theseus-fe.com

Lehrstuhl für Computation in Engineering

TECHNISCHE UNIVERSITÄT MÜNCHEN

Prof. Dr. rer. nat. Ernst Rank

Diplomarbeit

Parametrisches Modell zur interaktiven Simulation menschlicher Thermoregulation

In Kooperation mit P+Z Engineering GmbH, München

Jérôme Frisch

Betreuer	:	Dr.-Ing. Christoph van Treeck
Eingereicht	:	Juli 2008

Zusammenfassung

Die Betrachtung des menschlichen Temperatur- und thermischen Behaglichkeitsempfindens spielt in einer Vielzahl von Anwendungsfeldern eine bedeutende Rolle. Dabei genügt es häufig nicht, stationär die Behaglichkeit für den Körper als Ganzes zu bewerten, wie dies beim Modell von Fanger [17] der Fall ist. Beispielsweise sind in Fahrzeuginnenräumen lokale Effekte von Bedeutung, etwa die Absorption von Solarstrahlung auf der Haut bei gleichzeitiger Wirkung der Klimaanlage. Hierbei ist zwischen der sich auf der Haut einstellenden, resultierenden Oberflächentemperatur (RST), der (in Abhängigkeit vom Gesamtzustand des Körpers) dort lokal empfundenen Temperatur und letztendlich der thermischen Behaglichkeit zu unterscheiden.

Als Baustein einer Reihe von Arbeiten in einem übergeordneten Projekt ist es das Ziel dieser Arbeit, das thermoregulatorische Menschmodell von Fiala [18] zur Bestimmung von RST Werten mit einer interaktiven Simulation zu koppeln. Das Fiala-Modell wird dabei in Form des Produktes THESEUS-FE [12] von P+Z Engineering bereitgestellt. Zur Anbindung an THESEUS-FE ist eine geeignete Schnittstelle zu den Applikationen zu schaffen. Die reine Mensch-Geometrie wird dabei durch ein parametrisches Geometriemodell (Diplomarbeit von Yue [36]) bereitgestellt.

Die Funktionsweise des transienten Modells soll anhand der interaktiven Simulation einer sehr einfachen Szene („Mensch in rechteckigem Raum“) demonstriert werden. Hierfür soll eine „Virtuelle Klimakammer“ entworfen werden, in der vordefinierte Sichtfaktoren und Wärmeübergangskoeffizienten verwendet werden sollen; Randbedingungen werden gegeben.

Erklärung

Hiermit versichere ich, dass ich diese Diplomarbeit selbstständig verfasst, und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Stellen meiner Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen sind, habe ich in jedem Fall unter Angabe der Quelle als Anlehnung kenntlich gemacht. Dasselbe gilt sinngemäß für Tabellen und Abbildungen. Diese Arbeit hat in dieser oder ähnlicher Form noch nicht im Rahmen einer anderen Prüfung vorgelegen.

München im Juli 2008 Jérôme Frisch

Inhaltsverzeichnis

1 Einleitung **1**

2 Das Fiala-Modell **3**
2.1 Passives System . . . 3
2.1.1 Wärmeübertragung im Gewebe . . . 5
2.1.1.1 Bioheat Equation . . . 5
2.1.1.2 Metabolismus . . . 6
2.1.1.3 Blutzirkulation . . . 8
2.1.2 Wärmeaustausch mit der Umgebung . . . 8
2.1.2.1 Konvektion . . . 8
2.1.2.2 Strahlung . . . 9
2.1.2.3 Isolation durch Kleidung . . . 10
2.1.2.4 Verdunstung . . . 11
2.1.2.5 Atmung . . . 12
2.2 Aktives System . . . 13
2.2.1 Vasokonstriktion - Gefäßverengung . . . 14
2.2.2 Vasodilatation - Gefäßerweiterung . . . 15
2.2.3 Schwitzen (Sweating) . . . 15
2.2.4 Zittern (Shivering) . . . 15
2.3 Validierung . . . 16
2.4 Berechnung des Fiala-Modells mittels THESEUS-FE . . . 16
2.4.1 Gekoppelter und ungekoppelter Berechnungsmodus . . . 18
2.4.2 Erweiterungen des implementierten Fiala-Modells in THESEUS-FE . 20
2.4.3 Anwendung von THESEUS-FE in einem Computational Steering Umfeld 21
2.4.4 Validierung der Ergebnisse für eine Unterbrechung der Berechnung . 22
2.4.4.1 Beispiel 1: schlagartige Abkühlung . . . 22
2.4.4.2 Beispiel 2: zeitliche Anpassung . . . 24
2.4.4.3 Ergebnisse . . . 26
2.5 Schlußfolgerung . . . 26

3 Thermoregulations-Interface **28**
3.1 Anbindung an THESEUS-FE . . . 28
3.2 Benötigte Daten des Thermoregulations-Interfaces . . . 30
3.2.1 Datenstruktur des Körperaufbaus . . . 30
3.2.1.1 Elemente der Klasse BodyLayer . . . 32
3.2.1.2 Elemente der Klasse BodySector . . . 33
3.2.1.3 Elemente der Klasse BodyElement . . . 35

3.2.1.4 Elemente der Klasse Body 36
3.2.2 Datenstruktur des Thermoregulations-Interfaces 38
3.2.2.1 Elemente der Klasse TFESolverData 39
3.2.2.2 Elemente der Klasse TFEPostProcData 40
3.2.2.3 Elemente der Klasse TheseusModel (und ThermoModel) . . 41
3.2.3 Probleme bei der Auswahl der Zeitschrittweite 43

4 Parametrisches Geometriemodell 45
4.1 Globales Konzept 45
4.2 Umsetzung 45

5 Kopplungskonzepte 48
5.1 Allgemeines 48
5.2 Möglichkeiten 48
5.2.1 Virtual Climate Chamber Project 48
5.2.2 Kopplung an Fluidsolver 49
5.2.3 Kopplung an ein Ein- oder Mehrzonenmodell 50
5.2.4 Kopplung an Fluidsolver mit Überwachungsfunktion 51

6 Interaktive Simulation - Virtual Climate Chamber Project 53
6.1 Allgemeines 53
6.2 Programmierkonzept 53
6.3 Internes Kopplungskonzept 54
6.3.1 Berechnungsschleife 55
6.3.2 Datensynchronisation 55
6.3.3 Interner Ablauf in der graphischen Benutzeroberfläche 57
6.4 Funktionsweise der graphischen Benutzeroberfläche 58
6.4.1 OpenGL Viewport 59
6.4.2 Steuerungsmenüs 59
6.4.2.1 Geometrieinformationen 59
6.4.2.2 Randbedingungen 60
6.4.2.3 Displayeinstellungen 63
6.4.2.4 Darstellung der Randbedingungen als Tabelle 65
6.4.2.5 Darstellung der globalen Ergebnisse als Tabelle 66
6.4.3 Anmerkungen 66
6.4.3.1 Position des Manikins 66
6.4.3.2 LMV Darstellung 66

7 Zusammenfassung und Ausblick 68

A THESEUS-FE 70
A.1 NASTRAN Geometriedatei 70
A.2 THESEUS-FE Input File 71
A.3 HDF Ergebnisdatei 79
A.4 Äquivalenttemperatur 80
A.4.1 Definition 80

A.4.2 Berechnung 81
A.5 Änderung von 10 auf 15 Körperelemente 82

B Thermoregulations-Interface **83**
B.1 Reihenfolge der Sektoren in Temperaturfeldern 83
B.2 Kommunikation mittels Sockets 84
B.3 Header-Definitionen der Datenstruktur des Körperaufbaus 85
B.3.1 body_structure.h 85
B.3.2 fiala_structure.h 91
B.3.3 clothing.h 92
B.3.4 initial_conditions.h 92
B.3.5 current_data.h 93
B.4 Header-Definitionen des Thermoregulations-Interfaces 94
B.4.1 global_model.h 94
B.4.2 theseus_configuration.h 96
B.4.3 theseus_interface.h 98
B.4.4 socket_server.h 100

C Das Experiment von Raven (1970) **104**

Literaturverzeichnis **106**

Abbildungsverzeichnis

2.1 Schematische Darstellung des passiven Systems [18] 4
2.2 Wärmeaustausch mit der Umgebung über die Haut [28] 9
2.3 Blutzirkulation und Gefäßerweiterungs- und -verengungsmechanismen [28] . 14
2.4 Interner Aufbau von THESEUS-FE [29] 17
2.5 Übergang THESEUS-FE Shell-Modell zum internen Fiala Modell [28] 18
2.6 Computational Steering Konzept 21
2.7 Vergleich zwischen Ergebnissen Beispiel 1 23
2.8 Relativer Fehler in der metabolischen Wärmeerzeugung Q_m zwischen der ununterbrochenen Berechnung sowie den unterbrochenen Berechnungen mit unterschiedlichen Postprocessing-Schrittweiten 24
2.9 Temperaturrandbedinungen des Beispiels 2 25
2.10 Vergleich zwischen Ergebnissen Beispiel 2 26
2.11 Temperaturverläufe sowie relativer Fehler ΔQ_m in Beispiel 2 27

3.1 Anbindung des Thermoregulations-Interfaces an THESEUS-FE 29
3.2 Datenstruktur des Köperaufbaus 31
3.3 Datenstruktur des Thermoregulations-Interfaces 38
3.4 Probleme bei der Anpassung der Zeitschrittweite dt 44

4.1 Konzept des parametrischen Geometriemodells [32] 45
4.2 vom Fiala- zum VR-Modell [32] 46

5.1 Kopplungskonzept: Virtual Climate Chamber Project 49
5.2 Kopplungskonzept: Kopplung an Fluidsolver 50
5.3 Kopplungskonzept: Kopplung an ein Zonenmodell 51
5.4 Kopplungskonzept: Kopplung an Fluidsolver mit Überwachungsfunktion . . . 52

6.1 VCC: Menü: Geometrische Informationen 58
6.2 VCC: Menü: Randbedingungen (Teil 1) 60
6.3 VCC: Menü: Randbedingungen (Teil 2) 62
6.4 VCC: Menü: Displayeinstellungen 64
6.5 VCC: Menü: Tabelle für Darstellung der Randbedingungen 65

A.1 Screenshot des HDF-Viewers [7] 80
A.2 Definition der Äquivalenttemperatur [28] 81

C.1 Raven Experiment: Mittlere Hauttemperatur $T_{sk,m}$ 105
C.2 Raven Experiment: Metabolische Wärmeerzeugung Q_m 105

Tabellenverzeichnis

2.1 Randbedingungen des passiven Systems 4
2.2 Typische Werte für metabolische Aktivitäten (aus [15]) 7

3.1 Randbedingungen für thermische Neutralität 29
3.2 Ergebnisse bei thermischer Neutralität 30
3.4 Übersicht über alle öffentlichen Funktionen der Klasse `ThermoModel` 39
3.6 Übersicht über alle öffentlichen Funktionen der Klasse `TheseusModel` 41

C.1 Randbedingungen für Temperatur und rel. Luftfeuchte 104

1 Einleitung

Deetjen und Alzheimer [24] definieren die Aufgabe des Thermoregulationssystems folgendermaßen:

> „Die Aufgabe des Thermoregulationssystems besteht darin, die Körpertemperatur in engen Grenzen konstant zu halten. Die Steuerzentren dafür liegen im Hypothalamus. Die autonomen Mechanismen der Temperaturregulation (Hautdurchblutung, Schwitzen) erlauben unter extremen Umweltbedingungen in gewissem Umfang eine Akklimatisation, wenngleich in begrenztem Maße. [...] Bei extremen Umgebungsbedingungen und einem Versagen der Regulationsmechanismen sind – insbesondere bei Kindern und älteren Menschen – Hypo- und Hyperthermie möglich."

Die korrekte Vorhersage der menschlichen, thermoregulatorischen Antworten auf eine bestimmte Umgebung ist, wie der letzte Satz des Zitats schon aussagt, von großer Wichtigkeit. Bei zu gefährlichen, langwierigen oder unbequemen Versuchen mit Testpersonen ist eine numerische Simulation stets zu bevorzugen, falls sie mit realen Werten in irgendeiner Art und Weise validiert werden kann.

Ziel dieser Arbeit ist es, die Thermoregulation des menschlichen Körpers in einem digitalen Modell nachzubilden. Als Grundlage dient das von D. Fiala entwickelte Modell, das in Kapitel 2 detailliert erläutert wird. Hierzu wird das Programm THESEUS-FE von P+Z Engineering eingesetzt, und über geeignete Schnittstellen und Kopplungskonzepte für weitere Programme zur Verfügung gestellt.

Anschließend wird eine „Virtual Climate Chamber" implementiert, in der eine Versuchsperson unterschiedlichen Klimabedingungen, die interaktiv während der Simulation angepasst werden können, ausgesetzt wird. Anhand von Vergleichen mit Messungen die 1970 von Raven durchgeführt wurden, kann gezeigt werden, dass die Virtual Climate Chamber vergleichbare Ergebnisse liefert.

Diese Arbeit steht in Verbindung mit einer Reihe von Arbeiten eines übergeordneten Projektes, in denen interaktive Simulationen des thermischen Komforts in Innenräumen behandelt

werden [33]. Die menschliche Thermoregulation wird dabei durch geeignete Konzepte, die in dieser Arbeit entwickelt werden, simuliert.

Weiterhin werden Anbindungen zu anderen Systemen wie Fluidsolvern diskutiert und bewertet, und ein Ausblick über noch ausstehende Arbeiten und Punkte gegeben.

2 Das Fiala-Modell

Heutzutage können numerische Simulationen reale Versuche oder Untersuchungen ergänzen, da sie kostengünstiger, komfortabler, ungefährlicher und zerstörungsfrei sind. Allerdings muss darauf geachtet werden, dass ein physikalisch akkurates Modell richtig abgebildet und mathematisch korrekt gelöst wird, zumal sonst die Wirklichkeit nicht oder nicht richtig abgebildet werden kann. Zusätzlich müssen solche Modelle kalibriert und validiert werden, weil sie ansonsten für reale Anwendungen nicht brauchbar sind.

Das Fiala-Modell [18] als numerische Simulation menschlicher Thermoregulation wurde 1998 von D. Fiala an der De Montfort University in Leicester (UK) entwickelt und mit vielen Ergebnissen aus bestehender Literatur verglichen und validiert.

Es handelt sich um ein dynamisches Modell, das Vorhersagen über menschliche Reaktionen auf kalte, kühle, neutrale, warme und heiße Umgebung treffen soll. Es besteht aus einem passiven und einem aktiven System, welche anschließend im folgenden Kapitel näher erläutert werden.

Die Randbedingungen, die für die Berechnungen nötig sind, wie zum Beispiel Umgebungs- und Oberflächentemperaturen, relative Luftfeuchte oder Strömungsgeschwindigkeiten, werden aus der Realität oder aus anderen Simulationsdaten entnommen und an das Modell übergeben.

2.1 Passives System

Das passive, kontrollierte System ist in Abb. 2.1 dargestellt und wird durch die physikalische Beschreibung des menschlichen Körpers sowie dessen Wärmeaustausch mit seiner Umgebung modelliert.

Metabolische Wärme wird im Körper erzeugt und an verschiedene Regionen über Blutzirkulation verteilt. Anschließend wird die dort angekommene Wärme über Wärmeleitung zur Körperoberfläche transportiert und an die Umgebung durch Konvektion, Strahlung und

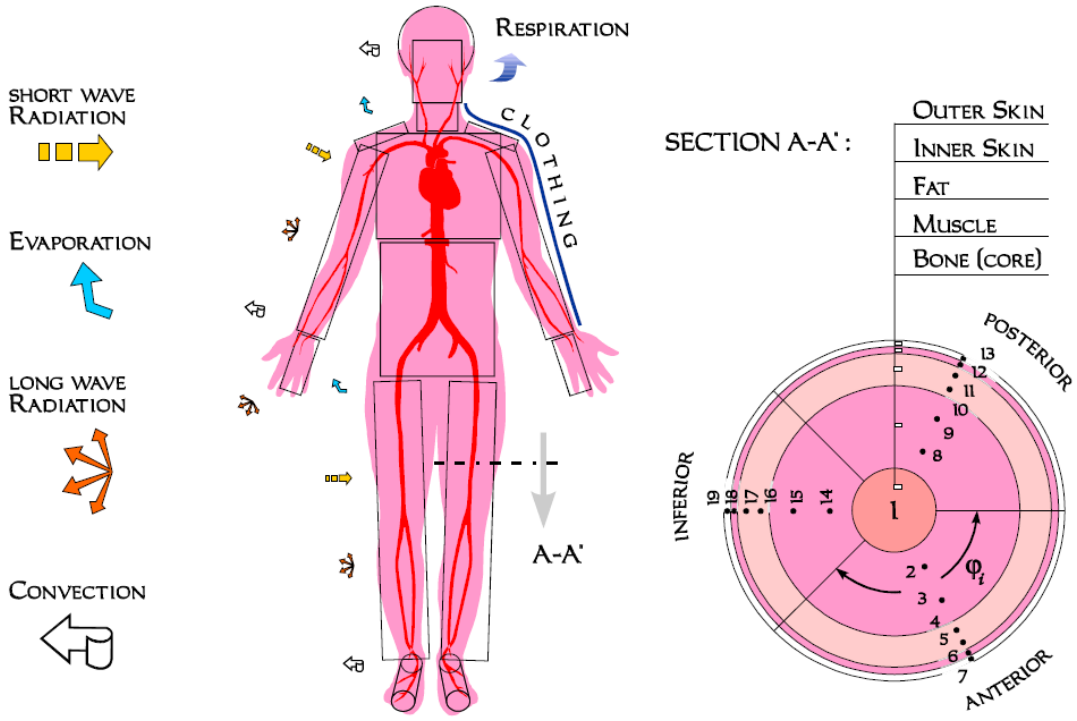


Abbildung 2.1: Schematische Darstellung des passiven Systems [18]

Verdunstung abgegeben. Kleidungsstücke wirken dabei lokal thermisch isolierend. Zusätzlich wird ebenfalls Wärme über die Atmung abtransportiert.

Um diese Effekte korrekt modellieren zu können, muss das passive System geometrisch wie auch anatomisch so gut wie möglich an den menschlichen Körper angepasst sein. Das Mensch-Modell ist in mehrere zylindrische oder sphärische Körperteile unterteilt. Diese bestehen wiederum aus mehreren Schichten (Layern), wie zum Beispiel Knochen, Muskeln, Fett, Haut, etc. Weiterhin sind die Elemente ebenfalls in unterschiedliche Sektoren wie *anterior*, *posterior* oder *inferior* unterteilt, um Asymmetrien im Körperaufbau besser modellieren zu können. Eine genaue Definition der Elemente inklusive Layer und Sektoren kann Tabelle 2 aus [19] entnommen werden, und ist in Abb. 2.1 sowie in Abb. 2.5 schematisch dargestellt. Das Originalmodell eines durchschnittlichen Mannes, das Fiala in [18] vorgeschlagen hat, besitzt folgende Randgrößen:

Tabelle 2.1: Randbedingungen des passiven Systems

m [kg]	BFP [%]	A_{sk} [m^2]	wt_{sk} [%]	CO [l/min]	$M_{bas,0}$ [W]
73.5	14.0	1.86	6.0	4.9	87.1

wobei m die Körpermasse, BFP den Körperfettanteil (body fat percentage), A_{sk} die Hautoberfläche des Körpers, wt_{sk} der nasse Anteil der Haut im Vergleich zur Gesamtoberfläche,

CO das grundlegende Herzminutenvolumen (basal cardiac output) sowie $M_{bas,0}$ den grundlegenden Metabolismus bezeichnet. Die Hautoberfläche bezieht sich auf die DuBois-Oberfläche des passiven Systems, und kann aus $A_{Du} = \int \mathrm{d}A_{sk}$ ermittelt werden, wobei $\mathrm{d}A_{sk}$ einzelne Teilfächen des Körpers bezeichnen [15, 19].

Zusätzlich wird die Haut beim Fiala-Modell mit Hilfe von zwei Layern modelliert. Die innere Hautschicht (ca. 1 mm dick) simuliert die Schicht, in der metabolische Wärme generiert wird und Blutzirkulation vorherrscht. Die äußere Hautschicht besitzt eine ähnliche Dicke, jedoch findet keine Wärmeerzeugung statt, und es existieren zudem keine thermisch relevanten Blutgefäße. Sie ist allerdings sehr wichtig für Verdunstungseffekte. Für weitere Details sei auf [18] und [19] verwiesen.

In den folgenden Punkten sollen nun knapp die physiologischen und thermischen Effekte des passiven Systems untersucht werden. Für weitergehende Betrachtungen sei auch hier auf den Artikel [19] verwiesen.

2.1.1 Wärmeübertragung im Gewebe

2.1.1.1 Bioheat Equation

Die Wärmeübertragung im menschlichen Gewebe kann für ein konzentrisches oder sphärisches Modell mit Hilfe der so genannten „Bioheat Equation“ nach Pennes [26] sehr gut nachgebildet werden:

$$k\left(\frac{\partial^2 T}{\partial r^2} + \frac{\omega}{r}\frac{\partial T}{\partial r}\right) + q_m + \rho_{bl} w_{bl} c_{bl}\left(T_{bl,a} - T\right) = \rho c \frac{\partial T}{\partial t} \tag{2.1}$$

wobei k den Wärmeleitkoeffizienten, T die Temperatur und r den Radius eines Körperelementes bezeichnet. ω ist ein dimensionsloser Parameter, der im Falle eines zylindrischen Körperelementes 1 ist, und im Falle eines sphärischen Elementes den Wert 2 besitzt. ρ_{bl} steht für die Blutdichte, c_{bl} für die spezifische Wärmekapazität von Blut, w_{bl} für die Durchblutungsrate und $T_{bl,a}$ für die arterielle Bluttemperatur. ρ und c stehen für die Dichte sowie die spezifische Wärmekapazität des betrachteten Gewebes.

Die Differentialgleichung (2.1) beschreibt die Wärmedissipation im homogenen, unendlich ausgedehnten Gewebe. Der erste Term der linken Seite repräsentiert die Wärmeleitung, q_m den Einfluß des Stoffwechsels und der letzte Term der linken Seite die arterielle Blutheizung. Dem gegenüber, auf der rechten Seite der Gleichung, steht die Wärmespeicherung in der Gewebemasse. Gleichung (2.1) wird auf alle Gewebe mit den entsprechenden Werten für

k, c und ρ angewendet, wobei zu beachten ist, dass q_m und w_{bl} durch das aktive System (siehe Abschn. 2.2) mit beeinflusst werden.

Die Wärmeleitung in (2.1) wird nur in radialer Richtung berücksichtigt; alle anderen Effekte, wie zum Beispiel Querleitung, können vernachlässigt werden. Auch wenn aufgrund von unterschiedlichen Temperaturgradienten, vor allem in den Hautbereichen, Wärme von einem Sektor in den angrenzenden übertragen wird, so ist dieser Effekt doch marginal im Vergleich zu der radial transportierten Wärme, da die Oberflächen für Querleitung wesentlich kleiner sind, als die für radiale Wärmeleitung. Weiterhin sind Einflüsse aus Querleitung insignifikant für die inneren Schichten, da dort die Wärmeübertragung von der Blutzirkulation und dem Metabolismus dominiert wird [19].

2.1.1.2 Metabolismus

Der Metabolismus eines finiten Gewebeelementes besteht in diesem Modell aus der Summe eines Basiswertes $q_{m,bas,0}$, welcher in Tabelle 2 in [19] aufgelistet ist, sowie aus einem zusätzlichen Term Δq_m, der von lokaler Thermoregulation erzeugt werden kann (zum Beispiel: Zittern):

$$q_m = q_{m,bas,0} + \Delta q_m \quad . \tag{2.2}$$

Der Wert für den totalen grundlegenden Metabolismus kann durch eine Integration über alle Körperteile ermittelt werden

$$M_{bas,0} = \int q_{m,bas,0} \, \mathrm{d}V \tag{2.3}$$

und muss sich gemäß Tabelle 2.1 zu 87 W ergeben. In Muskelschichten kann Δq_m aus bis zu drei Termen bestehen:

$$\Delta q_m = \Delta q_{m,bas} + q_{m,sh} + q_{m,w} \tag{2.4}$$

wobei die Terme $q_{m,sh} + q_{m,w}$ für Muskelzittern (shivering) und verrichtete Arbeit (working) für alle Nicht-Muskelschichten gleich Null sind. Für $q_{m,sh}$ gilt:

$$\Delta q_{m,sh} = a_{sh,j} \frac{Sh}{V_{mus,j}} \quad . \tag{2.5}$$

Sh wird durch das aktive System geregelt (Gleichung (2.31)). $V_{mus,j}$ steht für das Muskelvolumen des betrachteten Körperteils. Der Koeffizient $a_{sh,j}$ berücksichtigt die Hautempfind-

Tabelle 2.2: Typische Werte für metabolische Aktivitäten (aus [15])

Aktivität	met
Schlafen	0.7
Ausruhen	0.8
Gehen (3.2 km/h)	2.0
Gehen (6.4 km/h)	3.8
Lesen (sitzend)	1.0
Tippen	1.1

lichkeit unterschiedlicher Körperteile, und kann [18] oder [28] entnommen werden.

Der metabolische Anteil aus Arbeit berechnet sich damit zu

$$\Delta q_{m,w} = a_{m,w,j} \frac{H}{V_{mus,j}} \tag{2.6}$$

mit dem Koeffizienten $a_{m,w,j}$ aus Tabelle 2 in [19] und

$$H = act \cdot \frac{M_{bas,0}}{act_{bas}} \cdot (1 - \eta) - M_{bas,0} \qquad . \tag{2.7}$$

Hierbei bezeichnet act die Aktivität bei der jeweiligen Tätigkeit (siehe Tabelle 2.2), act_{bas} die Aktivität bei thermischer Neutralität und η kann durch einen Ansatz aus

$$\eta = 0.2 \cdot \tanh(0.39 \cdot act - 0.6) \tag{2.8}$$

gewählt werden. $(1 - \eta)$ repräsentiert den Anteil an metabolischer Wärmeproduktion, die für das Aufheizen des Organismus verantwortlich ist. Der Rest wird an die Umgebung abgegeben.

$\Delta q_{m,bas}$ bezeichnet die Differenz zwischen dem aktuellen Metabolismus und dem Metabolismus unter thermisch neutralen Bedingungen und kann über

$$\Delta q_{m,bas} = q_{m,bas,0} \cdot \left(2^{(T-T_0)/10} - 1\right) \tag{2.9}$$

berechnet werden, wobei T_0 die Setpoint-Temperatur bei thermischer Neutralität bezeichnet, das heißt die Temperatur, die sich einstellt, wenn mit thermisch neutralen Randbedingungen (siehe Tabelle 3.1) gerechnet wird.

2.1.1.3 Blutzirkulation

In Abschnitt 3.3.3. der Dissertation von Fiala [18] heisst es:

> "Blood circulation is of vital importance for the dissipation of heat within the human body. The dissipation effect was examined by simulations using the present model without blood circulation: the core temperature of the brain, which has a high metabolic rate, exceeded 73°C, when the passive system was exposed to steady neutral ambient temperatures of 30°C."

Daraus lässt sich erkennen, dass es von großer Bedeutung ist, den Blutkreislauf richtig zu modellieren, da ansonsten die Wärme nicht abgeführt werden kann. Dieser wird im System aus drei Komponenten zusammengesetzt: dem zentralen Blutvorrat (central blood pool) mit der Temperatur $T_{bl,p}$, dem Gegenstromwärmeaustauch Q_x (countercurrent heat exchange) zwischen venösem und arteriellem Blutkreislauf, und den Gefäßen zu den einzelnen Geweben. Details hierzu sind in Abbildung 2.3 dargestellt.

Die Formeln zur Berechnung der Bluttemperaturen $T_{bl,a}$, $T_{bl,v}$ und $T_{bl,p}$ können aus [28] entnommen werden.

2.1.2 Wärmeaustausch mit der Umgebung

Der Wärmeaustausch mit der Umgebung findet vor allem durch zwei Wege statt: Über die *Körperoberfläche* durch Konvektion, Strahlung und Verdunstung sowie über die *Atemwege* durch Konvektions- und Verdunstungseffekte.

Alle Wärmestromdichten, die auf der Oberfläche ankommen, werden durch die jeweiligen Schichten in die inneren Regionen geleitet und von der Blutzirkulation abtransportiert.

In den folgenden Punkten werden nun die einzelnen Wärmestromdichten detaillierter betrachtet.

2.1.2.1 Konvektion

Die Wärmestromdichte einer freien und erzwungenen Konvektion zwischen der Körperoberfläche eines Sektors mit Temperatur T_{sf} und der Umgebungstemperatur T_a kann nach folgendem Prinzip berechnet werden:

$$q_{conv} = h_{c,mix} \cdot (T_a - T_{sf}) \quad . \tag{2.10}$$

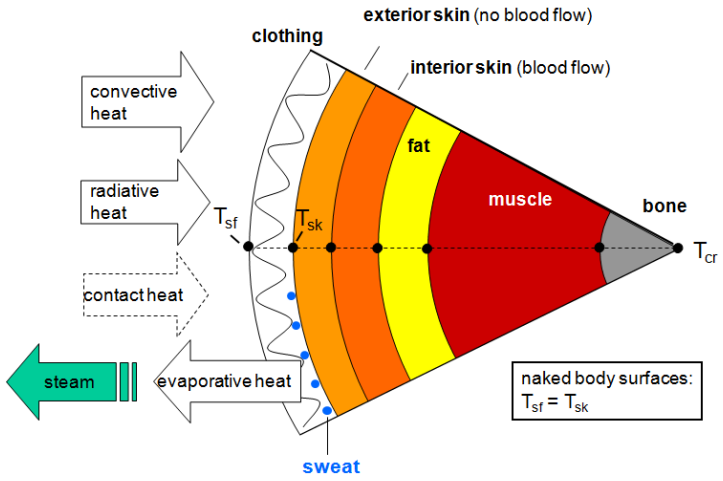


Abbildung 2.2: Wärmeaustausch mit der Umgebung über die Haut [28]

Der Koeffizient $h_{c,mix}$ ist nach dem Ansatz von Fiala zu

$$h_{c,mix} = \sqrt{a_{nat} \cdot \sqrt{|T_a - T_{sf}|} + a_{frc} \cdot v_{a,eff} + a_{mix}} \tag{2.11}$$

zu wählen. Hierbei sind die Koeffizienten a_{nat}, a_{frc} und a_{mix} in Tabelle 2 in [19] tabelliert, und $v_{a,eff}$ bezeichnet die effektiv herrschende Windgeschwindigkeit in m/s.

2.1.2.2 Strahlung

Unter Annahme eines schwarzen Strahlers wird die Wärmestromdichte aus Strahlung durch den linearisierten Ansatz nach

$$q_{rad} = h_r \cdot (T_w - T_{sf}) \tag{2.12}$$

bestimmt, wobei T_{sf}, wie im Falle der Konvektion auch, die Oberflächentemperatur eines Körpersektors und T_w die Hintergrundstrahlungstemperatur (der Wand) bezeichnet. Der Koeffizient h_r berechnet sich zu

$$h_r = \sigma \; \varepsilon_{sf} \; \varepsilon_w \; \psi_{sf-w} \; (T_{sf}^2 + T_w^2) \; (T_{sf} + T_w) \qquad . \tag{2.13}$$

Hierbei steht $\sigma = (5.670\,400 \pm 0.000\,040) \cdot 10^{-8} \, \frac{\mathrm{W}}{\mathrm{m^2 K^4}}$ für die Stefan-Boltzmann-Konstante, ε_{sf} für den Strahlungskoeffizienten der Körperoberfläche, ψ_{sf-w} für den Sichtfaktor des Dummy-Körpers, und ε_w für den Strahlungskoeffizienten der Wand, der als Randbedingung festgelegt

wird.

ε_{sf} und ψ_{sf-w} sind in Tabelle 2 in [19] von Fiala für 2 Standardfälle (stehend und sitzend) tabelliert worden.

Der Sichtfaktor ψ_{sf-w} für eine Körperoberfläche gibt an, wieviel Prozent der Umgebung w von der betrachteten Fläche sf aus sichtbar ist. Als Beispiel ist der Sichtfaktor des Inferior-Sektors eines Armes sehr klein, da dieser Sektor nahe am Oberkörper ist, durch den fast die gesamte Umgebung verdeckt ist.

2.1.2.3 Isolation durch Kleidung

Kleidungsstücke können als eine Isolationschicht aufgefasst werden, die zusätzlich zum Körperaufbau hinzukommt. Dabei ist zu beachten, dass die Hauttemperatur T_{sk} nicht mehr identisch mit der Oberflächentemperatur T_{sf} ist. Der Unterschied zwischen den beiden Temperaturen ist im Isolationswert R_{cl} der Bekleidung zu sehen:

$$R_{cl} = \frac{T_{sf} - T_{sk}}{q_{conv} + q_{rad}} \qquad \left[\frac{\mathrm{m^2\ K}}{\mathrm{W}} \right] \qquad . \tag{2.14}$$

In der Praxis wird der Kleidungs-Isolationswert R_{cl} gerne in der Einheit „clo“ angegeben. Die Umrechnung lautet dann laut [15]:

$$I_{cl} = 6.45 \cdot R_{cl} \qquad [\ \mathrm{clo}\ ] \qquad \left(\text{oder}\ \ 1\ clo = 0.155 \frac{m^2\ K}{W} \right) \qquad . \tag{2.15}$$

Wird nun die Wärmestromdichte durch die Kleidung betrachtet und als $q_{cond,cl}$ gekennzeichnet, so kann ein globaler Widerstand R_{cl} für den gesamten Körper bestimmt werden. Fiala hat zusätzlich R^*_{cl} eingeführt, mit dem lokal der Widerstand in jedem Sektor bezeichnet wird, wodurch es möglich ist, die Wärmestromdichten lokal für jeden Körpersektor zu bestimmen.

Wird ebenfalls eine eindimensionale Wärmeleitung

$$q_{cond,cl} = -k \cdot \frac{\Delta T}{\Delta x} = k \cdot \frac{T_{sf} - T_{sk}}{d} \tag{2.16}$$

in der Kleidungsschicht vorausgesetzt, so kann über eine Kleidungsschichtdicke d eine Wärmeleitfähigkeit k berechnet werden:

$$k = \frac{d}{R^*_{cl}} \qquad . \tag{2.17}$$

Sind mehrere Schichten $i = 1..n$ vorhanden, so kann die Wärmeleitfähigkeit durch folgenden Ansatz ermittelt werden

$$k = \frac{d}{\sum_{i=1}^{n} R^*_{cl,i}} \quad . \tag{2.18}$$

Ebenso kann der Wärmedurchgangskoeffizient U^*_{cl} definiert werden zu

$$U^*_{cl} = \frac{1}{\sum_{i=1}^{n} R^*_{cl,i}} = \frac{k}{d} \quad . \tag{2.19}$$

Sämtliche oben angegebenen Beziehungen gelten für den sogenannten „trockenen Wärmeverlust“. Spielen Verdunstungseffekte eine Rolle, so muss ebenfalls der latente Wärmeverlust mit einbezogen werden.

$q_{e,cl}$ bezeichnet die Wärmestromdichte des latenten Wärmeverlustes und berechnet sich aus

$$q_{e,cl} = -U^*_{e,cl} \cdot (p_{sk} - p_a) \tag{2.20}$$

wobei p_{sk} den Wasserdampfdruck an der Hautoberfläche, und p_a den Dampfdruck der Atmosphäre bestimmt. Der Wärmedurchgangskoeffizient $U^*_{e,cl}$ ergibt sich aus

$$U^*_{e,cl} = \frac{1}{\sum_{i=1}^{n} R_{ecl,i} + \frac{1}{f^*_{cl} \cdot h_{c,mix} \cdot L_a}} \tag{2.21}$$

mit der Lewis-Konstanten von Luft $L_a = 0.0165$ K/Pa und $R_{ecl,i}$ (für die Layer $i = 1..n$) als lokaler Verdunstungswiderstand in m^2Pa/W. In der Literatur wird auch öfters der Feuchtedurchlässigkeitsbeiwert i_{cl} angegeben, der in $R_{ecl,i}$ umgerechnet werden kann. f^*_{cl} definiert dazu den lokalen Bekleidungsanteil der Körperoberfläche ($f^*_{cl} = A_{sf}/A_{sk}$).

2.1.2.4 Verdunstung

Verdunstungseffekte an der Hautoberfläche führen ebenfalls zu einem Wärmeverlust des Körpers. Folgender Zusammenhang besteht zwischen der Wärmestromdichte aus Verdunstung q_e und dem Masseverlust aus Wasser $\partial m/\partial t$:

$$\frac{q_e}{\lambda_{H_2O}} = \frac{\partial m}{\partial t} \cdot \frac{1}{A_{sk}} = -\frac{\partial m_{sw}}{\partial t} \cdot \frac{1}{A_{sk}} - \frac{p_{sk,sat} - p_{sk}}{\lambda_{H_2O} \cdot R_{e,sk}} \tag{2.22}$$

mit $\lambda_{H_2O} = 2256$ kJ/kg als Verdampfungswärme von Wasser. p_{sk} steht für den effektiv vorhandenen Dampfdruck und $p_{sk,sat}$ für den Sättigungsdampfdruck an der Hautoberfläche. Das Modell von Fiala benutzt eine Hautfeuchtepermeabilität von $1/R_{e,sk} = 0.003$ W/m^{2}Pa. Die Schweißproduktion $\partial m_{sw}/\partial t$ wird durch das aktive System geregelt. In obiger Gleichung steht der linke Term der äußerst rechten Seite für Effekte aus Schwitzen und der rechte für Effekte aus Diffusion.

Übersteigt der effektiv vorhandene Dampfdruck p_{sk} den Sättigungsdampfdruck $p_{sk,sat}$, so kommt es zu einer Anlagerung von Schweiß an der Hautoberfläche

$$\frac{\partial m_{acc}}{\partial t} = \frac{\partial m_{sw}}{\partial t} - \frac{U^*_{e,cl} \cdot (p_{sk,sat} - p_a)}{\lambda_{H_2O}} \cdot A_{sk} \qquad (> 0!) \qquad . \tag{2.23}$$

Aufgrund von experimentellen Ergebnissen ist die Anlagerung von Schweiß m_{acc}/A_{sk} auf 35 g/m^2 beschränkt, da ein größerer Anteil Wasser einfach an der Hautoberfläche abläuft, und nicht mehr von der Simulation erfasst werden kann.

2.1.2.5 Atmung

Wie in 2.1.2 erwähnt, tauscht das Fiala-Modell nicht nur Wärmeströme über die Hautoberfläche aus, sondern auch über die Atmung. Ein Anteil besteht aus Konvektion $\dot{C}_{rsp}$ und ein Anteil aus Verdunstung $\dot{E}_{rsp}$. Sie berechnen sich zu

$$\dot{C}_{rsp} = -1.948 \cdot 10^{-3} \cdot Q_m \cdot (32.6 - 0.934 \cdot T_a + 1.96 \cdot 10^{-4} \cdot p_a) \tag{2.24}$$

$$\dot{E}_{rsp} = -4.373 \cdot Q_m \cdot (0.0277 - 6.5 \cdot 10^{-5} \cdot T_a - 4.91 \cdot 10^{-6} \cdot p_a) \tag{2.25}$$

$$\dot{Q}_{rsp} = \dot{C}_{rsp} + \dot{E}_{rsp} \tag{2.26}$$

wobei

$$Q_m = \int q_m \mathrm{d}V \tag{2.27}$$

Wärme aus dem Metabolismus, T_a die Raumtemperatur und p_a den Dampfdruck darstellt.

Die gesamte Wärme aus Atmung $\dot{Q}_{rsp}$ wird von Fiala nach einem bestimmten Muster um die Atemwege gelegenen Gewebe verteilt. Für weitere Details siehe [18].

2.2 Aktives System

Während sich das passive System mit physikalischen Effekten des Wärmeaustausches zwischen dem menschlichen Körper und seiner Umgebung, sowie dem dynamischen Wärmeaustausch innerhalb des Körpers beschäftigt, ist die Aufgabe des aktiven Systems die Regelung physiologischer Effekte. Dazu gehören Gefäßerweiterungen oder -verengungen, Schwitzen und Zittern einer nicht im thermischen Gleichgewicht stehenden Person. Wie aber schon in Punkt 2.1 zu erkennen war, ist das aktive mit dem passiven System eng verbunden.

Fiala hat anhand bereits vorliegender Ergebnisse viele Untersuchungen durchgeführt und durch Regressionsanalysen Funktionen bestimmt, mit denen die oben genannten, aktiven Regelungsmechanismen des menschlichen Körpers erfasst werden können. Dabei hat sich gezeigt, dass drei Variablen einen sehr großen Einfluss auf das aktive System besitzen:

- die Hypothalamustemperatur T_{hy} bzw. die Abweichung der Hypothalamustemperatur von der thermischen Neutralität $\Delta T_{hy} = T_{hy} - T_{hy,0}$
- die mittlere Hauttemperatur $T_{sk,m}$ bzw. die Abweichung der mittleren Hauttemperatur von der thermischen Neutralität $\Delta T_{sk,m} = T_{sk,m} - T_{sk,m,0}$
- die zeitliche Ableitung der mittleren Hauttemperatur $\partial T_{sk,m}/\partial t$ (in K/h)

Sämtliche Regelungsfunktionen sind daher von diesen drei Eingangsparametern abhängig, die wiederum vom passiven System beeinflusst werden. (siehe hierzu ebenfalls Abbildung 4.31 in [18]).

Da eine enge Verknüpfung zwischen aktivem und passivem System vorherrscht, ist es zunächst nicht auf Anhieb ersichtlich, wie bei einer Simulation mit aktiven Systemkomponenten zu beginnen ist. Daher ist hier die generelle Vorgehensweise geschildert:

Zuerst wird ausschließlich das passive System für einen stationären Fall bei Randbedingungen der thermischen Neutralität berechnet. Das Ergebnis ist eine Datei mit Sollwerten, in der für jeden Freiheitsgrad im Modell Temperaturen bei thermischer Neutralität vorliegen. Anschließend wird die transiente Simulation des kompletten Systems mit diesen Ausgangswerten gestartet. Nun werden mit dem passiven System für den ersten Zeitschritt sämtliche lokale und gemittelte Temperaturen bestimmt sowie deren Ableitungen gebildet. Damit sind die drei Eingangsgrößen für das aktive System bekannt, und deren Werte können berechnet, und auf die jeweiligen Körperelemente verteilt werden. Anschließend wird der nächste Zeitschritt, aufbauend auf dem vorherigen, bestimmt.

Befindet sich eine Person im thermischen Gleichgewicht, so sollten diese Funktionen ein Ergebnis von Null liefern, da im Falle thermischer Neutralität, laut Theorie, keine Regelung

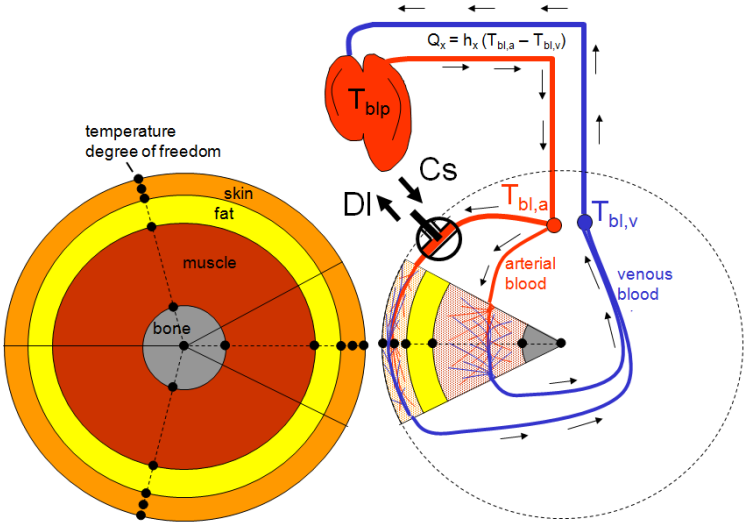


Abbildung 2.3: Blutzirkulation und Gefäßerweiterungs- und -verengungsmechanismen [28]

erforderlich ist. Ist die Funktion größer gleich Null, so sind die jeweiligen Effekte aktiv. Ein Wert kleiner Null ist nicht erlaubt und würde auf einen Fehler im System oder im Modell hindeuten.

In den folgenden Punkten werden nun die vier aktiven Regelungsfunktionen vorgestellt.

2.2.1 Vasokonstriktion - Gefäßverengung

Der Effekt der Gefäßverengung tritt generell bei kälterer Umgebung auf und berechnet sich zu

$$\begin{aligned} Cs &= 35 \cdot [\tanh(0.29 \cdot \Delta T_{sk,m} + 1.11) - 1] \cdot \Delta T_{sk,m} \\ &\quad -7.7 \cdot \Delta T_{hy} + 3 \cdot \Delta T_{sk,m} \frac{\partial T_{sk,m}}{\partial t} \quad . \end{aligned} \tag{2.28}$$

Der letzte Term, der die Ableitung der mittleren Hauttemperatur enthält, wird zu Null gesetzt, falls $\partial T_{sk,m}/\partial t > 0$ oder $\Delta T_{sk,m} > 0$ gilt.

Werte für $Cs > 0$ führen damit zu einer Verengung der Gefäße und zu einer reduzierten Blutzirkulation in der Hautschicht.

2.2.2 Vasodilatation - Gefäßerweiterung

Gefäßerweiterung tritt in der Regel bei wärmerer Umgebung auf und kann durch

$$\begin{aligned} Dl &= 16 \cdot [\tanh(1.92 \cdot \Delta T_{sk,m} - 2.53) + 1] \cdot \Delta T_{sk,m} \\ &\quad +30 \cdot [\tanh(3.51 \cdot \Delta T_{hy} - 1.48) + 1] \cdot \Delta T_{hy} \qquad [\mathrm{W/K}] \end{aligned} \tag{2.29}$$

beschrieben werden. Hier führen Werte größer Null zu einer Erweiterung der Gefäße und damit zu einer erhöhten Blutzirkulation in der Hautschicht.

2.2.3 Schwitzen (Sweating)

Schwitzen tritt hauptsächlich unter warmen Umgebungsbedinungen auf, und kann über folgende Funktion charakterisiert werden:

$$\begin{aligned} Sw &= [0.65 \cdot \tanh(0.82 \cdot \Delta T_{sk,m} - 0.47) + 1.15] \cdot \Delta T_{sk,m} \\ &\quad + [5.60 \cdot \tanh(3.14 \cdot \Delta T_{hy} - 1.83) + 6.4] \cdot \Delta T_{hy} \qquad [\mathrm{g/min}] \qquad . \end{aligned} \tag{2.30}$$

Im Modell ist die Schweißproduktion auf 30 g/min begrenzt. Weiterhin hat Schwitzen einen beträchtlichen Einfluss auf Verdunstungswärmeverluste.

2.2.4 Zittern (Shivering)

Der Effekt des Zitterns tritt typischerweise bei kalten Umgebungsbedingungen auf.

$$\begin{aligned} Sh &= 10 \cdot [\tanh(0.51 \cdot \Delta T_{sk,m} + 4.19) - 1] \cdot \Delta T_{sk,m} - 27.5 \cdot \Delta T_{hy} \\ &\quad -28.2 + 1.9 \cdot \Delta T_{sk,m} \cdot \frac{\partial T_{sk,m}}{\partial t} \qquad [\mathrm{W}] \end{aligned} \tag{2.31}$$

Dazu ist zu beachten, dass der letzte Term zu 0 gesetzt wird, falls $\partial T_{sk,m}/\partial t > 0$ oder $\Delta T_{sk,m} > 0$ ist. Zusätzlich ist die metabolische Wärmeproduktion bei Zittern auf 350 Watt beschränkt. Von einem phänomenologischen Standpunkt aus gesehen ist Zittern vergleichbar mit zusätzlich, von außen aufgebrachter Wärme durch den Menschen.

Es soll ebenfalls noch erwähnt werden, dass die in Gl. (2.28), (2.29), (2.30) und (2.31) definierten Funktionen nicht aus dem Artikel [20] von Fiala stammen sondern aus seiner Originaldissertation [18]. Die Funktionen sind mit den Koeffizienten im THESEUS-FE Solver implementiert worden (unter Mithilfe von D. Fiala).

2.3 Validierung

Fiala hat in [18] viele Validierungen für sein passives und aktives System angeführt. In [30] wurden von der Firma P+Z Engineering GmbH diese Validierungsergebnisse aufgegriffen und mit Hilfe ihres Produktes THESEUS-FE nachgerechnet. Deshalb wird an dieser Stelle das Fiala-Modell als solches, sowie dessen Implementierung in THESEUS-FE als validiert angesehen.

2.4 Berechnung des Fiala-Modells mittels THESEUS-FE

> "THESEUS-FE is a professional software tool for steady-state and fully transient thermal management applications like complete cabin interior analysis including passenger comfort studies. It is designed for the use in automotive, transportation and aerospace industry mainly but is applicable for other markets too." [12]

THESEUS-FE ist ein sehr mächtiges Werkzeug, mit dem komplette thermische Simulationen stationär oder transient berechnet werden können. Anschließend können die Ergebnisse interaktiv in einem graphischen Benutzerinterface (GUI) dargestellt werden, mit der ebenfalls am Anfang der Konfiguration das Problem interaktiv definiert werden konnte. Die Geometrie wird im NASTRAN Format eingelesen. Spezifikationen hierzu sind in Abs. A.1 des Anhangs dargelegt.

Zusätzlich enthält THESEUS-FE ab der Version 2 das neue thermoregulatorische Mensch-Modell FIALA-FE, das von D. Fiala entwickelt, und dessen Funktionsweise im ersten Teil dieses Kapitels erläutert wurde. Es wurde unter Mithilfe von Fiala in THESEUS-FE noch weiter ausgebaut und angepasst, sodass THESEUS-FE damit über ein hoch entwickeltes Modell zur Vorhersage von Oberflächentemperaturen und weiteren Kenngrößen eines Mensch-Modells verfügt. FIALA-FE soll im Folgenden benutzt und mit weiteren Programmen gekoppelt werden, um sämtliche relevanten Daten eines Dummys berechnen zu können.

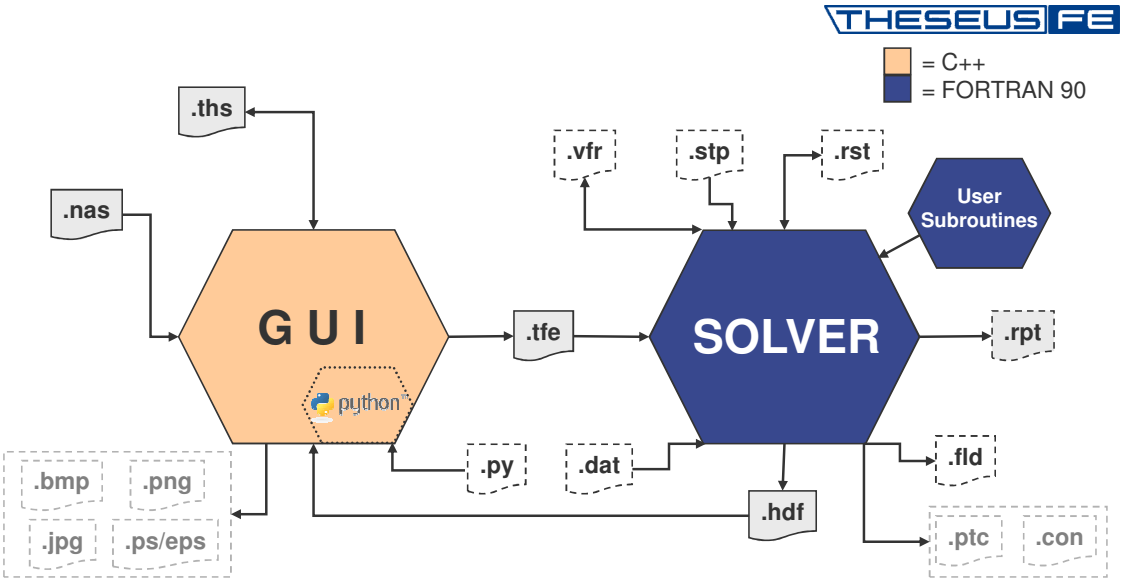


Abbildung 2.4: Interner Aufbau von THESEUS-FE [29]

Allerdings eignet sich eine eigene interaktive Eingabe des Manikins nicht im Sinne von Computational Steering (siehe Abschn. 2.4.3), da solche „sekundären“ Berechnungen ohne Interaktion des Benutzers stattfinden müssen. Dieser soll lediglich die Randdaten in der Computational Steering Oberfläche festlegen, welche dann intern automatisch an den THESEUS-FE Solver durchgereicht werden müssen. Deshalb ist es notwendig, alle benötigte Daten direkt an den Solver weiterzuleiten. Ein möglicher Lösungsweg wird nun geschildert.

Wird der interne Aufbau von THESEUS-FE (Abb. 2.4) genauer betrachtet, so ist erkennbar, dass die Benutzeroberfläche (GUI) umgangen werden kann, indem direkt die vom Solver benötigte `.tfe` Datei manipuliert wird, um dort Einstellungen am Dummy zu verändern. Die `.tfe` Datei ist eine ASCII kodierte Datei, in der sämtliche Einstellungen über *Keywords* vorgenommen werden, welche in [27] gut dokumentiert sind. Es ist hier ebenso wie bei der NASTRAN Geometriedatei darauf zu achten, dass die Daten in Blöcken von jeweils 8 Zeichen vorliegen müssen. Anschließend wird der Fortran-Solver mit dieser Datei gestartet und liefert die Ergebnisse in einer `.hdf` Datei zurück. Mehr Informationen zum HDF Datenformat befinden sich im Anhang unter Punkt A.3.

Im weiteren Verlauf sollen nun die Berechnungsmöglichkeiten dargestellt und diskutiert werden.

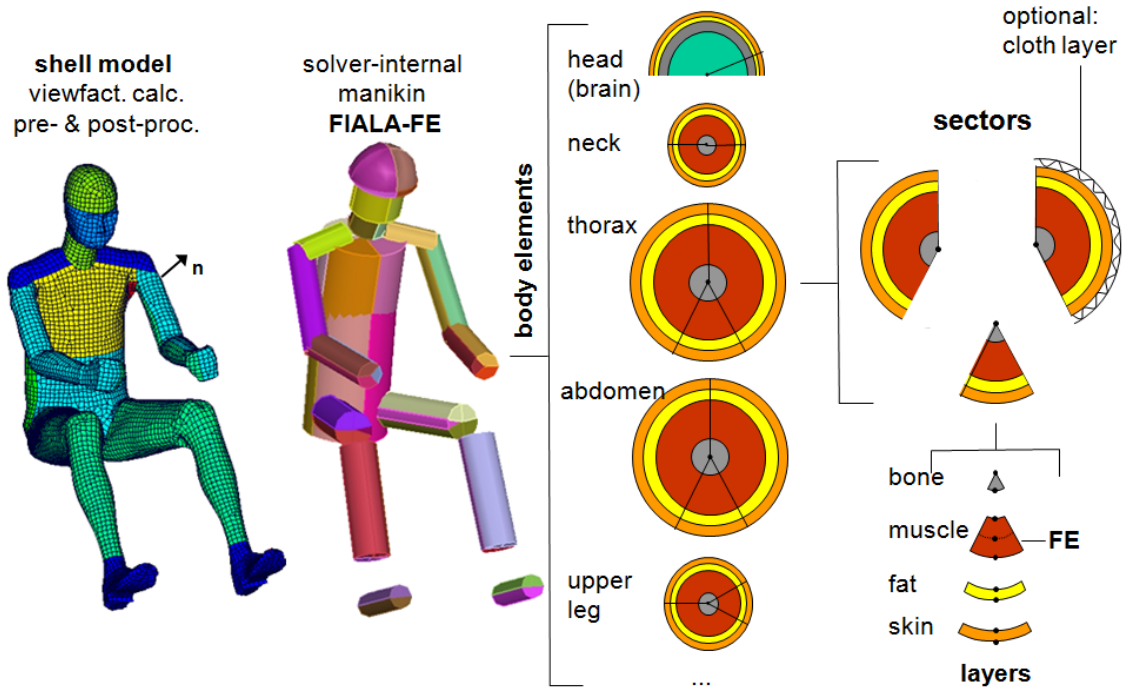


Abbildung 2.5: Übergang THESEUS-FE Shell-Modell zum internen Fiala Modell [28]

2.4.1 Gekoppelter und ungekoppelter Berechnungsmodus

Dem Benutzer von THESEUS-FE stehen zur Berechnung eines Dummy-Modells zwei wesentliche Möglichkeiten zur Verfügung: die gekoppelte und die ungekoppelte Berechnung.

Der gekoppelte Modus wird von den meisten Anwendern benutzt. Hierzu wird eine NASTRAN Geometriedatei geladen, die aus 48 einzelnen Gruppen besteht, welche eine Unterteilung in Elemente und Sektoren des Manikins ermöglichen (linkes Dummy-Modell in Abb. 2.5). In der graphischen Benutzeroberfläche von THESEUS-FE, die nur in der Lage ist, gekoppelte Berechnungen durchzuführen, können Randbedingungen mit den entsprechenden NASTRAN Gruppen verknüpft werden. Während der anschließenden Berechnungsphase werden Sichtbarkeitsfaktoren und Wärmeströme für alle Sektoroberflächen bestimmt. Diese werden an das interne Fiala-Modell übergeben, das nur aus zylindrischen und sphärischen Elementen besteht, wie im mittleren Teil der Abb. 2.5 zu erkennen ist.

Die Bestimmung der Sichtbarkeitsfaktoren und die Integration der Wärmeströme ist erforderlich, wobei im Falle von Computational Steering einige dieser Berechnungen doppelt ausgeführt werden würden. Aus diesem Grunde eignet sich eher die ungekoppelte Berechnung, bei der die oben beschriebenen Berechnungen nicht mit durchgeführt werden.

Der ungekoppelten Berechnung liegt, wie der Name schon nahe legt, kein Geometriemodell

zu Grunde, an das der Solver gekoppelt ist. Das Fiala-Modell wird dabei nur mit Randbedingungen stationär oder transient berechnet. Es kann sowohl das passive System alleine, als auch ein komplettes aktives System inklusive aller thermoregulatorischen Effekte bestimmt werden.

Dieser Modus bietet dabei den Vorteil, auch eigene Manikins definieren zu können. Dabei ist allerdings darauf zu achten, dass Validierungen neu durchgeführt werden müssen, da für eigenständig erzeugte Modelle nicht die gleichen Vorgaben zutreffen, wie für das validierte Fiala-Modell. Als Beispiel sei angeführt, dass für die Fettschicht eine andere Wärmeleitfähigkeit, Dichte, oder Dicke angenommen wurde. Um diese Probleme zu vermeiden, wird in diesem Rahmen das von P+Z validierte Modell benutzt.

Ein weiterer Vorteil ist, dass sämtliche Berechnungen sehr schnell durchgeführt werden können, zumal ein Manikin, je nach Diskretierungsstufe, nur über ca. 500 bis 1000 Freiheitsgrade verfügt, die sich sehr schnell lösen lassen.

Wird ein Manikin anders bekleidet, beispielsweise mit einer zusätzlichen Kleidungsschicht, oder würde ein Bereich noch weiter unterteilt werden, so ist es notwendig, eine neue sogenannte „Setpoint-Datei“ zu erzeugen, die sämtliche Informationen über ein Manikin bei thermischer Neutralität enthält und ausschließlich unter Verwendung des passiven Systems berechnet wurde. Allerdings ist beim Erzeugen einer solchen Datei darauf zu achten, dass die thermisch neutralen Bedingungen für ein unbekleidetes Modell definiert sind.

Würde ein Manikin ohne Bekleidung zur Erzeugung der Setpoint-Datei benutzt werden, so könnte kein bekleidetes Modell berechnet werden, da eine unterschiedliche Anzahl Freiheitsgrade im Modell und in der Setpoint-Datei vorliegen würde. Deshalb muss es sich bei der Erzeugung der Setpoint-Datei um das gleiche Modell handeln, mit dem auch später gerechnet werden soll. Die Bedingung „unbekleidet“ kann erzeugt werden, indem alle physikalischen Werte für Kleidungslayer zu Null gesetzt werden. Anschließend wird die Setpoint-Datei mit dem passiven System stationär bestimmt.

Der ungekoppelte Modus kann die Berechnung ohne Bestimmung der Sichtbarkeiten durchführen, da zwei Standardfälle (wie auch in [18] geschildert) vordefiniert sind. Dabei handelt es sich um einen stehenden und einen sitzenden Dummy. Es können allerdings auch eigene Sichtbarkeitsfaktoren für das Dummy-Modell übernommen werden, falls diese in anderen Berechnungen explizit bestimmt wurden.

2.4.2 Erweiterungen des implementierten Fiala-Modells in THESEUS-FE

Wie bereits erwähnt, wurde P+Z Engineering von D. Fiala während der Implementierung des Modells unterstützt. Durch die Zusammenarbeit konnten dabei diverse Unstimmigkeiten des in der Dissertation [18] beschriebenen Modells geklärt und berichtigt werden. Diese Veränderungen sollen im Folgenden kurz erläutert werden.

Fiala hat einen symmetrischen menschlichen Körpern mit 10 Elementen modelliert. In THESEUS-FE ist der Körper mit Hilfe von 15 Elementen so modelliert worden, dass zwischen Elementen der rechten und der linken Körperhälfte unterschieden werden kann. Die neuen 15 Elemente, mit den jeweils angepassten Regelungskoeffizienten des aktiven Systems, können aus Abbildung 4-9 in [28] entnommen werden. (siehe ebenfalls Anhang A.5)

Fiala hat die Bioheat Equation (2.1) mit Hilfe der Methode der finiten Differenzen berechnet. THESEUS-FE benutzt die Finite-Element-Methode. Der Benutzer kann für jede Schicht mehr als zwei Diskretisierungspunkte in radialer Richtung wählen. Dadurch wird ein nichtlinearer Temperaturverlauf innerhalb der Schicht ermöglicht. Standardmäßig sind drei Punkte pro Layer vordefiniert, was bedeutet, dass jeder Layer in zwei Sub-Layer unterteilt wird.

Es wurden ebenfalls Kontaktwärmeströme hinzugefügt, die allerdings nur in der gekoppelten Berechnung eine Rolle spielen.

Zudem wurde eine zusätzliche Kleidungsschicht mit eigenem Wärmeleitkoeffizienten, spezifischer Wärmeleitfähigkeit, Dichte und vorgegebener Dicke einbezogen, die die transiente Wärmeleitung durch die Kleidung berücksichtigt. Fiala hat diese Effekte für den stationären Fall durch eine Reduktion der Wärmeströme an der Hautoberfläche berücksichtigt.

Sämtliche Berechnungen von Fiala sind mit festen Zeitschritten von 300 Sekunden durchgeführt worden. In THESEUS-FE können die Simulationen sowohl mit einer beliebig festen, als auch mit einer adaptiven Zeitschrittweite berechnet werden. Aus diesem Grund wurde die zeitliche Ableitung der mittleren Hauttemperatur auch in ihrer Größe beschränkt. Wäre diese Sperre nicht eingebaut, so wäre es möglich, dass die Ableitung extreme Werte annimmt, für die das physiologische Modell nicht ausgelegt ist.

Der Gegenstromwärmeaustausch (countercurrent heat exchange) in den Schultern wurde zu Null gesetzt ($h_x = 0$), da sich andernfalls unrealistische Sollwert-Temperaturen ergeben würden.

2.4.3 Anwendung von THESEUS-FE in einem Computational Steering Umfeld

„Eine Computational-Steering Anwendung erlaubt das interaktive Steuern und Visualisieren einer laufenden Computersimulation“ [8]. Die grundlegende Idee ist dabei in Abb. 2.6 dargestellt.

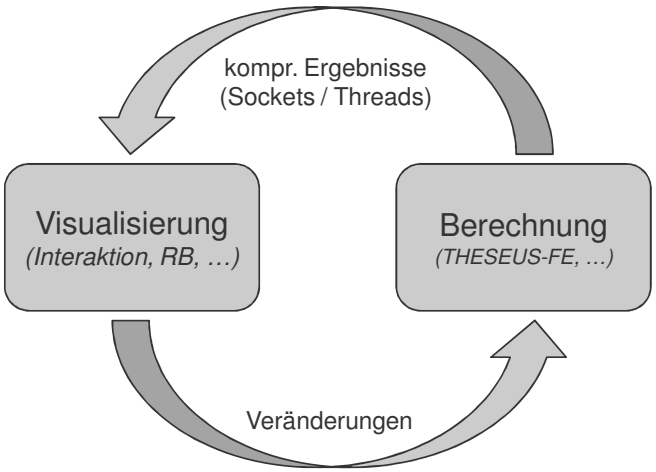


Abbildung 2.6: Computational Steering Konzept

Damit die Interaktivität so weit wie möglich ausgenutzt werden kann, muss die Übergabe der Randbedingungen, sowie das Liefern der Ergebnisse in sehr kurzen Abständen erfolgen. Dies war allerdings in THESEUS-FE nicht vorgesehen; bisher wurde ein langer Simulationszeitraum über mehrere Stunden fest vorgegeben, der anschließend ohne Benutzerinteraktion berechnet wurde.

Deshalb muss für das Computational Steering Umfeld ein anderer Weg gewählt werden. Es wird jeweils nur ein kleiner Berechnungsschritt von wenigen Minuten (oder Sekunden) durchgeführt, und anschließend die Berechnung unterbrochen. Anschließend ist es möglich die Ergebnisse zu visualisieren, die Randbedingungen zu ändern und die Berechnung neu zu starten.

In den nächsten Punkten wird untersucht, ob eine solche Unterbrechung möglich ist und annähernd die gleichen Ergebnisse liefert, wie eine durchgängige Berechnung.

2.4.4 Validierung der Ergebnisse für eine Unterbrechung der Berechnung

2.4.4.1 Beispiel 1: schlagartige Abkühlung

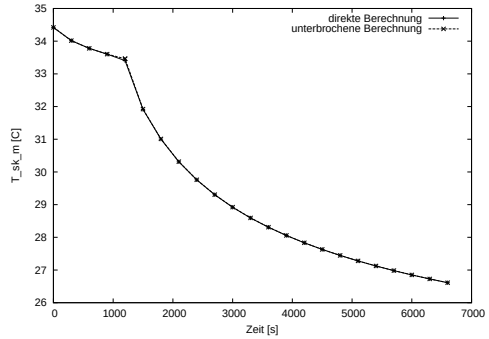
In diesem ersten Beispiel soll eine schlagartige Abkühlung von 24°C auf 5°C simuliert werden. Wie in Abschnitt 2.4.1 beschrieben, bietet THESEUS-FE die Möglichkeit eine ungekoppelte Berechnung durchzuführen. Dies wurde im Folgenden für ein Manikin mit 15 Körperelementen exemplarisch durchgeführt. Dabei wurden folgende Randbedingungen angenommen:

- aktives System
- stehende Position
- Bekleidung $I_{cl,global} = 0.5$ clo
- $T_a = T_w = 24$°C für 20 Minuten (preconditioning), anschließend $T_a = T_w = 5$°C für 90 Minuten (cool down)
- $\varepsilon_w = 0.93$
- relative Luftfeuchte 30%
- $v_a = 0.1$ m/s
- Aktivität 0.8 met

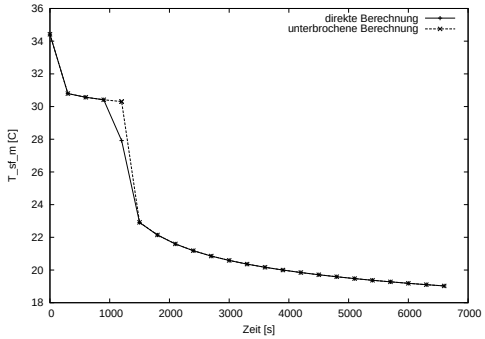
Die Schrittweite für die Berechnung wird intern adaptiv gesteuert, wohingegen das Liefern der Ergebnisse auf alle 300 Sekunden beschränkt wurde. Hierfür ergeben sich die Ergebnisse in Abb. 2.7.

In jedem Diagramm von Abb. 2.7 sind zwei Kurven dargestellt: Jeweils eine für eine nicht unterbrochene Berechnung, welche eine interne Zeittabelle für die Änderung der Wand- und Umgebungstemperatur benutzt, und jeweils eine für eine unterbrochene Berechnung, mit manueller Änderung der Wand- und Umgebungstemperatur und anschließendem Neustart.

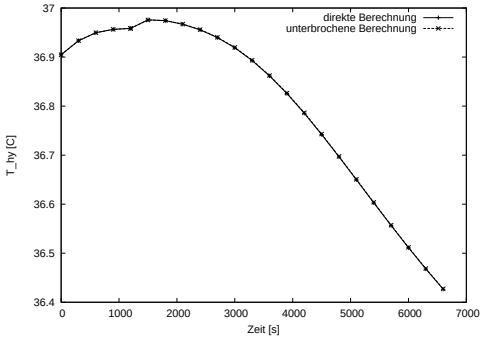
Es ist zu erkennen, dass die Kurven für die mittlere Haut- sowie die Hypothalamustemperatur sehr gut übereinstimmen. Für die metabolische Wärmeerzeugung gab es einen Unterschied beim Zeitpunkt $t = 1200$ s. Dies liegt daran, dass in dem „Restart-File“ für THESEUS-FE nur Temperaturen, nicht aber Ableitungen abgespeichert werden. Es werden also einige interne Zeitschritte benötigt, bis die Ableitungen wieder übereinstimmen und nicht mehr divergieren. Da die Wärmeerzeugung des Metabolismus unter anderem von der Ableitung der Temperatur (also dem Temperaturgradienten) abhängt, ergibt sich hier ein Unterschied, der jedoch zeitlich eingegrenzt werden kann, indem die Postprocessing-Schrittweite kleiner gestellt oder jeder Solver-Zeitschritt mitgeschrieben wird.



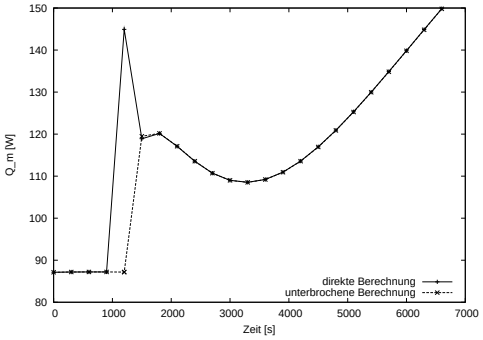
(a) mittl. Hauttemp. $T_{sk,m}$



(b) mittl. Oberflächentemp. $T_{sf,m}$



(c) Hypothalamustemperatur T_{hy}



(d) Metabolische Wärme Q_m

Abbildung 2.7: Vergleiche zwischen Ergebnissen einer direkten (ununterbrochen) Berechnung und einer unterbrochenen Berechnung

Für den Fall der Verkleinerung der Schrittweite ist in Abb. 2.8 der relative Fehler in der metabolischen Wärmeerzeugung aufgetragen. Es ist erkennbar, dass an der Unstetigkeitsstelle der Fehler sehr groß ist, dann aber schnell kleiner wird.

Auch wenn der Fehler anfangs sehr hoch erscheint, ist es doch etwas relativierter zu sehen, da in den Fällen, die für Computational Steering betrachtet werden, keine solchen Extremsituationen auftreten werden. In den meisten Fällen hat das System Zeit, sich langsam an die Umgebung, die sich voraussichtlich nicht schlagartig ändern wird, anzupassen. Zusätzlich könnte eine adaptive Zeitschrittweitensteuerung im Falle der unterbrochenen Berechnung eingebaut werden, um solche Fälle zu vermeiden.

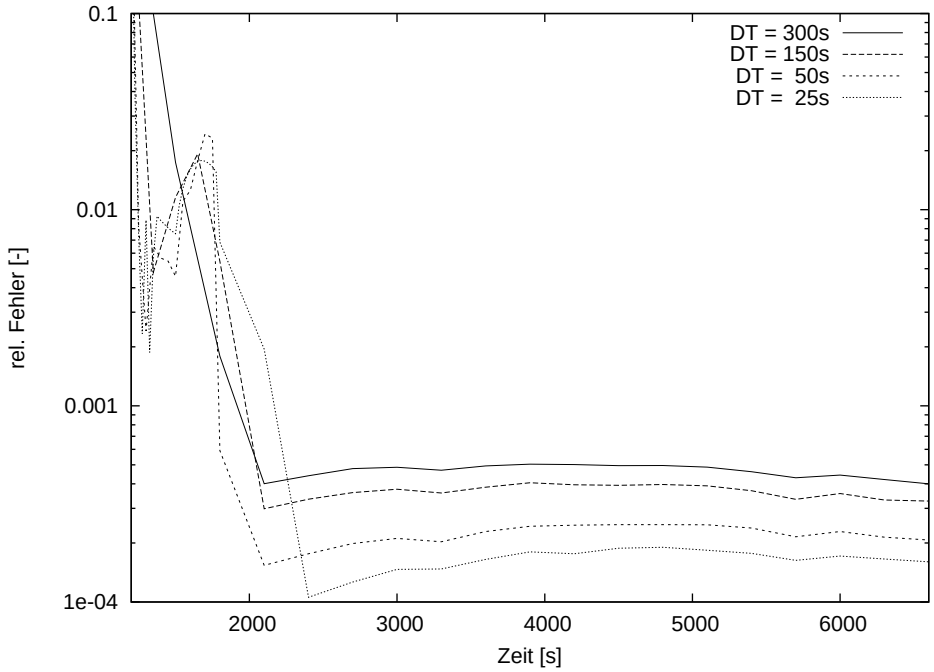


Abbildung 2.8: Relativer Fehler in der metabolischen Wärmeerzeugung Q_m zwischen der ununterbrochenen Berechnung sowie den unterbrochenen Berechnungen mit unterschiedlichen Postprocessing-Schrittweiten

2.4.4.2 Beispiel 2: zeitliche Anpassung

In einem zweiten Beispiel soll nun an einem realistischeren Fall nachgewiesen werden, dass die Unterbrechung und die anschließende Wiederaufnahme der Simulation eines Dummy-Modells, mit geänderten Randbedingungen für die Zwecke von Computational Steering, sehr gut mit Hilfe des ungekoppelten Fiala-Modells von THESEUS-FE berechnet werden kann.

Hierzu sind die Temperaturrandbedingungen T_A sowie T_W in Abb. 2.9 dargestellt. Die restlichen Randbedingungen wurden für dieses Beispiel wie folgt festgelegt (und während der Berechnung nicht verändert):

- aktives System
- stehende Position
- Bekleidung $I_{cl,global} = 0.5$ clo
- $\varepsilon_w = 0.93$
- relative Luftfeuchte 30%

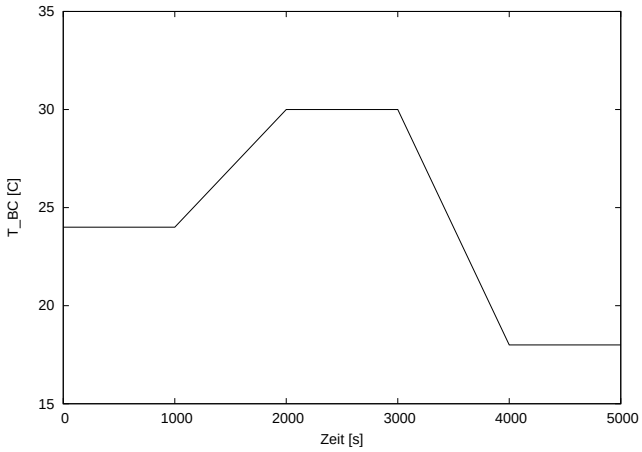


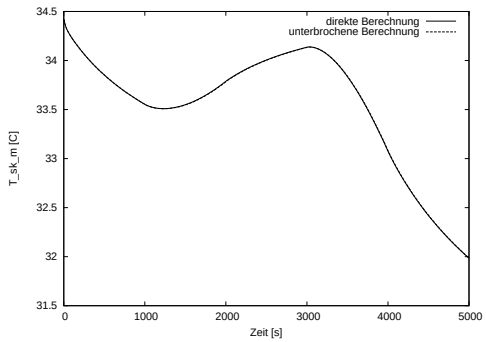
Abbildung 2.9: Temperaturrandbedinungen des Beispiels 2

- $v_a = 0.1$ m/s
- Aktivität 0.8 met

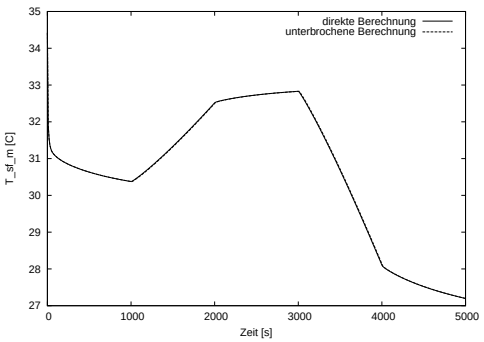
Bei der ununterbrochenen Berechnung wurden die Temperaturen mittels einer internen Zeittabelle festgelegt. Die Postprocessing-Schrittweite beträgt 10 Sekunden und die Endzeit 5000 Sekunden. Diese Lösung dient als Referenz für die im Folgenden stückchenweise ermittelten Ergebnisse.

Bei der unterbrochenen Berechnung wurde alle 10 Sekunden eine Unterbrechung mit anschließender Änderung der Temperaturen durchgeführt, und die Berechnung neu gestartet. Die entsprechenden Wand- und Umgebungstemperaturen wurden über lineare Interpolation aus den Eckdaten bestimmt. Es wurden ebenfalls alle 10 Sekunden Informationen in die Postprocessing-Datei geschrieben. Damit ist ersichtlich, dass für die Berechnungszeit von 10 Sekunden die Wand- und Umgebungstemperatur konstant war, was zu einem stufenförmigen Temperaturverlauf für die Randbedingungen führt (vergleiche ebenfalls Abschn. 3.2.3). Da diese Sprünge aber bei weitem nicht so groß sind, wie in Beispiel 1, gelingt eine bessere Übereinstimmung der Ergebnisse. Dies kann in Abb. 2.10 ersehen werden. Es besteht größtenfalls ein marginaler Unterschied, wie in der Darstellung des relativen Fehlers des Metabolismus in Abb. 2.11b zu erkennen ist.

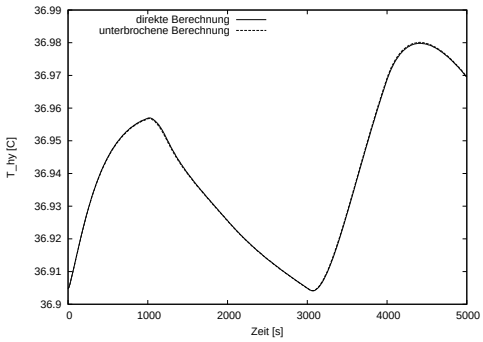
Um die Größenordnung besser abschätzen zu können, sind in Abb. 2.11a die Randbedingungstemperaturen T_W und T_A, die Hypothalamustemperatur T_{hy}, die mittlere Hauttemperatur $T_{sk,m}$ sowie die mittlere Oberflächentemperatur $T_{sf,m}$ dargestellt.



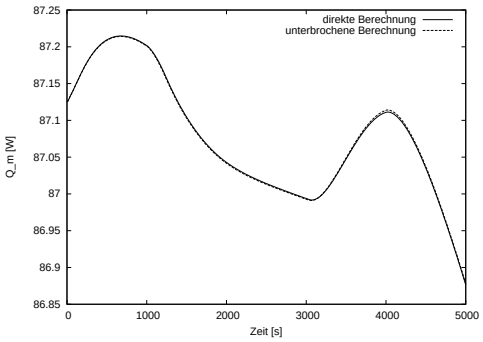
(a) mittl. Hauttemp. $T_{sk,m}$



(b) mittl. Oberflächentemp. $T_{sf,m}$



(c) Hypothalamustemperatur T_{hy}



(d) Metabolische Wärme Q_m

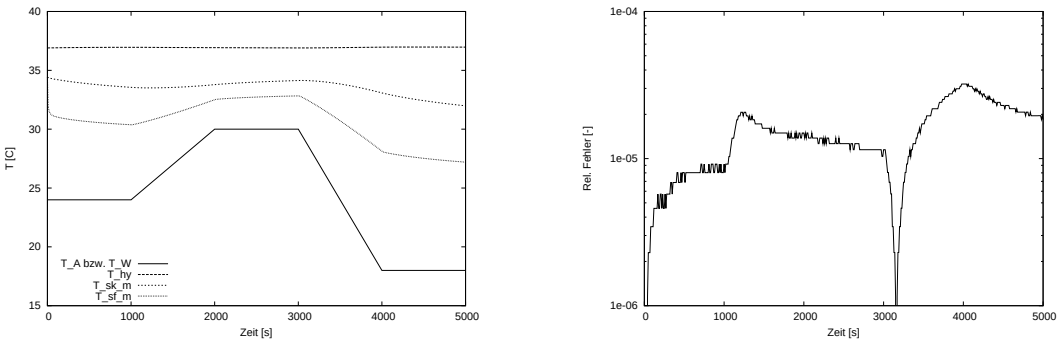
Abbildung 2.10: Vergleiche zwischen Ergebnissen einer direkten (ununterbrochen) Berechnung und einer unterbrochenen Berechnung

2.4.4.3 Ergebnisse

Insgesamt zeigen die Ergebnisse, dass es möglich ist, eine Berechnung zu unterbrechen, Randbedingungen zu ändern, und anschließend die Berechnung auf der Basis der vorherigen Werten unter Einbeziehung der neuen Randbedingungen wieder neu zu starten. Das ist besonders im Hinblick auf die Anwendung von Computational Steering eine der Grundvoraussetzungen, da ansonsten während der Laufzeit des Systems nichts geändert werden könnte.

2.5 Schlußfolgerung

Mit dem Fiala-Modell und dessen Implementierung in THESEUS-FE der Firma P+Z Engineering ist ein mächtiges Werkzeug gegeben, um eine numerische Simulation der menschlichen Thermoregulation auf gute Art und Weise nachzubilden und zu berechnen. In den



(a) Temperaturverläufe in Beispiel 2 **(b)** relativer Fehler in Q_m

Abbildung 2.11: Temperaturverläufe sowie relativer Fehler ΔQ_m in Beispiel 2

weiteren Kapiteln werden nun zusätzliche Teile des benötigten Modells geschildert, und eine Anbindung und Kopplung der einzelnen Komponenten beschrieben.

3 Thermoregulations-Interface

Wie bereits in Kapitel 2 erwähnt, wird das validierte Fiala-Modell mit der Software THESEUS-FE von P+Z Engineering zur Verfügung gestellt. Um dieses Modell voll nutzen zu können, muss eine Interaktion auf Programmierebene möglich sein. Dazu wurde eine Schnittstelle, genannt Thermoregulations-Interface, in `C++` implementiert, mit der THESEUS-FE für weitere `C++` Anwendungen zugänglich gemacht werden kann. Das Konzept, sowie der Aufbau der Schnittstelle wird nun in den folgenden Abschnitten erläutert und dokumentiert.

3.1 Anbindung an THESEUS-FE

Der schematische Aufbau von THESEUS-FE mit seiner eigenen graphischen Benutzeroberfläche und dem Solver wurde bereits in Abb. 2.4 dargestellt. Diese Struktur muss nun entsprechend abgeändert werden um im Rahmen von Computational Steering agieren zu können. Dazu muss sich die Interaktivität allerdings auf die oberste Ebene beschränken, zu der THESEUS-FE und das Thermoregulations-Interface jedoch nicht gehören. Deshalb ist das Anwenden der graphischen Benutzeroberfläche von THESEUS-FE auszuschließen. Das Interface muss demnach direkt an den THESEUS-FE-Solver angebunden werden. Zusätzlich käme noch hinzu, dass in der derzeitigen Version von THESEUS-FE die Benutzeroberfläche nur unter Microsoft Windows läuft. Damit wären sämtliche Linux Anwendungen wie zum Beispiel das am Lehrstuhl für Computation in Engineering der Technischen Universität München entwickelte Strömungssimulationspaket iFluids nicht direkt mit dem Thermoregulations-Interface zu koppeln. Weiterhin ist eine ungekoppelte Berechnung, wie in Absatz 2.4.1 erwähnt, mit der Benutzeroberfläche von THESEUS-FE nicht möglich.

In Abbildung 3.1 ist die Anbindung des Thermoregulations-Interface an THESEUS-FE dargestellt. Die graphische Benutzeroberfläche von Theseus wurde dabei ausgeblendet, da sie nicht verwendet wird. Das Prinzip sieht wie folgt aus:

Über eine externe Anbindung (siehe Kapitel 5) werden aktuelle Randbedingungen an das Thermoregulations-Interface übergeben. Diese werden vom Interface als ASCII kodierte Da-

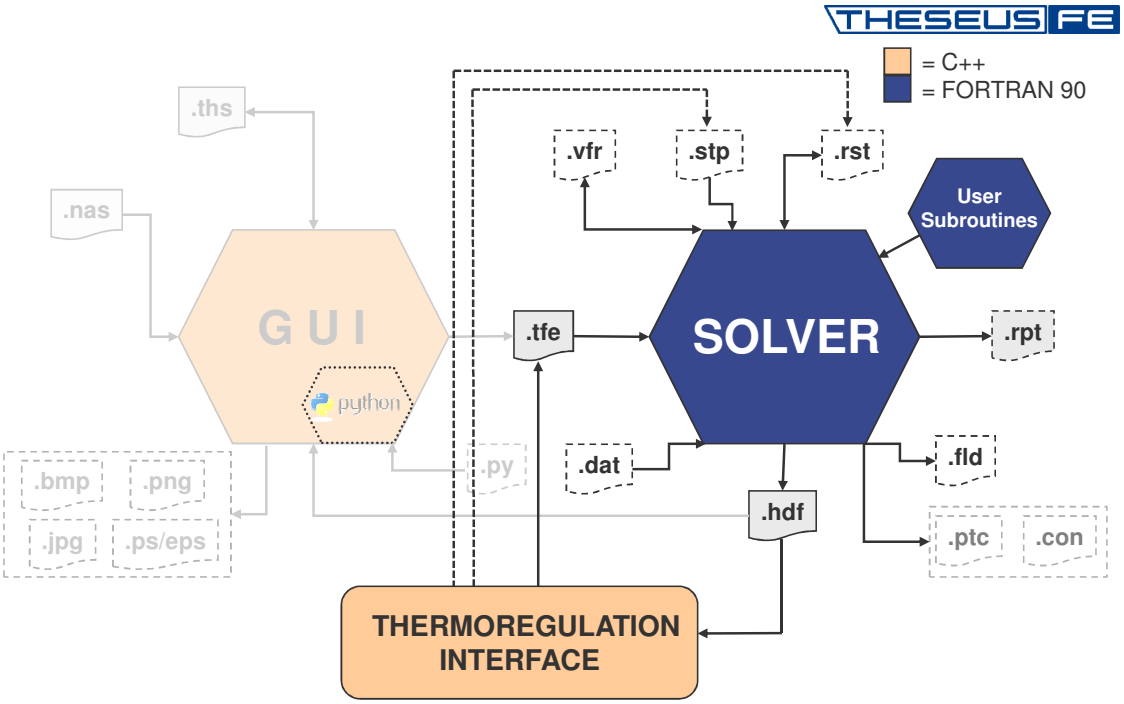


Abbildung 3.1: Anbindung des Thermoregulations-Interfaces an THESEUS-FE (In Anlehnung an [29])

Tabelle 3.1: Randbedingungen für thermische Neutralität

T_{air} [°C]	T_w [°C]	v_{air} [m/s]	rh [%]	ε_w [-]	act_{bas} [met]
30.0	30.0	0.05	40.0	0.93	0.80

tei mit der Endung `.tfe` herausgeschrieben. Dabei handelt es sich um eine speziell formatierte Eingabedatei für THESEUS-FE. Anschließend wird der Berechnungsvorgang gestartet, und der Solver schreibt als Ergebnis eine `.hdf` Datei im binären, komprimierten HDF5-Format (siehe Anhang A.3) heraus. Diese wird vom Thermoregulations-Interface eingelesen, interpretiert, und die Ergebnisse an die aufrufende Anwendung zurückgegeben.

Zusätzlich greift das Interface noch auf zwei weitere Dateien zu, die während vorhergehender Berechnungen vom Solver mitgeliefert wurden. Eine Datei hat die Endung `.stp` und bezeichnet die Setpoint-Temperaturen, sprich die Temperaturen des Körpers bei thermischer Neutralität. Diese werden in einer „steady-state-Berechnung“ unter ausschließlicher Verwendung des passiven Systems bei neutralen Randbedingungen und einem unbekleideten Manikin bestimmt. Jene Randbedingungen können Tabelle 3.1 entnommen werden.

Bei dieser Berechnung müssen die Ergebnisse mit denen in Tabelle 3.2 bis auf geringe Abweichungen übereinstimmen. (Weitere Referenzwerte können Tabelle 4 in [19] entnommen werden). Ist das nicht der Fall, wurde bei der Körperdefinition ein Fehler begangen, der zu

Tabelle 3.2: Ergebnisse bei thermischer Neutralität

T_{hy} [°C]	T_{re} [°C]	$T_{sk,m}$ [°C]	$T_{mus,m}$ [°C]
37.00	36.88	34.40	36.20

finden und zu verbessern ist. Werden die programminternen Werte im Thermoregulations-Interface allerdings nicht verändert, so stimmen die Werte zwangsläufig mit Tabelle 3.2 überein, da keine direkte Änderung der Körpergeometrie vorgenommen, und die Berechnung für die vorliegenden Werte bereits validiert wurde.

Die zweite Datei, auf die zugegriffen wird, trägt die Endung `.rst`. Dort sind Informationen bezüglich der letzten Berechnung abgespeichert, um bei einem Neustart auf den vorhergehenden Berechnungen aufbauen zu können. Nur bei der Ermittlung der Sollwert-Temperaturen und der allerersten transienten Berechnung wird diese Datei nicht benötigt.

Der ganze Ablauf kann anschließend zyklisch, das heißt unter sich ständig ändernden Randbedingungen, ablaufen. Damit sind die Voraussetzungen geschaffen um mit THESEUS-FE Computational Steering zu betreiben.

3.2 Benötigte Daten des Thermoregulations-Interfaces

Wie bereits in 3.1 erwähnt, schreibt das Thermoregulations-Interface die `.tfe` Steuerungsdatei für den THESEUS-FE Solver heraus. Dazu sind aber neben den aktuellen Randbedingungen auch die gesamten Körperdaten von Nöten. Dabei besteht ein großer Vorteil darin, dass Anpassungen des Körpers hinsichtlich Geometrie, physikalischer Parameter, Diskretisierung, etc. jederzeit möglich sind, da die Werte nicht im Solver „fest verdrahtet“ sind, sondern nach jeweiligem Wunsch verändert werden können. Dies zieht aber die schon mehrmals erwähnte Konsequenz mit sich, dass das Modell durch physikalische Messungen neu validiert werden muss.

Andererseits müssen nun auch sämtliche Körperdaten, die vom Solver benötigt werden, im Thermoregulations-Interface vorgehalten werden. Dazu wurde für diese Arbeit eine spezielle Datenstruktur entwickelt und implementiert, die nun im Folgenden dargestellt wird.

3.2.1 Datenstruktur des Körperaufbaus

Die Datenstruktur richtet sich im Grunde nach Abb. 3.2. Eine Hauptklasse `Body` enthält alle Daten, die benötigt werden, um eine `.tfe` Datei zu schreiben. Zusätzlich werden in dieser

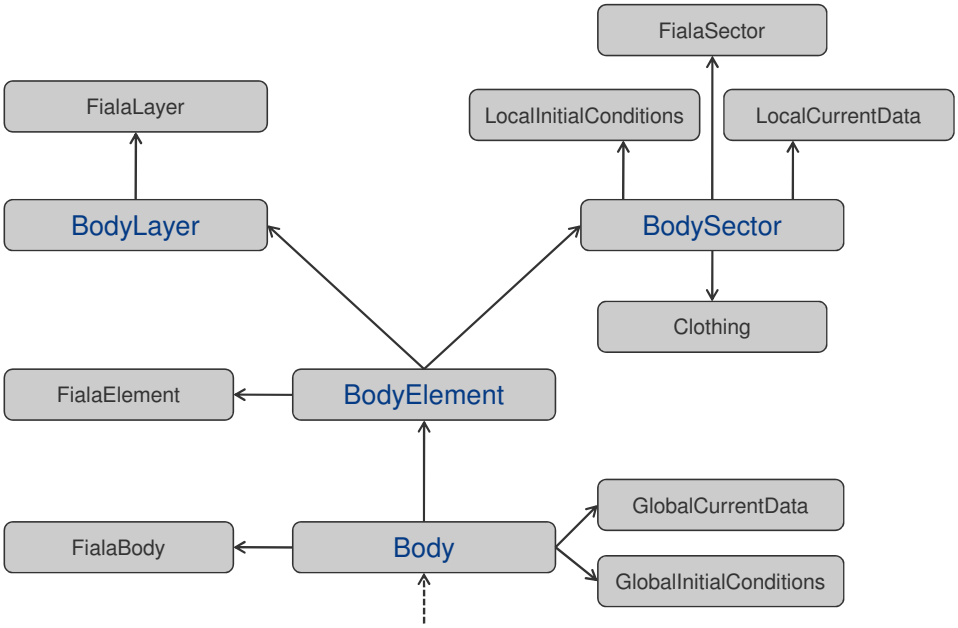


Abbildung 3.2: Datenstruktur des Köperaufbaus

Struktur auch Ergebnisse abgespeichert, die nach erfolgter Berechnung aus der `.hdf` Datei extrahiert wurden.

Die Unterklassen von `Body` wurden in Anlehnung an den Körperaufbau von Fiala entworfen. `Body` enthält eine Liste mit Referenzen zu 15 Klassen vom Typ `BodyElement`. Diese sind wiederum aus einer bestimmten Anzahl Klassen vom Typ `BodySector` und `BodyLayer` aufgebaut.

Wird die Abb. 2.5 auf Seite 18 betrachtet, so kann die Frage aufgeworfen werden, ob es nicht sinnvoller wäre, die Layer den Sektoren, und nicht den Elementen unterzuordnen. Der jeweilige Standpunkt zu dieser Entscheidung ist sicherlich eine Frage der Überzeugung, denn beide Versionen sind zu vertreten und bieten gewisse Vorteile. Die Gründe, weshalb im Rahmen dieser Arbeit folgende Struktur gewählt wurde, werden nun kurz erläutert.

Die Elemente sind in Sektoren unterteilt und die Sektoren in Layer. Alle Sektoren eines bestimmten Elementes enthalten aber die gleichen Layer. Demnach wären in der Datenstruktur die Layer mehrfach vorhanden oder mehrere Verweise (i.e. Pointer) zu einer Layerstruktur nötig, die am Schluss korrekt gelöscht werden müssten. Dies würde sich als schwierig erweisen, da bekannt sein müsste, ob noch ein Sektor auf ein Layer verweist, da ansonsten der Layer nicht gelöscht werden darf. Es ist wesentlich einfacher, wenn die Layer mit den Elementen assoziiert werden. Dort werden sie einmal erzeugt, und anschließend auch einmal gelöscht. Durch Verweise kennt jeder Layer das Element, an dem es hängt, und dadurch hat

er ebenso Zugriff auf die jeweiligen Sektoren und umgekehrt: jeder Sektor kennt demnach sein „Vaterelement“ und hat damit auch Zugang auf die entsprechenden Layer.

Da jetzt die grundlegende Klassenstruktur fest steht, können anschließend die Datencontainer mit den jeweilig benötigten Daten angehängt werden. Diese werden im Folgenden detailliert erläutert.

3.2.1.1 Elemente der Klasse BodyLayer

Die Klasse `BodyLayer` enthält nur ein größeres Unterelement vom Typ `FialaLayer`. Dort werden sämtliche physikalischen Parameter vorgehalten, die für die Definition der Layer benötigt werden. Das wären

name	: der Name des jeweiligen Layers (zum Beispiel: `BONE`, `FAT`, `SKIN`, ...)
disc	: Anzahl der Diskretisierungspunkte pro Layer (≥ 2)
r	: äußerer Radius des jeweiligen Layers [m]
k	: Wärmeleitfähigkeit [W/(m·K)]
rho	: Dichte [kg/m^3]
cp	: Wärmekapazität [J/(kg·K)]
w_bl	: Durchblutungsrate [m^3/(sec·m^3)] Anmerkung: Die Durchblutungsrate ist eine positive, reelle Zahl. Da sie im Thorax allerdings nicht durch eine Zahl, sondern durch `CARDOUT` definiert ist, wurde im Rahmen der Programmierung w_{bl} nicht als `float`, sondern als `const char*` implementiert. Damit musste der Thorax nicht als Spezialfall behandelt werden.
q_m	: Basiswert für den Metabolismus [W/m^3]
a_rsp	: Verteilungskoeffizient für die Atmung

Für weitere Details, siehe ebenfalls [27] Keyword `MANFLAY`.

Zusätzlich wird noch die ID des Layerelementes gespeichert sowie ein Pointer auf das jeweilige Element, das diesen Layer enthält.

Anmerkung zur ID: Die ID's sind aus dem THESEUS-FE Modell übernommen worden und beginnen stets mit 1.

3.2.1.2 Elemente der Klasse BodySector

Die Klasse `BodySector` enthält wesentlich mehr Daten. Zunächst werden dort ebenfalls die vom Fiala-Modell benötigten Daten in der Klasse `FialaLayer` abgespeichert:

name	: der Name des jeweiligen Sektors (zum Beispiel: `ANTERIOR`, ...)
phi	: Sektor-Winkel φ [°]
vf1	: Viewfaktor ψ_{sf-w} [-] für das *sitzenden* Modell (nach Fiala)
vf2	: Viewfaktor ψ_{sf-w} [-] für das *stehende* Modell (nach Fiala)
e_sf	: Emissivitätswert der Körperoberfläche [-]
l	: individuelle Sektorlänge [m] Anmerkung: Die individuelle Sektorlänge wird nur benötigt, falls ein Element in zwei oder mehrere Unterelemente aufgeteilt wird (Beispiel: linker Arm ist aufgeteilt in linker Oberarm und linker Unterarm). `-1` wird eingetragen, wenn der Wert nicht definiert, und die Standardlänge des Elementes herangezogen werden soll.

Für weitere Details, siehe ebenfalls [27] Keyword `MANFSEC`.

Zusätzlich werden auch Bekleidungsinformationen in der Klasse `Clothing` abgespeichert. Diese hängen aus folgendem Grund an den Sektoren und nicht an den Elementen:

Wie in obiger Anmerkung bereits erwähnt, ist es möglich, ein Element noch einmal zu unterteilen. Als Beispiel soll hier der Arm gewählt werden, der in einem Element definiert und in sechs Sektoren mit jeweils drei für Oberarm und drei für Unterarm aufgeteilt ist. Wäre die Bekleidung auf Elementebene definiert, so könnte immer nur der gesamte Arm bekleidet sein, oder auch nicht. Eine Unterscheidung in „nur Oberarm bekleidet“ für den Sommerfall wäre nicht möglich. Deshalb wird die Bekleidung auf Sektorebene definiert.

Zeigt der Verweis der Bekleidungsinformation auf `NULL`, so wird die Oberfläche als nicht bekleidet angesehen. Die Klasse `Clothing` enthält folgende physikalische Parameter:

thickn	: die Dicke der Bekleidungsschicht [m]
sicl	: $\sum I_{cl}^{*}$ [clo]
siicl	: $\sum(I_{cl}^{*}/i_{cl}^{*})$ [clo]
nsm	: körperunabhängige Masse (non-structural mass) [kg/m^2]
cp	: Wärmekapazität [J/(kg·K)]

Für weitere Details, siehe ebenfalls [27] Keyword `MANFSCL`.

Eventuelle Probleme mit Bekleidungsänderungen

Bei der Definition der Bekleidung muss noch auf ein weiteres Problem hingewiesen werden, was genaueren Überlegungen bedarf.

Die Bekleidungsschicht wird, genau wie die Layer, mit einer bestimmten Anzahl diskreter Punkte berechnet. Ist eine bestimmte Bekleidungsinformation vorhanden, so enthält das Modell dementsprechend auch Freiheitsgrade aus den Bekleidungsschichten. Diese würden bei der Bestimmung der neutralen Temperaturen mit berücksichtigt werden. Da die thermische Neutralität aber bei einer unbekleideten Person definiert ist, kann die Bekleidung nicht von Anfang an eingeschaltet werden. Das Dilemma allerdings ist, dass die Setpoint-Datei, die die thermisch neutralen Temperaturen enthält, genau so viele Freiheitsgrade haben muss, wie die spätere Berechnung. Weiterhin darf die Anzahl der Freiheitsgrade von einer Berechnung zur nächsten nicht variieren, da die Anzahl ansonsten in der Neustart-Datei (`.rst`) nicht übereinstimmt. Die Lösung dieses Problems, die im Thermoregulations-Interface implementiert wurde, lautet folgendermaßen:

Der gesamte Körper wird in jedem Sektor mit einer Bekleidungsschicht bedeckt. Allerdings werden alle physikalischen Werte auf $1 \cdot 10^{-5}$ gesetzt. Damit fällt die isolierende Wirkung der Bekleidung weg; thermisch gesehen ist sie nun nicht vorhanden, sodass die neutralen Temperaturen richtig berechnet werden können. Vom Standpunkt der Freiheitsgrade aus ist alles vorhanden, um Bekleidung während einer laufenden Simulation willkürlich über den gesamten Körper zu verteilen und anschließend wieder zu entfernen. Um die Bekleidung in einem Sektor einzuschalten, werden nur die korrekten physikalischen Werte eingesetzt.

Zusätzlich zu den globalen Randbedingungen in der Klasse `Body` können auch lokale Randbedingungen in der Klasse `LocalInitialConditions` definiert werden. Solche lokalen Randbedingungen werden beim Schreiben der `.tfe` Datei berücksichtigt, falls deren Verweis nicht auf `NULL` zeigt. Folgende Parameter können in dieser Klasse gesetzt werden:

t_w	: individuelle Wandtemperatur [°C]
t_a	: individuelle Raumtemperatur [°C]
v_a	: individuelle Windgeschwindigkeit [m/s]
rh	: individuelle relative Luftfeuchte [%]

Für weitere Details, siehe ebenfalls [27] Keyword `MANFSBC`.

Weiterhin hängt eine Klasse `LocalCurrentData` an den Sektoren, die die jeweiligen, gegenwärtigen Berechnungsergebnisse enthalten. Die Ergebnisse werden vom Thermoregulations-Interface aus der von THESEUS-FE geschriebenen `.hdf` Datei extrahiert und entsprechend

in die Datenstruktur eingeordnet. Allerdings werden nur die jeweils letzten Berechungsergebnisse in der Struktur abgespeichert. Sollen Daten zur Verfügung gestellt werden, um zum Beispiel eine zeitliche Entwicklung einer Temperatur zu beobachten, so müssen die dazu benötigten Werte außerhalb des Thermoregulations-Interface gesammelt und abgespeichert werden.

Das Grundprinzip beruht darauf, dass Berechnungsergebnisse, die auf Sektorebene definiert sind, in `LocalCurrentData`, und Informationen, die für den gesamten Körper gelten in `GlobalCurrentData` gespeichert werden (dazu später mehr).

Die Daten, die in `LocalCurrentData` abgelegt werden, können im Prinzip beliebig gewählt werden. Zur Zeit werden nur einige Daten in diese Struktur übernommen:

t_sf	: Oberflächentemperatur [°C]
t_sk	: Hauttemperatur [°C]
t_eq	: Äquivalenttemperatur [°C] (siehe Anhang A.4)
lmv	: Local Mean Vote [-] (nach dem Fraunhofer Institut für Bauphysik in Holzkirchen)

Sollen andere Werte später auch in der graphischen Benutzeroberfläche der „Virtual Climate Chamber“ (siehe Kapitel 6) mit angezeigt werden, so müssen die Namen noch in den entsprechenden Klassenfunktionen ergänzt werden. Es ist dabei darauf zu achten, dass die ID's mit 0 beginnen müssen, und die entsprechende Anzahl der Werte zudem in `getAmountLCDVal` angepasst werden muss. Zusätzlich müssen die Daten ebenso mit Ergebnissen belegt werden, was in die Funktion `TheseusModel::parseTheseusResults` im Thermoregulations-Interface mit eingebaut werden muss.

Außerdem enthält jeder Sektor noch eine ID (beginnend mit 1) mit der er eindeutig innerhalb des Elementes definiert wird, und einen Pointer auf das Element, welches diesen Sektor beinhaltet.

Damit wären sämtliche Unterklassen und Werte der Klasse `BodySector` beschrieben.

3.2.1.3 Elemente der Klasse BodyElement

Die Klasse `BodyElement` besitzt neben einer ID (beginnend mit 1), einen Pointer auf den Körper und eine Unterklasse `FialaElement`, die folgende Werte des Fiala-Modells für die Elemente enthält:

name	: der Name des Elementes (zum Beispiel: `HEAD`, `FACE`, ...)
type	: der Typ des Elementes (`SPHERE` oder `CYLINDR`)
l	: die Länge des Elementes [m] (bei Typ `SPHERE` : l = 0.0)
h_x	: Gegenstromwärmeaustauschkoeffizient (CCX) [W/K]
a_sk	: Verteilungskoeffizient für mittlere Hauttemp. (nach Fiala)
a_dl	: Verteilungskoeffizient für die Vasodilatation
a_cs	: Verteilungskoeffizient für die Vasokonstriktion
a_sh	: Verteilungskoeffizient für Zittern
a_m_w1	: Verteilungskoeffizient für mechanische Arbeit (sitzendes Modell)
a_m_w2	: Verteilungskoeffizient für mechanische Arbeit (stehendes Modell)
a_nat	: Regressionskoeffizient für natürliche Konvektion
a_frc	: Regressionskoeffizient für erzwungene Konvektion
a_mix	: Regressionskoeffizient für gemischte Konvektion

Für weitere Details, siehe wiederum [27] Keyword `MANFBEL`.

Sämtliche Koeffizienten können entweder [19] entnommen werden, oder wurden in [28] zusätzlich angepasst. Im Thermoregulations-Interface wurden die Werte von einem validierten 15-elementigen Körperaufbau aus THESEUS-FE übernommen.

Weiterhin enthält es Listen mit den oben definierten Klassen `BodyLayer` und `BodySector`. Wie ebenfalls oben beschrieben, sind die internen ID's der Layer und Sektoren aus dem THESEUS-FE Modell entlehnt worden und beginnen jeweils mit 1. Intern werden sie aus `C/C++` Gründen in der Liste jedoch mit 0 beginnend referenziert. Den Zugriffsfunktionen `getBodyLayer` sowie `getBodySector` werden allerdings die THESEUS-FE ID's übergeben, um einen Pointer auf den gewünschten Layer oder Sektor zu erhalten.

3.2.1.4 Elemente der Klasse Body

Zusätzlich zu der bereits oben beschriebenen Klasse `BodyElement`, enthält die Klasse `Body` noch folgende Informationen.

Wie in den anderen Klassen auch, existiert eine Klasse `FialaBody`, die noch zwei weitere Informationen für den Körper beinhaltet:

system	: zu berechnendes System: `ACTIVE` oder `PASSIVE`
position	: Position des Manikins: `STND` (stehend) oder `SED` (sitzend)

Außerdem gibt es noch zwei unentbehrliche Klassen namens `GlobalInitialConditions` und `GlobalCurrentData`, die ebenfalls an der Klasse `Body` hängen.

Die `GlobalInitialConditions` enthalten die, für die folgende Berechnung gültigen, globalen Randbedingungen. Es können (und müssen) folgende Werte definiert werden:

t_w	: Wandtemperatur (für Strahlungswärmeaustausch) [°C]
e_w	: Strahlungsemissivität der Wand [-]
t_a	: Raumtemperatur (für Konvektionswärmeaustausch) [°C]
v_a	: Windgeschwindigkeit [m/s]
rh	: relative Luftfeuchte [%]
act	: metabolische Körperaktivität [met]
global_icl	: globaler Kleidungsisolationswert [clo] (für PMV-Berechnung benötigt)

Für weitere Details, siehe [27] Keyword `MANIKIN`.

Wie bereits vorher erwähnt, werden sämtliche Werte, die für den gesamten Körper gelten in `GlobalCurrentData` abgespeichert. Wie bei `LocalCurrentData` können auch hier noch zusätzliche Daten angehängt werden. Dabei sind die Klassenfunktionen in `GlobalCurrentData`, die die Namen bereitstellen, entsprechend zu ergänzen, damit die Benutzeroberfläche automatisch die neuen Werte mit darstellen kann. Die Werte müssen zudem noch in `TheseusModel::parseTheseusResults` mit Berechnungswerten belegt werden. Zur Zeit werden folgende Ergebnisse im Thermoregulations-Interface aus der `.hdf` Datei extrahiert:

time	: gegenwärtige Zeit [s] (die am Ende der Berechnung herrscht)
t_hy	: Hypothalamustemperatur [°C]
t_sk_m	: gemittelte Hautoberflächentemperatur [°C]
t_sf_m	: gemittelte Oberflächentemperatur [°C] Anmerkung: wurde keine Kleidung definiert, so sind beide Temperaturen identisch
t_eq_m	: gemittelte Äquivalenttemperatur [°C] (siehe Anhang A.4)
Q_m	: gesamte metabolische Wärmeerzeugung [W]
PMV	: Predicted Mean Vote (nach Fanger) [-] (-3 .. 3)
PPD	: Predicted Percentage of Dissatisfied [%]
DTS	: Dynamic Thermal Sensation Index [-] (nach Fiala) (-3 .. 3)
TS	: Static Thermal Sensation Index [-] (nach Fiala) (-3 .. 3)

Soll ein Pointer auf ein Körperelement mit der THESEUS-FE ID geliefert werden, so kann die Funktion `getBodyElement` benutzt werden.

Des weiteren besitzt jeder Körper noch eine eindeutige ID. Es wäre theoretisch möglich, mehr als ein Manikin gleichzeitig zu berechnen. Viel zusätzlicher Aufwand würde nicht benötigt werden, um das Thermoregulations-Interface auf mehrere Körper auszudehnen. Zur Zeit wird allerdings nur mit einem Manikin gerechnet.

Für alle Header-Definitionen zu den oben aufgeführten Klassen, kann der gut kommentierte Quellcode eingesehen werden. (vgl. Anhang B.3 und Quellcode).

3.2.2 Datenstruktur des Thermoregulations-Interfaces

Im Folgenden wird der Aufbau des Thermoregulations-Interfaces an sich erläutert. Die Datenstruktur ist in Abb. 3.3 dargestellt.

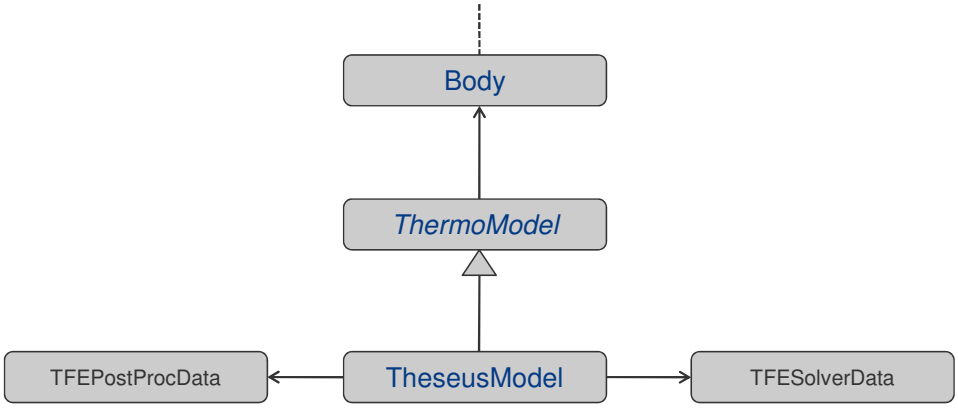


Abbildung 3.3: Datenstruktur des Thermoregulations-Interfaces

Es existiert eine abstrakte Klasse `ThermoModel`, die eine definierte, nach außen sichtbare Schnittstelle bieten soll. An dieser Stelle sei auch erwähnt, dass die Datenstruktur dafür ausgelegt sein soll, dass sie später eventuell an ein anderes Berechnungsmodell gekoppelt werden kann. Als Beispiel sei das einfache Zwei-Knoten-Modell von Gagge [15] erwähnt. Dies würde mehr oder weniger zum Vergleich verschiedener Modelle dienen.

Eine Übersicht über alle öffentlich zugänglichen Funktionen der Klasse befindet sich in Tabelle 3.4. Sie enthält eine virtuelle Funktion `computeUpdate`, die einen neuen Berechnungsschritt

Tabelle 3.4: Übersicht über alle öffentlichen Funktionen der Klasse `ThermoModel`

`computeUpdate`	eigentliche Funktion, in der die Berechnung der Thermoregulation durchgeführt wird
`setInitialConditions`	setzt globale Randbedingungen
`getThermoBody`	liefert einen Pointer auf die Datenstruktur des Körpers
`getAmountLocalCurrentData`	liefert Anzahl der Sektoren (standardmäßig 48)
`getCurrentGlobalData`	liefert einen Pointer auf die gegenwärtigen globalen Ergebnisse
`getCurrentLocalData`	liefert alle gegenwärtigen lokalen Daten (über 48 Sektoren) für den gewünschten Datentypen in einem Vektor
`getAllCurrentLocalData`	liefert alle gegenwärtigen lokalen Daten (über 48 Sektoren) für alle Datentypen in einer Matrix
`getCurrentTime`	liefert die gegenwärtige Zeit (i.e. gegenwärtiger Stand der Berechnungen)
`createThermoBody`	generiert den Körperaufbau für ein spezielles Datenkonzept

ausführt, eine virtuelle Funktion `createThermoBody`, die für den korrekten Körperaufbau zuständig ist, sowie die Körperdefinition für einen Typ `Body` und diverse Zugriffsfunktionen.

Von dieser abstraken Klasse wird die Klasse `TheseusModel` abgeleitet. Diese beinhaltet alle weiter notwendigen Spezifikationen. In den folgenden Punkten wird der Aufbau und die Funktionsweise dieser Klasse erläutert.

3.2.2.1 Elemente der Klasse TFESolverData

Die Klasse `TheseusModel` schließt die Unterklasse `TFESolverData` ein. Diese Klasse definiert die Spezifikationen, die für die Solver-Auswahl und Parameter benutzt werden. Sie befindet sich im THESEUS-FE Modell, da sie nur vom THESEUS-FE Solver benötigt wird, und logisch gesehen, nichts mit dem Körperaufbau zu tun hat. Die wichtigsten Elemente sind nachfolgend aufgezählt:

solverType : Art des Solvers: 0 (Trans2: Transienter, impliziter Solver zweiter Ordnung), 1 (Steady: steady-state Solver)

tend : Endzeit der Berechnung [s]
Anmerkung: Bei Anwendung des steady-state Solvers sollte diese Zeit 1.0 betragen

tinitfe : initiale Temperaturen von allen FE Elementen [°C]
Anmerkung: Diese Angabe ist nur bei einer steady-state Berechnung nötig und wird zur Ermittlung der neutralen Temperaturen auf 34°C gesetzt. Soll keine Temperatur definiert werden (Normalfall), so wird sie auf -1.0 gesetzt.

Die restlichen Einstellungen können dem kommentierten Quellcode (Anhang B.4) entnommen werden. Bei der Erzeugung der Klasse werden Standardwerte eingetragen, und brauchen für die Zwecke im Rahmen dieser Arbeit nicht weiter betrachtet werden. Weitere Informationen finden sich auch in [27] unter dem Keyword `SOL`.

Es ist ebenfalls darauf zu achten, dass in THESEUS-FE zwei Zeitskalen existieren. Die eine Skala gilt strikt nur für die laufende Instanz von THESEUS-FE, und beginnt stets bei Null. Die Endzeit der Berechnung `tend` bezieht sich auf diese Skala. Die zweite Skala hängt mit der Zeitausgabe in der Ergebnisdatei zusammen. Diese startet bei der ersten Berechnung bei Null und wird dann, anhand der Restart-Datei, jedesmal an die vorhergehende Berechnung angepasst. Folgendes Beispiel soll diesen Effekt zeigen:

Die erste Berechnung wird über 20 Sekunden durchgeführt. Dabei lautet die erste Konfiguration für `tend = 20`. In der Ergebnisdatei erscheint nun als Ende der Berechnung 20 Sekunden. Die zweite Berechnung, die aufbauend auf der ersten durchgeführt wird, soll über 30 Sekunden gehen. Dann ist für `tend = 30` einzutragen, obwohl am Ende in der Ergebnisdatei 50 Sekunden als Endzeit stehen wird.

3.2.2.2 Elemente der Klasse TFEPostProcData

Die Klasse `TFEPostProcData` enthält sonstige zusätzliche Einstellungen zum Verhalten des Solvers. Über `posFrqType` kann bestimmt werden, in welchen Intervallen Ergebnisse in die `.hdf` Datei geschrieben werden sollen. Standardmäßig wird am Ende einer Berechnung eine Ausgabe gemacht. Das reicht im vorliegenden Fall aus, da sowieso kleine Zeitschritte berechnet werden sollen. Ebenso wird ein Verlangsamen der Berechnung durch unnötiges Herausschreiben von Daten verhindert.

Weiterhin können hier noch Einstellungen vorgenommen werden, zum Beispiel: ob eine `.rst` Neustart-Datei herausgeschrieben, ob der PMV und der PPD Wert berechnet werden soll,

Tabelle 3.6: Übersicht über alle öffentlichen Funktionen der Klasse `TheseusModel`

`setClothingDefaultEmpty`	alle Kleidungsinformationen entfernen
`setClothingDefaultSummer`	Standard-Sommerbekleidung benutzen
`setClothingDefaultWinter`	Standard-Winterbekleidung benutzen
`setSetPointBaseName`	Name der Sollwert-Temperatur Datei
`setTfeBaseName`	Name der `.tfe` Datei, die für THESEUS-FE erstellt wird (Standard ist „`outfile`“)
`setGlobalPath`	globaler Pfad, in dem sämtliche, erstellten Dateien abgelegt werden
`switchPMVflag`	Einstellung, ob PMV und PPD berechnet werden sollen
`verboseTfe`	Einstellung, ob Kommentare in der `.tfe` Datei vorhanden sein sollen
`computeUpdate`	eigentliche Implementation der Berechnungsfunktion (speziell für THESEUS-FE)
`getCurrentClothSIcl`	Liefern von 48 I_{cl} Werten der Bekleidung in einem Vektor (für graphische Darstellung)

und ob der Manikin im Debug-Modus arbeiten soll, bei dem deutlich mehr Information herausgeschrieben werden.

3.2.2.3 Elemente der Klasse TheseusModel (und ThermoModel)

In Tabelle 3.6 ist eine Übersicht über alle Funktionen von `TheseusModel` zu sehen. Wie bereits oben erwähnt, ist die Klasse `TheseusModel` die Hauptklasse in der die gesamte Kopplung mit THESEUS-FE definiert wird. Da sie von der abstrakten Klasse `ThermoModel` abgeleitet ist, besitzt sie ebenfalls sämtliche Elemente dieser Klasse. Alle Elemente können im Quellcode im Anhang B.4 betrachtet werden. Hier werden nur die wichtigsten Funktionen mit ihren jeweiligen Wirkungsweisen noch einmal im Detail erläutert.

Im Konstruktor der Klasse wird die THESEUS-FE-Körperstruktur mit allen vom Solver benötigten Werten automatisch angelegt und in `thermoBody` abgespeichert. Über `getThermoBody` erhält man einen Pointer auf die Struktur. Standardmäßig ist das Manikin-Modell nackt, und besitzt weder lokale noch globale Randbedingungen. Die globalen Randbedingungen müssen an das Modell übergeben werden, bevor irgendeine Berechnung stattfinden kann.

Die Funktionen vom Typ `setClothing...` definieren die Standardkleidungslayer, die aus THESEUS-FE übernommen wurden. Da es sich nicht um einzelne Kleidungsstücke pro Sek-

tor, sondern um ganze Kleidungsensembles handelt, ist es nicht trivial die korrekten physikalischen Werte zu bestimmen. Deshalb wurde im Rahmen dieser Arbeit entschieden, nur zwei, unter Mithilfe von D. Fiala validierte Ensembles zu benutzen. Bei der Sommerkleidung beträgt die Dicke des T-Shirts an den Oberarmen zirka 3.5 cm. Das ist in so fern richtig, als dass nicht nur Kleidung berücksichtigt wurde, sondern ein globaler Wert für ein Ensemble an der Stelle angegeben ist, welches zudem eine Luftschicht mit berücksichtigt. Dabei wird davon ausgegangen, dass die Ärmel des T-Shirts von den Armen leicht abstehen.

Wird über `setSetPointBaseName` ein Dateiname angegeben, so wird dieser als Sollwert-Datei für die Berechnung verwendet. Wird keine Datei angegeben, so wird als erster Schritt die Sollwert-Datei für thermische Neutralität bestimmt. Das dauert zwar geringfügig länger (ca. 0.5 Sekunden), ist aber wesentlich robuster, da sichergestellt werden kann, dass die korrekte Anzahl Freiheitsgrade benutzt wird. Am Ende dieser Berechnung wird zudem eine Kontrolle der neutralen Berechnungsergebnisse durchgeführt und ein Protokoll im Berechnungsverzeichnis hinterlegt, das die Abweichungen von den validierten Fiala-Temperaturen aus Tabelle 3.2 angibt. Diese Datei mit der Endung „`.report_of_thermal_check`“ sollte immer überprüft werden, um sicher zu stellen, dass mit korrekten physikalischen Ansätzen gerechnet wurde. Als Beispiel ist eine Hypothalamustemperatur von 56.5°C nicht sehr sinnvoll.

Der globale temporäre Pfad, der mittels `setGlobalPath` gesetzt werden kann, gibt an, in welches Verzeichnis THESEUS-FE Dateien geschrieben werden sollen. Es muss (unter Linux) darauf geachtet werden, dass der Benutzter an der Stelle auch Schreibrechte besitzt. Wird kein Pfad gesetzt, so werden alle Dateien in das Verzeichnis geschrieben, von dem aus das Programm aufgerufen wurde. Das würde prinzipiell funktionieren, könnte aber für etwas Unordnung sorgen. Deshalb ist dringend anzuraten einen korrekten Pfad anzugeben.

Über die Funktion `verboseTfe` kann gesteuert werden, ob noch zusätzliche Kommentare in die `.tfe` Datei geschrieben werden sollen. Die Kommentare sind sehr nützlich für die Überprüfung der Datei. Vom Solver werden sie allerdings nicht benötigt. Standardmäßig werden die Kommentare mitgeschrieben und müssen explizit mit der oben beschriebenen Funktion unterdrückt werden, wenn keine erwünscht sind.

Die globalen Randbedingungen können über zwei verschiedene Wege gesetzt werden. Die einfachste Version benutzt die Funktion `setInitialConditions`. Dort können alle benötigten Werte übergeben werden. Die zweite Variante ist das manuelle Anhängen von `GlobalInitialConditions` an `thermoBody`, auf den über `getThermoBody` zugegriffen werden kann. Das ist zur Zeit ebenfalls die einzige Möglichkeit `LocalInitialConditions` auf Körpersektoren zu definieren.

Die eigentlichen Berechnungsschritte werden eingeleitet, sobald die Funktion `computeUpdate` aufgerufen wird. Als einziger Parameter wird die zu berechnende Zeitschrittweite *dt* benötigt. Die Anmerkungen zur Zeitschrittweite in Punkt 3.2.3 sind unbedingt zu beachten. Wird die Funktion aufgerufen, so werden zuerst die jeweiligen, gültigen Dateinamen für die `.tfe`, `.rst` und `.stp` Dateien bestimmt. Ist noch keine Sollwert Datei vorhanden, so wird in einem eigenen Schritt die thermische Neutralität berechnet, die für alle weiteren Berechnungen benutzt wird. Anschließend wird die `.tfe` Datei in das Verzeichnis, das vom globalen Pfad vorgegeben wurde, geschrieben, und THESEUS-FE mittels eines externen Programmaufrufs gestartet. Am Ende der Berechnung wird die vom Solver erzeugte `.hdf` Datei eingelesen, und die Ergebnisse in die Körperstruktur geschrieben. Damit sind die neuen `CurrentGlobalData` und `CurrentLocalData` mit aktuellen Werten belegt.

Über `getCurrentGlobalData` kann in der Schnittstelle direkt auf die globalen Ergebnisse zugegriffen werden. Es ist aber auch möglich, die Daten direkt aus der `thermoBody`-Struktur herauszulesen.

Für die lokalen Sektorergebnisse stehen in der Schnittstelle zwei Funktionen zur Verfügung. Über `getAllCurrentLocalData` werden alle, für diesen Zeitschritt gültigen Ergebnisse für alle Typen (Hauttemperatur, Oberflächentemperatur, Äquivalenttemperatur, etc.) und alle Sektoren (48 beim Standardmodell) in einer Matrix zurückgeliefert (siehe Anhang B.1 für eine Beschreibung der Reihenfolge). Die Matrix muss vorher bereits mit der entsprechenden Größe erstellt worden sein. Falls Unsicherheit über die Anzahl der Sektoren im Modell besteht, kann diese Anzahl mit der Funktion `getAmountLocalCurrentData` bestimmt werden. Wird nur eine Datenreihe (zum Beispiel die Oberflächentemperatur) benötigt, so kann diese mittels `getCurrentLocalData` in ein eindimensionales Feld gespeichert werden (das ebenfalls vorher erstellt werden muss). Dazu bedarf es unter anderem der ID der Datenreihe, die in der Definition der Klasse `LocalCurrentData` gefunden werden kann (siehe Anhang B.3.5).

3.2.3 Probleme bei der Auswahl der Zeitschrittweite

Besonderes Merkmal muss auf die Wahl der Berechnungszeitschrittweite gelegt werden. Einige Probleme wurden schon unter 2.4.4.1 angesprochen und sollen nun noch einmal aufgegriffen werden.

In Abb. 3.4 wird das Problem bei der Wahl der Zeitschrittweite deutlich. Die gedachte Verteilung einer Randbedingungstemperatur verläuft linear von einem Temperaturniveau auf das nächste. In Wirklichkeit bleiben jedoch sämtliche Randbedingungen, während einer Berechnung mit Zeitschritt *dt* konstant und steigen schlagartig von einem Wert auf den nächsten

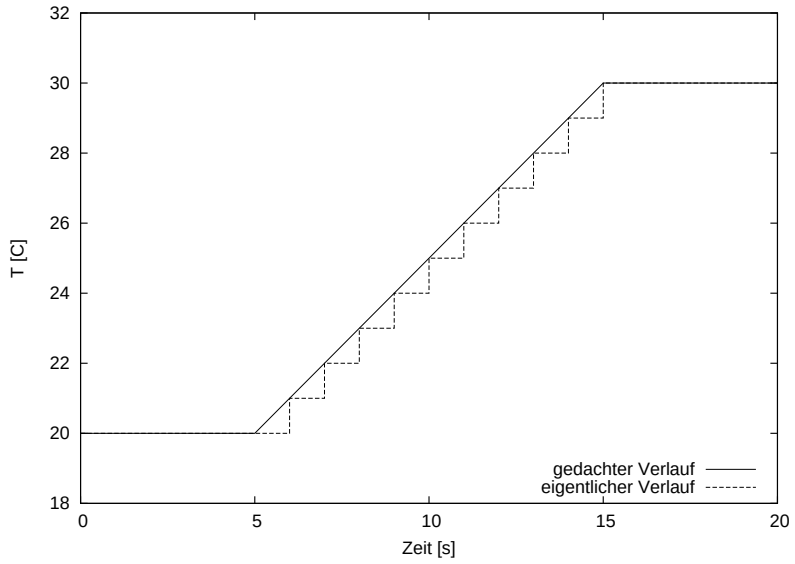


Abbildung 3.4: Probleme bei der Anpassung der Zeitschrittweite dt

an. Wird dt zu groß gewählt, so wird der Unterschied zum wirklichen Temperaturverlauf zu groß, und die Werte würden sehr ungenau werden. Wird der Zeitabstand zu klein festgelegt, so sind die Ergebnisse zwar genau, allerdings benötigt die Berechnung sehr lange.

Ist die Änderung der Temperatur nicht sehr groß oder annähernd konstant, so kann eine große Schrittweite gewählt werden. Sind jedoch große Gradienten vorhanden, so muss die Zeitschrittweite verkleinert werden. Welcher Wert angemessen ist, kann nicht pauschal angegeben werden, sondern muss von System zu System mit den jeweiligen Randbedingungen abgewogen und gegebenenfalls angepasst werden.

4 Parametrisches Geometriemodell

Z. Yue hat sich in seiner Masterarbeit “A Computational Human Model for Thermal Comfort Simulation” [36], die im Vorfeld am Lehrstuhl für Computation in Engineering der Technischen Universität München entstanden ist, mit einem parametrischen Geometriemodell beschäftigt, das im Folgenden kurz aufgegriffen und erläutert werden soll.

4.1 Globales Konzept

Die Grundidee eines parametrischen Geometriemodells besteht darin, eine topologische und geometrische Beschreibung eines Manikin-Modells zu haben, das anhand definierter Freiheitsgrade transformiert werden kann. Das Modell kann zudem zur Visualisierung von Bewegungsabläufen eingesetzt werden. Auf Basis des geometrischen Modells können ferner die im Rahmen dieser Arbeit relevanten Simulationen durchgeführt werden.

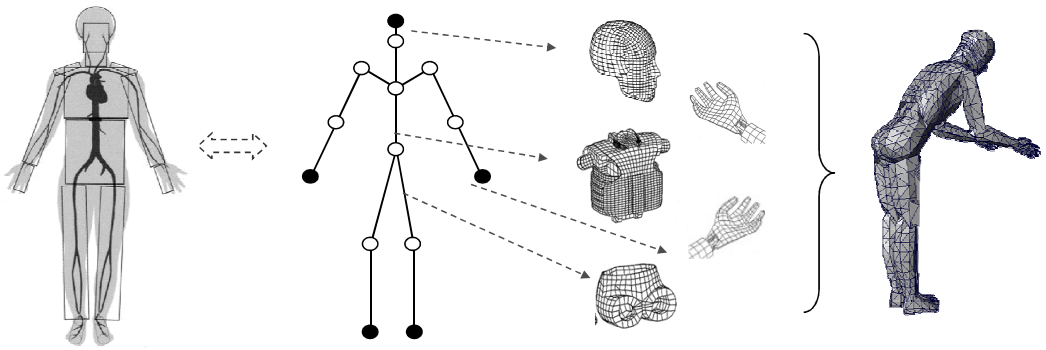


Abbildung 4.1: Konzept des parametrischen Geometriemodells [32]

4.2 Umsetzung

Um dieses Ziel zu erreichen, wurde eine Skelett-Struktur, ein sogenanntes Armature-Modell, in Blender [2] entworfen, die sich an den aus der Spiele-Industrie bekannten H-Anim Stan-

dard [6] anlehnt und in [36] dargestellt ist. Zusätzlich ist der Aufbau des passiven Systems von Fiala beim Erstellen der Struktur berücksichtigt worden, damit eine Unterteilung der Elemente in Sektoren, wie in Abschn. 2.1 erläutert, möglich ist (Abb. 4.1).

Die Skelett-Struktur kann anhand von Transformationsmatrizen in die gewünschte Form gebracht werden. Das topologische Modell verwaltet dabei sämtliche Abhängigkeiten. Das bedeutet, dass zum Beispiel bei einer Rotation des Oberarms um 20 Grad sämtliche verbundenen Elemente, wie Unterarm und Hand, automatisch mit transformiert werden, um eine konsistente Struktur aufrecht zu erhalten.

Dieser Skelett-Struktur wird anschließend ein Oberflächenmodell zugeordnet, das eine bestimmte Auflösung besitzt. Zur Zeit ist es möglich zwischen drei Stufen zu wählen: grob, mittel und fein. Weiterhin kann zwischen einem einfachen Zylindermodell, angelehnt an den einfachen Körperaufbau aus Zylindern und Sphären von Fiala und einem detaillierten Modell ausgewählt werden, das durch den von der NASA entwickelten "Man-System Integration Standard" [11] beschrieben ist.

Die Elemente wurden für jedes Modell und jede Auflösung einzeln vernetzt und liegen in einer Bibliothek bereit. Bei Bedarf werden die Elemente dann anhand der Skelett-Struktur zu einem kompletten Modell zusammengebaut und stehen als vollständige Geometrie eines Menschen für weitere Anwendungen, wie zum Beispiel thermische Simulationen, zur Verfügung.

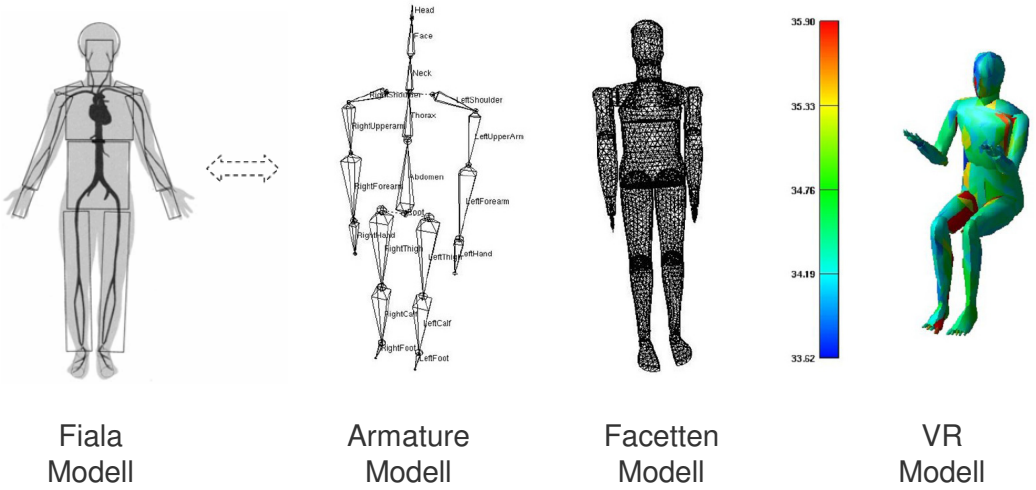


Abbildung 4.2: vom Fiala- zum VR-Modell [32]

Der Vorteil dieses parametrischen Modells liegt darin, dass für eine gewünschte Position eines Manikin-Modells nur sehr wenig Daten benötigt werden. Die Rotationswinkel, die benötigt werden um einen Menschen durch Koordinatentransformationen in die gewünschte Form

zu bringen reichen hierzu aus. Dies ist von großem Vorteil, wenn es darauf ankommt, die Datenübertragungsmenge über ein Netzwerk so klein wie möglich zu halten.

Ein Nachteil eines solchen Modells ist, dass es zwangsläufig bei den Verbindungen einzelner Elemente zu Unstimmigkeiten in der Netzgeometrie und eventuell zu einem inkonsistenten Netz kommen kann. Dies könnte einigen Programmen Probleme bereiten, die ein geschlossenes Oberflächenmodell als Basis für numerische Berechnungen verwenden, das im Wesentlichen klaffungsfrei sein muss. Ist das der Fall, so müssen Tools zur „Geometrieheilung“ eingesetzt werden, um diese Probleme zu beheben, was allerdings großen Zeitaufwand bedeuten kann.

Nach dem Generieren der Skelett-Struktur mittels Blender wurde das oben beschriebene Konzept in `C++` als Klassenstruktur umgesetzt. Damit ist eine Bibliothek entstanden, die entweder in ein eigenes Konzept eingebunden werden kann, oder die in einem eigenständigen Programm die Geometrie als STL oder als NASTRAN Datei liefern kann.

5 Kopplungskonzepte

5.1 Allgemeines

Die Grundlagen für eine interaktive Berechnung im Sinne von Computational Steering wurden in Kapitel 2 und 3 erläutert. Nun soll in den folgenden Punkten dargestellt werden, welche Möglichkeiten sich durch die Bereitstellung der neuen Schnittstelle eröffnen.

Allen Szenarien liegen folgende Prinzipien zu Grunde:

Die Randbedingungen für das Thermoregulations-System werden periodisch verändert, sei es durch reine Benutzerinteraktion oder durch neue Berechnungsergebnisse. Anhand dieser neuen Randbedingungen wird das Thermoregulations-System neu berechnet und liefert im Anschluss Berechnungsergebisse zurück. Diese Daten können dann wieder Ausgangspunkt für neue Überlegungen des Benutzers bezüglich Regelung sein. Der ganze Vorgang läuft zyklisch ab und wird während den gesamten Berechnungen bereits visualisiert.

5.2 Möglichkeiten

5.2.1 Virtual Climate Chamber Project

Das „Virtual Climate Chamber Projekt“ wurde anfangs im Rahmen dieser Arbeit nur als Testumgebung für das Thermoregulations-Interface konstruiert. Anschließend wurde es allerdings als eigenständige Anwendung weiterentwickelt. Details dazu sind in Kapitel 6 zu finden. Das grundlegende Kopplungskonzept ist in Abb. 5.1 dargestellt.

Ein Benutzer startet die Virtual Climate Chamber (VCC). Dort lädt er Geometrieinformationen aus dem parametrischen Geometriemodell (Kapitel 4), die zur Darstellung der Ergebnisse benutzt werden, in das System. Es ist hier wichtig zu erwähnen, dass in diesem Kopplungskonzept keine eigentlichen Berechnungen auf dem Geometriemodell ausgeführt werden. Prinzipiell könnte eine willkürlich gewählte Geometrie im NASTRAN Datei-Format

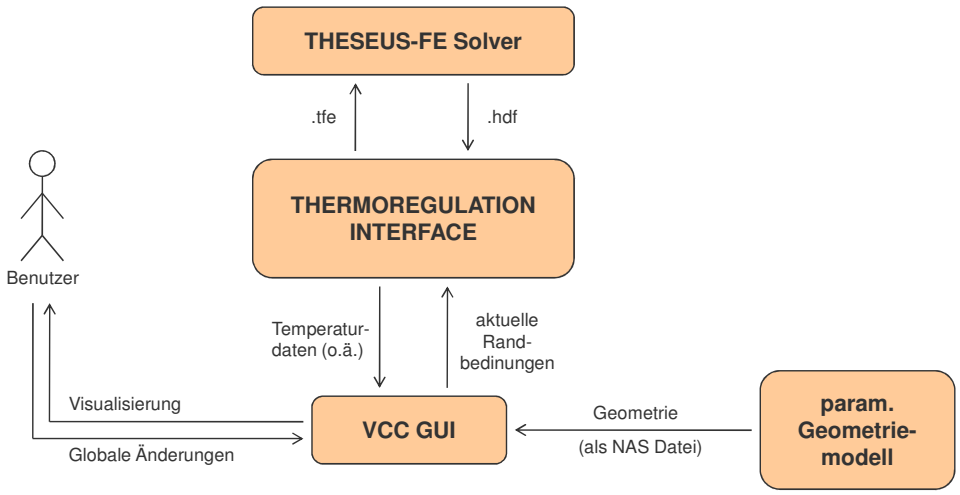


Abbildung 5.1: Kopplungskonzept: Virtual Climate Chamber Project

(siehe Anhang A.1) eingefügt werden, solange sie über 48 Gruppen (für die Darstellung der 48 Sektoren) verfügt.

Anschließend kann der Benutzer Randbedingungen für die Berechnung setzen. Intern leitet die VCC diese Randbedingungen an das Thermoregulations-Interface weiter, welches sie wiederum in das THESEUS-FE Format umwandelt und berechnen lässt. Die Ergebnisse werden vom Thermoregulations-Interface dann zurück an die Virtual Climate Chamber geliefert, wo sie visualisiert werden. Der Benutzer kann sich an dem Punkt entscheiden, ob er die Randbedingungen ändern will oder nicht.

Diese Version wurde im Rahmen der Arbeit implementiert und getestet.

5.2.2 Kopplung an Fluidsolver

Eine weitere Möglichkeit ist die direkte Kopplung an einen Fluidsolver. Dabei ist die eigentliche Geometrie des Dummys durch ein parametrisches Geometriemodell als „Hindernis“ in den Strömungsbereich mit eingebunden. Vom Fluidsolver werden dem Thermoregulations-Interface die Randbedingungen übergeben. Anschließend wird die Berechnung durchgeführt, und die Ergebnisse zurück an den Fluidsolver übermittelt. Dieser setzt die neuen Oberflächentemperaturen an der Manikin-Oberfläche als neue Randbedingungen an und berechnet anschließend den Strömungsbereich erneut.

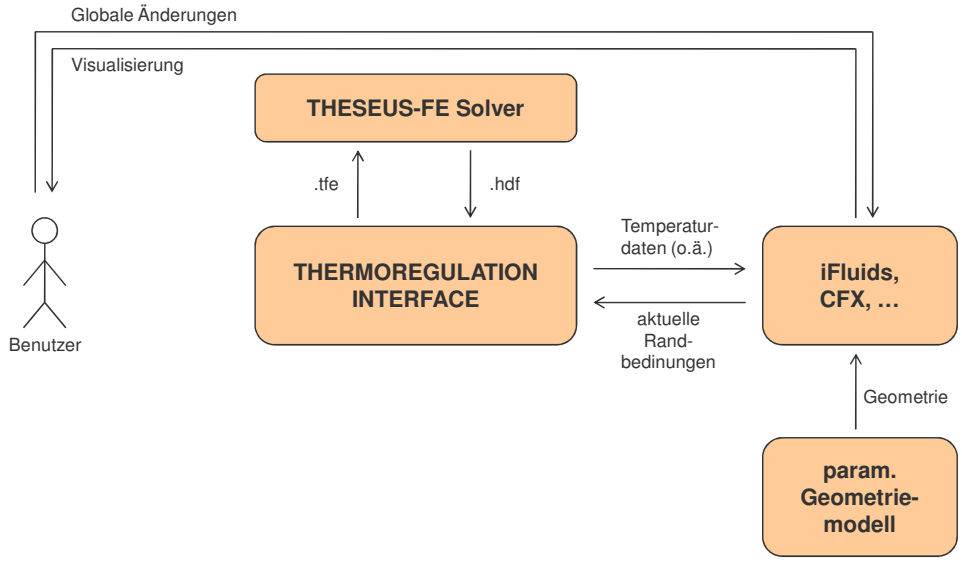


Abbildung 5.2: Kopplungskonzept: Kopplung an Fluidsolver

Dieses Konzept ist in Abb. 5.2 dargestellt. Dabei kann als Fluidsolver die am Lehrstuhl für Computation in Engineering der Technischen Universität München entwickelte, interaktive Simulationsumgebung „iFluids“ [34], oder eine kommerzielle Lösung wie CFX [1] benutzt werden. In beiden Fällen sind jedoch noch Anpassungen zu treffen, um die Daten korrekt an das Thermoregulations-Interface übergeben zu können.

Weiterhin müssen einige Probleme beachtet werden, die bei einer solchen Kopplung auftreten können. Ein Problem ist der Unterschied in den Zeitskalen. Das Thermoregulations-Modell reagiert sehr träge im Vergleich zur Strömungsberechnung. Während sich bei „normalem“ Raumklima die Skala der Thermoregulationsberechung ca. im Minutenintervall befindet, so reicht das bei weitem nicht für eine ausreichende Auflösung des Strömungsbereichs aus. Soll eine solche Kopplung implementiert werden, so müssen vorher Probleme dieser Art vernünftig durchdacht und gelöst werden.

5.2.3 Kopplung an ein Ein- oder Mehrzonenmodell

In Unterschied zu der in Punkt 5.2.2 dargestellten Möglichkeit wird hier nicht ein Fluidsolver zur Bestimmung der Randbedingungen angebunden, sondern ein Ein- oder Mehrzonenmodell [16, 31].

Der strukturelle Aufbau dazu ist in Abb. 5.3 dargestellt.

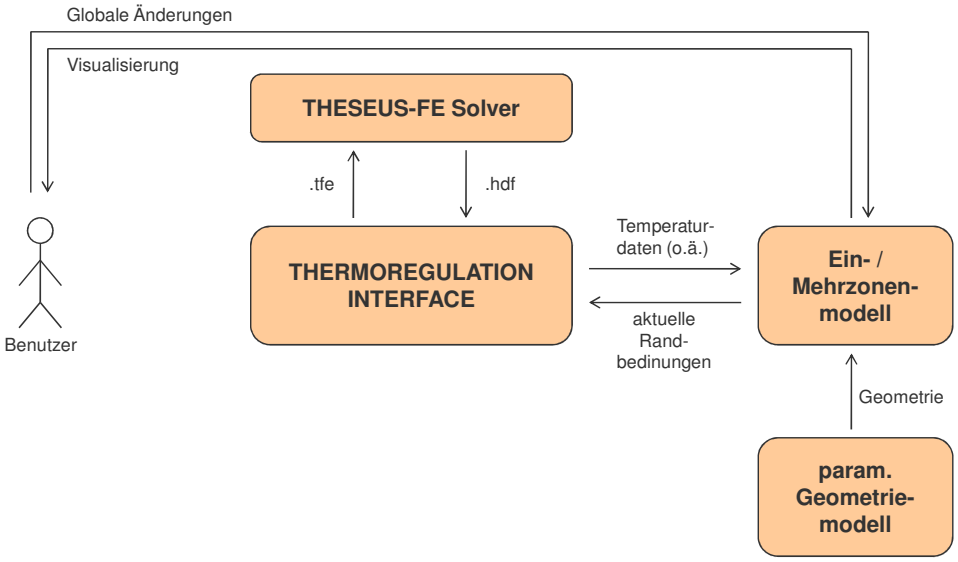


Abbildung 5.3: Kopplungskonzept: Kopplung an ein Zonenmodell

Ein Benutzer konfiguriert ein Ein- oder Mehrzonenmodell. Während der anschließenden Berechnungen werden Temperaturdaten für die Randbedingungen des Mensch-Modells, das ebenfalls über eine geometrische Darstellung (beispielsweise anhand des parametrischen Geometriemodells) im Modell abgebildet ist, bestimmt. Diese werden an das Thermoregulations-Modell übergeben, und die Oberflächentemperaturen werden zurückgeliefert.

Auch hier könnte ein Problem mit den Zeitskalen auftreten, da ein zonales Modell eine Langzeitsimulation beinhalten kann. Beispielsweise soll das Innenklima eines Bürogebäudes über ein ganzes Jahr hinweg simuliert, die Komfortbedingungen bestimmt, und eine Statistik der Überschreitungstage angefertigt werden. Es ist ersichtlich, dass in diesem Falle die Zeitschrittweite im Thermoregulations-Modell anders angepasst werden muss, als beim Fluidsolver.

5.2.4 Kopplung an Fluidsolver mit Überwachungsfunktion

Der Ablauf ist, wie in Abb. 5.4 zu erkennen, ähnlich zu dem in Punkt 5.2.2 dargestellten Fall. Auch hier berechnet ein Fluidsolver, unter anderem, die neuen Randbedingungen für das Thermoregulations-Modell.

Als neuer Punkt kommt nun die zusätzliche Kopplung mit einer „lite-Version“ der Virtual Climate Chamber hinzu. Unter „lite-Version“ versteht man hier, dass ausschließlich eine

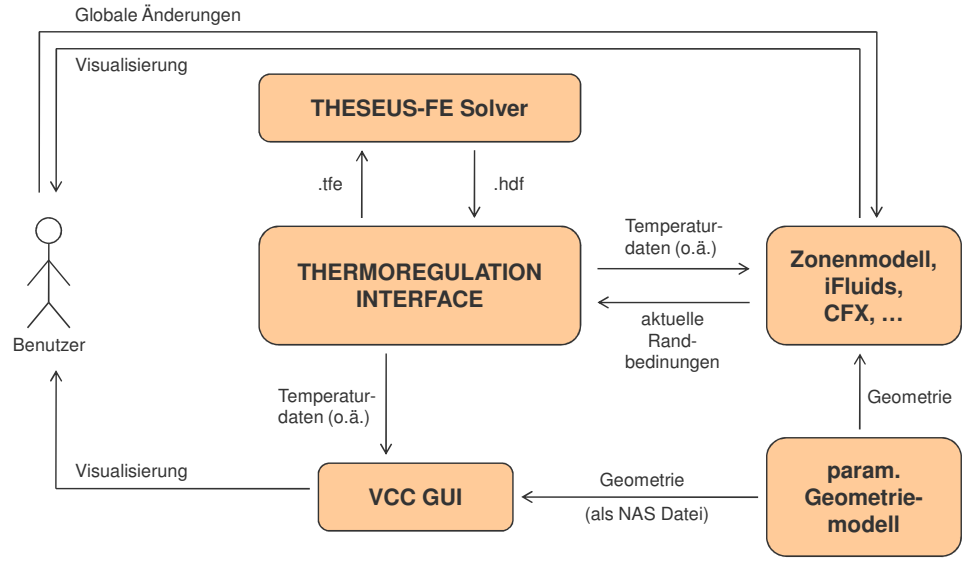


Abbildung 5.4: Kopplungskonzept: Kopplung an Fluidsolver mit Überwachungsfunktion

Visualisierung vorhanden ist. Sämtliche aktiven Steuerungsmechanismen und Regler sind nicht verfügbar.

Der Ablauf sieht dann wie folgt aus: Der Benutzer startet die interaktive Umgebung (zum Beispiel iFluids) und die „lite-Version“ der Virtual Climate Chamber. Die aktuellen Randbedingungen werden vom Benutzer an den Fluidsolver übermittelt, der eine Lösung für den Strömungsbereich, inklusive Manikin-Berandung, berechnet. Diese Daten werden an das Thermoregulations-Modell übergeben, welches mit THESEUS-FE die Oberflächentemperatur berechnet.

Die Ergebnisse werden nun sowohl an den Solver, als auch an die VCC Umgebung weitergeleitet. Der Solver kann in der interaktiven Umgebung ebenfalls eine Visualisierung des gesamten Domains, inklusive Manikin, vornehmen. Die VCC Umgebung allerdings zeigt ausschließlich die Daten des Manikins, die in vielen verschiedenen Formen visualisiert werden können.

Damit wurde eine Oberfläche geschaffen, in der sämtliche simulierte Vitalwerte in Analogie zu einem Überwachnungsmonitor auf einer Intensivstation dargestellt werden können. Für weitere Informationen über die Möglichkeit der Visualisierung in der Virtual Climate Chamber siehe Kapitel 6.

6 Interaktive Simulation - Virtual Climate Chamber Project

6.1 Allgemeines

Da die Virtual Climate Chamber auf dem Thermoregulations-Interface aufbaut, das wiederum THESEUS-FE benötigt, ist es erforderlich, dass ThesesuFE mit einer gültigen Lizenz installiert sein muss. Um mit der Virtual Climate Chamber Umgebung rechnen zu können, werden die Lizenzinformationen bei jedem Berechnungsschritt überprüft.

THESEUS-FE wird vom Thermoregultions-Interface über ein internes Systemkommando aufgerufen, das unter Windows `TheseusCL.exe` und unter Linux `theseusfe` lautet. Damit dieser Aufruf problemlos funktioniert, müssen in beiden Betriebssystemen der `TheseusFE` Ordner, sowie der `TheseusFE/bin` Ordner im aktuellen Pfad vorhanden sein. Ein FLEXlm Server [9] muss lokal laufen, oder es muss eine Netzwerk-Verbindung zu einem Rechner mit FLEXlm Server bestehen. Die entsprechenden Firewall-Ports müssen offen und erreichbar sein.

6.2 Programmierkonzept

Das Projekt wurde mit `wxDev-C++` [13], einer `C/C++` IDE[1] für Windows, entwickelt. Die graphische Benutzeroberfläche basiert auf der freien `wxWidgets` Bibliothek [14]. `wxDev-C++` benutzt standardmäßig `minGW-C++` [10], eine Windows Version des GNU Compilers [4].

`wxWidgets` [14] ist ein in `C++` geschriebenes Framework zur plattformunabhängigen Entwicklung von Anwendungen. Es bietet alle Möglichkeiten zum Programmieren von graphischen Benutzeroberflächen und besitzt eine leicht modifizierte LGPL Lizenz, die es erlaubt, damit

[1] Integrated Development Environment - integrierte Entwicklungsumgebung

sowohl proprietäre, als auch freie Software zu entwickeln. Weitere Details dazu sind auf der entsprechenden Homepage im Literaturverzeichnis zu finden.

Die Tatsache, dass der Code für `minGW` konzipiert ist, und eine Portierung der `wxWidgets` Bibliotheken für Linux und Windows existiert, ermöglicht es, den gleichen Code ohne große Änderungen unter Linux mit `gcc` und unter Windows mit `minGW` zu übersetzen.

Da auch der Solver von THESEUS-FE in seiner gegenwärtigen Version unter Linux und Windows läuft, können vollwertige Applikationen des Virtual Climate Chamber Projektes unter Linux und Windows kompiliert und benutzt werden.

6.3 Internes Kopplungskonzept

Wie bereits in 5.2.1 erwähnt, ist das Thermoregulations-Interface aus Kapitel 3 in dem Virtual Climate Chamber Projekt integriert. Obwohl die Berechnung mittels THESEUS-FE für jeden Schritt schnell durchführbar ist, ist eine Entkopplung der Berechnung von der graphischen Benutzeroberfläche nötig, da diese ansonsten für die Berechnungszeiträume blockiert wäre, was eine Interaktion im Sinne von Computational Steering unmöglich machen würde.

Aus diesem Grunde ist die Virtual Climate Chamber mit Hilfe von Threads programmiert worden. Dazu wurde die Threadklasse von `wxWidgets` benutzt. Eine gute Dokumentation hierzu befindet sich in [22]. Ein Thread lässt sich als „Ausführungsstrang“ durch ein Programm sehen. Manchmal werden sie fälschlicherweise auch als eigene kleine Prozesse bezeichnet. Das ist in so fern nicht korrekt, als dass sich alle Threads eines Prozesses den gleichen Adressraum im Speicher teilen, wohingegen der Adressraum von einzelnen Prozessen streng getrennt ist. Obwohl der gemeinsame Speicher es einfach macht Daten zwischen einzelnen Threads auszutauschen, können große Probleme auftreten, falls zwei Threads zur gleichen Zeit versuchen auf die gleichen Daten zuzugreifen. In den meisten Fällen ist deshalb eine gewisse Synchronisation unvermeidlich.

Im Falle der Virtual Climate Chamber wurden zwei Threads benutzt. Der Hauptthread enthält die gesamte Benutzerumgebung. Der zweite sogenannte Workerthread beinhaltet die Berechnungsschleife, die ab dem Zeitpunkt des Starts durchgehend läuft, bis sie durch den Benutzer abgebrochen wird. Durch Synchronisationmechanismen, die die Datenintegrität gewährleisten, kommunizieren die beiden Threads zu festgelegten Zeiten.

6.3.1 Berechnungsschleife

Der sehr einfache Pseudocode für die Berechnungsschleife, die zugleich den einzigen Inhalt des Workerthreads darstellt, sieht wie folgt aus:

```
setze globale Zeit zu Null
wiederhole bis zum Abbruch
        prüfe ob Thread weiterlaufen soll
        transferiere Eingangsdaten aus GUI-Thread in diesen Thread
        übergebe Randbedingungen an Thermoregulations-Interface
        führe eigentliche Berechnung aus
        schreibe Ergebnisse zurück in GUI-Thread
        erhöhe globale Zeit um letzten Berechnungszeitschritt
wiederhole_ende
```

Wie zu erkennen ist, ist die Berechnungsschleife nicht weiter kompliziert. Die Datensynchronisation ist schwieriger und wird im Folgenden erläutert.

6.3.2 Datensynchronisation

In der Berechnungsschleife ist eine Datensynchronisation an mindestens einem Punkt notwendig. Es wurde versucht, die Anzahl der kritischen Sektionen soweit wie möglich zu reduzieren. Unter einer kritischen Sektion versteht man einen bestimmten Teil des Codes, indem sich nur jeweils ein Thread aufhalten darf und exklusiven Zugriff auf alle Daten dieser Sektion hat. Eine gängige Methode, die auch im Rahmen der Programmierung der Virtual Climate Chamber benutzt wurde, macht Gebrauch vom sogenannten Mutex-Verfahren. Mutex steht für „mutual exclusion“ und bedeutet soviel wie gegen- oder wechselseitiger Ausschluss. Im Grunde hat ein Mutex drei Funktionen sowie einen Zustand:

Lock: Ein Thread möchte gerne einen kritischen Bereich betreten und exklusiven Zugriff auf dessen Daten bekommen. Bevor er diesen jedoch betreten darf, muss er erst die Funktion `Lock` ausführen. Befindet sich kein anderer Thread in der kritischen Sektion (i.e. hat der Mutex den Zustand „unlocked“) so kann der Thread diese kritische Sektion betreten und setzt den Mutex anschließend auf „locked“, so dass kein anderer Thread Zutritt hat. Steht der Mutex bereits auf „locked“ wenn der Thread die Sektion betreten will, so wartet er so lange, bis die Sektion freigegeben wird (i.e. bis ein anderer Thread die Funktion `Unlock` aufruft). Anschließend betritt er sie und sperrt sie sofort hinter sich ab.

Unlock: Verlässt ein Programm eine kritische Sektion, so hat es die Funktion `Unlock` aufzurufen. Wird dieser Punkt an irgendeiner Stelle vergessen, so wird das schwerwiegende Folgen für das gesamte Programm haben, da kein anderer Thread die kritische Sektion betreten kann.

TryLock: Mit der Funktion `TryLock` kann überprüft werden, ob der Mutex bereits im Zustand „locked“ ist. Diese Funktion liefert nur einen booleschen Wert zurück, wartet allerdings im Gegensatz zur `Lock` Funktion nicht, um den Bereich wirklich betreten zu können.

Bei der Benutzung des Mutex-Verfahrens ist allerdings auf die Probleme von „Deadlock“ und „Starvation“ zu achten. Eine genaue Beschreibung dieser Phänomene findet sich in [23].

Nach einigen Überlegungen und Tests konnte festgestellt werden, dass kein Mutex gesetzt werden muss, wenn die Daten aus dem GUI-Thread kopiert werden, da vom Workerthread kein Schreibzugriff auf den GUI-Thread erfolgt.

Selbst wenn nicht alle Daten sofort benötigt werden, wird der Lesezugriff gebündelt durchgeführt, um eine möglichst konsistente Abbildung aller Daten für einen Berechnungsschritt zu erzielen. Eine Blockade der Benutzeroberfläche für den Zeitraum dieser Kopie wäre nicht sinnvoll und wird auch nicht benötigt. Um aber eine Angabe über die, für die Berechnung benutzten Randbedingungen zu erhalten, wird zu einem späteren Zeitpunkt eine Auflistung sämtlicher Eingangswerte durchgeführt.

Nun werden die Ergebnisse an das Thermoregulations-Interface weitergeleitet. Auch hier ist kein Schutz erforderlich, da während der Berechnungen nur der Workerthread Zugriff auf das Thermoregulations-Interface hat.

Die Ergebnisse müssen anschließend vom Thread an die graphische Benutzeroberfläche zurück geliefert werden. Hier werden nun Schutzmechanismen benötigt. Würde der Thread versuchen, direkt auf die Klassenstrukturen der Benutzeroberfläche zuzugreifen, so müsste eine komplette Blockade während des Kopierprozesses erfolgen. Dies erwies sich in Tests als nicht sinnvoll, da die Zeit, die vergeht, während der Workerthread darauf wartet, exklusiven Zugriff auf die GUI-Klassen zu bekommen, zu lange dauert.

Deshalb wurde hier eine andere Variante gewählt:

Es existieren drei Warteschlangen, die in Form von `C++ stl::queue` implementiert sind, ein Datenfeld, das 48 Fließkommazahlen enthält, sowie 2 Integer-Werte. Nur diese, oben genannten Objekte, werden vom Mutex-Verfahren vor konkurrierendem Zugriff geschützt.

Die erste Warteschlange beinhaltet die aktuell gültigen Randbedingungen, wie sie auch an das Thermoregulations-Interface übergeben wurden. Diese Informationen werden benötigt,

um die Auflistung der Randbedingungsgeschichte zu erstellen. Die zweite Warteschlange enthält globale Ergebnisse für die durchgeführte Berechnung (i.e. Hypothalamus-Temperatur, gemittelte Hauttemperatur, etc...). Eventuelle Protokollnachrichten, die vom Workerthread abgegeben wurden, werden von der dritten Schlange behandelt. Das Datenfeld beschreibt die gegenwärtig gültigen, lokalen Temperaturwerte, und ein Integerwert gibt an, um welchen Datentyp es sich bei den Fließkommazahlen handelt.

In diesem Sinne bedeutet der Befehl `schreibe Ergebnisse zurück in GUI-Thread` nicht, dass die Ergebnisse sofort in die Benutzeroberfläche kopiert werden, sondern, dass eine Kopie an das Ende der entsprechenden FIFO-Queue[2] [35] angehängt wird. Die lokalen Daten werden, unter Verwendung von Mutex-Konstrukten, direkt in das entsprechende Feld der Benutzeroberfläche kopiert. Dies ist möglich, da die Implikationen für die lokalen Felder direkt abgeschätzt werden können, und nicht im Hintergrund von einer `wxWidgets` Klasse beeinflusst werden, in der nicht sichergestellt werden kann, dass sie „threadsicher“ ist. Im GUI-Thread werden die Warteschlangen dann entsprechend weiterverarbeitet. Hier ist von großer Bedeutung, dass die Wartezeit minimiert wird, indem eine Kopie an die Warteschlage angehängt wird. Der Workerthread kann im Prinzip die ganze Zeit Arbeit ausführen und hat somit fast keine Leerlaufzeit.

6.3.3 Interner Ablauf in der graphischen Benutzeroberfläche

Nun, da Ergebnisse in Warteschlangen stehen, müssen diese auch vom GUI-Thread behandelt werden. Der OpenGL-Viewport führt in sehr kurzen Abständen regelmäßig eine Neuzeichnung seiner Oberfläche aus. In genau dieser Prozedur (Repaint) wird auch die Behandlung der Warteschlangen durchgeführt.

Bei Aufruf überprüft eine Prozedur, ob in irgendeiner der drei Warteschlangen Daten vorhanden sind. Falls dort Daten existieren, werden diese nacheinander (unter Verwendung eines Mutex für das Löschen von Daten) aus der Warteschlange entfernt und behandelt. In dem Fall von Randbedingungen und Berechnungsergebnissen bedeutet das, dass diese an eine Liste vergangener Daten angehängt werden, mit denen eine durchgängige Geschichte von Beginn der Berechnungen bis zum jetztigen Zeitpunkt gezeichnet werden kann. Zusätzlich können die Ergebnisse in 2 Fenstern als Tabellen dargestellt und noch während der Berechnung betrachtet werden. Die Protokolldateien werden in dem entsprechenden Textfenster ausgegeben. Sind keine weiteren Daten vorhanden, so wird die Prozedur beendet.

[2] FIFO: „First In, First Out“ Order

Diese Art der Datenbehandlung und Synchronisation funktioniert sehr gut im Falle der Berechnung mittels THESEUS-FE, bei dem eine Berechnung im Zeitbereich von ca. 0.5 bis 1.0 Sekunden liegt. Wird die reelle Berechnungszeit wesentlich kleiner, so muss darüber nachgedacht werden, ob weiterhin ein optimaler Datenfluss vorliegt, da die Zeit der Verarbeitung in der Benutzeroberfläche vielleicht länger ist, als das Berechnen der Daten im Workerthread.

Zudem wird noch vor jedem Repaint-Event das Datenfeld, das die lokal darzustellenden Daten enthält, exklusiv kopiert, um zu verhindern, dass eine Inkonsistenz der Daten während dem Repaint auftritt. Dies führt zum Vorhalten von zwei Kopien der lokalen Temperaturen, was aber nicht als sonderlich schlimm erachtet wird, da es sich nur um jeweils 48 Floatwerte und 2 Integerwerte handelt.

6.4 Funktionsweise der graphischen Benutzeroberfläche

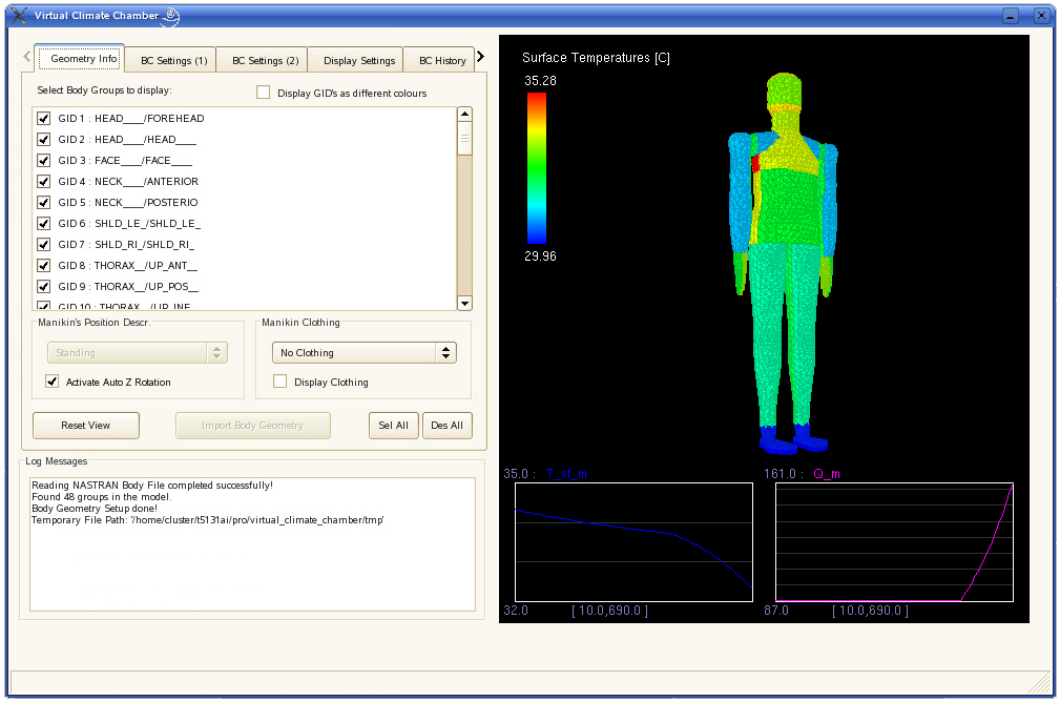


Abbildung 6.1: VCC: Menü: Geometrische Informationen

Wie in Abb. 6.1 zu erkennen, besteht die graphische Benutzeroberfläche im Wesentlichen aus drei Teilen: dem OpenGL Viewport, in dem eine dreidimensionale Darstellung des Mensch-Modells zu sehen ist, den Steuerungsmenüs, sowie einem Fenster in dem Protokollnachrichten angezeigt werden.

6.4.1 OpenGL Viewport

Für das Rendern des 3D-Mensch-Modells existiert in `wxWidgets` ein Wrapper, mit dem OpenGL Befehle direkt umgesetzt werden können. Mit dieser Unterstützung wurde die Grafik dargestellt.

Der OpenGL-Viewport besteht aus zwei wesentlichen Teilen. Im oberen Bereich ist die 3D-Modell-Geometrie dargestellt. Diese ist intuitiv mit der Maus dreh-, verschieb- und skalierbar. Links wird eine Legende und ein Titel angezeigt, der angibt, welcher Datentyp gerade dargestellt wird, und was die Randwerte mit ihren jeweiligen Einheiten sind. Im unteren Bereich befinden sich zwei Vorschaufenster, mit denen der zeitliche Verlauf der globalen Daten angezeigt werden kann (mehr dazu in Punkt 6.4.2.3).

6.4.2 Steuerungsmenüs

Die Steuerungsmenüs sind in Registerkarten angeordnet, und ihr Inhalt wird in den nächsten Punkten dargestellt und erläutert.

6.4.2.1 Geometrieinformationen

Abb. 6.1 zeigt das Menü zur Geometrieinformationseingabe.

Mittels `Import Body Geometry` werden die Geometrieinformationen, sprich das Mensch-Modell, im NASTRAN Geometrieformat eingelesen. Wie bereits vorher erwähnt, ist die Art der Geometrie im Prinzip willkürlich wählbar, solange sie über 48 Gruppen verfügt, deren Nummerierung bei eins beginnt. Sobald die Geometrie eingelesen ist, wird der Manikin grau im Viewport dargestellt. Liegen Berechnungsergebnisse vor, so wird er dementsprechend eingefärbt.

Um die Gruppenzuordnung überprüfen zu können, existiert auf der linken Seite die Möglichkeit einzelne Teile aus- bzw. einzuschalten. Damit kann überprüft werden, ob die Zuordnungen richtig getroffen wurden. Wird die CheckBox `Display GID's as different colors` angeklickt, so werden die Körperteile je nach GID[3] eingefärbt.

Zusätzlich muss an dieser Stelle angegeben werden, ob der Mensch sitzt oder steht (siehe ebenfalls Anmerkung 6.4.3.1). Dies ist mit der ChoiceBox in `Manikin's Position Desc.`

[3]GID: Group ID Number (hier von 1 bis 48)

möglich. `Activate Auto Z Rotation` gibt an, ob das Modell während der Berechnungsphase automatisch um die z-Achse rotieren soll.

Unter `Manikin Clothing` kann die Kleidung des Manikins ausgewählt werden. Dieser Punkt kann auch während der Berechnung verändert werden (siehe Bemerkungen in 3.2.2). Es kann zwischen der Standard-Sommer- und der Standard-Winterbekleidung gewählt werden. Bei einem Klick auf `Display Clothing` werden die I_{cl}-Werte der Kleidung auf dem Manikin-Modell dargestellt.

Erst wenn das Geometriemodell korrekt geladen wurde, können weitere Einstellungen vorgenommen, und Berechnungen durchgeführt werden.

6.4.2.2 Randbedingungen

In den in Abb. 6.2 und Abb. 6.3 gezeigten Menüs können Einstellungen bezüglich der globalen Randbedingungen vorgenommen werden.

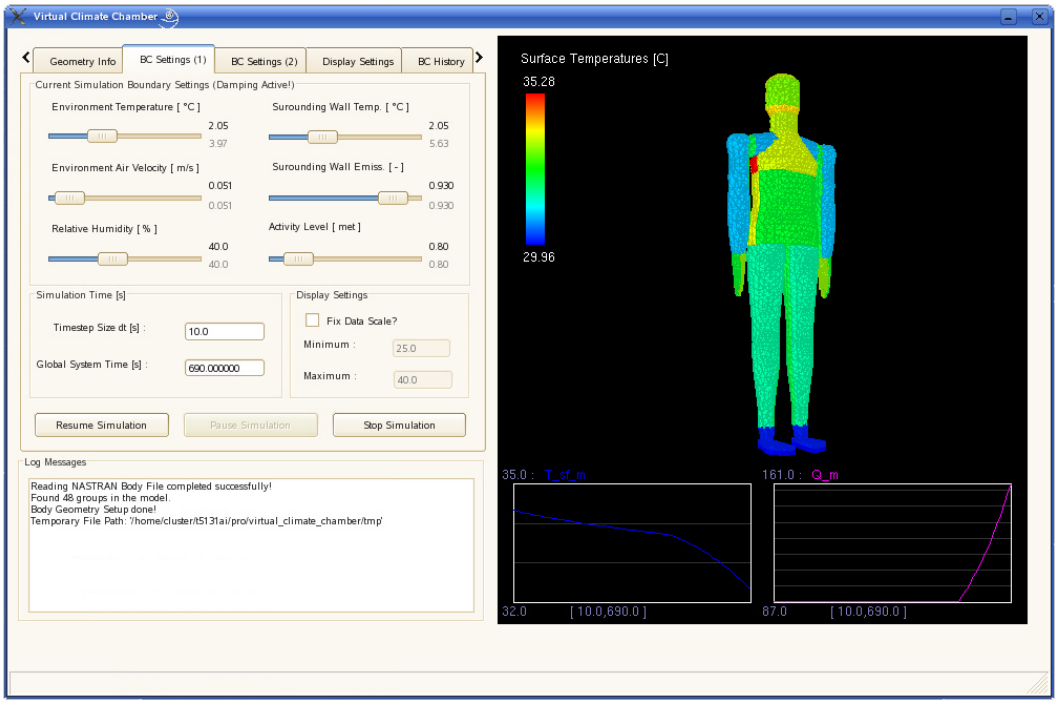


Abbildung 6.2: VCC: Menü: Randbedingungen (Teil 1)

Es stehen sechs Schieberegler zur Verfügung um die globalen Einstellungen für das Thermoregulations-Modell interaktiv, während einer Berechnung zu ändern. Die veränderbaren Werte sind

T_a	Umgebungstemperatur	[-20.0	..	50.0]	°C
T_w	Strahlungstemperatur	[-20.0	..	50.0]	°C
v_a	Windgeschwindigkeit	[0.001	..	1.000]	m/s
ε_w	Emissivität	[0.30	..	1.00]	-
rh	relative Luftfeuchte	[0.0	..	100.0]	%
act_{met}	metabolische Aktivität	[0.5	..	3.0]	met

Die Werte können nach Belieben in bestimmten, angegebenen Bereichen variiert werden.

Neben den Schiebereglern befinden sich jeweils zwei Zahlenwerte. Der obere (schwarze) gibt den gegenwärtigen Stand des dazugehörigen Schiebereglers an. Dieser Wert verändert sich, sobald der Schieberegler bewegt wird, unabhängig davon, ob gerade eine Berechnung durchgeführt wird oder nicht. Die untere (graue) Zahl gibt an, welche Werte bei der letzten Berechnung benutzt wurden. Das wird später wichtig, wenn der Effekt der Dämpfung hinzukommt.

Bei `Timestep Size` kann der Simulationszeitschritt in Sekunden angegeben werden. Dies ist im Prinzip auch während der Simulation möglich. Es ist allerdings darauf zu achten, ob die Ergebnisse den gewollten entsprechen, wenn die Zeitschrittweite während einer laufenden Simulation geändert wird. Man stelle sich folgendes Szenario vor: Während die Zeitschrittweite gerade bearbeitet wird, leitet die Berechnungsschleife ein Kopieren der Randbedingungen ein. Dabei wird dann als Zeitschrittweite nicht der gewünschte Wert übernommen, sondern irgend ein Zwischenwert. Soll dieser Effekt vermieden werden, sollte die Simulation angehalten (Pause), die Zeitschrittweite geändert, und die Simulation anschließend wieder aufgenommen werden (Resume). Damit kann das oben beschriebene Problem nie auftreten.

Die Zeit, die gegenwärtig im System herrscht, kann unter `Global System Time` abgelesen werden. Sie lässt sich manuell nicht verändern, sondern wird bei fortschreitender Simulation automatisch angepasst. Wie bei der „realen“ Zeit auch, kann sie sich nur in eine Richtung ausbreiten.

Im Normalfall passen sich die Randwerte der Skala für die lokalen Werte automatisch an den größten und kleinsten Betrag an. Ist dies nicht erwünscht, so kann die Skala mit der CheckBox `Fix Data Scale` fixiert werden, und die Randwerte manuell an die Erwünschten angepasst werden.

Anschließend stehen drei Buttons für die Simulationskontrolle zur Verfügung. Mit `Start Simulation` kann die Simulation gestartet werden. Anschließend wird nach einem temporären Verzeichnis gefragt, in dem sämtliche THESEUS-FE Zwischendateien etc. abgelegt werden (vergleiche ebenfalls 3.2.2). Hier sollte drauf geachtet werden, dass der Benutzer in

dem gewählten Verzeichnis Schreibrechte besitzt (vor allem unter Linux), da THESEUS-FE dort Daten ablegen muss.

Dieser Ablauft tritt nur beim ersten Mal auf. Der Button wird anschließend automatisch in `Resume Simulation` umbenannt. Mit `Pause Simulation` kann eine Simulation angehalten werden. In diesem Falle wird der Workerthread nicht gelöscht, sondern nur temporär gestoppt. Die Simulation kann wieder mit `Resume Simulation` aufgenommen werden.

Beim Betätigen von `Stop Simulation` wird die Simulation komplett gestoppt, und der Workerthread gelöscht. Eine Wiederaufnahme der Simulation ist anschließend nicht mehr möglich. Es sollte also gut überlegt werden, ob die Simulation wirklich gestoppt, oder nur unterbrochen werden soll.

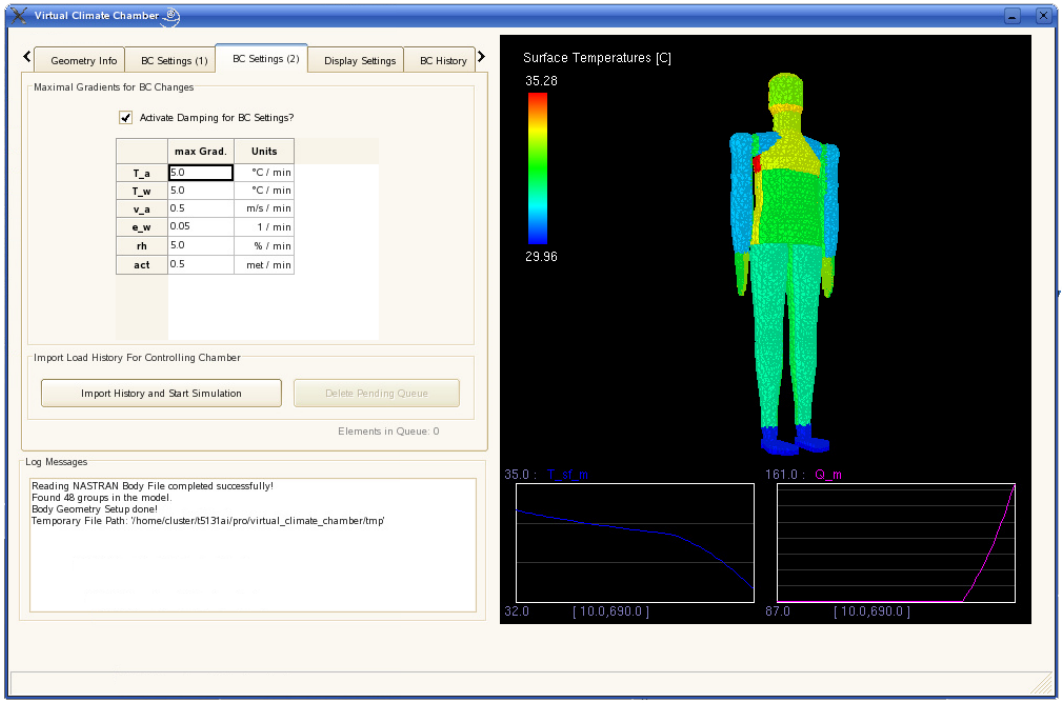


Abbildung 6.3: VCC: Menü: Randbedingungen (Teil 2)

Auf der zweiten Seite der Randbedinungseinstellungen (Abb. 6.3) kann die Dämpfung eingestellt werden. Sie ist von Haus aus aktiviert, könnte aber mit der CheckBox `Activate Damping for BC Settings` deaktiviert werden.

Unter Dämpfung versteht man hier den Effekt, dass keine sprunghaften Veränderungen der Werte möglich sind. Auch in einer realen Umgebung fällt die Temperatur normalerweise nicht innerhalb von einer Sekunde, von zum Beispiel 30°C auf 2°C. Die maximalen Gradienten können in der angegebenen Einheit dort den jeweiligen Wünschen angepasst werden.

Wird bei aktiver Dämpfung ein Temperaturregler, der bei 30°C steht auf 2°C gestellt, so wird sich die schwarze Zahl daneben sofort auf 2°C stellen. Die graue Zahl hingegen, wird sich mit zunehmender Simulationszeit mit einem maximalen Gradienten von 5 °C/min (im Standardfall) auf 2°C zu bewegen. Wenn die am Schieber eingestellte Temperaturänderung kleiner als der angegebene maximale Gradient ist, wird die eingestellte Temperatur sofort benutzt.

Es ist ebenfalls möglich, mit der Virtual Climate Chamber vordefinierte Profile abzufahren. Diese Profile können mit dem Button `Import History and Start Simulation` eingelesen werden.

Die Form dieser „Loadcase-Datei“ ist identisch mit der Randbedingungsdatei, die unter Punkt 6.4.2.4 herausgeschrieben werden kann. Es ist also möglich ein bereits gefahrenes Profil einfach erneut nachzufahren. Es gibt nur einen kleinen Unterschied: Die globale Zeit wird vernachlässigt, und es werden nur die Zeitschrittweiten *dt* berücksichtigt. Das ist insofern interessant, als dass nun das gleiche Profil mehrmals hintereinander abgearbeitet werden kann, und das System automatisch die globale Zeit richtig anpasst.

Wird eine solche Datei geladen, so wird sie zunächst von der graphischen Benutzeroberfläche komplett eingelesen, die Daten interpretiert und in einer Warteschlage abgelegt. Diese Warteschlange wird dann Schritt für Schritt abgearbeitet, während die Simulation läuft. Unter den Buttons erkennt man eine Zeile `Elements in Queue: 0`. Dort wird die gegenwärtige Anzahl der noch abzuarbeitenden Randbedinungen angegeben.

Während eine Warteschlange abgearbeitet wird, sind sämtliche Kontrollen deaktiviert, und die Randbedingungen können nicht mehr manuell verändert werden. Soll an einem bestimmten Zeitpunkt doch wieder eingegriffen werden, so kann der Rest der Warteschlange mit dem Button `Delete Pending Queue` gelöscht werden. Ist die Warteschlange durch manuelles Löschen, oder Abarbeiten der Schritte leer, so werden sämtliche Kontrollen wieder an den Benutzer übergeben.

6.4.2.3 Displayeinstellungen

Die in Abb. 6.4 dargestellten Displayeinstellungen werden häufig benötigt, denn hier kann eingestellt werden, was im OpenGL-Viewport zu sehen ist.

Mit der ChoiceBox `Data to Display on Manikin` können die lokalen Werte ausgewählt werden, die dargestellt werden sollen. Es kann dann zwischen den in `LocalCurrentData` (siehe Abschn. 3.2.1.2) definierten Werten für die Anzeige gewählt werden. Die Änderung

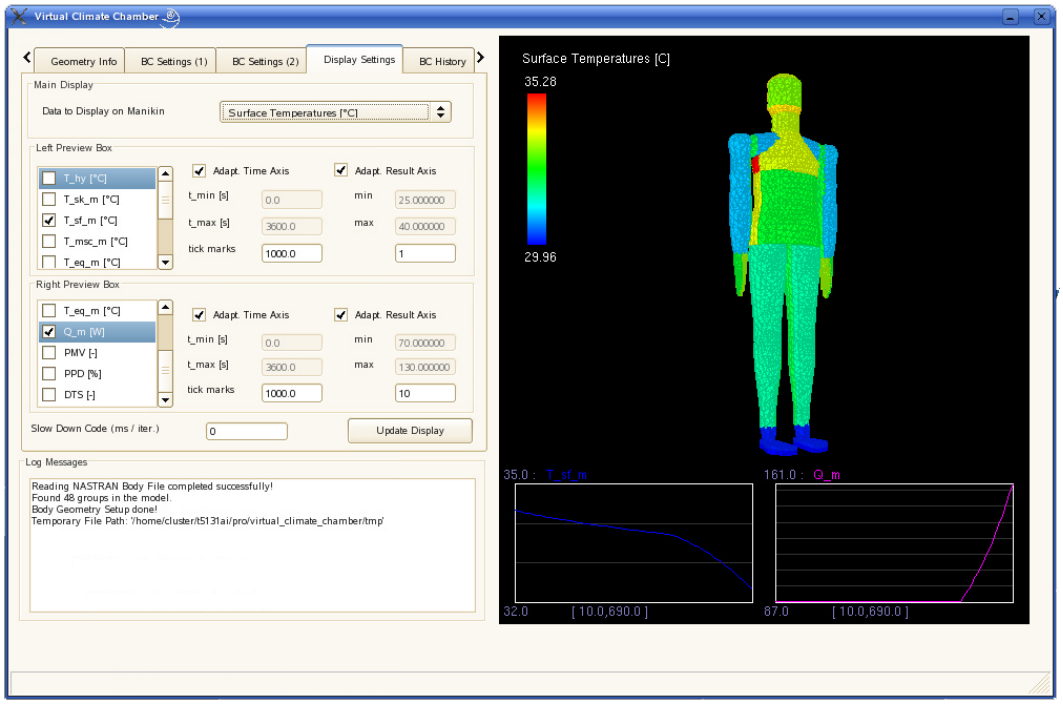


Abbildung 6.4: VCC: Menü: Displayeinstellungen

tritt allerdings erst nach dem nächsten Berechnungsschritt in Kraft, da die Auswahl des anzuzeigenden Datentyps erst beim nächsten Berechnungsschritt übermittelt wird und die Benutzeroberfläche die Daten erst dann zurückgesendet bekommt.

In den `Left` und `Right Preview Box`en können zeitliche Verläufe von globalen Werten während des Simulationsverlauf dargestellt werden. Welche Daten angezeigt werden sollen, kann in der entsprechenden CheckListBox bestimmt werden. Das System lässt mehrere Auswahlmöglichkeiten auf einmal zu. Es ist allerdings darauf zu achten, dass nur mehrere Datensätze abgebildet werden können, deren Skalen auch kompatibel sind. So kann zum Beispiel der Metabolismus in Watt nicht gleichzeitig mit der Hypothalamustemperatur in °C in einer Box dargestellt werden. Es können aber sehr wohl die Hypothalamustemperatur und die mittlere Hauttemperatur in °C gleichzeitig in einer Box angezeigt werden. Auch PMV, DTS und TS können gleichzeitig angezeigt werden, da sie sich auf die gleiche sieben Punkte ASHRAE Komfortskala beziehen [15].

Sowohl die Zeit- als auch die Werteachse passen sich automatisch an die benötigten Größen an. Ist das nicht erwünscht, so kann die entsprechende CheckBox ausgeschaltet, und die Werte manuell eingetragen werden.

Ist die Berechnung zu schnell, um alle gewünschten Änderungen an den Randbedingungen zu vollführen, so kann die Berechnung künstlich verlangsamt werden, indem eine Drosselung

in Millisekunden pro Iteration in den `Slow Down Code` eingetragen wird. Wird beispielsweise 500 angegeben, so wird nach jeder Berechnung 500 Millisekunden (also 0.5 Sekunden) gewartet, bevor die Berechnungsschleife wieder ausgeführt wird.

Technisch gesehen wird der Workerthread für die angegebene Zeit „schlafen“ gelegt. Das bedeutet, dass er keine (oder nur sehr wenige) Systemressourcen verbraucht.

6.4.2.4 Darstellung der Randbedingungen als Tabelle

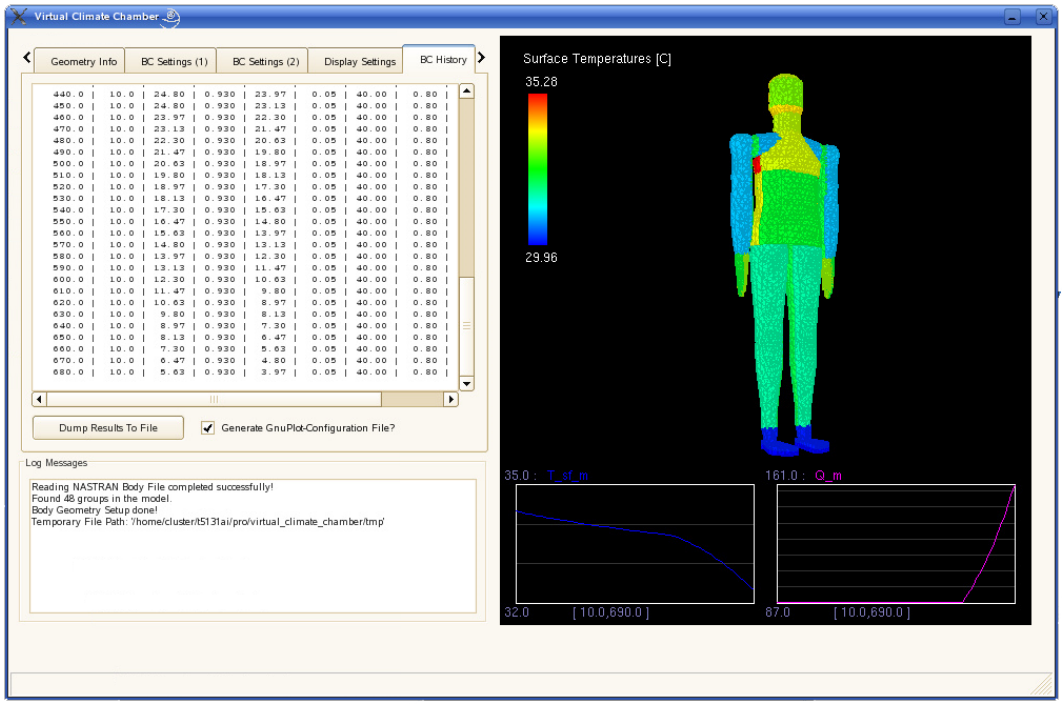


Abbildung 6.5: VCC: Menü: Tabelle für Darstellung der Randbedingungen

Die Darstellung der Randbedingungen als Tabelle ist in Abb. 6.5 zu erkennen. Damit ist es möglich, während der Berechnungen zu überprüfen, ob die richtigen Werte an das Thermoregulations-Interface übergeben wurden.

Zusätzlich kann über `Dump Results To File` eine Ausgabe erstellt werden. Ist dabei die CheckBox `Generate GnuPlot-Configuration File` aktiviert, so wird noch eine zweite Datei mit einem GnuPlot Konfigurationsskript ausgegeben, mit dem es möglich ist, über das Open-Source Programm GnuPlot [5] Diagramme der Verläufe darstellen zu lassen. Sämtliche Diagramme von Berechnungsergebnissen in dieser Arbeit wurden mittels GnuPlot erstellt.

Die dort herausgeschriebene Datei kann auch als VCC „Loadcase-Datei“ eingelesen werden, um ein Profil nachzufahren (siehe 6.4.2.2).

6.4.2.5 Darstellung der globalen Ergebnisse als Tabelle

Die globalen Ergebnisse, die auch in den Preview-Boxen visualisiert werden können, werden ebenfalls während der Simulation in eine Tabelle geschrieben, die mit `Dump Results To File` in einer Datei abgelegt werden können (analog zu 6.4.2.4).

6.4.3 Anmerkungen

6.4.3.1 Position des Manikins

Zur Zeit werden nur zwei Standardfälle für das Mensch-Modell betrachtet: stehend und sitzend. Diese Beschränkung liegt an der Angabe der Viewfaktoren ψ_{sf-w}, die zur Zeit aus der Tabelle von D. Fiala übernommen werden. Das bedeutet, dass egal wie sich das Modell im Raum befindet, gerechnet wird, als ob es stehen oder sitzen würde.

Die Berechnung der Viewfaktoren ist ein Punkt, der in Zukunft implementiert werden muss, um ein konsistentes Modell für sämtliche Mensch-Geometrien zu bekommen.

6.4.3.2 LMV Darstellung

Im Rahmen des ComfSim Projektes [33] hat das Fraunhofer-Institut für Bauphysik in Holzkirchen [3] lokale Komfortanalysen in Innenräumen durchgeführt und einen Zusammenhang zwischen lokalen Äquivalenttemperaturen und Local Mean Votes (LMV) hergestellt [21].

Die LMV Werte berechnen sich nach folgendem Schema:

$$\mathrm{LMV} = a \cdot T_{eq} + b \qquad . \tag{6.1}$$

Hierbei stellen a und b Regressionskoeffiziente dar, die in Versuchen für unterschiedliche Bekleidungen in einem gewissen Wertebereich ermittelt wurden. Gegenwärtig ist die Sommer- und Winterbekleidung in der VCC mit LMW Werten verknüpft. Für ein unbekleidetes Modell existieren keine Werte, und es wird dementsprechend auch nichts dargestellt.

Ist die lokale Äquivalenttemperatur größer oder kleiner als der jeweils gültige Wertebereich, so wird eine Warnung `OoB` (Out of Bounds) im OpenGL-Viewport angegeben. Die GID Nummer für die betreffende Gruppe wird ebenfalls angezeigt, sodass herausgefunden werden kann, welcher Bereich nicht mehr validiert ist.

Weiterhin gelten die LMV Werte nur für stationäre Zustände. Ändern sich die Äquivalenttemperaturen im Modell stark, so sind die Werte nicht mehr validiert. Sie werden trotzdem abgebildet; allerdings wird eine Warnung angezeigt, dass keine „quasi-stationären“ Zustände vorherrschen würden.

7 Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit wurde ein Thermoregulations-Interface, das auf der Software THESEUS-FE aufbaut, entworfen, um das von D. Fiala entwickelte menschliche Thermoregulations-Modell, welches ein passives und aktives System beinhaltet, zu verwenden. Hinter dem Thermoregulations-Interface verwaltet eine Datenstruktur sämtliche Werte, die für die Berechnungen benötigt werden. Diese können nach belieben variiert und verändert werden.

Es wurden verschiedene Kopplungskonzepte entwickelt, und anschließend diskutiert, wie bestehende und zukünftige Programme mit dem Thermoregulations-Interface umgehen können, um Temperaturen und sonstige Daten für ein Mensch-Modell zu ermitteln. Dabei wurden einige Szenarien genauer betrachtet und Probleme geschildert, die bei einem Kopplungsversuch auftauchen können.

Ein Konzept wurde mit dem Projekt „Virtual Climate Chamber“ realisiert und in `C++` implementiert. Anhand dieses eigenständigen Tools ist es möglich, eine virtuelle Testperson unter verschiedenen Bedingungen zu beobachten sowie die thermoregulatorischen Antworten des Körpers auf die Umgebungsbedingungen auf Anhieb zu erkennen. Steuernde Maßnahmen können eingeleitet werden, um einer negativen Entwicklung vorzubeugen. Mit diesem Schritt in Richtung Computational Steering konnte gezeigt werden, dass solche Systeme im Hinblick auf Thermoregulation möglich sind, und sich auch ohne Probleme auf Laptop-Rechnern zeitnah berechnen und interaktiv steuern lassen.

In weiteren Schritten können nun die bereits erwähnten, unbehandelten Konzepte aus Kapitel 5 implementiert und untersucht werden. Dabei sind bereits geschilderte Probleme wie zum Beispiel Viewfaktorberechnung für jede beliebige menschliche Position, sowie das Thematisieren der unterschiedlichen Zeitskalen zu behandeln und effizient zu lösen.

Als weiterer Punkt ist eine Anpassung der Körperdaten von großem Interesse. Fiala benutzt in seinem Modell einen „Durchschnittsmenschen“. Wie stark sich jedoch die unterschiedliche Physiologie von weiblichen, männlichen, dicken, dünnen, großen, kleinen oder sportlichen Menschen auf die thermoregulatorischen Antworten auswirkt, ist noch sehr wage erforscht. Hier herrscht dringender Bedarf, auch von Seiten der messenden Institutionen, Validierungen

für numerisch gerechnete Modelle zu liefern um die zu benutzenden Koeffizienten gegebenenfalls anzupassen und zu justieren.

Weiterhin kann das Thermoregulations-Interface, sowie das Virtual Climate Chamber Projekt intensiver getestet und verbessert werden, frei nach dem Zitat von Edsger Wybe Dijkstra (1930-2002):

> "Program testing can be a very effective way to show the presence of bugs, but is hopelessly inadequate for showing their absence."

In diesem Sinne bleibt auf diesem äußert spannenden und hochaktuellen Feld noch viel zu tun!

Anhang A

THESEUS-FE

A.1 NASTRAN Geometriedatei

Die ASCII NASTRAN Geometriedatei, als Eingabe für THESEUS-FE, benötigt eine strikt definierte Struktur. Diese soll anhand eines einfachen Beispiels einer geschlossenen Kiste mit 6 Seiten illustriert werden.

```
$------------------------------------------------------------------------
$ BOX WITH 6 FACES
$
$ Total number of nodes    : 8
$ Total number of elements : 6 (6 quads / 0 trias)
$ Total number of groups   : 6
$   [   21,   22,   23,   24,   25,   26 ]
$
$+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
$1      2       3       4       5       6       7       8       9
$23456781234567812345678123456781234567812345678123456781234567812345678
$+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
BEGIN BULK
GRID           1        -0.33333-0.33333-0.33333
GRID           2         0.33333-0.33333-0.33333
GRID           3        -0.33333 0.33333-0.33333
GRID           4         0.33333 0.33333-0.33333
GRID           5        -0.33333-0.33333 0.33333
GRID           6         0.33333-0.33333 0.33333
GRID           7         0.33333 0.33333 0.33333
GRID           8        -0.33333 0.33333 0.33333
CQUAD4         1      21       1       2       4       3
CQUAD4         2      22       2       1       5       6
CQUAD4         3      23       4       2       6       7
CQUAD4         4      24       3       4       7       8
CQUAD4         5      25       1       3       8       5
CQUAD4         6      26       6       5       8       7
ENDDATA
```

Wie im Code eindeutig zu erkennen ist, können Kommentarzeilen mittels $-Zeichen eingeführt werden. Leerzeilen sind nicht erlaubt. Die Datentrennung beim Einlesen der NASTRAN Datei beruht auf abgezählten Abständen, weshalb es nicht nötig ist, Leerzeichen zwischen einzelnen Daten zu lassen. Das mag etwas befremdlich wirken, ist aber dringend einzuhalten.

Die Geometriedaten stehen zwischen `BEGIN BULK` und `ENDDATA`. Die Eingabe ist in Blöcken von jeweils 8 Zeichen unterteilt. In der Regel enthält der erste Block das entsprechende Keyword, das beschreibt, welche Daten in der Zeile folgen werden. Für die Geometrieeingabe in THESEUS-FE werden von dieser Arbeit nur die Schlüsselwörter `GRID` für die Punkteingabe, `CTRIA3` für die Definition von dreieckigen Oberflächen, und `CQUAD4` für viereckige Oberflächen benutzt. Den Schlüsselwörtern folgt anschließend die jeweils entsprechende Anzahl an Datenblöcken.

```
$+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
$1      2       3       4       5       6       7       8       9
$23456781234567812345678123456781234567812345678123456781234567812345678
$+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
$
GRID          NID          X-COORD Y-COORD Z-COORD
$
CTRIA3        EID     GID    NID1    NID2    NID3
$
CQUAD4        EID     GID    NID1    NID2    NID3    NID4
```

Hierbei bezeichnen `NID` die Knotennummer, `EID` die Elementnummer sowie `GID` die Gruppennummer, und werden jeweils als ganze Zahlen größer Null angegeben. Die im Block 2 angegebenen Knoten- und Elementnummern müssen eindeutig sein, und dürfen nicht mehrfach erscheinen. Zusätzlich dürfen keine Knoten vorhanden sein, die nicht von einem Element referenziert werden. Ist das trotzdem der Fall, bricht der Solver die Berechnung ab.

Die Koordinaten der Punkte werden in Fließkommadarstellung angegeben, und es muß darauf geachtet werden, dass eine Koordinate inklusive Vorzeichen und Komma nicht mehr als 8 Zeichen belegen darf.

Für weitere Informationen über andere Keywords, die von THESEUS-FE unterstützt werden, sei auf [27] und [29] verwiesen.

A.2 THESEUS-FE Input File

Folgendes Beispiel zeigt eine THESEUS-FE `.tfe` Datei, mit der der Solver gesteuert wird. Sie enthält eine Manikin-Definition des Fiala-Modells mit Winterbekleidung für eine unge-

koppelte Berechnung, und wurde vom Thermoregulations-Interface automatisch für ein Set Randbedingungen geschrieben und soll hier nur als Beispiel dienen, wie eine solche Datei auszusehen hat.

```
$
$ THESEUS-FE input file
$
$
$ Generated by Thermoregulation Interface
$
$-------------------------------------------------------------------------------
$ Title Definitions
$
TITLE1  Thermoregulation Interface File for an
TITLE2  Uncoupled Computation of the Fiala-Model
TITLE3  J. Frisch
TITLE4  CiE TUM
TITLE5  2008-05-15
$-------------------------------------------------------------------------------
$1      2       3       4       5       6       7       8       9      10     11
$2345678123456781234567812345678123456781234567812345678123456781234567812345678
$-------------------------------------------------------------------------------
$
$  Manikin Definition
$
$      |MANID  |TYPE   |MOD/ACT|SYSTEM |POSI   |BE     |ACT    |LINK   |
$      |T_W    |E_W    |T_A    |V_A    |RH     |T_BLA  |LEWIS  |       |
MANIKIN 1       FIALA-FEUNCPLD  ACTIVE  STND    15      8.1250-1
+       24.8    9.3000-124.8    5.0950-240
$
$      |MANID  |ICL    |       |       |       |       |       |       |
MANFPMV 1       1.33
$
$      |MANID  |BE_ID  |NAME   |TYPE   |NSEC   |       |L      |H_X    |
$      |A_SK   |A_SW   |A_DL   |A_CS   |A_SH   |A_M_W1 |A_M_W2 |       |
$      |A_NAT  |A_FR   |A_MIX  |       |       |       |       |       |
MANFBEL 1       1       HEAD    SPHERE  2               0.0     0.0
+       8.3500-29.5000-25.5000-23.0000-20.0     0.0     0.0
+       3       113     -5.7
$
$      |MANID  |BE_ID  |NAME   |NLAY   |ANGLE  |VF_SED |VF_STND|E_SF   |L      |
MANFSEC 1       1       FOREHEAD8       10      1       1       9.9000-1
+                       HEAD    8       170     9.0000-19.0000-18.0000-1
$
$      |MANID  |BE_ID  |NAME   |DISC   |R      |K      |RHO    |C      |W_BL   |Q_M    |A_RSP  |
MANFLAY 1       1       BRAIN1  3       3.0000-24.9000-11080    3850    0.01013 1.3400+40.0
+                       BRAIN2  3       7.6000-24.9000-11080    3850    0.01013 1.3400+40.0
+                       BRAIN3  3       8.6000-24.9000-11080    3850    0.01013 1.3400+40.0
+                       BONE    3       1.0050-11.16    1500    1591    0.00    0.0     0.0
+                       FAT     3       1.0200-11.6000-1850     2300    3.6000-658      0.0
+                       SKIN    3       1.0300-14.7000-11085    3680    0.01096 736     0.0
+                       XSKIN   3       1.0400-14.7000-11085    3680    0.00    0.0     0.0
$
$      |MANID  |BE_ID  |SEC_ID |DISC   |THICKN |SICL   |SIICL  |NSM    |C      |
MANFSCL 1       1       1       3       1.0000-51.0000-51.0000-51.0000-51.0000-5
```

```
MANFSCL 1       1       2       3       1.0000-51.0000-51.0000-51.0000-51.0000-5
$
$       |MANID  |BE_ID  |NAME   |TYPE   |NSEC   |       |L      |H_X    |
$       |A_SK   |A_SW   |A_DL   |A_CS   |A_SH   |A_M_W1 |A_M_W2 |       |
$       |A_NAT  |A_FR   |A_MIX  |       |       |       |       |       |
MANFBEL 1       2       FACE    CYLINDR 1               9.8400-20.0
+       4.1800-25.4000-24.6000-23.3000-22.0000-30.0     0.0
+       3       113     -5.7
$
$       |MANID  |BE_ID  |NAME   |NLAY   |ANGLE  |VF_SED |VF_STND|E_SF   |L      |
MANFSEC 1       2       FACE    7       210     9.0000-19.0000-19.9000-1
$
$       |MANID  |BE_ID  |NAME   |DISC   |R      |K      |RHO    |C      |W_BL   |Q_M    |A_RSP  |
MANFLAY 1       2       MUSCLE1 3       2.6800-24.2000-11085    3768    5.3800-4684     2.0000-1
+                       BONE    3       5.4200-21.16    1500    1591    0.00    0.0     0.0
+                       MUSCLE2 3       6.8000-24.2000-11085    3768    5.3800-4684     2.5000-1
+                       FAT     3       7.6000-21.6000-1850     2300    3.6000-658      0.0
+                       SKIN    3       7.7000-24.7000-11085    3680    0.02234 736     0.0
+                       XSKIN   3       7.8000-24.7000-11085    3680    0.00    0.0     0.0
$
$       |MANID  |BE_ID  |SEC_ID |DISC   |THICKN |SICL   |SIICL  |NSM    |C      |
MANFSCL 1       2       1       3       1.0000-51.0000-51.0000-51.0000-51.0000-5
$
$       |MANID  |BE_ID  |NAME   |TYPE   |NSEC   |       |L      |H_X    |
$       |A_SK   |A_SW   |A_DL   |A_CS   |A_SH   |A_M_W1 |A_M_W2 |       |
$       |A_NAT  |A_FR   |A_MIX  |       |       |       |       |       |
MANFBEL 1       3       NECK    CYLINDR 2               8.4200-20.0
+       4.1700-24.2000-23.1000-22.5000-22.0000-33.0000-21.0000-2
+       1.6     130     -6.5
$
$       |MANID  |BE_ID  |NAME   |NLAY   |ANGLE  |VF_SED |VF_STND|E_SF   |L      |
MANFSEC 1       3       ANTERIOR6       180     7.0000-17.0000-19.9000-1
+                       POSTERIO6       180     7.5000-17.5000-19.9000-1
$
$       |MANID  |BE_ID  |NAME   |DISC   |R      |K      |RHO    |C      |W_BL   |Q_M    |A_RSP  |
MANFLAY 1       3       BONE    3       1.9000-27.5000-11357    1700    0.00    0.0     0.0
+                       MUSCLE  3       5.4600-24.2000-11085    3768    5.3800-4684     1.2500-1
+                       FAT     3       5.5600-21.6000-1850     2300    3.6000-658      0.0
+                       SKIN    3       5.6100-24.7000-11085    3680    0.01360 736     0.0
+                       XSKIN2  3       5.6700-24.7000-11085    3680    0.00    0.0     0.0
$
$       |MANID  |BE_ID  |SEC_ID |DISC   |THICKN |SICL   |SIICL  |NSM    |C      |
MANFSCL 1       3       1       3       2.6000-21.2216  1.7134  1.40255 1150
MANFSCL 1       3       2       3       2.6000-21.2216  1.7134  1.40255 1150
$
$       |MANID  |BE_ID  |NAME   |TYPE   |NSEC   |       |L      |H_X    |
$       |A_SK   |A_SW   |A_DL   |A_CS   |A_SH   |A_M_W1 |A_M_W2 |       |
$       |A_NAT  |A_FR   |A_MIX  |       |       |       |       |       |
MANFBEL 1       4       SHLD LE CYLINDR 1               1.6000-10.0
+       1.5000-21.8500-21.0000-21.0000-21.0000-42.5000-21.0000-2
+       5.9     216     -10.8
$
$       |MANID  |BE_ID  |NAME   |NLAY   |ANGLE  |VF_SED |VF_STND|E_SF   |L      |
MANFSEC 1       4       SHLD LE 6       130     9.0000-19.0000-19.9000-1
$
```

```
$      |MANID  |BE_ID  |NAME   |DISC   |R      |K      |RHO    |C      |W_BL   |Q_M    |A_RSP  |
MANFLAY 1       4       BONE    3       3.7000-27.5000-11357    1700    0.00    0.0     0.0
+                       MUSCLE  3       3.9000-24.2000-11085    3768    5.3800-4684     0.0
+                       FAT     3       4.4000-21.6000-1850     2300    3.6000-658      0.0
+                       SKIN    3       4.5000-24.7000-11085    3680    2.0200-3736     0.0
+                       XSKIN   3       4.6000-24.7000-11085    3680    0.00    0.0     0.0
$
$      |MANID  |BE_ID  |SEC_ID |DISC   |THICKN |SICL   |SIICL  |NSM    |C      |
MANFSCL 1       4       1       3       2.4000-22.1982  3.6185  1.5374  1200
$
$      |MANID  |BE_ID  |NAME   |TYPE   |NSEC   |       |L      |H_X    |
$      |A_SK   |A_SW   |A_DL   |A_CS   |A_SH   |A_M_W1 |A_M_W2 |       |
$      |A_NAT  |A_FR   |A_MIX  |       |       |       |       |       |
MANFBEL 1       5       SHLD RI CYLINDR 1               1.6000-10.0
+       1.5000-21.8500-21.0000-21.0000-21.0000-42.5000-21.0000-2
+       5.9     216     -10.8
$
$      |MANID  |BE_ID  |NAME   |NLAY   |ANGLE  |VF_SED |VF_STND|E_SF   |L      |
MANFSEC 1       5       SHLD RI 6       130     9.0000-19.0000-19.9000-1
$
$      |MANID  |BE_ID  |NAME   |DISC   |R      |K      |RHO    |C      |W_BL   |Q_M    |A_RSP  |
MANFLAY 1       5       BONE    3       3.7000-27.5000-11357    1700    0.00    0.0     0.0
+                       MUSCLE  3       3.9000-24.2000-11085    3768    5.3800-4684     0.0
+                       FAT     3       4.4000-21.6000-1850     2300    3.6000-658      0.0
+                       SKIN    3       4.5000-24.7000-11085    3680    2.0200-3736     0.0
+                       XSKIN   3       4.6000-24.7000-11085    3680    0.00    0.0     0.0
$
$      |MANID  |BE_ID  |SEC_ID |DISC   |THICKN |SICL   |SIICL  |NSM    |C      |
MANFSCL 1       5       1       3       2.4000-22.1982  3.6185  1.5374  1200
$
$      |MANID  |BE_ID  |NAME   |TYPE   |NSEC   |       |L      |H_X    |
$      |A_SK   |A_SW   |A_DL   |A_CS   |A_SH   |A_M_W1 |A_M_W2 |       |
$      |A_NAT  |A_FR   |A_MIX  |       |       |       |       |       |
MANFBEL 1       6       THORAX  CYLINDR 6               3.0600-10.0
+       1.2900-11.0100-11.4100-15.0000-46.3050-11.2000-17.0000-2
+       5.0000-1180     -7.4
$
$      |MANID  |BE_ID  |NAME   |NLAY   |ANGLE  |VF_SED |VF_STND|E_SF   |L      |
MANFSEC 1       6       UP ANT  7       150     8.0000-19.0000-19.9000-12.0400-1
+                       UP POS  7       150     9.5000-19.5000-19.9000-12.0400-1
+                       UP INF  7       60      5.0000-21.0000-19.9000-12.0400-1
+                       LO ANT  7       150     8.0000-19.0000-19.9000-11.0200-1
+                       LO POS  7       150     9.5000-19.5000-19.9000-11.0200-1
+                       LO INF  7       60      5.0000-21.0000-19.9000-11.0200-1
$
$      |MANID  |BE_ID  |NAME   |DISC   |R      |K      |RHO    |C      |W_BL   |Q_M    |A_RSP  |
MANFLAY 1       6       LUNG    3       7.7300-22.8000-1550     3718    CARDOUT 600     5.0000-2
+                       BONE    3       8.9100-27.5000-11357    1700    0.00    0.0     0.0
+                       MUSCLE  3       1.2340-14.2000-11085    3768    5.3800-4684     0.0
+                       FAT     3       1.2680-11.6000-1850     2300    3.6000-658      0.0
+                       SKIN    3       1.2790-14.7000-11085    3680    3.1600-3736     0.0
+                       XSKIN   3       1.2900-14.7000-11085    3680    0.00    0.0     0.0
$
$      |MANID  |BE_ID  |SEC_ID |DISC   |THICKN |SICL   |SIICL  |NSM    |C      |
MANFSCL 1       6       1       3       2.4300-22.7748  4.3421  1.5374  1200
```

```
MANFSCL 1       6       2       3       2.4300-22.7748  4.3421  1.5374  1200
MANFSCL 1       6       3       3       2.4300-22.7748  4.3421  1.5374  1200
MANFSCL 1       6       4       3       2.4300-22.7748  4.3421  1.5374  1200
MANFSCL 1       6       5       3       2.4300-22.7748  4.3421  1.5374  1200
MANFSCL 1       6       6       3       2.4300-22.7748  4.3421  1.5374  1200
$
$      |MANID  |BE_ID  |NAME   |TYPE   |NSEC   |       |L      |H_X    |
$      |A_SK   |A_SW   |A_DL   |A_CS   |A_SH   |A_M_W1 |A_M_W2 |       |
$      |A_NAT  |A_FR   |A_MIX  |       |       |       |       |       |
MANFBEL 1       7       ABDOMEN CYLINDR 3               5.5200-10.0
+       1.2100-11.8100-11.6100-12.0500-22.4000-14.6000-12.0000-1
+       1.2     180     -9
$
$      |MANID  |BE_ID  |NAME   |NLAY   |ANGLE  |VF_SED |VF_STND|E_SF   |L      |
MANFSEC 1       7       ANTERIOR7       150     8.0000-19.0000-19.9000-1
+                       POSTERIO7       150     9.5000-19.5000-19.9000-1
+                       INFERIOR7       60      2.0000-13.0000-19.9000-1
$
$      |MANID  |BE_ID  |NAME   |DISC   |R      |K      |RHO    |C      |W_BL   |Q_M    |A_RSP  |
MANFLAY 1       7       CORE    3       7.8500-25.3000-11000    3697    4.3100-34100    0.0
+                       BONE    3       8.3400-27.5000-11357    1700    0.00    0.0     0.0
+                       MUSCLE  3       1.0900-14.2000-11085    3768    5.3800-4684     0.0
+                       FAT     3       1.2440-11.6000-1850     2300    3.6000-658      0.0
+                       SKIN    3       1.2520-14.7000-11085    3680    2.8800-3736     0.0
+                       XSKIN   3       1.2600-14.7000-11085    3680    0.00    0.0     0.0
$
$      |MANID  |BE_ID  |SEC_ID |DISC   |THICKN |SICL   |SIICL  |NSM    |C      |
MANFSCL 1       7       1       3       2.3700-22.9354  4.6721  2.12528 1300
MANFSCL 1       7       2       3       2.3700-22.9354  4.6721  2.12528 1300
MANFSCL 1       7       3       3       2.3700-22.9354  4.6721  2.12528 1300
$
$      |MANID  |BE_ID  |NAME   |TYPE   |NSEC   |       |L      |H_X    |
$      |A_SK   |A_SW   |A_DL   |A_CS   |A_SH   |A_M_W1 |A_M_W2 |       |
$      |A_NAT  |A_FR   |A_MIX  |       |       |       |       |       |
MANFBEL 1       8       ARM LE  CYLINDR 6               6.3700-12.065
+       9.0000-26.6500-24.7500-21.9450-12.0000-29.5000-24.0000-2
+       8.3     216     -10.8
$
$      |MANID  |BE_ID  |NAME   |NLAY   |ANGLE  |VF_SED |VF_STND|E_SF   |L      |
MANFSEC 1       8       UP ANT  6       150     7.5000-18.5000-19.9000-13.1850-1
+                       UP POS  6       120     8.0000-19.0000-19.9000-13.1850-1
+                       UP INF  6       90      1.0000-12.0000-19.9000-13.1850-1
+                       LO ANT  6       150     7.5000-18.5000-19.9000-13.1850-1
+                       LO POS  6       120     8.0000-19.0000-19.9000-13.1850-1
+                       LO INF  6       90      1.0000-12.0000-19.9000-13.1850-1
$
$      |MANID  |BE_ID  |NAME   |DISC   |R      |K      |RHO    |C      |W_BL   |Q_M    |A_RSP  |
MANFLAY 1       8       BONE    3       1.5300-27.5000-11357    1700    0.00    0.0     0.0
+                       MUSCLE  3       3.4300-24.2000-11085    3768    5.3800-4684     0.0
+                       FAT     3       4.0100-21.6000-1850     2300    3.6000-658      0.0
+                       SKIN    3       4.0950-24.7000-11085    3680    2.2000-3736     0.0
+                       XSKIN   3       4.1800-24.7000-11085    3680    0.00    0.0     0.0
$
$      |MANID  |BE_ID  |SEC_ID |DISC   |THICKN |SICL   |SIICL  |NSM    |C      |
MANFSCL 1       8       1       3       2.4100-21.6996  2.4478  1.5374  1200
```

```
MANFSCL 1       8       2       3       2.4100-21.6996  2.4478  1.5374  1200
MANFSCL 1       8       3       3       2.4100-21.6996  2.4478  1.5374  1200
MANFSCL 1       8       4       3       2.4100-21.6996  2.4478  1.5374  1200
MANFSCL 1       8       5       3       2.4100-21.6996  2.4478  1.5374  1200
MANFSCL 1       8       6       3       2.4100-21.6996  2.4478  1.5374  1200
$
$      |MANID  |BE_ID  |NAME   |TYPE   |NSEC   |       |L      |H_X    |
$      |A_SK   |A_SW   |A_DL   |A_CS   |A_SH   |A_M_W1 |A_M_W2 |       |
$      |A_NAT  |A_FR   |A_MIX  |       |       |       |       |       |
MANFBEL 1       9       ARM RI  CYLINDR 6               6.3700-12.065
+       9.0000-26.6500-24.7500-21.9450-12.0000-29.5000-24.0000-2
+       8.3     216     -10.8
$
$      |MANID  |BE_ID  |NAME   |NLAY   |ANGLE  |VF_SED |VF_STND|E_SF   |L      |
MANFSEC 1       9       UP ANT  6       150     7.5000-18.5000-19.9000-13.1850-1
+                       UP POS  6       120     8.0000-19.0000-19.9000-13.1850-1
+                       UP INF  6       90      1.0000-12.0000-19.9000-13.1850-1
+                       LO ANT  6       150     7.5000-18.5000-19.9000-13.1850-1
+                       LO POS  6       120     8.0000-19.0000-19.9000-13.1850-1
+                       LO INF  6       90      1.0000-12.0000-19.9000-13.1850-1
$
$      |MANID  |BE_ID  |NAME   |DISC   |R      |K      |RHO    |C      |W_BL   |Q_M    |A_RSP  |
MANFLAY 1       9       BONE    3       1.5300-27.5000-11357    1700    0.00    0.0     0.0
+                       MUSCLE  3       3.4300-24.2000-11085    3768    5.3800-4684     0.0
+                       FAT     3       4.0100-21.6000-1850     2300    3.6000-658      0.0
+                       SKIN    3       4.0950-24.7000-11085    3680    2.2000-3736     0.0
+                       XSKIN   3       4.1800-24.7000-11085    3680    0.00    0.0     0.0
$
$      |MANID  |BE_ID  |SEC_ID |DISC   |THICKN |SICL   |SIICL  |NSM    |C      |
MANFSCL 1       9       1       3       2.4100-21.6996  2.4478  1.5374  1200
MANFSCL 1       9       2       3       2.4100-21.6996  2.4478  1.5374  1200
MANFSCL 1       9       3       3       2.4100-21.6996  2.4478  1.5374  1200
MANFSCL 1       9       4       3       2.4100-21.6996  2.4478  1.5374  1200
MANFSCL 1       9       5       3       2.4100-21.6996  2.4478  1.5374  1200
MANFSCL 1       9       6       3       2.4100-21.6996  2.4478  1.5374  1200
$
$      |MANID  |BE_ID  |NAME   |TYPE   |NSEC   |       |L      |H_X    |
$      |A_SK   |A_SW   |A_DL   |A_CS   |A_SH   |A_M_W1 |A_M_W2 |       |
$      |A_NAT  |A_FR   |A_MIX  |       |       |       |       |       |
MANFBEL 1       10      HAND LE CYLINDR 2               3.1000-12.8500-1
+       4.5000-22.4500-26.0500-21.1000-11.0000-31.0000-25.0000-3
+       8.3     216     -10.8
$
$      |MANID  |BE_ID  |NAME   |NLAY   |ANGLE  |VF_SED |VF_STND|E_SF   |L      |
MANFSEC 1       10      HANDBACK6       180     8.0000-18.0000-19.9000-1
+                       PALM    6       180     1.0000-12.0000-19.9000-1
$
$      |MANID  |BE_ID  |NAME   |DISC   |R      |K      |RHO    |C      |W_BL   |Q_M    |A_RSP  |
MANFLAY 1       10      BONE    3       7.0000-37.5000-11357    1700    0.00    0.0     0.0
+                       MUSCLE  3       1.7400-24.2000-11085    3768    5.3800-4684     0.0
+                       FAT     3       2.0400-21.6000-1850     2300    3.6000-658      0.0
+                       SKIN    3       2.1500-24.7000-11085    3680    9.0800-3736     0.0
+                       XSKIN   3       2.2600-24.7000-11085    3680    0.00    0.0     0.0
$
$      |MANID  |BE_ID  |SEC_ID |DISC   |THICKN |SICL   |SIICL  |NSM    |C      |
```

```
MANFSCL 1       10      1       3       1.0000-51.0000-51.0000-51.0000-51.0000-5
MANFSCL 1       10      2       3       1.0000-51.0000-51.0000-51.0000-51.0000-5
$
$       |MANID  |BE_ID  |NAME   |TYPE   |NSEC   |       |L      |H_X    |
$       |A_SK   |A_SW   |A_DL   |A_CS   |A_SH   |A_M_W1 |A_M_W2 |       |
$       |A_NAT  |A_FR   |A_MIX  |       |       |       |       |       |
MANFBEL 1       11      HAND RI CYLINDR 2               3.1000-12.8500-1
+       4.5000-22.4500-26.0500-21.1000-11.0000-31.0000-25.0000-3
+       8.3     216     -10.8
$
$       |MANID  |BE_ID  |NAME   |NLAY   |ANGLE  |VF_SED |VF_STND|E_SF   |L      |
MANFSEC 1       11      HANDBACK6       180     8.0000-18.0000-19.9000-1
+                       PALM    6       180     1.0000-12.0000-19.9000-1
$
$       |MANID  |BE_ID  |NAME   |DISC   |R      |K      |RHO    |C      |W_BL   |Q_M    |A_RSP  |
MANFLAY 1       11      BONE    3       7.0000-37.5000-11357    1700    0.00    0.0     0.0
+                       MUSCLE  3       1.7400-24.2000-11085    3768    5.3800-4684     0.0
+                       FAT     3       2.0400-21.6000-1850     2300    3.6000-658      0.0
+                       SKIN    3       2.1500-24.7000-11085    3680    9.0800-3736     0.0
+                       XSKIN   3       2.2600-24.7000-11085    3680    0.00    0.0     0.0
$
$       |MANID  |BE_ID  |SEC_ID |DISC   |THICKN |SICL   |SIICL  |NSM    |C      |
MANFSCL 1       11      1       3       1.0000-51.0000-51.0000-51.0000-51.0000-5
MANFSCL 1       11      2       3       1.0000-51.0000-51.0000-51.0000-51.0000-5
$
$       |MANID  |BE_ID  |NAME   |TYPE   |NSEC   |       |L      |H_X    |
$       |A_SK   |A_SW   |A_DL   |A_CS   |A_SH   |A_M_W1 |A_M_W2 |       |
$       |A_NAT  |A_FR   |A_MIX  |       |       |       |       |       |
MANFBEL 1       12      LEG LE  CYLINDR 6               6.9500-13.45
+       1.0400-11.3050-11.1500-12.0000-14.0650-25.5000-23.0000-1
+       5.3     220     -11
$
$       |MANID  |BE_ID  |NAME   |NLAY   |ANGLE  |VF_SED |VF_STND|E_SF   |L      |
MANFSEC 1       12      UP ANT  6       180     8.5000-19.0000-19.9000-13.4750-1
+                       UP POS  6       115     9.5000-19.0000-19.9000-13.4750-1
+                       UP INF  6       65      1.0000-16.5000-19.9000-13.4750-1
+                       LO ANT  6       150     8.5000-19.0000-19.9000-13.4750-1
+                       LO POS  6       150     9.5000-19.0000-19.9000-13.4750-1
+                       LO INF  6       60      1.0000-16.5000-19.9000-13.4750-1
$
$       |MANID  |BE_ID  |NAME   |DISC   |R      |K      |RHO    |C      |W_BL   |Q_M    |A_RSP  |
MANFLAY 1       12      BONE    3       2.2000-27.5000-11357    1700    0.00    0.0     0.0
+                       MUSCLE  3       4.8000-24.2000-11085    3768    5.3800-4684     0.0
+                       FAT     3       5.3300-21.6000-1850     2300    3.6000-658      0.0
+                       SKIN    3       5.4300-24.7000-11085    3680    2.1000-3736     0.0
+                       XSKIN   3       5.5300-24.7000-11085    3680    0.00    0.0     0.0
$
$       |MANID  |BE_ID  |SEC_ID |DISC   |THICKN |SICL   |SIICL  |NSM    |C      |
MANFSCL 1       12      1       3       1.7400-21.0901  2.4158  4.3914-11200
MANFSCL 1       12      2       3       1.7400-21.0901  2.4158  4.3914-11200
MANFSCL 1       12      3       3       1.7400-21.0901  2.4158  4.3914-11200
MANFSCL 1       12      4       3       1.7400-21.0901  2.4158  4.3914-11200
MANFSCL 1       12      5       3       1.7400-21.0901  2.4158  4.3914-11200
MANFSCL 1       12      6       3       1.7400-21.0901  2.4158  4.3914-11200
$
```

```
$       |MANID  |BE_ID  |NAME   |TYPE   |NSEC   |       |L      |H_X    |
$       |A_SK   |A_SW   |A_DL   |A_CS   |A_SH   |A_M_W1 |A_M_W2 |       |
$       |A_NAT  |A_FR   |A_MIX  |       |       |       |       |       |
MANFBEL 1       13      LEG RI  CYLINDR 6               6.9500-13.45
+       1.0400-11.3050-11.1500-12.0000-14.0650-25.5000-23.0000-1
+       5.3     220     -11
$
$       |MANID  |BE_ID  |NAME   |NLAY   |ANGLE  |VF_SED |VF_STND|E_SF   |L      |
MANFSEC 1       13      UP ANT  6       180     8.5000-19.0000-19.9000-13.4750-1
+                       UP POS  6       115     9.5000-19.0000-19.9000-13.4750-1
+                       UP INF  6       65      1.0000-16.5000-19.9000-13.4750-1
+                       LO ANT  6       150     8.5000-19.0000-19.9000-13.4750-1
+                       LO POS  6       150     9.5000-19.0000-19.9000-13.4750-1
+                       LO INF  6       60      1.0000-16.5000-19.9000-13.4750-1
$
$       |MANID  |BE_ID  |NAME   |DISC   |R      |K      |RHO    |C      |W_BL   |Q_M    |A_RSP  |
MANFLAY 1       13      BONE    3       2.2000-27.5000-11357    1700    0.00    0.0     0.0
+                       MUSCLE  3       4.8000-24.2000-11085    3768    5.3800-4684     0.0
+                       FAT     3       5.3300-21.6000-1850     2300    3.6000-658      0.0
+                       SKIN    3       5.4300-24.7000-11085    3680    2.1000-3736     0.0
+                       XSKIN   3       5.5300-24.7000-11085    3680    0.00    0.0     0.0
$
$       |MANID  |BE_ID  |SEC_ID |DISC   |THICKN |SICL   |SIICL  |NSM    |C      |
MANFSCL 1       13      1       3       1.7400-21.0901  2.4158  4.3914-11200
MANFSCL 1       13      2       3       1.7400-21.0901  2.4158  4.3914-11200
MANFSCL 1       13      3       3       1.7400-21.0901  2.4158  4.3914-11200
MANFSCL 1       13      4       3       1.7400-21.0901  2.4158  4.3914-11200
MANFSCL 1       13      5       3       1.7400-21.0901  2.4158  4.3914-11200
MANFSCL 1       13      6       3       1.7400-21.0901  2.4158  4.3914-11200
$
$       |MANID  |BE_ID  |NAME   |TYPE   |NSEC   |       |L      |H_X    |
$       |A_SK   |A_SW   |A_DL   |A_CS   |A_SH   |A_M_W1 |A_M_W2 |       |
$       |A_NAT  |A_FR   |A_MIX  |       |       |       |       |       |
MANFBEL 1       14      FOOT LE CYLINDR 2               2.4000-11.7
+       3.7500-22.3500-25.0000-23.7650-11.0000-31.0000-25.0000-3
+       6.8     210     -10.5
$
$       |MANID  |BE_ID  |NAME   |NLAY   |ANGLE  |VF_SED |VF_STND|E_SF   |L      |
MANFSEC 1       14      INSTEP  6       180     9.0000-19.0000-19.9000-1
+                       SOLE    6       180     1       1       9.9000-1
$
$       |MANID  |BE_ID  |NAME   |DISC   |R      |K      |RHO    |C      |W_BL   |Q_M    |A_RSP  |
MANFLAY 1       14      BONE    3       2.0000-27.5000-11357    1700    0.00    0.0     0.0
+                       MUSCLE  3       2.5000-24.2000-11085    3768    5.3800-4684     0.0
+                       FAT     3       3.2600-21.6000-1850     2300    3.6000-658      0.0
+                       SKIN    3       3.3800-24.7000-11085    3680    3.0000-3736     0.0
+                       XSKIN   3       3.5000-24.7000-11085    3680    0.00    0.0     0.0
$
$       |MANID  |BE_ID  |SEC_ID |DISC   |THICKN |SICL   |SIICL  |NSM    |C      |
MANFSCL 1       14      1       3       7.7000-31.5106  2.3839  12.3061 1550
MANFSCL 1       14      2       3       7.7000-31.5106  2.3839  12.3061 1550
$
$       |MANID  |BE_ID  |NAME   |TYPE   |NSEC   |       |L      |H_X    |
$       |A_SK   |A_SW   |A_DL   |A_CS   |A_SH   |A_M_W1 |A_M_W2 |       |
$       |A_NAT  |A_FR   |A_MIX  |       |       |       |       |       |
```

```
MANFBEL 1       15      FOOT RI CYLINDR 2               2.4000-11.7
+       3.7500-22.3500-25.0000-23.7650-11.0000-31.0000-25.0000-3
+       6.8     210     -10.5
$
$      |MANID  |BE_ID  |NAME   |NLAY   |ANGLE  |VF_SED |VF_STND|E_SF   |L      |
MANFSEC 1       15      INSTEP  6       180     9.0000-19.0000-19.9000-1
+                       SOLE    6       180     1       1       9.9000-1
$
$      |MANID  |BE_ID  |NAME   |DISC   |R      |K      |RHO    |C      |W_BL   |Q_M    |A_RSP  |
MANFLAY 1       15      BONE    3       2.0000-27.5000-11357    1700    0.00    0.0     0.0
+                       MUSCLE  3       2.5000-24.2000-11085    3768    5.3800-4684     0.0
+                       FAT     3       3.2600-21.6000-1850     2300    3.6000-658      0.0
+                       SKIN    3       3.3800-24.7000-11085    3680    3.0000-3736     0.0
+                       XSKIN   3       3.5000-24.7000-11085    3680    0.00    0.0     0.0
$
$      |MANID  |BE_ID  |SEC_ID |DISC   |THICKN |SICL   |SIICL  |NSM    |C      |
MANFSCL 1       15      1       3       7.7000-31.5106  2.3839  12.3061 1550
MANFSCL 1       15      2       3       7.7000-31.5106  2.3839  12.3061 1550
$
$-------------------------------------------------------------------------------
$  Solver Settings
$
$      |SOLTYP |TTYP   |DTO    |TEND   |ADPTOL |MXDT   |SSCOND |CONVCRT|MINITER|
$      |MAXITER|MAXSTEP|RELAX  |PRED   |DIVISOR|CRIT2  |MINDT  |DIVITER|MINQ   |
SOL     TRANS2  ADAPTIV 1.0000-110      1.0000-160              1.0000-21
+       100     1000000 0.0             1000    1.0000-11.0000-72       100
$
$-------------------------------------------------------------------------------
$  Postprcessor Settings
$
$      |DTPOST |OPT1   |OPT2   |
POSTFRQ END
$
$-------------------------------------------------------------------------------
$  Writing Restart Data to File
$
RESTWR
$
$----------------------- End of THESEUS-FE input file ------------------------
```

A.3 HDF Ergebnisdatei

Der THESEUS-FE Solver liefert sämtliche Ergebnisse in einer `.hdf` Datei zurück. Eine komplette Dokumentation kann unter [7] gefunden werden. HDF ist ein OpenSource Projekt und steht für *Hierarchical Data Format*. Es kann extrem große und komplexe Datenmengen handhaben, und es existieren APIs[1] für `C++` und `Fortran`.

[1] API: Application Programming Interface

Bei sämtlichen Betrachtungen im Rahmen dieser Arbeit, wurden die Quellen von HDF5 in der Version 1.8.0 unter Linux und Windows übersetzt und als Bibliothek an die Programme gelinkt, die HDF Funktionen ausführen sollen.

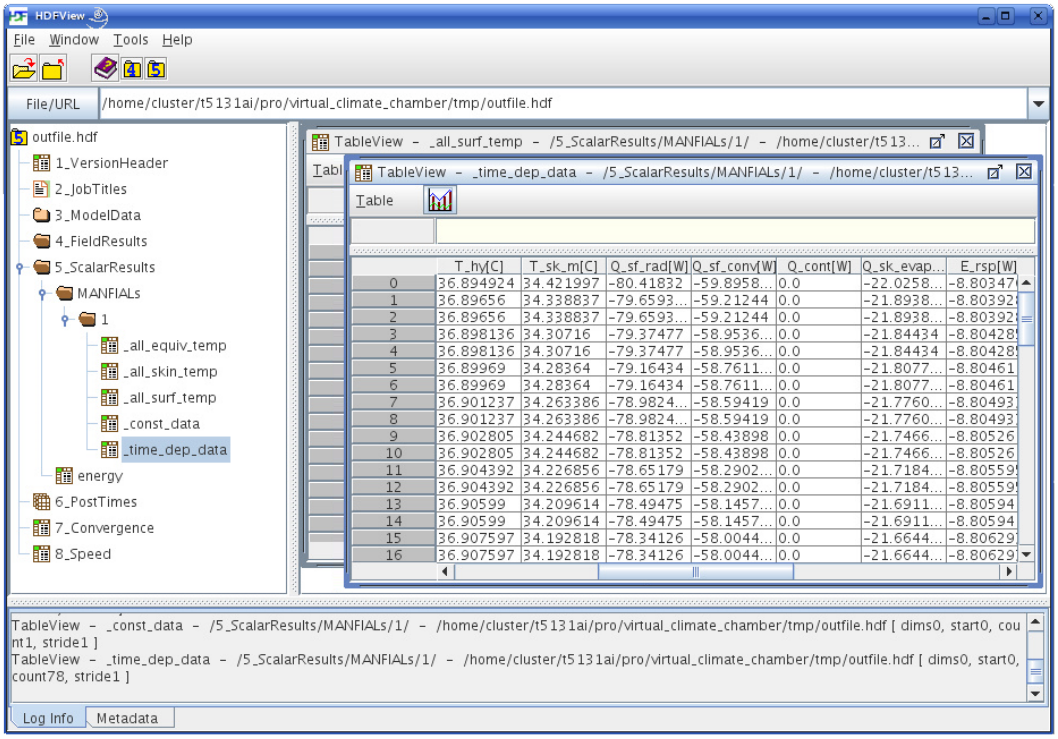


Abbildung A.1: Screenshot des HDF-Viewers [7]

In Abb. A.1 ist ein Screenshot des HDF-Viewers [7] zu sehen, mit dem die von THESEUS-FE erzeugte `.hdf` Datei betrachtet werden kann. Die interne Speicherstruktur ist zu erkennen und wird in [29] detailliert erläutert.

A.4 Äquivalenttemperatur

A.4.1 Definition

In DIN EN ISO 14505-1:2006 Abs. 3.2 ist die Äquivalenttemperatur definiert durch:

Temperatur eines homogenen „Raumes“ mit einer mittleren Strahlungstemperatur gleich der Lufttemperatur und einer Luftgeschwindigkeit von Null, in dem der Wärmeverlust einer Person durch Konvektion und Strahlung gleich ist, wie unter den tatsächlichen Bedingungen.

Unter Abs. 3.5 ist die lokale Äquivalenttemperatur definiert:

Temperatur eines homogenen „Raumes“ mit einer mittleren Strahlungstemperatur gleich der Lufttemperatur und einer Luftgeschwindigkeit von Null, in dem der Wärmeverlust eines bestimmten Bereiches der Körperoberfläche einer Person durch Konvektion und Strahlung gleich ist, wie unter den tatsächlichen Bedingungen.

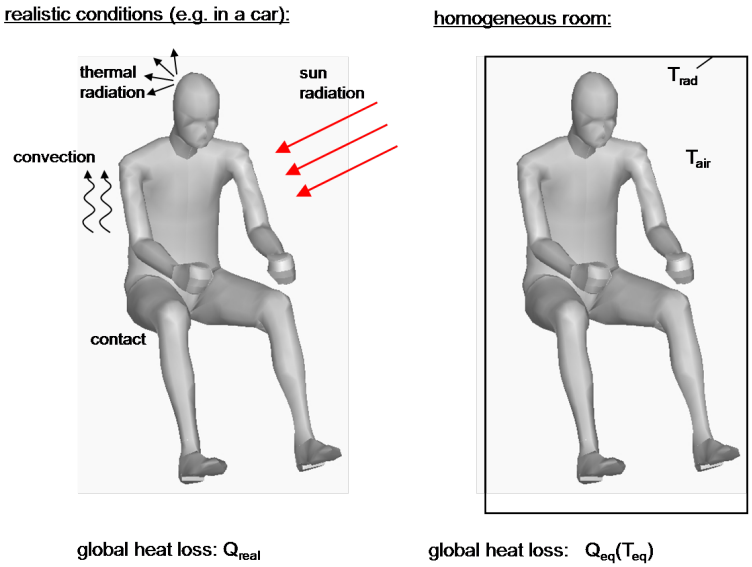


Abbildung A.2: Definition der Äquivalenttemperatur [28]

Würde die Windgeschwindigkeit ganz zu Null gesetzt werden, könnten numerische Probleme auftreten. Deshalb wird ein Wert von 0.1 m/s gewählt. (möglich nach DIN EN ISO 14505-2:2006 Abs. 4.2)

A.4.2 Berechnung

Umgesetzt bedeutet die Definition

$$Q_{real} = Q_{eq}(T_{eq}) \quad . \tag{A.1}$$

Hierbei bezeichnet Q_{real} den tatsächlichen (oder simulierten) Wärmestrom und $Q_{eq}(T_{eq})$ den Wärmestrom, der bei einer Äquivalenttemperatur T_{eq} fließt. Umgeformt nach T_{eq} ergibt sich:

$$T_{eq} = T_{sf} - \frac{Q_{real}}{A_{sf} \cdot (h_{c,mix} + h_r)} \tag{A.2}$$

$$T_{eq} = T_{sf} - \frac{q_{real}}{h_{c,mix} + h_r} \quad . \tag{A.3}$$

Hierbei steht T_{sf} für die Oberflächentemperatur des betrachteten Segmentes, h_r (Gleichung 2.13) und $h_{c,mix}$ (Gleichung 2.11) für die von Fiala definierten Koeffizienten für Strahlung und gemische Konvektion unter der Verwendung von $v_{a,eff} = 0.1$ m/s und $\varepsilon_w = 0.93$. Die Gleichung wird anschließend mittels Fixpunktiteration gelöst.

Siehe hierzu ebenfalls [28].

A.5 Änderung von 10 auf 15 Körperelemente

Wie bereits in Punkt 2.4.2 erwähnt, besitzt THESEUS-FE eine erweiterte Definition der Körperelemente im Vergleich zum Original. Es werden 15 Elemente benutzt, bei der eine Unterscheidung in Elemente der rechten und linken Körperhälfte möglich ist. Bei der Erweiterung der 10 Elemente-Version auf 15 Elemente ist auf folgende Punkte zu achten:

- Die Elemente Schulter, Arm, Bein, Hand und Fuß werden in jeweils zwei Teile aufgeteilt: rechte Schulter, linke Schulter, etc.
- Die Körperlängen der Elemente werden im Vergleich zum Original halbiert (als Beispiel: Das Bein hat nach dem Original Fiala-Modell laut Tabelle 2 in [19] eine Länge von 139 cm. Die Länge der Beine nach dem neuen Modell ergeben sich demnach zu 69.5 cm.)
- Die Gegenstromwärmeaustauschkoeffizenten (countercurrent heat exchange) h_x werden halbiert (außer bei der Schulter, wo sie aus dem in 2.4.2 erwähnten Grunde zu 0 gesetzt werden).
- Die Regelungskoeffizienten des aktiven Systems werden halbiert (a_{cs} allerdings nicht). Siehe hierzu ebenfalls Abb. 4-9 in [28].

Sind diese Änderungen vorgenommen, so wird eine Kontrollrechnung durchgeführt, bei der es *keinerlei* Abweichung zu dem 10 elementigen Modell geben darf. Schon die kleinste Abweichung zeigt auf einen Fehler im aufgeteilten Modell hin.

Wird zusätzlich im Debug-Modus von THESEUS-FE gearbeitet (einstellbar durch das Keyword `MANFDBG 1` in der `.tfe` Datei), so werden die Summen der jeweiligen Koeffizienten in die Protokolldatei geschrieben, die sich *immer* zu 1.0000 ergeben müssen.

Anhang B

Thermoregulations-Interface

B.1 Reihenfolge der Sektoren in Temperaturfeldern

Sämtliche Temperaturfelder, die vom Thermoregulations-Interface zurückgeliefert werden, besitzen in der gegenwärtigen Form 48 Einträge für 48 Sektoren. Die Reihenfolge ist festgelegt, da zum Befüllen der Felder zwei verschachtelte Schleifen abgearbeitet werden. In der äußersten Schleife wird über 15 Körperelemente iteriert, und in der inneren Schleife über die Anzahl der Sektoren dieses Elementes.

In der unten abgebildeten Auflistung ist die Reihenfolge des gegenwärtig implementierten Körperaufbaus dargestellt. Eine aktuelle Liste kann vom Thermoregulations-Interface erstellt werden, wenn die Funktion `printSectorNames` in der `TheseusModel` Klasse aufgerufen wird.

```
id  (i,j) ELEMENT_/SECTOR__
--------------------------
 0) (0,0) HEAD____/FOREHEAD
 1) (0,1) HEAD____/HEAD____
 2) (1,0) FACE____/FACE____
 3) (2,0) NECK____/ANTERIOR
 4) (2,1) NECK____/POSTERIO
 5) (3,0) SHLD_LE_/SHLD_LE_
 6) (4,0) SHLD_RI_/SHLD_RI_
 7) (5,0) THORAX__/UP_ANT__
 8) (5,1) THORAX__/UP_POS__
 9) (5,2) THORAX__/UP_INF__
10) (5,3) THORAX__/LO_ANT__
11) (5,4) THORAX__/LO_POS__
12) (5,5) THORAX__/LO_INF__
13) (6,0) ABDOMEN_/ANTERIOR
14) (6,1) ABDOMEN_/POSTERIO
15) (6,2) ABDOMEN_/INFERIOR
16) (7,0) ARM_LE__/UP_ANT__
17) (7,1) ARM_LE__/UP_POS__
```

```
18) (7,2) ARM_LE__/UP_INF__
19) (7,3) ARM_LE__/LO_ANT__
20) (7,4) ARM_LE__/LO_POS__
21) (7,5) ARM_LE__/LO_INF__
22) (8,0) ARM_RI__/UP_ANT__
23) (8,1) ARM_RI__/UP_POS__
24) (8,2) ARM_RI__/UP_INF__
25) (8,3) ARM_RI__/LO_ANT__
26) (8,4) ARM_RI__/LO_POS__
27) (8,5) ARM_RI__/LO_INF__
28) (9,0) HAND_LE_/HANDBACK
29) (9,1) HAND_LE_/PALM____
30) (10,0) HAND_RI_/HANDBACK
31) (10,1) HAND_RI_/PALM____
32) (11,0) LEG_LE__/UP_ANT__
33) (11,1) LEG_LE__/UP_POS__
34) (11,2) LEG_LE__/UP_INF__
35) (11,3) LEG_LE__/LO_ANT__
36) (11,4) LEG_LE__/LO_POS__
37) (11,5) LEG_LE__/LO_INF__
38) (12,0) LEG_RI__/UP_ANT__
39) (12,1) LEG_RI__/UP_POS__
40) (12,2) LEG_RI__/UP_INF__
41) (12,3) LEG_RI__/LO_ANT__
42) (12,4) LEG_RI__/LO_POS__
43) (12,5) LEG_RI__/LO_INF__
44) (13,0) FOOT_LE_/INSTEP__
45) (13,1) FOOT_LE_/SOLE____
46) (14,0) FOOT_RI_/INSTEP__
47) (14,1) FOOT_RI_/SOLE____
Total Amount of Sectors : 48
```

B.2 Kommunikation mittels Sockets

Für die in Abschn. 5.2.4 erwähnte Kopplung wird eine Interprozesskommunikation benötigt, die von Sockets [22] übernommen werden kann. Die Header-Definition des SocketServers ist in Abs. B.4.4 dargestellt.

Soll diese Kopplungsmöglichkeit verwendet werden, so muss das Thermoregulations-Interface mit der Definition `-DSOCKETS` im Makefile generiert werden. Anschließend kann der SocketServer auf einem offenen Port gestartet werden. Hierzu steht die Funktion `startServer` zur Verfügung, die als Argument den zu verwendenden Port übergeben bekommt. Zugriff auf dem SocketServer wird über die Funktion `getSocketServer` im Thermoregulations-Interface bereitgestellt. Weitere Anpassungen für Kommunikation seitens des Interfaces sind nicht notwendig.

Der Server wartet anschließend auf eine Verbindung eines Clients an dem angegebenen Port. Sobald diese hergestellt ist, können Daten übertragen werden; solange keine Verbindung besteht wird auch kein Kommunikationsversuch unternommen.

Die Kommunikation ist komplett in einem eigenen Thread abgekapselt, um eine Reduktion der Geschwindigkeit der Berechnung zu vermeiden. In Analogie zu Abschn. 6.3 werden Warteschlangen für die Datenübergabe benutzt.

B.3 Header-Definitionen der Datenstruktur des Körperaufbaus

B.3.1 body_structure.h

```
1  #ifndef BODY_STRUCTURE_H
2  #define BODY_STRUCTURE_H
3
4  #include "clothing.h"
5  #include "current_data.h"
6  #include "initial_conditions.h"
7  #include "fiala_structure.h"
8
9  // forward declaration
10 class BodyElement;
11 class Body;
12
13 /*************************************************************************\
14  * Body Structure Definition                                             *
15  *                                                                       *
16  * with all necessary links to temperatures, initial conditions,         *
17  * clothing information, etc...                                          *
18  *                                                                       *
19 \*************************************************************************/
20
21 class BodyLayer
22 {
23 public:
24         // Constructor and Destructor
25         BodyLayer(BodyElement* father);
26         ~BodyLayer();
27
28         //..Interfaces..( Set / Create )..............................
29
30         // set ID of BodyLayer
31         void setId(const int& _lay_id);
32
33         // set Fiala Layer Information
34         void setFialaLayer(FialaLayer* _fl);
35
```

```
36
37          //..Interfaces..( Get )....................................
38
39          // get ID of BodyLayer
40          int getId();
41
42          // get Fiala Layer Information
43          FialaLayer* getFialaLayer();
44
45          // get father element containing this layer
46          BodyElement* getFatherElement();
47
48          //.........................................................
49
50  private:
51          // ID of this BodyLayer
52          int lay_id;
53
54          // Fiala Information for this BodyLayer
55          FialaLayer* fl;
56
57          // father element containing this layer
58          BodyElement* el;
59  };
60
61
62  //=============================================================================================
63
64
65  class BodySector
66  {
67  public:
68          // Constructor and Destructor
69          BodySector(BodyElement* father);
70          ~BodySector();
71
72          //..Interfaces..( Set / Create )..........................
73
74          // set LocalCurrentData
75          int setLocalCurrentData(LocalCurrentData* _lcd);
76
77          // set LocalInitialConditions
78          int setLocalInitialConditions(LocalInitialConditions* _lic);
79
80          // set Clothing Information
81          int setClothing(Clothing* _clo);
82
83          // set ID of this BodySector
84          int setId(const int& _sec_id);
85
86          // set Fiala Sector Information for this Sector
87          int setFialaSector(FialaSector* _fs);
88
89
90          //..Interfaces..( Get )....................................
```

```
91
92          // get Local Current Data attached to this sector
93          LocalCurrentData* getLocalCurrentData();
94
95          // get Local Initial Conditions
96          LocalInitialConditions* getLocalInitialConditions();
97
98          // get Clothing Information
99          Clothing* getClothing();
100
101         // get Sector ID
102         int getId();
103
104         // get Fiala Sector Information
105         FialaSector* getFialaSector();
106
107         // get father element containing this sector
108         BodyElement* getFatherElement();
109
110         //.....................................................
111
112         // compute skin area of this sector (if with_cloth = 1, the surface
113         // area will be computed. With with_cloth = 0, the skin area is computed.)
114         float computeSectorArea(const int& with_cloth = 0);
115
116         // compute the local eqivalent temperature (only if local current
117         // data available and in the case a fiala body is used)
118         int computeCurrentEquivalentTemp(float& t_eq_sector, float q_real, float t_sf);
119
120         // strip this sector from clothes (i.e. delete clothing information)
121         int stripClothes();
122
123         //.....................................................
124
125
126 private:
127         // data attached to this sector
128         LocalCurrentData* lcd;
129
130         // initial conditions attached to this element (if available)
131         LocalInitialConditions* lic;
132
133         // clothing information attached to this sector (if available)
134         Clothing* clo;
135
136         // ID of this BodySector
137         int sec_id;
138
139         // Fiala Information for this BodySector
140         FialaSector* fs;
141
142         // father element containing this sector
143         BodyElement* el;
144 };
145
```

```
146
147 //=============================================================================================
148
149
150 class BodyElement
151 {
152 public:
153         // Constructor and Destructor
154         BodyElement(Body* father);
155         ~BodyElement();
156
157         //..Interfaces..( Set / Create )..............................
158
159         // create an array of BodySectors
160         int createArrayOfBodySectors(const int& _n_sec);
161
162         // copy BodyElement* to array
163         int setBodySector(const int& sec_id, BodySector* _bs);
164
165         // create array of BodyLayers
166         int createArrayOfBodyLayers(const int& _n_lay);
167
168         // copy BodyLayer* to bl array
169         int setBodyLayer(const int& lay_id, BodyLayer* _bl);
170
171         // set BodyElement ID
172         void setId(const int& _be_id);
173
174         // set Fiala Element Information for this BodyElement
175         int setFialaElement(FialaElement* _fe);
176
177
178         //..Interfaces..( Get )......................................
179
180         // get amount of BodySectors in this Element
181         int getNSectors();
182
183         // get BodySector sec_id from array (sec_id starts from 1 !)
184         BodySector* getBodySector(const int& sec_id);
185
186         // get amount of BodyLayers in this Sector
187         int getNLayers();
188
189         // get BodyLayer lay_id from array (lay_id starts from 1 !)
190         BodyLayer* getBodyLayer(const int& lay_id);
191
192         // get BodyElement ID
193         int getId();
194
195         // get Fiala Element Information for this BodyElement
196         FialaElement* getFialaElement();
197
198         // get Body containing the element
199         Body* getBody();
200
```

```
201         //.....................................................
202
203
204 private:
205         // a body element contains n_sec BodySectors
206         int n_sec;
207         BodySector** bs;
208
209         // a body element contains n_lay BodyLayers
210         int n_lay;
211         BodyLayer** bl;
212
213         // ID of this BodyElement
214         int be_id;
215
216         // Fiala Information for this BodyElement
217         FialaElement* fe;
218
219         // Body containing this BodyElement
220         Body* bd;
221 };
222
223
224 //=============================================================================================
225
226
227 class Body
228 {
229 public:
230         // Constructor and Destructor
231         Body();
232         ~Body();
233
234         //..Interfaces..( Set / Create )..............................
235
236         // create an array of BodyElements
237         int createArrayOfBodyElements(const int& _n_be);
238
239         // copy BodyElement* to array
240         int setBodyElement(const int& be_id, BodyElement* _be);
241
242         // set Body ID
243         int setId(const int& _body_id);
244
245         // set Global Current Data attached to the whole Body
246         int setGlobalCurrentData(GlobalCurrentData* _gcd);
247
248         // set Global Initial Conditions for the whole Body
249         int setGlobalInitialConditions(GlobalInitialConditions* _gic);
250
251         // set Fiala Information for this Body
252         int setFialaBody(FialaBody* _fb);
253
254
255         //..Interfaces..( Get )......................................
```

```
256
257         // get Amount of BodyElements
258         int getNElements();
259
260         // get BodyElement be_id from array (be_id starts from 1 !)
261         BodyElement* getBodyElement(const int& be_id);
262
263         // get Body ID
264         int getId();
265
266         // get Global Current Data attached to the whole Body
267         GlobalCurrentData* getGlobalCurrentData();
268
269         // get Global Initial Conditions for the whole Body
270         GlobalInitialConditions* getGlobalInitialConditions();
271
272         // get Fiala Information for this Body
273         FialaBody* getFialaBody();
274
275
276         //..Other Member Functions (public).....................
277
278         // compute skin global area of manikin (without clothing)
279         float computeGlobalSkinArea();
280
281         // compute global area of manikin (with current clothing)
282         float computeGlobalSurfaceArea();
283
284         // compute global icl value
285         float computeGlobalICL();
286
287         // check if there is any clothing information at all attached to the body
288         bool clothingExists();
289
290         // strip all clothes (i.e. delete all clothing information)
291         int stripBodyFromClothes();
292
293         //.......................................................
294
295
296 private:
297         // a body is composed of n_be BodyElements
298         int n_be;
299         BodyElement** be;
300
301         // ID of this body
302         int body_id;
303
304         // Data attached globally to the whole Body
305         GlobalCurrentData* gcd;
306
307         // Initial Conditions for the whole Body
308         GlobalInitialConditions* gic;
309
310         // Fiala Information for this Body
```

```
311         FialaBody* fb;
312 };
313
314 #endif //BODY_STRUCTURE_H
```

B.3.2 fiala_structure.h

```
1  #ifndef BODY_FIALA_STRUCTURE_H
2  #define BODY_FIALA_STRUCTURE_H
3
4  /*****************************************************************************\
5   * Fiala Information Necessary for Computation with Theseus-FE               *
6   *                                                                           *
7  \*****************************************************************************/
8
9  // for further information about the meaning of the variables see
10 // Theseus-FE Keyword Manual p. 34ff
11
12
13 // Layer Information (MANFLAY)
14 typedef struct {
15         const char* name;
16         int disc;
17         float r;
18         float k;
19         float rho;
20         float cp;
21         const char* w_bl; // normally float. Only here const char* because of CARDOUT Problem.
22         float q_m;
23         float a_rsp;
24 } FialaLayer;
25
26
27 // Sector Information (MANFSEC)
28 typedef struct {
29         const char* name;
30         float phi;
31         float vf1; // viewfactor sitting
32         float vf2; // viewfactor standing
33         float e_sf;
34         float l;   // -1 if undefined
35 } FialaSector;
36
37
38 // Element Information (MANFBEL)
39 typedef struct {
40         const char* name;
41         const char* type;
42         float l;
43         float h_x;
44         float a_sk;
45         float a_sw;
46         float a_dl;
```

```
47          float a_cs;
48          float a_sh;
49          float a_m_w1;
50          float a_m_w2;
51          float a_nat;
52          float a_frc;
53          float a_mix;
54  } FialaElement;
55
56
57  // Body Information (MANIKIN)
58  typedef struct {
59          const char* system;     // "ACTIVE" or "PASSIVE"
60          const char* position;   // "STND" (standing) or "SED" (sitting)
61  } FialaBody;
62
63  #endif // BODY_FIALA_STRUCTURE_H
```

B.3.3 clothing.h

```
1   #ifndef BODY_CLOTHING_H
2   #define BODY_CLOTHING_H
3
4   /*****************************************************************************\
5    * Clothing Information                                                      *
6    *                                                                           *
7   \*****************************************************************************/
8
9   // for further information see Theseus-FE Keywordmanual (Keyword MANFSCL)
10
11  typedef struct {
12          float thickn;   // thickness of the clothing layer
13          float sicl;
14          float siicl;
15          float nsm;      // non structural mass [kg/m^2]
16          float cp;       // heat capacity [J/kg/C]
17  } Clothing;
18
19  #endif //BODY_CLOTHING_H
```

B.3.4 initial_conditions.h

```
1   #ifndef BODY_INITIAL_CONDITIONS_H
2   #define BODY_INITIAL_CONDITIONS_H
3
4   /*****************************************************************************\
5    * Initial Conditions Definition                                             *
6    *                                                                           *
7   \*****************************************************************************/
8
```

```
9
10 // Global Initial Conditions
11 typedef struct {
12         float t_w;
13         float e_w;
14         float t_a;
15         float v_a;
16         float rh;
17         float act;
18         float global_icl;
19 } GlobalInitialConditions;
20
21
22 // Local Initial Conditions
23 typedef struct {
24         float t_w;
25         float t_a;
26         float v_a;
27         float rh;
28 } LocalInitialConditions;
29
30 #endif //BODY_INITIAL_CONDITIONS_H
```

B.3.5 current_data.h

```
1  #ifndef BODY_CURRENT_DATA_H
2  #define BODY_CURRENT_DATA_H
3
4  #include <string>
5
6  /*************************************************************************\
7   * Current Data Attached to the Body Structure                           *
8   *                                                                       *
9  \*************************************************************************/
10
11 // ---------- Data attached globally to the whole body ------------------------
12 class GlobalCurrentData {
13 public:
14     // get global current data of id=0 (t_hy), id=1 (t_sk_m), etc...
15     float getGCDVal(const int& id);
16
17     // set global current data of id=0 (t_hy), id=1 (t_sk_m), etc...
18     int setGCDVal(const int& id, const float& val);
19
20     // enumeration of GCD Value types
21     enum { TEMP = 1, MET = 2, PMV = 3, PPD = 4};
22
23     // get amount of GCD Values
24     static int getAmountGCDVal();
25
26     // get the name of the GCD Val corresponding to ID id
27     static std::string getGCDName(const int& id, const bool& long_form = false);
28
```

```
29      // get the type of the GCD Val corresponding to ID id
30      // (all elements of the same type can be selected simultaniously)
31      static int getGCDType(const int& id);
32
33
34      // members (actual data)
35          float time;     // time has no id, as it is accessed via -> time
36          float t_hy;     // id = 0
37          float t_sk_m;   // id = 1
38          float t_sf_m;   // id = 2
39          float t_eq_m;   // id = 3
40          float q_m;      // id = 4
41          float pmv;      // id = 5
42          float ppd;      // id = 6
43          float dts;      // id = 7
44          float ts;       // id = 8
45  };
46
47
48  // ---------- Data attached locally to every body sector -----------------------
49  class LocalCurrentData {
50  public:
51      // get local current data of id 0 (t_sf), id 1 (t_sk) , etc
52      float getLCDVal(const int& id);
53
54      // get the amount of LCD values
55      static int getAmountLCDVal();
56
57      // get the name of the LCD Val corresponding to ID id
58      static std::string getLCDName(const int& id);
59
60
61      // members (actual data)
62          float t_sf;         // id 0
63          float t_sk;         // id 1
64          float t_eq;         // id 2
65          float lmv;          // id 3
66  };
67
68  #endif //BODY_CURRENT_DATA_H
```

B.4 Header-Definitionen des Thermoregulations-Interfaces

B.4.1 global_model.h

```
1  #ifndef GLOBAL_MODEL_H
2  #define GLOBAL_MODEL_H
3
```

```
4  #include "body_structure.h"
5
6  #ifdef SOCKETS
7          // forward declaration of socket server class
8          class SocketServer;
9  #endif
10
11 /*************************************************************************\
12  * Global Model for Thermoregulation Model                               *
13  *                                                                       *
14  *                                                                       *
15  * Abstract Class for building up Models etc...                          *
16  *                                                                       *
17  *                                                                       *
18 \*************************************************************************/
19
20
21 class ThermoModel
22 {
23 public:
24         // Constructor and Destructor
25         ThermoModel();
26         virtual ~ThermoModel();
27
28         //..Interfaces..( Set )....................................
29
30         // set the initial conditions for the body
31         int setInitialConditions(const float& t_w, const float& e_w, const float& t_a,
32                                  const float& v_a, const float& rh, const float& act,
33                                  const float& global_icl);
34
35
36         //..Interfaces..( Get )....................................
37
38         // get thermoBody*
39         Body* getThermoBody();
40
41         // get amount of local current data attached currently to the body structure
42         int getAmountLocalCurrentData();
43
44         // get local current data
45         int getCurrentLocalData(const int& data_type, const int& n_data, float* array_data);
46
47         // get all local current data (in matrix)
48         int getAllCurrentLocalData(int& n_type, const int& n_data, float** matrix_data);
49
50         // get global current data
51         GlobalCurrentData* getCurrentGlobalData();
52
53         // get the current time from the model
54         float getCurrentTime();
55
56 #ifdef SOCKETS
57         // get pointer to socket server class
58         SocketServer* getSocketServer();
```

```
59  #endif
60
61          // check if sockets are enabled
62          bool socketsEnabled();
63
64
65          //..Interacting Member Functions.............................
66
67          // compute results of thermoBody at time (t+ dt)
68          virtual int computeUpdate(const float& dt) = 0;
69
70
71          //...........................................................
72
73  protected:
74          // Body Model Holding all Data
75          Body* thermoBody;
76
77          // pure virtual function creating thermoBody with all necessary Information
78          virtual int createThermoBody() = 0;
79
80  #ifdef SOCKETS
81          // Socket Server to communicate with other applications
82          SocketServer* sserver;
83
84          // declare socket server as friend class (has access to private members of ThermoBody class)
85          friend class SocketServer;
86  #endif
87
88  };
89
90  #endif //GLOBAL_MODEL_H
```

B.4.2 theseus_configuration.h

```
1   #ifndef THESEUS_MODEL_CONFIGURATION_H
2   #define THESEUS_MODEL_CONFIGURATION_H
3
4   /*************************************************************************\
5    * Theseus-FE Model Configuration Utilities                              *
6    *                                                                       *
7   \*************************************************************************/
8
9   // TFESolver for configuration of the Theseus Solver. See also Keywordmanual (Keyword SOL)
10
11  class TFESolverData {
12  public:
13          // constructor and destructor
14          TFESolverData();
15          ~TFESolverData();
16
17          //..Public Members.....................................
18
```

```
19          // Enumerations for Solver Type and Timestep Type
20          enum tfeSolverType { Trans2 = 0, Steady = 1 };
21          enum tfeTimeStepType {Adaptive = 0, Fixed = 1, Empty = 2 };
22
23
24          int solverType;         // solver type (0 for Trans2 and 1 for Steady)
25          int timeStepType;       // time step type (0 for adaptive, 1 for fixed and 2 if empty)
26          float dt0;              // initial timestep for transient and steady state solver
27          float tend;             // ending time of computation
28          float adptol;           // adaptive tolerance
29          float mxdt;             // maximum time step
30          float sscond;           // steady state condition
31          float crit1;            // maximum relative energy residual
32          int   minitr;           // minimum numbers of nonlinear iterations
33          int   maxitr;           // maximum numbers of nonlinear iterations
34          long  maxstp;           // maximum numbers of timesteps
35          float relax;            // relaxation parameter
36          const char* pred;       // order of explicit predicting temperatures
37          float divisor;          // divisor for the linear iterations
38          float crit2;            // maximum temperature correction
39          float mindt;            // minimum time step
40          int   divgitr;          // number of nonlinear iterations to check 1st der.
41          float minq;             // minimum average heat flux
42
43          float tinitfe;          // write initial fe temperature (only required for setpoint generation) (-1.0
        if not defined!)
44
45  };
46
47
48  // TFEPostProc for configuration of postprocessing steps and all kind of writeouts
49
50  class TFEPostProcData {
51  public:
52          // constructor and destructor
53          TFEPostProcData();
54          ~TFEPostProcData();
55
56          // enumeration for methods of postprocessing
57          enum tfePostFrqType { Interval = 0, End = 1 };
58
59          int postFrqType;        // type of write out (in intervals (= 0) or at the end (= 1) )
60
61          int n_interv;           // if intervals are chosen, this gives the amount of entries in the tables
62          float* t_int;           // timetable containing the proceding time where the writeout changes
63          float* dt;              // table containing the calls every dt
64
65          bool write_restart_file;// writing restart file for continued computation
66          bool manikin_debug;     // manikin in debug mode (MANFDBG Keyword!)
67
68          bool compute_pmv;       // should the PMV be computed?
69
70  };
71
72  #endif // THESEUS_MODEL_CONFIGURATION_H
```

B.4.3 theseus_interface.h

```
#ifndef THESEUS_INTERFACE_H
#define THESEUS_INTERFACE_H

#include "theseus_configuration.h"
#include "global_model.h"
#include "body_structure.h"

#include <string>
using std::string;

/*****************************************************************************\
 * Theseus-FE Interface for Thermoregulation Model                           *
 *                                                                           *
 * Library containing initial settings for Fiala Values etc...               *
 *                                                                           *
\*****************************************************************************/

class TheseusModel : public ThermoModel
{
public:
        // Constructor and Destructor
        TheseusModel();
        ~TheseusModel();

        //..Interfaces..( Set / Create )..............................

        // Default Clothing Values can be loaded with this functions to the Model
        int setClothingDefaultEmpty();
        int setClothingDefaultSummer();
        int setClothingDefaultWinter();

        // set Basename for SetPoint File (Thermal Neutrality) to use for computation
        int setSetPointBaseName(const string& _setpointBaseName);

        // set Basename for all tfe files
        int setTfeBaseName(const string& _tfeBaseName);

        // set global Path Name for Theseus-FE File Storage etc.
        int setGlobalPath(const char* _globalPath);


        //..Interfaces..( Get )......................................

        // get current clothing thickness and icl values
        int getCurrentClothThickness(const int& n_d, float* array_d);
        int getCurrentClothSIcl(const int& n_d, float* array_d);

        // print names of bodyelements and sectors on screen (for the right indexing)
        void printSectorNames();


        //..Interacting Member Functions.............................

```

```
54          // switch the flag deciding if PMV values shall be computed
55          int switchPMVflag(const bool& enable);
56
57          // write debug information to the .tfe file
58          int verboseTfe(const bool& flag = true);
59
60          // compute results of thermoBody at time (t + dt)
61          int computeUpdate(const float& dt);
62
63          //.....................................................
64
65
66  private:
67          // BaseNames of Files necessary for Theseus-FE
68          string tfeBaseName;
69          string currentTfeName;
70          string currentRestartName;
71          string setpointBaseName;
72          string globalPath;
73
74      // verbose flag
75      bool verbose;
76
77          // counter over all computations (for name generation)
78          int ct_compute;
79
80          // Solver configuration data
81          TFESolverData* solver;
82
83          // Postprocessing configuration data
84          TFEPostProcData* postproc;
85
86
87          //..Private Member Functions.............................
88
89          // create TheseusBody with all default values from Fiala Model
90          int createThermoBody();
91
92          // write TheseusModel to .tfe-File
93          int writeTheseusModel(const char* tfe_file, const bool& debug = true);
94
95          // parse Theseus-FE Result File (.hdf) and save results in Body Structure
96          int parseTheseusResults(const char* hdf_file);
97
98          // call for starting computation with Theseus-FE
99          int computeTheseus(const std::string& currentTfeBase, const std::string& currentRestartBase, const std
     ::string& setPointBase);
100
101         // compute Thermal Neutrality
102         int computeThermalNeutrality(string setpointBaseName);
103
104         // backup Body Clothing in a new body structure
105         Body* backupForNeutralComputation();
106
107         // restore Clothing Information from a storing body to the current Body
```

```
108         int restoreFromNeutralComputation(Body* storing_temp_body);
109
110         // set clothing values to non-isolation and deleting local initial conditions(for generating thermal
        neutrality values)
111         int prepareForNeutrality();
112
113         // check passive system information after computing set point temperatures
114         int checkNeutralData(const char* report_name);
115 };
116
117 #endif //THESEUS_INTERFACE_H
```

B.4.4 socket_server.h

```
1  #ifndef SOCKET_SERVER_H
2  #define SOCKET_SERVER_H
3
4  /*************************************************************************\
5   *                                                                       *
6   * This two classes used for communication are based on the wxWidgets-Library *
7   * If you want to use this classes, the wxWidgets librarys must be installed  *
8   * and ready to use. Otherwise the code won't compile.                   *
9   *                                                                       *
10  * The communication is done via blocked sending (and receiving) commands.    *
11  * The whole communication is encapsulated in threads so that the        *
12  * blocking call does not interfere with the computation itself.         *
13  *                                                                       *
14  * If no communication ability is desired, just remove the -DSOCKETS     *
15  * definition in the Makefile. In this case no calls to a wxWidget-Library    *
16  * are raised.                                                           *
17  *                                                                       *
18 \*************************************************************************/
19
20
21 // include this class only if sockets shall be used (defined in the Makefile)
22 #ifdef SOCKETS
23
24 #include "wx/wx.h"
25 #include "wx/socket.h"
26 #include "wx/thread.h"
27 #include <queue>
28
29 // forward declaration
30 class ThermoModel;
31 class SocketServer;
32
33 // struct for temporary storing ServerData
34 typedef struct {
35         int    fsize;
36         float* field;
37 } ServerData;
38
39 // ----------------------------------------------------------------------
```

```
40 // Socket Server Thread Class transfering data
41 // ----------------------------------------------------------------------------
42 class SocketServerThread : public wxThread
43 {
44 public:
45         // Constructor and Destructor
46         SocketServerThread(SocketServer* _sserver);
47         ~SocketServerThread();
48 
49         // thread execution starts here
50         virtual void *Entry();
51 
52         // called when the thread exits - whether it terminates normally or is
53         // stopped with Delete() (but not when it is Kill()ed!)
54         virtual void OnExit();
55 
56 private:
57         SocketServer* sserver;
58 
59 };
60 
61 // ----------------------------------------------------------------------------
62 // Socket Server Class handling all Communications for external visualization
63 // ----------------------------------------------------------------------------
64 class SocketServer : public wxEvtHandler
65 {
66 public:
67         // Constructor and Destructor
68         SocketServer(ThermoModel* _thermoModel);
69         ~SocketServer();
70 
71         //--- Server Issues ------------------------------------------------------
72 
73         // start the Socket Server (to listen to on port 'port' for communication)
74         int startServer(const int& port = 30001);
75 
76         // stop the Socket Server
77         int stopServer();
78 
79         //--- Send Data ----------------------------------------------------------
80 
81         // send current data over to a connected client socket (only if connected)
82         int sendDataToClient();
83 
84 
85         //--- Data Conversion ----------------------------------------------------
86 
87         // convert current tmodel data to float field
88         float* convertCurrentTModelToField(int& size);
89 
90         // convert field to tmodel data (all data needs to be initialized!!!)
91         static int convertFieldToData(float* input_field, int& input_size, GlobalCurrentData* gcd,
92                                       GlobalInitialConditions* gic, const int& size_local,
93                                       float* local, int& data_type, float* clothing);
94 
```

```
95          //--- Queue Access -- ( protected by Mutex ) ------------------------------------
96
97          // check if queue is empty
98          bool getSendingQueueState();
99
100         // get data from queue (and remove it from there)
101         ServerData getServerDataFromQueue();
102
103         // push ServerData to queue
104         int pushServerDataToQueue(const ServerData& sdata);
105
106
107
108 private:
109
110         // ThermoModel Instance containing this socket (for data access)
111         ThermoModel* tmodel;
112
113         // wxSocket Server
114         wxSocketServer* m_server;
115
116         // wxSocket Base (to where the server is connected)
117         wxSocketBase *sock;
118
119         // flag if server is connected to client
120         bool is_connected;
121
122         // flag if the server is up and running
123         bool server_up;
124
125         // which data type for local data shall be transmitted
126         int local_data_type;
127
128         // SocketServerTransferThread
129         SocketServerThread* ssthread;
130         bool thread_state;
131
132         // queue for transfering server data to client
133         std::queue<ServerData> severQueue;
134
135         // Mutex for protecting queue
136         wxMutex mut_squeue;
137         wxMutex mut_transfer;
138
139         // --- private Member functions
140         void OnServerEvent(wxSocketEvent& event);
141         void OnSocketEvent(wxSocketEvent& event);
142
143         // Event handler declarations
144         DECLARE_EVENT_TABLE()
145
146         // IDs for the event handler etc
147         enum
148         {
149                 // id for sockets
```

```
150                 SERVER_ID = 100,
151                 SOCKET_ID
152         };
153
154         friend class SocketServerThread;
155 };
156
157
158 #endif //SOCKETS
159
160 #endif //SOCKET_SERVER_H
```

Anhang C

Das Experiment von Raven (1970)

P. R. Raven [25] hat 1970 ein Experiment mit elf, nur mit Shorts bekleideten, Versuchspersonen durchgeführt. Sie wurden bei 28°C für 25 Minuten akklimatisiert, anschließend in den nächsten 5 Minuten auf 5°C Umgebungstemperatur gebracht, und für die nächsten 2 Stunden dort belassen. Während der ganzen Zeit wurden Messungen durchgeführt.

Tabelle C.1: Randbedingungen für Temperatur und rel. Luftfeuchte

t [s]	t [min]	T_a, T_w [°C]	rh [%]
0	0	28.0	45.0
1500	25	28.0	45.0
1800	30	5.0	70.0
9000	150	5.0	70.0

Weitere Randbedingungen sind $\varepsilon_w = 0.93$, $v_a = 0.1$ m/s und $act_{bas} = 0.8$ met.

Fiala hat gezeigt, dass die Ergebnisse seiner Arbeit gut mit den Versuchsergebnissen übereinstimmen. Weiterhin wurde im Rahmen der Validierung von THESEUS-FE [30], sowie mit der Virtual Climate Chamber das Experiment nachgerechnet. Die Ergebnisse sind in Abb. C.1 und Abb. C.2 zu sehen.

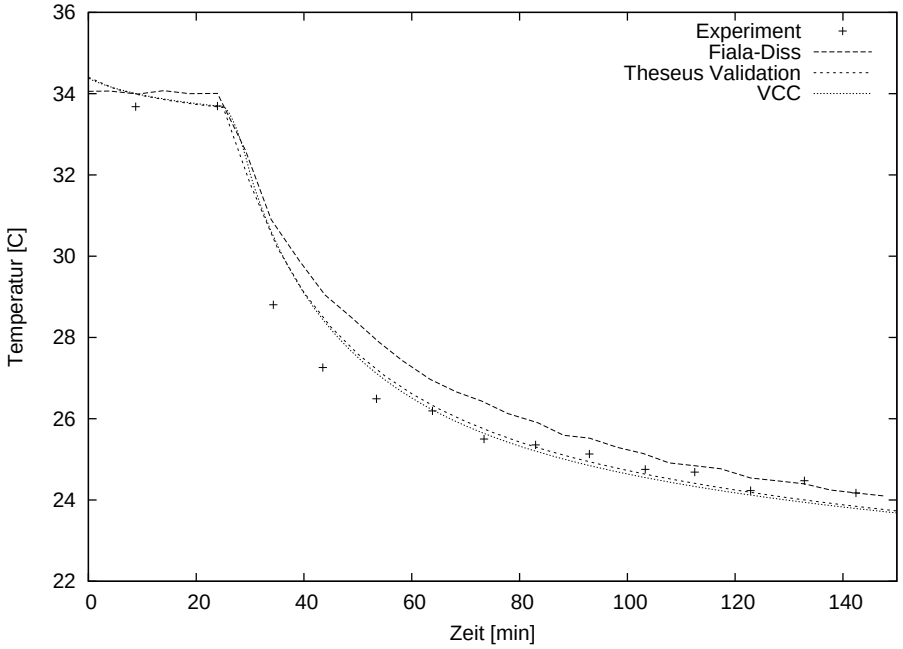


Abbildung C.1: Raven Experiment: Mittlere Hauttemperatur $T_{sk,m}$

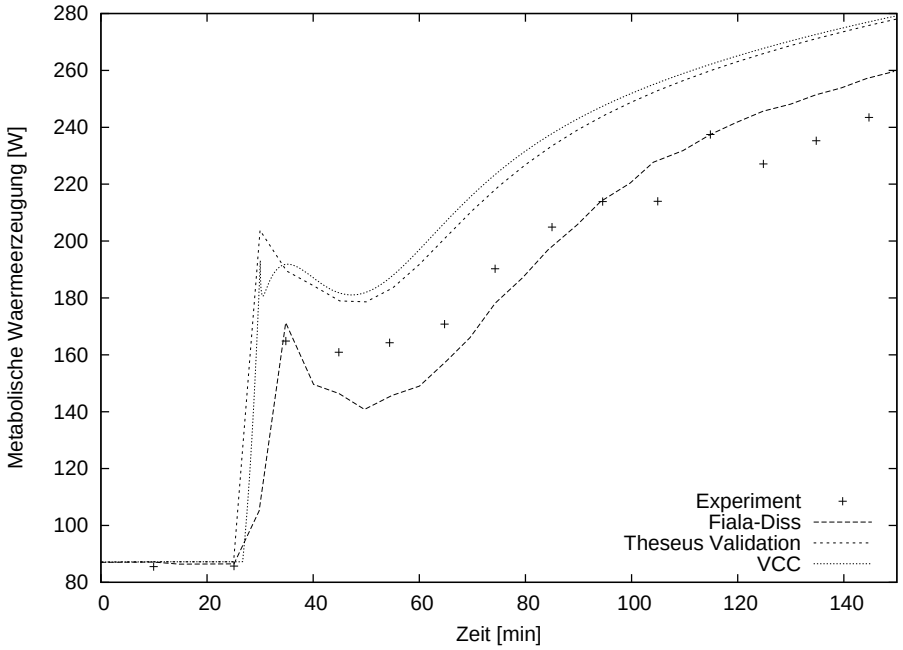


Abbildung C.2: Raven Experiment: Metabolische Wärmeerzeugung Q_m

Literaturverzeichnis

[1] *Ansys CFX*. http://www.cfx-germany.com.

[2] *Blender - free open source 3D content creation suite*. http://www.blender.org.

[3] *Fraunhofer-Institut für Bauphysik IBP Holzkirchen*. http://www.hoki.ibp.fhg.de.

[4] *GNU Compiler*. http://gcc.gnu.org.

[5] *GnuPlot Homepage*. http://www.gnuplot.info.

[6] *H-Anim - Humanoid Animation Working Group*. http://www.h-anim.org.

[7] *The HDF Group*. http://hdfgroup.com/HDF5.

[8] *LRZ München*. http://www.lrz-muenchen.de/services/peripherie/vr/ComputationalSteering.html.

[9] *Macrovision FLEXlm Server*. http://www.macrovision.com.

[10] *minGW - Minimalist GNU for Windows*. http://www.mingw.org.

[11] *NASA MSIS - Man-Systems Integration Standards*. http://msis.jsc.nasa.gov.

[12] *THESEUS-FE*. http://www.theseus-fe.com.

[13] *wxDev-C++*. http://wxdsgn.sourceforge.net.

[14] *wxWidgets Cross-Platform GUI Library*. http://wxwidgets.org.

[15] *ASHRAE Handbook Fundamentals*, 2005. American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc.

[16] J.A. Clarke. *Energy simulation in building design*. Butterworth-Heinemann, 2nd edition, 2001.

[17] P. O. Fanger. Calculation of thermal comfort: Introduction of a basic comfort equation. *ASHRAE Transactions*, 73(2):III4.1–III4.20, 1967.

[18] D. Fiala. *Dynamic Simulation of Human Heat Transfer and Thermal Comfort*. PhD Thesis, De Montfort University, Leicester, 1998.

[19] D. Fiala, K. J. Lomas, and M. Stohrer. A computer model of human thermoregulation for a wide range of environmental conditions: the passive system. *J Appl Physiol*, 87(5):1957–1972, Nov 1999.

[20] D. Fiala, K. J. Lomas, and M. Stohrer. Computer prediction of human thermoregulatory and temperature responses to a wide range of environmental conditions. *Int J Biometeorol*, 45(3):143–159, Sep 2001.

[21] Fraunhofer-Institut für Bauphysik IBP. *ComfSim - Lokale Komfortanalyse in Innenräumen - Zusammenhang lokale Äquivalenttemperaturen und Local Mean Vote - Interner Bericht*, Feb. 2008.

[22] J. Smart, K. Hock, S. Csomor. *Cross-Platform GUI Programming with wxWidgets*. Prentice Hall, 2005.

[23] R.-P. Mundani. *Parallel Computing, Lecture Notes, Technische Universität München*, 2007.

[24] P. Deetjen, C. Alzheimer. *Physiologie*. Elsevier, Urban & Fischer, 4 edition, 2005.

[25] P. R. Raven, S. M. Horvath. Variability of physiological parameters of unacclimatized males during a two-hour cold stress of 5 degrees C. *Int J Biometeorol*, 14(3):309–320, Sep 1970.

[26] H. H. Pennes. Analysis of tissue and arterial blood temperatures in the resting human forearm. 1948. *J Appl Physiol*, 85(1):5–34, Jul 1998.

[27] P+Z Engineering GmbH, Munich. *THESEUS-FE Keyword Manual (Version 2.1)*, Oct. 2007.

[28] P+Z Engineering GmbH, Munich. *THESEUS-FE Theory Manual (Version 2.1)*, Oct. 2007.

[29] P+Z Engineering GmbH, Munich. *THESEUS-FE User's Manual (Version 2.1)*, Oct. 2007.

[30] P+Z Engineering GmbH, Munich. *THESEUS-FE Validations (Version 2.1)*, Oct. 2007.

[31] C. van Treeck. *Building Energy Simulation, Lecture Notes, Technische Universität München*, Nov. 2007.

[32] C. van Treeck, P. Wenisch, A. Borrmann, M. Pfaffinger, N. Cenic, and E. Rank. Utilizing high performance supercomputing facilities for interactive thermal comfort assessment. In *Proc. 10th IBPSA Conf. Building Simulation*, Bejing, China, September 3-6 2007.

[33] C. van Treeck, P. Wenisch, A. Borrmann, M. Pfaffinger, O. Wenisch, and E. Rank. ComfSim - Interaktive Simulation des thermischen Komforts in Innenräumen auf Höchstleistungsrechnern. *Bauphysik*, 29(1):2–7, 2007. DOI: 10.1002/bapi.200710000.

[34] P. Wenisch. *Computational Steering of CFD Simulations on Teraflop-Supercomputers*. PhD thesis, Technische Universität München, 2008.

[35] D. Yang. *C++ and Objectoriented Numeric Computing for Scientists and Engineers*. Spinger, 2001.

[36] Z. Yue. *A Computational Human Model for Thermal Comfort Simulation*. Master Thesis, Technische Universität München, Dec. 2007.