov

Dieses File enthält im wesentlichen die über den Optimierungszeitraum gebildeten Mittelwerte der generalisierten Kosten sowie der Umweltdaten und Anlagenauslastungsfaktoren (sov = scenario output value).

Allgemeines Zeilenformat:

(*scanParaVal* ,) *outputId* , *Mean* , *Dev* , *Min* , *Max*

mit *outputId* = Kurzbezeichnung der angegeben Kostenart, der Komponente des Umweltdatenvektors bzw. die Bezeichnung des Anlagenauslastungsfaktors, *Mean* = Stichproben-Mittelwert, *Dev* = Stichproben-Varianz, *Min* = Minimum und *Max* = Maximum. Wird aufgrund entsprechender Angaben in der Datei *projectPath/scenId*.siv der Parameterwert *scanParaName* des Prozesses *scanParaProcName* durchgescannt, so wird dieser Parameterwert *scanParaVal* am Anfang jeder Zeile angegeben.

2.1.5 Szenarien–Zeitreihen–Ausgabedaten: *resPath/scenId*.sot

Sofern im Szenario-Eingabedatenfile eine extensive Ergebnisausgabe gewünscht wird (Zeile: "E , 1"), werden in diesem File für alle Zeitintervalle die generalisierten Kosten wiedergegeben (sot = scenario output time series).

Allgemeines Zeilenformat:

intCount , *outputId* , *outputVal* , *com*

outputId ist die Abkürzung der Ausgabedatenbezeichnung (= Kurzbezeichnung der generalisierten Kosten) und *outputVal* stellt den entsprechenden Ausgabewert dar.

Beispiel: BSP.sot

1	,	M	,	1	monetäre Kosten
1	,	SO2	,	2	
1	,	NOX	,	3	
1	,	CO2	,	4	
1	,	E	,	5	Primärenergie
2	,	M	,	1.5	monetäre Kosten
2	,	SO2	,	2.5	
2	,	NOX	,	3.5	
2	,	CO2	,	4.6	
2	,	E	,	5.6	Primärenergie
...	,	...	,	...	

2.2 Prozeß–Dateien

2.2.1 Prozeß–Definition: *projectPath/projectName*.pd

Alle Prozesse, die während eines Programmlaufes betrachtet werden können, werden in der Datei *projectName*.pd (pd = process definition) mit folgendem Zeilenformat definiert:

procId , *procType* , *select* , *com*

Dabei steht *procId* für die Prozeßkurzbezeichnung, *procType* für die Prozeßtypkurzbezeichnung (vordefiniert), *select* ist $\neq 0$, falls der Prozeß zugelassen ist (0 sonst). Am Ende jeder Zeile darf ein Kommentar *com* angegeben sein. Das szenarienunabhängige Prozeß-Definitionsfile *projectPath/projectName*.pd kann durch ein szenarienspezifisches File *scenPath/scenId.pd* gleichen Formats überschrieben werden.

2.2.2 Prozeß–Eingabedaten (standard): *projectPath/ projectName*.piv

Die Parameterstandardwerte jedes (zugelassenen) Prozesses sind in der Datei *projectPath/projectName*.piv (piv = process input value) wiedergegeben. Jede Zeile dieser Datei bezitzt folgendes Format:

procId , *inputId* , *inputValue*, *com*

Dabei steht *procId* für die Prozeßkurzbezeichnung, *inputId* für die Kurzbezeichnung des betrachteten Prozeßparameters und *inputValue* für den Wert des Parameters. Die Bezeichnungen *inputId* sind prozeßtypspezifisch vordefiniert.

Das szenarienunabhängige Prozeß-Eingabedatenfile *projectPath/projectName*.piv kann durch ein szenarienspezifisches File *scenPath/scenId*.piv gleichen Formats modifiziert werden. Die Werte in diesem File überschreiben die entsprechenden Standardwerte in *projectPath/projectName*.piv. Sie sind nur anzugeben, wenn die Standardwerte geändert werden sollen.

2.2.3 Prozeß–Zeitreihen–Eingabedaten: *procInTsFileName*.pit

In diesem File sind die prozeßspezifischen Datenzeitreihen (z.B. Energiebedarfsdaten) abgelegt (pit = process input time series) . Es besitzt Datenbankformat; alle Zeileneinträge werden durch Kommata getrennt. Auf eine nicht-abfallende Reihenfolge der Zeitintervallnummern ist zu achten.

Allgemeines Zeilenformat:

intCount , *procId* , *inputId* , *inputVal* , *com*

2.2.4 Prozeß–Ausgabedaten: *resPath/ scenId*.pov

Dieses File enthält die über den Optimierungszeitraum gebildeten Mittelwerte der prozeßinternen Zustandsgrößen und Parameter (Koeffizienten der Nebenbedingungen, "RHS"'s der Nebenbedingungen, Koeffizienten der Zielfunktion, etc) (pov = process output value).

Allgemeines Zeilenformat:

(*scanParaVal* ,) *procId* , *outputId* , *Mean* , *Dev* , *Min*, *Max*

mit *outputId* = Kurzbezeichnung der auszugebenden Größe, *Mean* = Stichproben-Mittelwert, *Dev* = Stichproben-Varianz, *Min* = Minimum und *Max* = Maximum. Wird aufgrund entsprechender Angaben in der Datei *projectPath/scenId*.siv der Parameterwert *scanParaName* des Prozesses *scanParaProcName* durchgescannt, so wird dieser Parameterwert *scanParaVal* am Anfang jeder Zeile angegeben.

2.2.5 Prozeß–Zeitreihen–Ausgabedaten: *resPath/scenId*.pot

Sofern im Szenario-Eingabedatenfile eine extensive Ergebnisausgabe gewünscht wird (Zeile: "E , 1"), werden in diesem File für alle Zeitintervalle die prozeßinternen Zustandsgrößen (z.B. die Speichertemperaturen) wiedergegeben (pot = process output time series). Es besitzt Datenbankformat; alle Zeileneinträge werden durch Kommata getrennt.

Allgemeines Zeilenformat:

intCount , *procId* , *outputId* , *outputVal* , *com*

outputId ist die Abkürzung der Ausgabedatenbezeichnung (= Kurzbezeichnung der Zustandsvariable) und *outputVal* stellt den entsprechenden Ausgabewert dar.

2.3 Bilanzpunkt–Dateien

2.3.1 Bilanzpunkt–Definition: *projectPath/projectName*.bd

Alle (Energie-)bilanzpunkte, die während eines Programmlaufes betrachtet werden können, werden in der Datei *projectName*.bd (bd = balance definition) mit folgendem Zeilenformat definiert:

balanId , *energyFlowType* , *linkType* , *select* , *com*

Dabei steht *balanId* für die Bilanzpunktkurzbezeichnung, *energyFlowType* für die Kurzbezeichnung der Art (El,Mech,H,...) und *linkType* für die Kurzbezeichnung der Typen (0, 1, 2, 3, oder 4 gemäß Typeneinteilung) der Energieströme, die am Bilanzpunkt bilanziert werden (vordefiniert); *select* ist $\neq 0$, falls der Bilanzpunkt zugelassen ist (0 sonst). Am

Ende jeder Zeile darf ein Kommentar *com* angegeben sein.

Das szenarienunabhängige Bilanzpunkt-Definitionsfile *projectPath*/*projectName*.bd kann durch ein szenarienspezifisches File *scenPath*/*scenId*.bd gleichen Formats überschrieben werden.

2.4 Energiestrom–Dateien

2.4.1 Energiestrom–Definition: *projectPath*/*projectName*.cd

Alle Energieströme, die Prozesse und Bilanzpunkte miteinander verbinden und die während eines Programmlaufes betrachtet werden können, werden in der Datei *projectName*.cd (cd = connection definition) mit folgendem Zeilenform definiert:

connectId , *procId* , *energyFlowNumber* , *balanId* , *energyFlowType*, *linkType* , *direct* , *com*

Dabei steht *connectId* für die Kurzbezeichnung des betrachteten Energiestroms, der den Prozeß *procId* am (energetischen) Ausgang (Eingang) mit der Nummer (als Integer einzugeben) *energyFlowNumber* verläßt (in diesen eintritt) und mit dem Bilanzpunkt *balanId* verbindet. *energyFlowType* steht für die Kurzbezeichnung der Art und *linkType* für den Typ (gemäß Typeneinteilung) des Energiestromes (vordefiniert) und *direct* ist 1, falls der Energiestrom vom Prozeß zum Bilanzpunkt fließt, -1 im umgekehrten Fall sowie 0, falls die Verknüpfung nicht erlaubt sein soll. Am Ende jeder Zeile darf ein Kommentar *com* angegeben sein.

Das szenarienunabhängige Energiestrom-Definitionsfile *projectPath*/*projectName*.cd kann durch ein szenarienspezifisches File *scenPath*/*scenId*.cd gleichen Formats überschrieben werden.

2.5 (Prozeß-) Aggregat-Dateien

2.5.1 (Prozeß-) Aggregat-Definition: *projectPath*/*projectName*.ad

Alle Prozeßaggregate, die während eines Programmlaufes betrachtet werden können, werden in der Datei *projectName*.ad (ad = (process) aggregate definition) mit folgendem Zeilenformat definiert:

aggId , *select* , *com*

Dabei steht *aggId* für die (Prozeß-) Aggregatkurzbezeichnung, *select* ist $\neq 0$, falls das (Prozeß-) Aggregat zugelassen ist (0 sonst). Möchte man darüber hinaus eine Maximalleistung für das Aggregat angeben, so ist *select* = 1 zu wählen. Am Ende jeder Zeile

darf ein Kommentar *com* angegeben sein. Das szenarienunabhängige (Prozeß-) Aggregat–Definitionsfile *projectPath/projectName*.ad kann durch ein szenarienspezifisches File *scenPath/scenId*.ad gleichen Formats überschrieben werden.

2.5.2 Prozeß–Aggregat–Eingabedaten (standard): *projectPath/projectName*.aiv

Die Parameterstandardwerte jedes (zugelassenen) (Prozeß-) Aggregates sind in der Datei *projectPath/projectName*.aiv (aiv = (process) aggregate input value) wiedergegeben. Jede Zeile dieser Datei bezitzt folgendes Format:

aggId , *procId* , *inputValue*, *com*

Dabei steht *procId* für die Prozeßkurzbezeichnung eines Prozesses, der am Aggregat beteiligt ist, *aggId* für die Kurzbezeichnung des betrachteten (Prozeß-) Aggregates und *inputValue* für den Wert des Parameters (=Maximalleistung des Prozeßaggregates). Diese Maximalleistung beschränkt die Energieströme der am Aggregate beteiligten Prozesse, die zur Bestimmung der fixen Kosten des Prozesses herangezogen werden; sie ist für alle Prozesse eines Aggegates identisch (aber dennoch jeweils anzugeben). Die Zeile ist auch anzugeben, wenn keine Maximalleistung vorgegeben werden soll, um die Verknüpfung von Aggregat und Prozeß zu gewährleisten. Als "Maximalleistung" ist in diesem Falle -1 zu verwenden!

Das szenarienunabhängige (Prozeß-) Aggregat–Eingabedatenfile *projectPath/projectName*.aiv kann durch ein szenarienspezifisches File *scenPath/scenId*.aiv gleichen Formats modifiziert werden. Die Werte in diesem File überschreiben die entsprechenden Standardwerte in *projectPath/projectName*.aiv. Sie sind nur anzugeben, wenn die Standardwerte geändert werden sollen.

2.5.3 Prozeß–Aggregat–Ausgabedaten: *resPath/ scenId*.aov

Dieses File enthält die über den Optimierungszeitraum gebildeten Mittelwerte der (Prozeß-) Aggregatleistung (aov = (process) aggregate output value).

Allgemeine Zeilenformat:

(*scanParaVal* ,) *aggId* , *Mean* , *Dev* , *Min*, *Max*

mit *aggId* = Kurzbezeichnung des Aggregates, *Mean* = Stichproben-Mittelwert, *Dev* = Stichproben-Varianz, *Min* = Minimum und *Max* = Maximum. Wird aufgrund entsprechender Angaben in der Datei *projectPath/scenId*.siv der Parameterwert *scanParaName* des Prozesses *scanParaProcName* durchgescannt, so wird dieser Parameterwert *scanParaVal* am Anfang jeder Zeile angegeben.

2.6 Zusammenfassung der Input-Output-Daten

Im folgenden werden die Endungen der verschiedenen Datenfiles übersichtlich zusammengestellt. Mit Ausnahme von .siv - Files besitzen alle Files ASCII-delimited Datenbankformat.

Objekte	Definitionen	Kenngrößen-Input	Zeitreihen-Input	Kenngrößen-Output	Zeitreihen-Output
Szenarien	.sd	.siv	.sit	.sov	.sot
Prozesse	.pd	.piv	.pit	.pov	.pot
Bilanzen	.bd	-	-	-	-
Energieströme	.cd	-	-	-	-
Aggregate	.ad	.aiv	-	.aov	-

Hinweis: deeco kann erst dann (fehlerfrei) gestartet werden, wenn ein vollständiger Satz szenarienunspezifischer Kontrollfiles existiert, d.h. die .sd,.pd,.bd,.cd,.ad,.siv,.piv,.aiv,.sit und .pit Files müssen vorhanden sein (unabhängig davon, ob sie Werte enthalten oder nicht)! Die Kurzbezeichungen in den .sd, .pd, .bd und .cd Files sollten nicht mehr als 5 Buchstaben enthalten!

Kapitel 3

Fehlermeldungen

1 Programming failure:
10 ERRORS OCCURED !!!!!!!!!!!!!!
11 Freestore exceeded :
12 Can't read file :
13 Can't write in file :
14 Run stopped at interval :
15 Missing TS-Data for interval :
16 Error reading file :
17 Error writing in file :
18 Something wrong in interval :
19 Scanning parameter not used :
20 Scanning and creating extensive result files is incompatible:
21 Use of not-fully defined virtual function:
22 Reading of expected "," impossible:
23 Can't clear file :
24 Can't insert process in graph:
"insert" wird angewendet auf einen nicht-existierenden Prozeß;
genauer auf einen nicht-existierenden Pointer auf einen Prozeß
(z.B. weil kein Platz mehr vorhanden ist, einen solchen Pointer dynamisch zu erzeugen).
25 Can't find process type :
26 Can't insert balance in graph:
"insert" wird angewendet auf einen nicht-existierenden Bilanzpunkt;
genauer auf einen nicht-existierenden Pointer auf einen Bilanzpunkt
(z.B. weil kein Platz mehr vorhanden ist, einen solchen Pointer dynamisch zu erzeugen).
27 Can't find balance type :
28 Can't insert flow in graph :
"insert" wird angewendet auf einen nicht-existierenden Energiestrom;
genauer auf einen nicht-existierenden Pointer auf einen Energiestrom
(z.B. weil kein Platz mehr vorhanden ist, einen solchen Pointer dynamisch zu erzeugen).
29 Can't find balance of connect:
30 Can't find process of connect:

31 Cycle found including process:
32 No connection found :
33 More than 1 connection found:
34 More than 1 process connected:
35 Edge not connected to a process entry or exit:
36 Explicit connection with “El“:
37 Wrong “linkType“ of connection:
38 Edges of the cycle :
39 Energy flow type not compatible with balance type:
40 More than one entering energy flow not allowed for:
41 More than one leaving energy flow not allowed for:
42 More than one leaving or entering energy flow not allowed for:
43 No entering or no leaving energy flow detected for:
44 Balance type is not a legal one:
45 (Simplex), Unbounded objective function detected at interval:
46 (Simplex), No solutions satisfy constraints given in interval:
47 Process of process aggregate not found in process list:
48 Some objectiv function coefficients are missing:
Es gibt Prozesse, bei denen die generalisierten Kosten
nicht für alle Kostenkomponenten berechenbar sind,
die im File .siv angegeben sind.
49 No constraint or only one energyflow detected
Es sollten wenigstens eine Nebenbedingung und zwei Energieströme
angegeben sein, damit Simplex sinnvoll funktioniert.
50 Math-library function error:
“DOMAIN“: Bereichsfehler; z.B. sqrt(-2),
“SINGULARITY“: Singularität (starke Variation des Funktionswertes
bei schwacher Variation des Arguments); z.B. tan(pi/2),
“OVERFLOW“: Bereichsüberschreitung; z.B. 1/0,
“TOTAL LOSS OF SIGNIFICANCE“ und “PARTIAL LOSS OF SIGNIFICANCE“:
Probleme mit den signifikanten Stellen; z.B. cos(1e100),
51 Floating point error (invalid operation, divide-by-zero or overflow)
Verwende “xdb“ und den erzeugten core, um den Typ festzustellen und
um die Stelle des Fehlers ausfindig zu machen.
52 Freestore exceeded
Übersetze mit _TEST_ in “TestFlag.h“ und starte das Programm erneut.
Wenn keine Fehlermeldung mehr vor dem Absturz generierte wird, findet
der Fehler des “new“-Aufrufs vermutlich in einer Bibliotheksfunktion
z.B. in Map< , > etc. statt!
53 Negative inequalities are not allowed for process aggregates:
54 Missing attribut detected :
55 Missing process input parameter:
56 Some input parameters are zero or negativ :
57 Missing environment data :
58 Missing process input time series data:
59 Process aggregation forbidden:
60 Return temperature is larger than flow temperature:
61 Inconsistent input parameter:

62 Environment data out of range:
63 Return temperature is not determined (heat flow usage):
64 Routine should not be used:
65 Unexpected negativ or zero value during run:
67 Input parameter is negativ:
68 Time series input data is negativ:
69 Attribut is zero or negativ:
70 No solution found for:
501 Can't run scenario :
502 Unexpected end of scenario :
503 Standard version used of (virtual function not overwritten?):
In diesem Falle werden nur die zeitunabhängigen Prozeßparameter verwendet!
504 Math-library function problem:
505 Input parameters are zero or negativ (a floating point error may occur):
506 Divide by a zero approaching value:
507 Tried to take log of a zero approaching value:
1000 :
1001 Constructor :
1002 Destructor :
1003 User time :
1004 Real time :
1005 End of
1006 End of scenario :
1009 Definition values :
1010 Input parameter values :
1011 Input file read :
1012 Output file written :
1013 Old input overwritten with :
1014 Vertices of the graph :
1015 Edges of the graph :
1016 Sorted process list :
1017 You interrupted the program. If you think the program is hanging in an endless loop, see the user manual.
Versuche die Werte EPSOPT (in deecoApp.h), EPS (in simp2.c and simplx.c) und REL_EPS (in Data.c), die zur Auswertung von a == 0 bei floats verwendet werden, zu verändern.
S.a. Press: Numerical Recipes,p.339 (1988). und
Künzi, Mathematische Optimierung, Teubner, Stuttgart (1966), S.80;
1018 Interval number:

Kapitel 4

Hinweise

4.1 Schattenpreise

Die mit Hilfe von deeco bestimmbaren Schattenpreise geben an, um wieviel der Optimalwert der Zielfunktion zunimmt, wenn eine $\geq$ Nebenbedingungs-RHS ("Rechte Seite") um eine Einheit erhöht bzw. eine $\leq$ Nebenbedingungs-RHS um eine Einheit erniedrigt wird. Schattenpreise lassen sich nur dann sinnvoll interpretieren, wenn die Nebenbedingung im ganzen Zeitraum immer eine $\geq$ bzw. $\leq$ Bedingung ist!

4.2 Mittelwerte

Die in den Ausgabefiles angegebenen Mittelwerte der Nebenbedingungs- und Zielfunktionskoeffizienten geben die auslastungsabhängigen Mittelwerte (z.B. Wirkungsgrade) an. Will man auslastungsunabhängige Mittelwerte, so muß man durch geeignete Szenariendefinition Bedingungen herstellen, die zu einer Norm-Auslastung führen (z.B. bestimmte Techniken nur alleine einsetzen).

4.3 Zeitunabhängige Parameter

Zeitabhängige Attribute (z.B. Temperaturen), die einem Energiestrom zugeordnet sind, werden von Prozeß zu Prozeß weitergegeben. Für zeitunabhängige Parameter eines Energiestromes (z.B. Wärmekapazitäten eines Enthalpiestromes) erfolgt diese Weitergabe nicht. Die auftretenden Parameter stellen Prozeßparameter der Prozesse dar, die durch den Energiestrom verbunden sind, wobei für jeden Prozeß ein eigener Parameter (z.B. Wärmekapazität) angegeben wird. Der Benutzer muß selbst auf die Konsistenz dieser Angaben achten!

4.4 Zielgewichtung

Bei Verwendung einer Zielfunktion, die neben dem Primärenergieeinsatz weitere Zielfunktionskomponenten enthält, ist zu beachten, daß

1. nicht überprüft wird, ob für alle Module tatsächlich die entsprechenden spezifischen Kosten oder Emissionsfaktoren angegeben sind.

2. nicht überprüft wird, ob für jedes Modul Informationen über die fixen (verallgemeinerten) Kosten vorhanden sind.

Somit besteht die Gefahr, daß bei unvorsichtiger Vorgehensweise gewisse Kostenanteile verloren gehen. Bei den spezifischen fixen (verallgemeinerten) Kosten ist darüber hinaus zu beachten, daß sich diese nur i.allg. auf (einen!) maximalen Energiestrom des Prozesses beziehen.[1] Es gibt Prozeßmodule bei denen sich diese Kosten auf andere Größen (z.B. Speichervolumen, Kollektorfläche) beziehen. Dies ist aber in der Modulbeschreibung explizit angegeben.

[1] Auf welchen Energiestrom sich die spezifischen fixen (verallgemeinerten) Kosten beziehen, wird durch die Programmroutine "showPower" des entsprechenden Programmoduls angegeben.

Kapitel 5

Bekannte Bugs

1. Die Verwendung von Tabulatoren in den Eingabefiles ist i.allg. hilfreich und problemlos möglich. Im .siv-File sollte aber darauf verzichtet werden (Fehlermeldung 48, 49 oder Verwendung falscher Zielfunktion möglich! Weitere Kontrolle erforderlich:)

2. Bei mehr als einem Gewicht in der Zielfunktion kann es zur Fehlermeldung 48 bzw. 49 kommen. Lineare Zielgewichtung bis auf weiteres nicht verwenden (d.h. immer nur eine "1" als Gewicht einsetzen). (Weitere Kontrolle erforderlich!)

3. Die bei den Ausgabedaten häufig mit angegebene Standardabweichung ist numerisch sehr sensitiv, da sie aus der Differenz zweier großer Zahlen berechnet wird. (Auf die Ausgabe dieser Standardabweichung sollte in Zukunft verzichtet werden.)