Institut für Hydrologie der Albert-Ludwigs-Universität Freiburg i. Br.

Joachim Herrmann

OGRUT GIS-Integration von FEFLOW in das Freiburger Wasserqualitätsinformations-system WAQIS

Diplomarbeit unter der Leitung von Prof. Dr. Ch. Leibundgut
Freiburg i. Br., im August 2000

Institut für Hydrologie der Albert-Ludwigs-Universität Freiburg i. Br.

Joachim Herrmann

OGRUT GIS-Integration von FEFLOW in das Freiburger Wasserqualitätsinformations-system WAQIS

Referent: Prof. Dr. Ch. Leibundgut
Koreferent: Prof. Dr. S. Demuth

Diplomarbeit unter der Leitung von Prof. Dr. Ch. Leibundgut
Freiburg i. Br., im August 2000

Danksagung

An erster Stelle gebührt mein herzlichster Dank meinem Betreuer Stephen Schrempp, dessen grosse Geduld und Hilfsbereitschaft sehr viel zur angenehmen Arbeitsatmosphäre während der Diplomarbeit beigetragen haben.

Krischan Eberle danke ich für seine Nachsicht mit meinen anfänglich geringen Kenntnissen in der Programmierung mit Magik und für seine Bereitschaft, zur kontinuierlichen Verbesserung dieser Kenntnisse beizutragen.

Herzlich danke ich meiner Freundin Andrea Rhyn für ihre moralische Unterstützung und der aufopferungsvollen Bereitschaft zum Korrekturlesen der Arbeit.

Schliesslich bedanke ich mich besonders bei meinen Eltern Peter und Liselotte Herrmann, die mich während des Studiums stets unterstützt haben.

Freiburg, den 10.08.2000 Joachim Herrmann

Inhaltsverzeichnis

Verzeichnis der Abbildungen im Text

Verzeichnis der Tabellen im Text

Verzeichnis der Abbildungen im Anhang

Verzeichnis der Tabellen im Anhang

Liste der verwendeten Symbole und Abkürzungen

A	Fläche (in m^2)
a	Beschleunigung ($m\ s^{-2}$)
ACP	alien co-processor
BOMET	Bodenkundlich-meteorologisches Teilinformationssystem
C	Expansions- und Kontraktionsverlust-Koeffizient
CASE	Computer-Aided Software Engineering
DBMS	Datenbank-Managementsystem
DGM	digitales Geländemodell
div	Divergenz
err	Fehler
ET_a	Aktuelle Evapotranspiration (in $mm\ d^{-1}$)
ET_P	Potentielle Evapotranspiration (in $mm\ d^{-1}$)
ET_R	Reale Evapotranspiration (in $mm\ d^{-1}$)
f	Infiltrationsrate, oder Funktion
f_0	Anfangsinfiltrationsrate
f_∞	Endinifiltrationsrate
F_a	beschleunigende Kraft (in N)
FEFLOW	Finite Elemente Flow Simulation Programm
FEW	Freiburger Energie- und Wasserversorgungs-AG
F_f	Reibungskraft (in N)
F_G	Schwerkraft (in N)
FK	Feldkapazität (in Vol-% oder *mm*/Bezugstiefe)
F_p	Druckkraft (in N)
g	Gravitationskonstante (in $m\ s^{-2}$)
GIS	Geographisches Informationssystem
grad	Gradient (∇)
GUI	graphical user interface
GWNB	Grundwasserneubildung (in *mm*)
G_x	Gewichtskraft, Komponente in x-Richtung (in N)
h	Mächtigkeit (in m)
H	Gesamtenergiehöhe (in m)
h_e	Energiehöhenverlust
HEC-RAS	Hydrologic Engineering Center's River Analysis System
IHF	Institut für Hydrologie, Albert-Ludwigs-Universität Freiburg
ISO	International Standardization Organization
k	ungesättigte hydraulische Leitfähigkeit (in $cm\ d^{-1}$)
K	Transportleistung
k_f	gesättigte hydraulische Leitfähigkeit (in $cm\ d^{-1}$)
k_{St}	STRICKLER-Rauhigkeitsbeiwert
L	Abstand (in m)
LIWIS	Lahmeyer International Wasserinformationssystem
m	Masse (in kg)
nFK	nutzbare Feldkapazität (in Vol.-% oder *mm*/Bezugstiefe)
NN	Normal-Null
Nq	Niedrigwasserabflussspende (in $l\ s^{-1}km^{-1}$)

OGRUT	Oberflächen- und Grundwasser Teilinformationssystem
OOP	object-oriented programming
P	Niederschlag (in $mm\ d^{-1}$)
p	Druck (in *hPa*)
PWP	permanenter Welkepunkt (in Vol-% oder *mm*/Bezugstiefe)
Q	Abfluss (in $m^3\ s^{-1}$)
R	hydraulischer Radius (in *m*)
RDBMS	Relationales Datenbank-Managementsystem
RIS	raumbezogenes Informationssystem
S	Sorptivität (Wasseraufnahmefähigkeit des Bodens)
S_0	Längsgefälle
S_f	Energieliniengefälle
SF	spezifische Kraft
SQL	structured query language
t	Zeitvariable
TK	Topographische Karte
U	benetzter Umfang (in *m*)
UWSP	Ungleichförmiges Wasserspiegelprogramm
V	Volumen (in m^3)
v	Geschwindigkeit (in $m\ s^{-1}$)
v_f	Filtergeschwindigkeit (in $m\ s^{-1}$)
W_a	Kinetische Energie (in *J*)
WAQIS	Wasserqualitätsinformationssystem
W_G	Lageenergie (in *J*)
W_{gesamt}	Gesamtenergie (in *J*)
W_p	Druckenergie (in *J*)
WS	Wasserstand (in *m*)
z	Vertikalabstand, Höhe (in *m*)
α	Geschwindigkeitskoeffizient
β	Impulskoeffizient
ρ	Dichte (in $kg\ m^{-3}$)
γ	Wichte (in $kg\ m^{-2}s^{-2}$)
Δ	Differenz, $\Delta x = x_1 - x_2$
ε	Fehlertoleranz für Abbruchbedingung
Ψ	Gesamtpotential des Bodens
Ψ_M	Matrixpotential
τ	Scherkraft (in *N*)
θ	Winkel (in °)
Θ	Bodenwassergehalt (in Vol.-% oder *mm*/Bezugstiefe)
Θ_∞	maximaler Bodenwassergehalt (in Vol.-% oder *mm*/Bezugstiefe)
Θ_{nFK}	pflanzenverfügbarer Bodenwassergehalt (Θ-*PWP*), also prozentualer Anteil der *nFK*
$\forall$	Allquantor
∂	Differentiationsoperator
|	Alternative, a | b ist "a oder b"

Zusammenfassung

Der Einsatz von Geoinformationssystemen (GIS) erlangt zunehmend an Bedeutung für weite Gebiete der Hydrologie. Das *WAQIS*-Projekt hat zum Ziel, ein Wasserqualitäts-Informationssystem (*WAQIS*) für den regionalen Wasserversorger Freiburger Energie- und Wasserversorgungs-AG (*FEW*) aufzubauen. Das Projekt wird von der *FEW* in Zusammenarbeit mit dem Institut für Hydrologie der Albert-Ludwigs-Universität Freiburg (*IHF*) bearbeitet.
In einem Wasserqualitätsinformationssystem ist für den Schutz des Grundwassers eine möglichst genaue Kenntnis der verschiedenen Grundwasserparameter erforderlich. Um die Leistung von Grundwassermodellen zu erhöhen, bedarf es neben möglichst genauen und umfangreichen Eingabedaten auch der Fähigkeit, das Simulationsmodell mit Daten und Randbedingungen flexibel und weitgehend automatisch zu parametrisieren.

Die Verbesserung der Leistung und Genauigkeit von Grundwassermodellen, insbesondere des im *WAQIS*-Projekt verwendeten Programms *FEFLOW* (Finite Elemente Flow Simulation Programm) ist die Motivation für diese Arbeit, die den Aufbau eines Oberflächen- und Grundwasser-Teilinformationssystems (*OGRUT*) für *WAQIS* zum Zweck hat.
Die Aufgabe von *OGRUT* besteht dabei nicht nur im Verwalten sämtlicher für die Modellanwendung erforderlichen Daten in einer Geodatenbank. Vielmehr muss es auch Auswertungswerkzeuge bereitstellen, die die erfassten Daten verarbeiten und somit einen Informationszuwachs gewährleisten. Im Falle des *OGRUT* sollen diese Werkzeuge die Daten zur Parametrisierung des Grundwassermodells aufbereiten bzw. berechnen. Durch Rücknahme und Aufbereitung der Modellergebnisse soll eine Visualisierung der räumlichen und zeitlichen Variablilität der Grundwasserdaten möglich werden.
Von grossem Interesse ist dabei die Grundwasserneubildung als relevante Grösse für die Erneuerung der Grundwasservorkommen. Diese Randbedingung gilt es möglichst genau zu erfassen, da sie als Inputgrösse grossen Einfluss auf die Ergebnisse des Grundwassermodells hat. Neben der direkten Grundwassserneubildung aus flächenhaftem Niederschlag kommt im Anwendungsgebiet Zartener Becken als weitere Grösse die indirekte Grundwasserneubildung aus Überflutungsflächen hinzu. Um die Überflutung durch Oberflächengewässer möglichst genau zu Erfassen, wurde ein Programm zur Berechnung von Wasserspiegellinien an *OGRUT* angekoppelt.
Der operationelle Einsatz und die notwendige Fähigkeit zur Berechnung von Zeitreihen der Abflussdaten ziehen die zentrale Forderung nach sich, dass die Parametrisierung des Wasserspiegellinienprogramms durch den in *WAQIS* enthaltenen Datenbestand realisierbar sein muss. Daher wurde beschlossen, ein entsprechendes Programm selbst zu implementieren.
Als hydraulische Grundlage für das Programm fiel die Wahl auf die Simulationssoftware *HEC-RAS* (Hydrologic Engineering Center's River Analysis System; BRUNNER, 1998), da hier nicht nur der Fall eines strömenden Abflussregimes sondern auch die Berechnung des schiessenden Abflusses beschrieben wird. Das neu entwickelte Programm *UWSP* (ungleichförmigen Wasserspiegellinienprogramm) lehnt sich an die in *HEC-RAS* angewandte Hydraulik an.

OGRUT wurde wie das als Basis des *WAQIS* dienende Wasserinformationssystem *LIWIS* der Firma *Lahmeyer International* als Softwareanwendung im GIS *Smallworld* entwickelt und programmiert.
Das GIS *Smallworld* besteht aus der Entwicklungsumgebung *Magik*, einer objektorientierten Programmiersprache und einer objektorientierten relationalen Datenbank. Diese wird mit einem *CASE*-Tool erstellt, die Funktionalität wird dann mit *Magik* programmiert.

Die Hauptkomponenten des *OGRUT* sind ein Oberflächengewässer-Modul, ein Finite-Elemente Modul, ein Agrar-Modul und ein Modul für digitale Geländemodelle. Hinzu kommen Werkzeuge zur Auswertung und Berechung abgeleiteter Daten. Das Programm *UWSP* wird über eine Schnittstelle zu *OGRUT* in das GIS eingebunden.

Das Oberflächengewässermodul verwaltet die Geometrie- und Abflussdaten an Gerinnen. Das Programm *UWSP* wird aus dem GIS heraus parametrisiert und berechnet die Wasserstandsdaten von Gerinnen. Die Überflutung durch Hochwasser wird anhand eines digitalen Geländemodells ermittelt. Zusammen mit Daten aus Hydropedotopen wird die Grundwasserneubildung für die einzelnen Geländepunkte des Höhenmodells berechnet. Diese Zeitreihendaten der Grundwasserneubildung werden dann auf Katasterflächen übertragen, wo sie auf die Fläche bezogen werden. Hinzu kommen an diesser Stelle Grundwasserneubildungsdaten aus Niederschlag. Diese Daten werden aus *BOMET*, einem vom *IHF* entwickelten bodenkundlich-meteorologisches Teilinformationssystem entnommen. Zuletzt wird eine Verschneidung zwischen Katasterflächen und Elementflächen des finiten Elemente-Netzes vorgenommen. Die Grundwasserneubildungs-Zeitreihen für das finite Elemente-Netz dienen dann neben anderen Daten als Parameter für das Grundwassersimulationsmodell *FEFLOW*.
Neben Funktionen zur Datenmanipulation und Datenorganisation werden von *OGRUT* weitere Werkzeuge bereitgestellt, die die nahtlose Steuerung der einzelnen Berechnungsschritte von einer zentralen Benutzeroberfläche und aus einem zentralen Datenbestand erlauben.
Erste Tests mit den neuen Anwendungen zeigen, dass die Vorhersagegenauigkeit von *FEFLOW* durch die Parametrisierung aus *OGRUT* für einige Situationen gesteigert werden kann. Insbesondere die bessere Berücksichtigung der Grundwasserneubildung aus Überflutung führt im Anwendungsgebiet zu einer Verbesserung der Modellergebnisse.

Abstract

OGRUT - Creation of Groundwater and Surface Water Information System

The use of Geographical Information Systems (GIS) gains increasingly in importance for a lot of fields of hydrology. The goal of the *WAQIS*-project is to set up a water quality information system (*WAQIS*) for the regional water supplier Freiburger Energie- und Wasserversorgungs-AG (*FEW*). The project is worked on by the *FEW* in cooperation with the institute of hydrology of the Albert-Ludwigs-Universität Freiburg (*IHF*).
To protect the ground water, a water quality system needs to know the different parameters of groundwater as exactly as possible. Besides very exact and extensive input data also the ability to flexibly and to a great extend automatically parametrizise the simulation model with data and border conditions is needed to improve the output of groundwater models.

The improvement in power and accuracy of groundwater models, especially of the *FEFLOW* program (Finite Elements Flow Simulation Program) which is used in the *WAQIS* project initiated this work with its purpose of constructing a surface and groundwater information system (*OGRUT*).
OGRUTs task does not just consist of managing all the data in a geographical database which are required in a model application. Rather it has to provide evaluation tools which are processing the available data and therefore guarantee an increase in information. In the case of *OGRUT* these tools should edit and calculate the data for parametrizing the groundwater model. A visualisation of the spatial and temporal variability of the groundwater data should be possible by retraction and preparation of the model results.
The groundwater recharge as a relevant parameter for the renewal of groundwater ressources is of great interest. As an input variable this border condition must be registered as exactly as possible as it has a significant effect on the results of the groundwater model. Besides the direct groundwater recharge because of rain a further parameter in the application area *Zartener Becken* is the indirect groundwater recharge from flood areas. To register the flooding by rivers as accurately as possible a program to calculate water surface profiles was linked to *OGRUT*.
The operational use and the necessary ability to calculate time series of runoff data demand the realisation of the parametrization of the water surface profile program by the *WAQIS* datapool. It was therefore decided to implement an appropriate program. As hydraulic basis for the program the simulation software *HEC-RAS* (Hydrologic Engineering Center's River Analysis System; BRUNNER, 1998) was chosen as it describes the subcritical flow regime as well as the supercritical runoff. The newly developped program *UWSP* (ungleichförmiges Wasserspiegellininenprogramm) is modelled on the hydraulics which are used in *HEC-RAS*.

OGRUT as well as the water information system *LIWIS* by the *Lahmeyer International* company which serves as basis for *WAQIS*, were developped and programmed as software applications for GIS *Smallworld*.
GIS *Smallworld* consists of the development environment *Magik*, an object-oriented programming language and an object-oriented relational database. The latter is constructed with a CASE-tool, the functionality is then programmed with *Magik*.
The main components of *OGRUT* are an river module , a finite elements module, an agricultural module and a module for digital terrain models, furthermore tools for evaluating and calculating derived data. The program *UWSP* is integrated into *OGRUT* by an interface.

The river module is managing the geometrical and runoff data of channels. The *UWSP* programme is parametrizised by the GIS and calculates the water surface data of channels. Floodings are calculated by a digital terrain model. Together with data from hydropedotops the groundwater recharge for particular points of the terrain model are calculated. These time series data are then transferred to land register planes.
Furthermore groundwater recharge from rain are taken from *BOMET* a pedological-meteorological information system which was developped by the *IHF*. In the end an intersection between land register planes and element areas of the finite elements network is carried out. Next to other data the groudwater recharge time series for the finite elements network are then serving as parameters for the groundwater simulation model *FEFLOW*.
Besides functions for data manipulation and organisation *OGRUT* provides further tools which allow the seamless control of single calculation steps from a central user surface and a central datapool.
First tests with the new applications show that the parametrization by *OGRUT* leads to an increase in the *FEFLOW* prediction accuracy of some situations. Especially the stronger consideration of groundwater recharge by flooding leads to an improvement of the model results in the application area.

Keywords

Geographical Information System
Data Modelling
Object-Oriented Programming
GIS *Smallworld*
Ground Water Regeneration
Ground Water Model Parametrization
Water Surface Calculation
Flooding Areas

1 Einleitung

Unter den Süsswasservorräten der Erde entfällt etwa die dreissigfache Menge des Oberflächenwassers auf das unterirdische Wasser (DYCK&PESCHKE, 1995). Aber nicht so sehr die grössere Menge als vielmehr die bessere Qualität des Grundwassers machen es zur wichtigeren Quelle für die Trinkwasserversorgung. Andererseits ist auch das Grundwasser einer zunehmenden anthropogenen Belastung ausgesetzt. Dadurch gerät sein Schutz vor Erschöpfung und Verschmutzung zur zentralen Aufgabe der Qualitätssicherung eines Trinkwasserversorgers.

Die Beantwortung hydrologischer Fragestellungen im Sinne der Nachhaltigkeit erfordert zunehmend integrative, raumbezogene und zeitvariante Beschreibungen der bewirtschafteten Grundwassersysteme und ihrer Einflussgrössen. Aus einer Zusammenarbeit mit der regionalen Wasserversorgung *Freiburger Energie- und Wasserversorgungs-AG* (*FEW*) bei Fragen des Ressourcenmanagements und der Qualitätssicherung der bewirtschafteten Trinkwasservorkommen im Freiburger Raum entwickelte sich das Forschungsprojekt *WAQIS* (*Wasserqualitätsinformationssystem*).
Forschungsgegenstand dieses Projekts bildet der Wasserhaushalt der regionalen Grundwassergewinnungsgebiete Zartener Becken und Staufener Bucht. Gekoppelt wird dies mit der Entwicklung eines raumbezogenen, wasserwirtschaftlichen Informationssystems *WAQIS*. Dieses Informationsystem soll zukünftig eine zentrale Plattform der regionalen Wasserwirtschaft und hydrologischen Forschung bilden wird.

Im Zartener Becken, einem der beiden grossen Trinkwassereinzugsgebiete der Freiburger Energie- und Wasserversorgungs-AG (*FEW*), besteht wie vielerorts die Notwendigkeit zur Lösung eines Nutzungskonfliktes zwischen Trinkwasserversorgung und intensiver Landwirtschaft. Ein Schritt in diese Richtung ist das *WAQIS*-Projekt.
Die Aufgabe des Projektes besteht im zentralen Monitoring aller für den Aquifer- und Bodenschutz relevanter Daten in einer Geodatenbank. Dieses Informationsystem ermöglicht dann die räumliche und zeitliche Verknüpfung der Daten um einen Informationsmehrwert zu erhalten. Hierfür stellt das *WAQIS* eine Reihe von Werkzeugen bereit, die die Kenntnisse über die Verhältnisse im Boden und in den Grundwasserleitern erweitern. Anhand dieser Werkzeuge lassen sich z.B. Beurteilungen und Massnahmen zur Reduktion der Schadstoffbelastung durchführen.

WAQIS will der heute geforderten integrativen Betrachtungsweise gerecht werden. Dafür bietet es neben den eigentlichen Wasserwerksinformationen die Fähigkeit der qualitativen und quantitativen Beschreibung aller relevanten Wasserhaushaltsgrößen und ihrer Steuerfaktoren.
Diese Aufgaben werden von in die Strukturen der GIS-Anwendung *WAQIS* integrierten hydrologische Modelle und Methoden wahrgenommen. Die GIS-Anwendung liefert dabei sowohl eine automatisierte Parametrisierung der Modellrandbedingungen durch integrierte Preprocessing-Methoden, als auch die Auswertung, Verwaltung und Visualisierung der Modellergebnisse.

2 Problemstellung und Zielsetzung

2.1 Problem und Motivation

Das im Rahmen von *WAQIS* entwickelte Grundwassermodell für das Zartener Becken (JUNGHANS, 1998) muss für eine realistische Beschreibung des Grundwassersystems mit zeitlich und räumlich hochauflösenden Randbedingungen versorgt werden. Die Steigerung der Leistungsfähigkeit und der operationellen Anwendbarkeit des Grundwassermodells bilden die Motivation für diese Arbeit. In ihr wird für das *WAQIS* ein Oberflächengewasser- und Grundwasser-Teilinformationssystem (*OGRUT*) entwickelt, dessen Grundlage eine Geodatenbank aus hydraulischen, bodenkundlich-meteorologischen und geographischen Daten ist. Darauf aufbauend stellt *OGRUT* Werkzeuge zur Verfügung, die z.B. die Berechung der indirekte Grundwasserneubildung aus Überflutungsflächen und die Anbindung eines externen Programms zur Berechnung von Wasserspiegellinien ermöglichen.

Die Wichtigkeit des Bodenraums für die Sicherung der Trinkwasserqualität waren Anlass für die Entwicklung des *bodenkundlich-meteorologischen Informationsmoduls BOMET* im Rahmen des *WAQIS*-Projekts (EBERLE, 1999).
Den zweiten Schwerpunkt des Forschungsprojektes bilden die aktuellen, langfristigen, dreidimensionalen Grundwasserströmungs- und Stofftransportmodelle, sowie die für ihre Parametrisierung verwendeten Werkzeuge. Der GIS-Software obliegt dabei die Aufgabe der Verzahnung der verschiedenen hydrologischen Teilsysteme bzw. Modellsysteme (z.B. Bodenwasser – Grundwasser, Oberflächengewässer – Grundwasser). Anlass für diese Arbeit waren somit die Verringerung des Aufwands bei der Modellpflege, die zentrale Verfügbarkeit von Modellen und Ergebnissen und die vereinfachte Gewährleistung der Aktualität der Modelle.
Diese Diplomarbeit liefert somit einen Beitrag zur Weiterentwicklung des *WAQIS*.

2.2 Zielsetzung

Gestützt auf ein grundlegendes Modellsystem der Firma *Lahmeyer International*, dem Wasserinformationsystem *LIWIS* gilt es ein Modul zur Anbindung des Grundwassersimulationsprogramms *FEFLOW* zu entwickeln. Insgesamt zeigt es sich, dass das bestehende *LIWIS* in erheblichem Umfang an die Bedürfnisse der Freiburger Wassergewinnung angepasst werden muss. Eine Analyse der hydrologischen Situation zeigte die Bedeutung der folgenden dynamischen Steuergrössen:

- Randbedingung der flächenhaften Grundwasserneubildung aus Niederschlag. Deren Bereitstellung wird von *BOMET* wahrgenommen.
- Randbedingung des linienhaften Grundwasseraustausches über Vorfluter. Diese Randbedingung wird gegenwärtig ohne weitere Differenzierung direkt in *FEFLOW* definiert.
- Randbedingung der indirekten Grundwasserneubildung aus Überflutungsflächen. Hier zeigte sich ein hohes Mass an Anpassungsbedarf, um alle benötigten Daten bereitstellen zu können. Diese Aufgabe wird Kern der Arbeit sein.

Teil der Aufgabe wird auch sein, eine automatische Parametrisierung bei variabel definierbarer zeitlicher Auflösung (Monat bis Stunden) zu ermöglichen. Dies wird den bisher hohen Aufwand für die Bereitstellung von Daten verringern.

Aufgabe dieser Arbeit ist somit die Integration des Programms *FEFLOW* in das GIS *Smallworld*. Die Koppelung wird die automatische Parametrisierung der Randbedingungen und Eingabedaten für *FEFLOW* aus *Smallworld* ermöglichen.
Dies umfasst die Berechnung und Bereitstellung von Daten, die *FEFLOW* als Eingabe dienen, insbesondere Zeitreihen von Grundwasserneubildungsdaten. Gefordert werden einerseits Daten der Grundwasserneubildung aus flächenhaftem Niederschlag und andererseits Daten der Grundwasserneubildung aus Überflutung durch Gewässer. Diese Daten sollen ermittelt bzw. berechnet werden und *FEFLOW* in geeigneter Form übermittelt werden. Die Berechnung der Grundwasserstands-Zeitreihen in *FEFLOW* ist Aufgabe einer anderen Diplomarbeit (BOLD, 2000), die parallel durchgeführt wird. Die berechneten Modellergebnisse, z.B. Grundwasserspiegeldaten sollen wieder in *Smallworld* eingelesen werden und weiterverarbeitet werden können.
Zur Umsetzung der genannten Aufgaben sind in *Smallworld* geeignete Datenstrukturen zu definieren, die die verschiedenen Daten aufnehmen können. Es sind Funktionen zu implementieren, die die erforderlichen Berechnungen durchführen. Weiterhin ist eine graphische Benutzeroberfläche (*graphical user interface*, *GUI*) zu erstellen die es dem Anwender erlaubt, auf einfache Weise und ohne Kenntnis der konkreten Algorithmen und Datenstrukturen die gewünschten Bearbeitungsschritte durchzuführen.

Ziel ist, die Leistungsfähigkeit des GIS *Smallworld* als integrierendes System zu erhöhen. Es dient als Kontrollzentrum für eine Fülle von Daten, die im Zusammenhang mit der Erforschung und Nutzung von Komponenten des Wasserkreislaufs anfallen und soll deren zentrale Verwaltung, Verarbeitung und Visualisierung ermöglichen. Im Rahmen des Forschungsprojektes *WAQIS* sollen die Ergebnisse dieser Arbeit als weiter Komponente in ein Gesamtkonzept einfliessen.

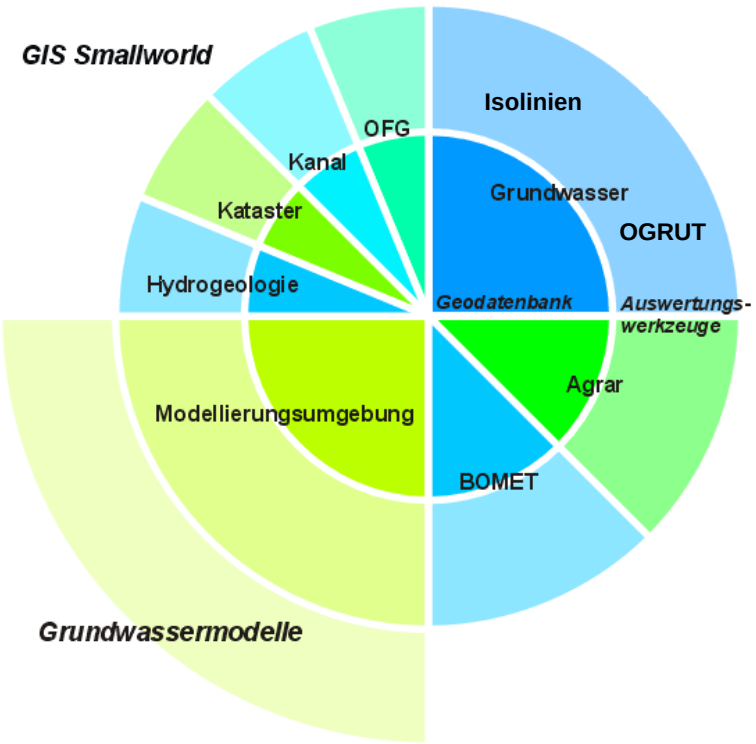


Abb. 2.1: Schematischer Aufbau des WAQIS (nach WAQIS, 2000)

Thema dieser Arbeit ist also die Entwicklung einer Anzahl von Komponenten für die *WAQIS*-Geodatenbank in GIS *Smallworld*. In einer flexiblen, modular aufgebauten Struktur

sollen Datenmodelle und Auswertungswerkzeuge zu allen relevanten Themen angeboten werden (s. Abb. 2.1). Das Ergebnis dieser Arbeit wird das integrative Konzept des *WAQIS* um weitere Bausteine, insbesondere im Bereich Oberflächengewässer, Isolinien-Modell und Grundwasser ergänzen.

3 Anwendungsgebiet und Datengrundlage

3.1 Anwendungsgebiet

Das Anwendungsgebiet liegt im Südwesten von Baden-Württemberg, östlich der Stadt Freiburg im Landkreis Breisgau-Hochschwarzwald (TK 25-Blätter 7913, 8013, 8014). Es umfasst das von der Dreisam entwässerte Gebiet oberhalb des amtlichen Gewässerpegels Ebnet an der Dreisam. Der Pegel umfasst eine Fläche von etwa 260 km^2 wovon 90% zum Grundgebirgsschwarzwald und 10% zum intramontanen, mit Schottern gefüllten Zartener Becken gehören (METZ, 1959). Die höchste Erhebung des Einzugsgebiets ist mit 1493 *m* der Feldberg im Südschwarzwald, der niedrigste Geländepunkt mit 308.29 *m* der Pegel Ebnet (HERDEG, 1993).
Das Einzugsgebiet besteht aus drei Teilgebieten, nämlich dem Zartener Becken, einem stark reliefierten Gebiet und einem Teil aus Hochflächen und Mulden. Diese Dreiteilung findet man auch bei den Bachläufen. Die Unterläufe haben sich je nach Lage und Erosionskraft in die breite Schotterebene des Zartener Beckens eingetieft. Die Mittelläufe der Bäche sind in enge, steilwandige Kerbtäler eingezwängt. Im Oberlauf können diese Täler in wannenförmige Hochtäler übergehen. Meist sind aber auch die Oberläufe steile und schottererfüllte Tobel mit einem unausgeglichenen Gefälle (FRIEG, 1987).
Das Anwendungsgebiet beinhaltet das Grundwassereinzugsgebiet des Wasserwerks Ebnet, das ca. 1 *km* oberhalb des Pegels Ebnet im Zartener Becken liegt.

3.1.1 Klima

An der Niederschlagsstation Ebnet betrug der mittlere Niederschlag der hydrologische Jahre 1958 bis 1984 1080 *mm* (FRIEG, 1987). An der Klimastation Buchenbach, die ca. 120 *m* höher als die Station Ebnet liegt, sind zwischen den einzelnen Jahren Niederschlagsschwankungen bis zu 400 *mm* in der Jahresniederschlagssumme festzustellen (s. Tab. 3.1). Der Gebietsniederschlag für das Gesamteinzugsgebiet der Dreisam (bis zum Pegel Ebnet) betrug im Mittel der hydrologischen Jahre 1952 bis 1984 1473 *mm* (FRIEG, 1987).

Tab. 3.1: Niederschlag, aktuelle Evapotranspiration und Grundwasserneubildung (GWNB) bei 100 mm nFK. Hydrologische Jahre 1988-1993, Station Buchenbach (aus ROLKE, 1994).

	Niederschlag [$mm\ a^{-1}$]	ET_a (Gras) [$mm\ a^{-1}$]	GWNB (Gras) [$mm\ a^{-1}$]
1988	1330	657	664
1989	957	579	448
1990	928	598	220
1991	962	529	401
1992	1045	555	531
1993	1043	551	421
Mittel 1988 bis 1993	1044	578	448

Die Verdunstung im Zartener Becken wurde von DOMMERMUTH&TRAMPF (1990) nach HAUDE-RENGER (RENGER, 1974) ermittelt Wie in Tab. 3.1 zu sehen ist, variiert die aktuelle Evapotranspiration ET_a zwischen den einzelnen Jahren erheblich.

3.1.2 Geologie

Das Zartener Becken ist ein von Grundgebirge (Granite, Gneise und Anatexite) umgebenes und mit quartären fluvioglazialen Schottern gefülltes intramontanes Becken. Der Schotterkörper wird zu 90% aus Gneisen gebildet. Der Schotterkörper wurde von Bächen unter Bildung von Niederterrassenflächen durchschnitten. An den Ausgängen der Seitentäler in die Haupttäler wurden postglazial Schwemmkegel aufgeschüttet (FRIEG, 1987).
Nach HERDEG (1993) lässt sich der Schotterkörper in zwei Lagen gliedern:
Verwitterte Schotter und Sande des älteren Pleistozän mit hohem Schluff- und Tonanteil und somit geringer Durchlässigkeit sind von unverwitterten, geringschluffigen Schottern und Sanden des Würmglazials höherer Durchlässigkeit überlagert.
Die würmeiszeitlichen Schotter weisen im Südteil des Zartener Beckens einen höheren Schluffanteil auf als im Nordteil. Die Mächtigkeit nimmt von Osten nach Westen und von Norden mit bis zu 35 *m* Mächtigkeit nach Süden mit 5 bis 15 *m* Mächtigkeit ab (HERDEG, 1993).
Der Schotterkörper bildet das Hauptgrundwasserreservoir für die Freiburger Trinkwasserversorgung.

3.1.3 Pedologie

Bezüglich der Bodenentwicklung kann man im Zartener Becken drei Bodengesellschaften unterscheiden: Böden der Talauen, Böden der Niederterrassen und Böden der Hangschuttsedimente (GLOMB&ZWÖLFER, 1990).
Grosse Teile der Schotter der Niederterrassenflächen wurden im Pleistozän von Hochflutlehmen unterschiedlicher Mächtigkeit überlagert.. Dadurch konnten sich bei bis zu einem Meter mächtigen Lehmüberdeckungen mittel- bis tiefgründige Braunerden ausbilden, bei Lehmüberdeckungen bis 40 *cm* flachgründige Braunerden. Teile der Niederterrassen sind von geringmächtigen holozänen Auenlehmen überdeckt mit nachfolgender Bildung von Auenbraunerden und braunen Auenböden (ROLKE, 1994).
Periodische Überflutungen der Talebenen lagerten im Holozän in den Flussauen schluffige humose Lehme ab. Auch hier wurde die Bodenbildung durch die Mächtigkeit der Deckschichten und zusätzlich durch die Nähe zum Fliessgewässer und somit durch Grundwasserbeeinflussung gesteuert. Bei fehlenden oder sehr geringmächtigen Deckschichten entwickelten sich Auenregosole, ansonsten Auenbraunerden. Bei Grundwasserbeeinflussung entstanden Auengleye. Nach flächenhaftem holozänen Bodenabtrag von den Hängen bildeten sich an Hangfüssen, in Seitentälern und auf Schwemmfächern Kolluvien und Gleye (ROLKE, 1994).

Allgemein ist der Schotterkörper des Anwendungsgebiets weitgehend von Auen- und Hochflutlehmen aus mehreren Sedimentationslagen unterschiedlicher Körnung bedeckt. Die Mächtigkeit dieser Lehme bestimmt somit das Wasserspeichervermögen (nutzbare Feldkapazität) des Bodens im Anwendungsgebiet.

3.1.4 Hydrologie

3.1.4.1 Oberflächengewässer

Oberirdisch wird das Zartener Becken von der Dreisam mit den Hauptzuflüssen Brugga, Krummbach, Rotbach, Wagensteigbach und Eschbach entwässert (s. Abb. 3.1).

Die intensive Zertalung des Einzugsgebietes bedingt kleine getrennte Einzugsgebiete der Oberflächengewässer und damit ein relativ dichtes Gewässernetz. Die Flussdichte beträgt 1.66 $km\ km^{-2}$ (HOFIUS&NIPPES, 1985).Die meist kurzen, wasserreichen Bachläufe entwässern das Gebiet zum Rhein. Sie sind durch grosses Gefälle (etwa 3%) und stark wechselnde Wasserführung charakterisiert (KIRWALD, 1980). Durch ihre starke Erosionskraft haben sie ihre Oberläufe und Quellen immer mehr nach Osten verlegt und sind so den flachgeneigten, auf der Leeseite des Gebirges der Donau zuströmenden Flüssen überlegen (FRIEG, 1987).
Die Niedrigwasserabflussspende *Nq* der grösseren Vorfluter liegt auch in extremen Trockenjahren bei 5 bis 10 $l\ s^{-1}km^{-1}$. Grund sind die auch dann noch relativ hohen Niederschläge und das geringe Speichervermögen im Untergrund der Ober- und Mittelläufe der Bäche. Das starke Gefälle führt zudem zu einem schnellen Leerlaufen des Grundwasserleiters (FRIEG 1987).

Tab. 3.2: Mittlere Jahresabflusswerte (aus FRIEG, 1987, teilweise aus WUNDT, 1965).

Nr	Gewässer	Pegel (Berechnungszeitraum)	m^3s^{-1}	$ls^{-1}\ km^{-2}$	mm
1	Dreisam	Ebnet (1947 bis 1984)	5.63	21.79	688
2	Brugga	Oberried-Ibrecht (1946 bis 1983)	1.53	38.52	1214
3	St. Wilhelmer Talbach	St.Wilhelm (1958 bis 1983)	0.66	43.51	1372
4	Zastlerbach	Zastler	0.62	33.83	1067
5	Rotbach	Hirschsprung (1950 bis 1955)	1.1	29.7	936
6	Wagensteigbach	Wiesneck (1947 bis 1983)	1.22	24.09	758
7	Ibenbach	Unteribental (1941 bis 1955)	0.4	24.4	769

Für die Abflussmessung stehen eine Reihe von Pegeln zur Verfügung (s. Tab. 3.2 und Abb. 3.1). Unteribental und Hirschsprung wurden wieder stillgelegt. Der oberirdische Abfluss beträgt am Pegel Ebnet im Jahresmittel 5.63 $m^3\ s^{-1}$. Dies entspricht einer Abflussspende von 21.79 $ls^{-1}\ km^{-2}$. Wenn man diesen Wert mit dem Gebietsniederschlag von 1473 *mm* vergleicht, so fliessen 48 % (688 *mm*) davon oberflächlich ab. Der Abfluss im Winterhalbjahr beträgt etwa 70 % des Gesamtjahresabflusses. Im langjährigen Durchschnitt erreicht die Dreisam im April mit 8.37 m^3s^{-1} bei Pegel Ebnet den höchsten Monatswert. Auch die übrigen Messstellen haben im April ihre höchsten Abflüsse zu verzeichnen. Die Pegel Zastler und St.Wilhelm mit den hochgelegenen Einzugsgebieten erreichen jedoch auch im Mai aufgrund der späteren Schneeschmelze fast die Aprilwerte. Die geringsten Abflüsse findet man bei allen Pegeln im September, wo sie ungefähr bei einem Drittel des Aprilwertes liegen (FRIEG 1987).
Wenn man die Abflusswerte aller Teileinzugsgebiete zusammenzählt , ergibt sich gerade der Abflusswert der Dreisam, obwohl das Kappeler Tal und der Eschbach mit seinen Nebenbächen noch nicht berücksichtigt sind. Die Wasserführung der Dreisam müsste 1 bis 2 $m^3\ s^{-1}$ höher sein. Es versickert also ein Teil der Bäche im Zartener Becken. (WUNDT, 1965).

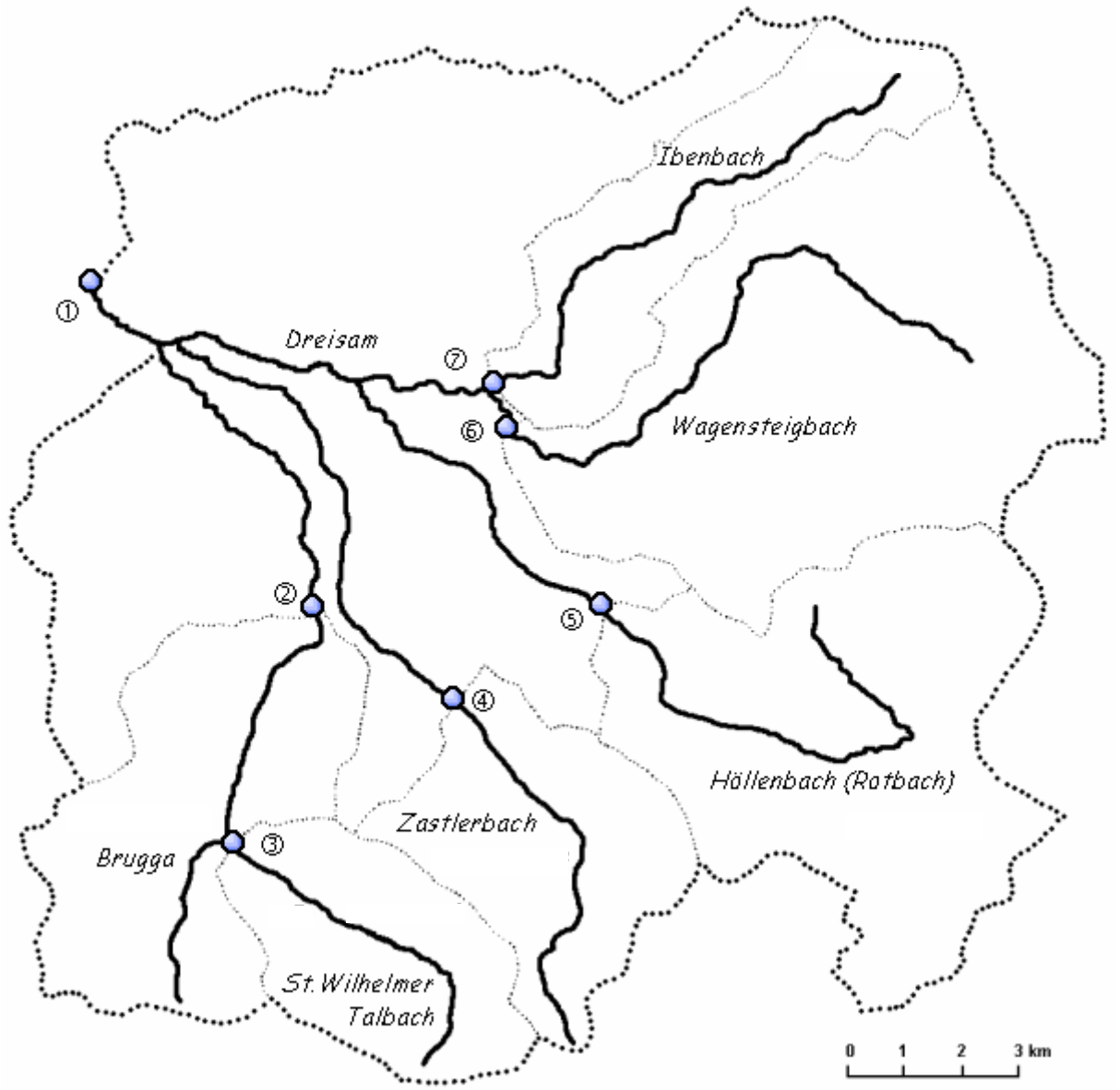


Abb. 3.1: Gewässerpegel und Teileinzugsgebiete des Dreisameinzugsbiets bis zum Hauptpegel Ebnet (aus FRIEG, 1987).

3.1.4.2 Grundwasser

Die Grundwasserspeicherfähigkeit des Zartener Beckens ist durch die Gliederung in zwei Schotterkörper geprägt. Die liegenden, stark schluffigen Schotter wirken als Grundwasserhemmer; die hangenden, unverwitterten und schluffarmen Schotter bilden einen Porengrundwasserleiter.

Im Nordteil des Beckens entwickelte sich eine Rinne mit bis zu 35 *m* mächtigen (EHRMINGER & HERDEG, 1992), gut durchlässigen Sedimenten (k_f = $1.28 \cdot 10^{-3}$ $m\ s^{-1}$) und einem hohen Grundwasserspeichervermögen bei einer mittleren Abstandsgeschwindigkeit von 4 bis 6 $m\ d^{-1}$ (FRIEG, 1987).
Im südlichen Becken ist die Mächtigkeit der unverwitterten Schotter geringer (5 bis 15 *m*) (EHRMINGER & HERDEG, 1992). Der Schluffgehalt ist höher und die Durchlässigkeit liegt bei $k_f = 1.99 \cdot 10^{-4}$ $m\ s^{-1}$. Das Grundwasserspeichervermögen ist geringer und die mittlere Abstandsgeschwindigkeit liegt bei 3.8 $m\ d^{-1}$ (FRIEG, 1987).

Die Hauptgrundwassermenge fliesst in der Nordrinne, und zwar von Osten nach Westen mit Zuflüssen von Südost nach Nordwest. Daher fällt auch der Grundwasserspiegel von Süd nach Nord. Die mittleren Flurabstände liegen im Nordteil bei 10 bis 20 *m* und 3 bis 10 *m* im Südteil des Beckens (EHRMINGER&HERDEG, 1992). Von Ost nach West nimmt der Flurabstand ab, da die Festgesteinsoberfläche im Vergleich zum Grundwasserspiegel im Westen des Beckens schwächer einfällt und gleichzeitig eine Verengung des Talquerschnittes erfolgt (HERDEG, 1993).
Nach Niederschlägen findet man in Gebieten mit hohen Flurabständen nach 3 bis 4 Wochen Grundwasseranstiege, bei geringen Flurabständen bereits nach 1 bis 2 Wochen (EHRMINGER & HERDEG, 1992).

3.1.4.3 Grundwasserneubildung

Bei niedrigen Grundwasserständen erfolgt vorwiegend im südlichen Becken eine Grundwasseranreicherung mit Uferfiltrat (HERDEG, 1993), bei hohen Grundwasserständen exfiltriert Grundwasser in die Flüsse. Folglich sind die Grundwasserstände der Messstellen in der Nähe von Oberflächengewässern ausgeglichener als die der übrigen Grundwassermessstellen (ROLKE, 1994).

Nach einer Kartierung von GLOMB&ZWÖLFER (1989) beträgt die mittlere Grundwasserneubildung aus Niederschlag der Jahre 1959 bis 1988 je nach *nFK* der Böden 320 bis 450 $mm\ a^{-1}$.
Innerhalb der Jahre schwankt die Grundwasserneubildung je nach Niederschlagssumme und Jahresniederschlagsverteilung erheblich (s. Tab. 3.1). Im Sommerhalbjahr können aufgrund der hohen Evapotranspiration meist nur geringe Mengen Bodenwasser in die gesättigte Zone sickern (ROLKE, 1994).

3.2 Datengrundlage

Aus dem Untersuchungsgebiet wurden für diese Arbeit eine Reihe von Daten verwendet. Hierbei handelt es sich um:

- Gerinnegeometrie. Für die Oberflächengewässer im Einzugsgebiet wurden Querprofildaten verwendet, die in LEIBUNDGUT&GÄßLER (1998) angegeben sind.
- Abflusszeitreihen an Gerinnen. Es wurden die Abflussmessungen der *LfU* (Landesanstalt für Umweltschutz) an den Landespegel verwendet. Es werden sowohl Stundenwerte als auch Tageswerte verwendet.
- Katasterflächen. Die Daten über Katasterflächen liegen bereits digital vor und stammen von der Freiburger Energie- und Wasserversorgungs-AG.
- Hydropedotope. Die Einteilung des Anwendungsgebietes in Hydropedotope erfolgte durch das Insitut für Hydrologie, Albert-Ludwigs-Universität Freiburg.
- Für die Simulation von Grundwassersgrössen wird ein finites Elemente-Netz verwendet, das von JUNGHANS (1998) erstellt wurde. Dieses Netz wird auch als Grundlage zur Verwaltung von Grundwasserdaten verwendet.
- Für das digitale Geländemodell (DGM) des Anwendungsgebietes wurden Höhenlinien der Flurkarten 1:5000 vom Vermessungsamt Freiburg verwendet und digitalisiert. Die Äquidistanz (Vertikalabstand) der Isolinien beträgt hier bis hinab zu 0.25 *m*. Auf dieser Grundlage steht ein hochauflösenden DGM zur Verfügung.

4 Theoretische Grundlagen

4.1 Theorie der Gerinnehydraulik

Die Gerinnehydraulik beruht auf den grundlegenden Erhaltungssätzen der Physik, insbesondere die Erhaltung der Energie, des Impulses und der Masse. Durch Kombination dieser Grundgleichungen können Gleichungssystem aufgestellt werden, die von einfachen Fliessverhältnissen (gleichförmiger Normalabfluss) über ungleichförmiges Fliessverhalten bis hin zu instationärem Wellenablauf (flood routing) prinzipiell jeden Fliessvorgang im offenen Gerinne beschreiben können. Der Aufwand zur Berechnung steigt selbstverständlich mit den Anforderungen.
Während für den Normalabfluss lediglich ein Querprofil unabhängig vom vorhergehenden oder nachfolgenden Querprofil betrachtet werden muss, ist bei ungleichförmigem Abfluss eine Iterationslösung unter Einbeziehung zweier Querprofile erforderlich. Für instationäre Verhältnisse muss auf das SAINT-VENANT-Gleichungssystem zurückgegriffen werden, das sich wiederum aus der allgemeineren NAVIER-STOKES-Gleichung ableitet.

4.1.1 Bewegungsgleichungen

Bei instationären Bewegungen wird die NAVIER-STOKES-Bewegungsgleichung für inkompressible Flüssigkeiten verwendet. Sie enthält die hydrodynamische Grundgleichung und die innere Reibungskraft. Durch die NAVIER-STOKES-Gleichung kann der Ablauf von Wellen exakt beschrieben werden. Problematisch ist hierbei, dass die Formel nicht integrierbar ist. Somit müssen vereinfachende Annahmen getroffen werden.
Vernachlässigt man die Eigenschaften realer Fluide und betrachtet lediglich die Strömung idealer Fluide, so erhält man die EULER-Bewegungsgleichung.
Bleibt man bei realen Fluiden, vereinfacht aber das dreidimensionale Strömungsproblem auf ein eindimensionales Strömungsproblem, indem man die Geschwindigkeit und andere Parameter über den Gerinnequerschnitt mittelt, so ergeben sich die SAINT-VENANT-Gleichungen.
Berücksichtigt man weder Reibung, noch den mehrdimensionalen Fall, so kommt man unter der zusätzlichen Bedingung, dass stationäre Verhältnisse vorliegen zur BERNOULLI-Gleichung. Diese ist für stationäre, d.h. zwar räumlich, jedoch nicht zeitlich veränderliche Geschwindigkeiten geeignet. Diese Einschränkung macht sie für Flood-Routing unbrauchbar, jedoch nicht für die stationäre Wasserspiegellinienberechnung, bei der ein zeitlich konstanter Durchfluss für jeden Punkt vorausgesetzt wird.
Der Rechenaufwand und die Genauigkeit nimmt in der Reihenfolge

BERNOULLI < EULER < SAINT-VENANT < NAVIER-STOKES

zu. Die Übersicht in Tab. 4.1 verdeutlicht dies.

Tab. 4.1: Systematik der Bewegungsgleichungen.

	Ideales Fluid	***Reales Fluid***
1-dimensional	***stationär:BERNOULLI-Gleichung***	***SAINT-VENANT-Gleichung***
3-dimensional	***EULER-Gleichung***	***NAVIER-STOKES-Gleichung***

4.1.2 BERNOULLI-Gleichung

Bei stationären Verhältnissen kann die BERNOULLI-Gleichung verwendet werden. Sie drückt die Energieerhaltung für den Fall aus, dass die Geschwindigkeit an einem gegebenen Punkt zeitlich konstant ist. Man kann die BERNOULLI-Energiegleichung aus der allgemeineren SAINT-VENANT-Gleichung ableiten, indem man den vereinfachten Fall der Stationarität ($\frac{\partial v}{\partial t} = 0$) und der Reibungsfreiheit betrachtet.

Eine andere Möglichkeit ist, die Energieinhalte eines ohne Reibungsverluste bewegten Flüssigkeitsvolumens zu berechnen. Dies sind die Lageenergie W_G, die Druckenergie W_p und die kinetische Energie W_a . Für die einzelnen Kräfte gilt

$$W_G = \int_0^z F_G \cdot dx = \int_0^z m \cdot g \cdot dx = m \cdot g \cdot z \qquad \textit{Gl. 4.1}$$

$$W_p = \int_0^z F_p \cdot dx = \int_0^z p \cdot A \cdot dx = p \cdot A \cdot z = p \cdot V = p \cdot \frac{m}{\rho} \qquad \textit{Gl. 4.2}$$

$$W_a = \int_0^z F_a \cdot dx = \int_0^z m \cdot a \cdot dx = \int_0^z m \cdot \frac{dv}{dt} \cdot dx = \int_0^v m \cdot v \cdot dv = \frac{1}{2} \cdot m \cdot v^2 \qquad \textit{Gl. 4.3}$$

Mit $W_{gesamt} = W_G + W_p + W_a$ folgt

$$W_{gesamt} = m \cdot g \cdot z + p \cdot \frac{m}{\rho} + \frac{1}{2} \cdot m \cdot v^2 = \text{konstant} \qquad \textit{Gl. 4.4}$$

bzw. nach Division durch das Volumen

$$\rho \cdot g \cdot z + p + \frac{1}{2} \cdot \rho \cdot v^2 = \text{konstant} \qquad \textit{Gl. 4.5}$$

$$z + \frac{p}{\rho \cdot g} + \frac{v^2}{2 \cdot g} = \text{konstant} . \qquad \textit{Gl. 4.6}$$

Dies ist eine Form der BERNOULLI-Gleichung.

Immer dann, wenn der Wasserstand die kritische Tiefe unter- oder überschreitet, wird die auf der BERNOULLI-Gleichung basierende Energiegleichung als nicht geeignet angesehen. Sie ist lediglich anwendbar bei graduellen Änderungen der Abflussverhältnisse, und der Übergang von subkritischem (strömendem) zu superkritischem (schiessendem) Abfluss oder umgekehrt ist eine Situation, bei der sich die Abflussverhältnisse schnell ändern. Solche Übergänge können in verschiedenen Fällen auftreten. Hierzu gehören rasche Änderungen der Hangneigung im Verlauf des Gerinnebetts, Brückenbauwerke, Wehre und Zusammenflüsse zweier Gerinne. In einigen dieser Fälle können empirische Formeln eingesetzt werden (z.B. bei Wehren). in anderen Fällen ist es dagegen nötig, die Impulsgleichung anzuwenden um eine Lösung zu erhalten.

4.1.3 Impulsgleichung

Die Impulsgleichung kommt bei schiessendem Abflussregime zum Einsatz. Um zu zeigen, wie die Impulsmethode eingesetzt wird, wird hier eine Herleitung aus BRUNNER (1998) gezeigt.

Die Impulsgleichung wird aus NEWTONs zweitem Bewegungssatz abgeleitet:

Impulsänderung bzw. Kraft = Masse · Beschleunigung

$$\sum F_i = m \cdot a \qquad \textit{Gl. 4.7}$$

Wird NEWTONs zweiter Bewegungssatz auf einen Wasserkörper angewendet, der sich zwischen zwei Querprofilen an den Stellen 1 und 2 befindet (s. Abb. 4.1), dann kann folgender Ausdruck für die Änderung des Impulses über eine Zeiteinheit aufgestellt werden:

$$p_2 - p_1 + G_x - F_f = Q \cdot \rho \cdot \Delta v_x \qquad \textit{Gl. 4.8}$$

Mit:
- $p_{1,2}$ Hydrostatischer Druck an den Stellen 1 und 2
- G_x Gewichtskraft des Wassers, Komponente in *x*-Richtung
- F_f Kraft durchReibungsverluste zwischen 2 und 1
- Q Abfluss
- ρ Dichte des Wassers
- Δv_x Änderung der Geschwindigkeit zwischen 2 und 1, in *x*-Richtung

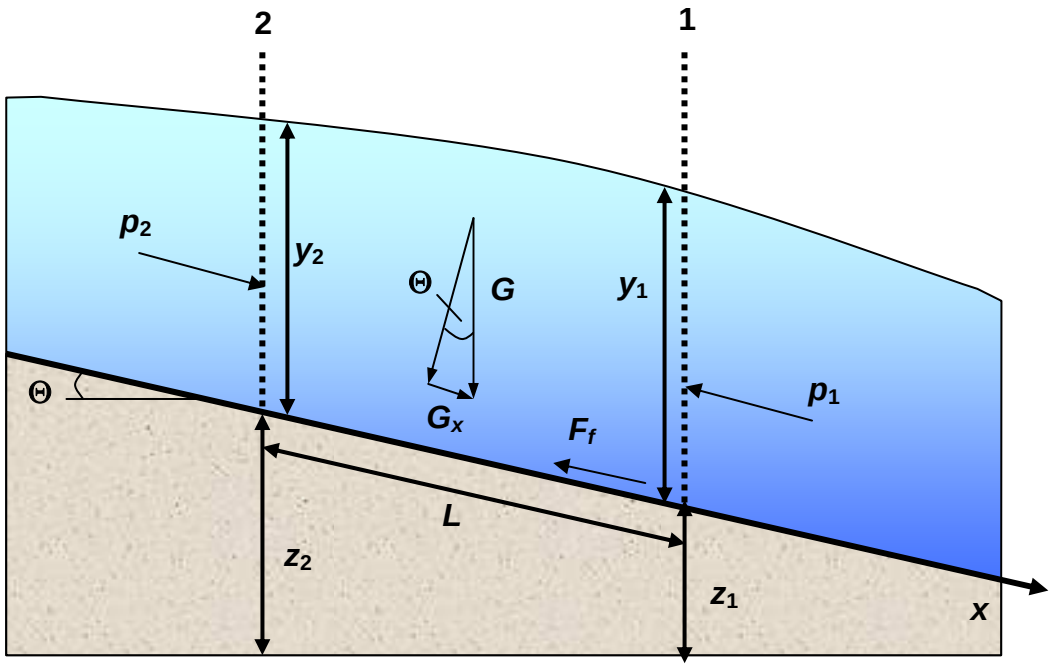


Abb. 4.1: Anwendung der Momentenmethode (nach BRUNNER, 1998).

Hydrostatische Druckkräfte

Die Kraft in *x*-Richtung aus dem hydrostatischen Druck ist:

$$p - \rho \cdot g \cdot A \cdot \bar{y} \cdot \cos\theta \qquad \textit{Gl. 4.9}$$

Die Annahme einer hydrostatischen Druckverteilung ist nur bei einem Längsgefälle mit einer geringeren Neigung als 1:10 erfüllt. Der *cos* θ für eine Längsgefälle von 1:10 (≈ 6°) ist gleich 0.995. Da die Neigung von Gerinnen gewöhnlich weit unter 1:10 ist, kann *cos* θ ≈ 1 angenommen werden (CHOW, 1959). Daher lautet die Gleichung für den hydrostatischen Druck an den Querprofilen 1 und 2 wie folgt:

$$p_1 = \rho \cdot g \cdot A_1 \cdot \bar{y}_1$$ *Gl. 4.10*

$$p_2 = \rho \cdot g \cdot A_2 \cdot \bar{y}_2$$ *Gl. 4.11*

Mit: ρ Dichte des Wassers
g Gravitationskonstante
A_i Durchflussquerschnittsfläche der Querprofile 1 und 2
y_1, y_2 Wassertiefe an den Querprofilen 1 und 2

Gewichtskraft des Wassers

Es gilt: Masse = Dichte · Volumen, also

$$G = \rho \cdot g \cdot V$$ *Gl. 4.12*

$$G = \rho \cdot g \cdot \left(\frac{A_1 + A_2}{2} \right) \cdot L$$ *Gl. 4.13*

$$G_x = G \cdot \sin\theta$$ *Gl. 4.14*

$$\sin\theta = \frac{z_2 - z_1}{L} = S_0$$ *Gl. 4.15*

$$G_x = \rho \cdot g \cdot \left(\frac{A_1 + A_2}{2} \right) \cdot L \cdot S_0$$ *Gl. 4.16*

Mit: L Abstand zwischen den Querprofilen 1 und 2 entlang der x-Achse
S_o Gefälle des Gerinnes, gestützt auf mittlere Gerinnehöhen
z_1, z_2 Mittlere Gerinnebetthöhe an den Querprofilen 1 und 2

Reibungskraft

$$F_f = \tau \cdot \overline{U} \cdot L$$ *Gl. 4.17*

Mit: τ Scherkraft
U Mittlerer benetzter Umfang zwischen den Querprofilen 1 und 2

$$\tau = \rho \cdot g \cdot \overline{R} \cdot \overline{S}_f$$ *Gl. 4.18*

Mit: $\overline{R}$ Mittlerer hydraulischer Radius ($R = A/U$)
$\overline{S}_f$ Energieliniengefälle

$$F_f = \rho \cdot g \cdot \frac{\overline{A}}{\overline{U}} \cdot \overline{S}_f \cdot \overline{U} \cdot L$$ *Gl. 4.19*

$$F_f = \rho \cdot g \cdot \left(\frac{A_1 + A_2}{2} \right) \cdot \bar{S}_f \cdot L \qquad \text{Gl. 4.20}$$

Beschleunigungskraft

$$m \cdot a = Q \cdot \rho \cdot \Delta v_x \qquad \text{Gl. 4.21}$$

$$\rho = \frac{\gamma}{g} \quad \text{und} \quad \Delta v_x = (\beta_1 \cdot v_1 - \beta_2 \cdot v_2) \qquad \text{Gl. 4.22}$$

$$m \cdot a = \frac{Q \cdot \gamma}{g} \cdot (\beta_1 \cdot v_1 - \beta_2 \cdot v_2) \qquad \text{Gl. 4.23}$$

Mit: β Impulskoeffizient, der eine variable Geschwindigkeitsverteilung in unregelmässigen Gerinnen berücksichtigt.

Einsetzen der einzelnen Kraftkomponenten in Gl. 4.8 und die Annahme, dass der Abfluss Q zwischen den Querprofilen 2 und 1 variieren kann, führt zu:

$$\gamma \cdot A_2 \cdot \bar{Y}_2 - \gamma \cdot A_1 \cdot \bar{Y}_1 + \gamma \cdot \left(\frac{A_1 + A_2}{2} \right) \cdot L \cdot S_0 - \gamma \cdot \left(\frac{A_1 + A_2}{2} \right) \cdot L \cdot \bar{S}_f = \frac{Q_1 \cdot \gamma}{g} \cdot \beta_1 \cdot v_1 - \frac{Q_2 \cdot \gamma}{g} \cdot \beta_2 \cdot v_2$$

$$\frac{Q_2^{\,2} \cdot \beta_2}{g \cdot A_2} + A_2 \cdot \bar{y}_2 + \left(\frac{A_1 + A_2}{2} \right) \cdot L \cdot S_0 - \left(\frac{A_1 + A_2}{2} \right) \cdot L \cdot \bar{S}_f = \frac{Q_1^{\,2} \cdot \beta_1}{g \cdot A_1} + A_1 \cdot \bar{y}_1 \qquad \text{Gl. 4.24}$$

Diese Gleichung ist die operationelle Form der Impulsgleichung. Sie findet verbreitete Anwendung, um den Wasserspiegel bei schiessendem Abflussregime zu berechnen. In Kap. 5.1.1.7 kommt sie im Gerinnehydraulikprogramm *UWSP* zum Einsatz.

4.2 Smallworld GIS

Das GIS *Smallworld* besteht aus der Entwicklungsumgebung *Magik*, einer *Smallworld*-eigenen objektorientierten Programmiersprache und einer ebenfalls objektorientierten relationalen Datenbank. Das Datenmodell wird mit einem *CASE*-Tool erstellt.

4.2.1 Das GIS

Geographisches Informationssystem (Geoinformationssystem, GIS) ist die Bezeichnung für ein raumbezogenes Informationssystem (RIS) seitens der Geographen, Forstwirte, Ökologen, Raumplaner und Demoskopen (WHATIS, 2000).

4.2.1.1 Dateneingabe und Datenmanipulation

Ein GIS ermöglicht die Verwaltung, Manipulation und Visualisierung einer Menge von geographischen Daten. Es erlaubt die Abfrage und Analyse einer relationalen Datenbank und liefert die Ergebnisse in Form einer Karte. Geographische Informationen werden explizit durch geographische Koordinaten (Länge und Breite eines nationalen Raster-Koordinatensystems) oder implizit z.B. durch Strassennamen und Postleitzahl, oder durch Kartensymbole für Wald, Weide, Wasserflächen usw. beschrieben. Ein geographisches Informationssystem beinhaltet die Fähigkeit, implizite geographische Daten in eine explizite Ortsangabe im Koordinatensystem der Karte umzuwandeln. Häufig erhält man als GIS-Entwickler Daten von

öffentlichen Stellen oder Firmen, die auf das Sammeln und Organisieren geographischer Informationen spezialisiert sind. Die Umwandlung von impliziten in explizite Daten wird *geocoding* genannt.

4.2.1.2 Vektorformate und Rasterformate

Geographische Daten können in einem Vektorformat oder in einem Rasterformat gespeichert werden. Bei einem *Vektorformat* werden zweidimensionale Daten durch (*x*,*y*)-Koordinaten gespeichert. Das Vektormodell ist geeignet, um linearisierbare Merkmale darzustellen (Abb. 4.2).
Ein *Rasterformat* beschreibt Daten als eine kontinuierliche Menge von Rasterzellen. Dieses Format ist geeignet, um kontinuierliche Änderungen, wie z.B. Bodentyp-Muster darzustellen (s. Abb. 4.2).
Die meisten geographischen Informationsysteme machen folglich Gebrauch von beiden Datenformaten.

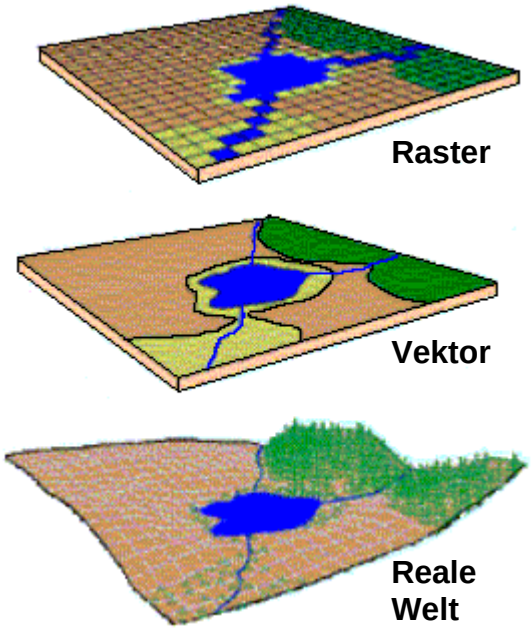


Abb. 4.2: Vektorformat und Rasterformat (nach ESRI, 1999)

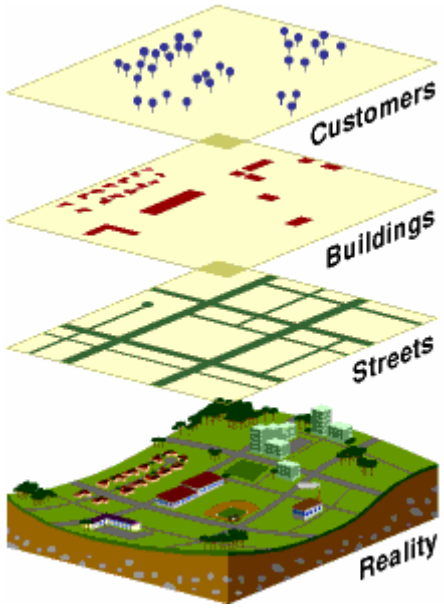


Abb. 4.3: Ebenen mit verschiedenen Daten (ESRI, 1999).

4.2.1.3 Datenanalyse

In einem GIS sind Daten für einfache Abfragen (*query*) wie komplexe Analysen zugänglich. Ein GIS speichert seine Informationen über die Welt als eine Sammlung thematischer Kategorien oder Ebenen, die über ihre geographische Position verbunden sein können (s. Abb. 4.3).
Die Integration unterschiedlicher Daten-Ebenen (*layers*) erfordert die Überlagerung (*overlay*) von Daten verschiedener Ebenen. Im einfachsten Fall könnte dies eine rein visuelle Operation sein. Analyseverfahren bauen jedoch oft darauf auf, dass eine oder mehrere Datenebenen physikalisch verknüpft werden. Diese Überlagerung oder räumliche Verknüpfung kann z.B. Daten über Boden, Hangneigung, Vegetation zusammenführen (s. Abb. 4.4).

Abb. 4.4: Überlagerung (Verschneiden) verschiedener Ebenen (ESRI, 1999).

4.2.1.4 Visualisierung

Für viele Arten von geographischen Operationen wird das Ergebnis am besten als Karte oder Diagramm präsentiert. Ein GIS erweitert diese Möglichkeiten häufig um spezielle Fähigkeiten, z.B. Report-Möglichkeiten, dreidimensionale Ansichten, bewegte Abläufe (Animationen).

Zusammenfassend kann man feststellen, dass ein GIS folgende Aufgaben umfasst (WHATIS, 2000):

- Es erhält einen geographischen Input in Form eingescannter oder digitalisierter Kartenbilder.
- Es skaliert und manipuliert diese geographische Daten.
- Es beinhaltet einen Datenbank-Manager, im Allgemeinen ein relationales Datenbank-Managementsystem (relational database management system, RDBMS).
- Es beinhaltet Abfrage- und Analyseprogramme, die einfache Fragen, z.B. nach dem Abstand zwischen zwei Punkten auf einer Karte beantworten und komplexere Analysen durchführen, z.B. das Verkehrsaufkommen an einer bestimmten Strassenkreuzung, ermitteln.
- Die Antworten werden bei Bedarf visuell präsentiert, im algemeinen als Karten oder Diagramme.

4.2.1.5 Spezielle Eigenschaften des GIS Smallworld

Im Gegensatz zu den meisten anderen GIS-Systemen arbeitet *Smallworld* nicht layer-orientiert sondern objekt-orientiert. Die graphische Oberfläche besteht zum einen aus dem GIS-Fenster, das die geometrischen Daten zeigt. Die Sicht auf die nicht-geometrischen Daten in der Datenbank erfolgt hingegen über *Editoren*. Für jede Tabelle der Datenbank kann kann ein solcher Editor geöffnet werden, der stets einen Datensatz der Tabelle anzeigt. Das Navigieren durch die Datensätze der Tabelle erfolgt durch betätigen entsprechender *Buttons* in den Editoren. Ebenso können Beziehungen zwischen den Tabellen über Buttons von den Editoren aus verfolgt werden. Für komplexe Zugriffe auf die Datenbank kann ein Abfragewerkzeug aufgerufen werden, mit dem z.B. SQL-Abfragen (*structured query language*) möglich sind (SMALLWORLD SYSTEMS, 1994b).

4.2.1.5.1 Topologie

Neben der reinen Darstellung können Geometrien in *Smallworld* topologischen Regeln unterliegen. Sie bestimmen die Interaktion zwischen den Geometrien. Je nach Regel können zum Beispiel zwei sich schneidende Linien einen Schnittpunkt erzeugen, oder sich

ungehindert überlagern. Im ersten Fall wären die Linien topologisch miteinander verknüpft, im zweiten Fall wüssten die beiden Linien nichts voneinander.
In anderen GIS-Systemen wie *ARC/INFO* (ESRI, 2000) existiert nur jeweils eine topologische Regel für die verschiedenen Geometrietypen Punkt, Linie und Fläche.
In *Smallworld* können hingegen explizite Regeln zwischen den Geometriefeldern der Objekte definiert werden. Somit können unterschiedliche topologische Interaktionen zwischen Geometrien des gleichen Typs realisiert werden. Dies ermöglicht beispielsweise die Verwaltung eines Leitungsnetz unabhängig von einem Strassennetz und einem Gewässernetz. Strassen dürfen sich z.B. untereinander schneiden, haben aber keine Verbindungspunkte zu Leitungen (SMALLWORLD SYSTEMS, 1994a)

4.2.1.5.2 Geometrische Attribute

Im Gegensatz zu gewöhnlichen Datenbanken können in *Smallworld* in den Datenbanktabellen die zugehörigen Geometriedaten über ein spezielles Konzept direkt mitverwaltet werden, so dass sich für den Nutzer die Geometrie eines Objektes als Feld wie jedes andere präsentiert. Diese Struktur ermöglicht es auch, dass in einer Objektklasse mehrere Geometriefelder existieren. So kann eine Klasse `Strasse-Eisenbahn-Kreuzung` nützlicherweise ein Punkt-Geometriefeld besitzen, dessen Topologie mit Strassen verknüpft ist, und ein zweites Punkt-Geometriefeld, dessen Topologie mit Eisenbahnstecken verbunden ist.

4.2.2 Die Geodatenbank

Eine *Datenbank* ist eine Sammlung von nicht-redundanten Daten, die von mehreren Anwendungsprogrammen benutzt werden (EBNER, 1999). Darüberhinaus werden die folgenden Begriffe verwendet:
Ein *Datenbanksystem* ist der Mittler zwischen Datenbestand und Anwendungsprogramm. Die Bezeichnung *Datenbank-Management-System* (DBMS) wird meist synonym für Datenbanksysteme verwendet. Es kann aber auch ein Programm sein, welches Zusatzaufgaben wie beispielsweise das Sichern (backup) der Daten erledigt.
Der *Datenbestand* umfasst die Daten, welche in die Datenbank geschrieben wurden.

4.2.2.1 Redundanz, Konsistenz und Integrität

Um das Funktionieren einer Datenbank zu gewährleisten müssen einige Bedingungen erfüllt sein. Dazu gehört, dass *Redundanzen* vermieden werden. Diese entstehen, wenn identische Datensätze mehrfach vorhanden sind. So etwas vermehrt nicht nur den Bedarf an Speicherplatz, es birgt auch die Gefahr der *Inkonsistenz*, wenn an nur einem Datensatz Änderungen durchgeführt werden.

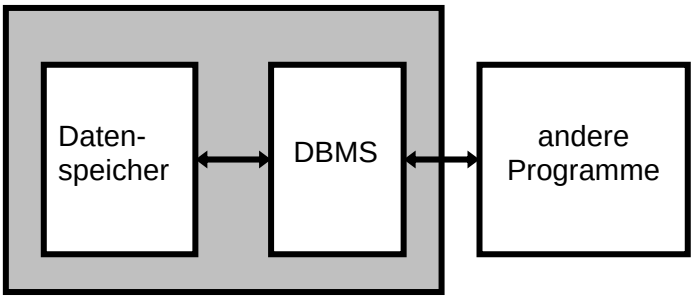


Abb. 4.5: Aufbau einer Datenbank (nach BARTELME, 1989).

Wenn Anwendungsprogramme auf einen Datenbestand direkt zugreifen können, dann müssen Massnahmen getroffen werden, dass die Daten von diesen Programmen nicht fehlerhaft

verändert werden. Dazu müssen in jedem Anwendungsprogramm Sicherungen eingebaut sein. Dies ist sowohl unökonomisch als auch fehleranfällig. Es hat sich deshalb durchgesetzt, dass die Anwendungsprogramme nicht direkt auf die Datenbestände zugreifen, sondern Aufträge an ein Datenbank-Management-System erteilen (s. Abb. 4.5). Dieses greift als einziges direkt auf die Daten zu und kann so die *Integrität* der Daten sicherstellen.

4.2.2.2 Relationale Datenbanken

Der Begriff *relationale Datenbank* geht zurück auf CODD (1970). Inzwischen sind von CODD 333 Kriterien erstellt worden, die ein Datenbank-Management-System (DBMS) erfüllen muss, damit es sich relational nennen "darf". Nach der Ansicht von EBNER (1999) erfüllt derzeit kein einziges Datenbanksystem alle 333 Kriterien. In der Praxis wird ein DBMS *relational* genannt, wenn es die wesentlichen Bedingungen erfüllt.

Eine *Relation* ist eine Tabelle. Relationale Datenbanken kann man als *auf Tabellen basierende Datenbanken* bezeichnen. Sämtliche Daten werden in Relationen, also in Tabellen gespeichert. Eine *Tabelle* ist eine logische Verbindung von einer festen Anzahl von *Attributen* (Spalten) und einer variablen Anzahl von *Tupeln* (Zeilen, Reihen). Die Tupel werden auch als *Datensätze* bezeichnet und bestehen aus einzelnen *Datenfeldern*. Jedes Datenfeld enthält genau ein *Datum* (Datenelement).

4.2.2.3 Schlüssel

Eine relationale Datenbank besteht im Allgemeinen aus vielen Tabellen. Um Verbindungen zwischen Datensätzen aus verschiedenen Tabellen herstellen zu können werden *Keys* (Schlüssel) verwendet.

Die Tabellen einer relationalen Datenbank sind *nicht* sequentiell organisiert, d.h., es besteht keine Ordnung unter den Datensätzen, dass man vom ersten, 385., letzten Datensatz sprechen könnte. Vielmehr muss man aus Sicht des Programmierers davon ausgehen, das die Daten Elemente in einer ungeordneten Menge sind. Um einen Datensatz eindeutig bestimmen zu können, ist ein *Candidate Key* (Primärschlüssel), also ein eindeutiger Schlüssel nötig, der in keinem anderen Datensatz derselben Tabelle vorkommt.

Unter einem *Primärschlüssel* versteht man eine Attribut-Menge (also eine Menge von einer oder mehreren Spalten), die eindeutig ist. So ein Primärschlüssel ist im einfachsten Fall eine fortlaufende Nummer, z.B. eine Kundennummer oder eine Bestellnummer.
Zwingend nötig ist die Verwendung einer Nummer nicht. Eine Person könnte man auch durch die Kombination mehrerer Personendaten eindeutig identifizieren - es würden allerdings viele Personendaten nötig sein. Die Kombination von Vor- und Nachname reicht bei weitem nicht. Je grösser die Datenmengen sind, desto wahrscheinlicher wird das Auftreten zweier gleicher Primärschlüssel. Die Adresse eignet sich nicht, da sie sich schnell ändern kann. Nimmt man zum Vor- und Nachnamen weitere Attribute hinzu (z.B. Geburtsdatum, Geburtsort), so ist eine Übereinstimmung zweier Personen recht unwahrscheinlich. Sollte ein solcher Fall dann aber doch einmal vorkommen, so weigert sich das Datenbanksystem schlicht, den zweiten Datensatz anzunehmen.
Darüberhinaus sind zusammengesetzte Schlüssel ungünstig, da Schlüssel zur Herstellung einer Referenz verwendet werden. Dies bedeutet, dass sie in die Tabelle aufgenommen werden, die eine Referenz beinhaltet. Wird eine Referenz nicht über eine einzelne Nummer, sondern über eine Kombination mehrerer Felder vorgenommen, so werden alle Felder in die Tabelle aufgenommen, sowohl der Speicherbedarf als auch der Suchaufwand wird dadurch grösser.

Als Fazit kann man feststellen, dass Tabellen in der Regel eine durchlaufende Nummer als Primärschlüssel haben sollten.

Gerade bei relationalen Datenbanken werden viele *Master-Detail*-Verknüpfungen (1:*n*-Verbindungen) erstellt. Ein Beispiel dafür wäre die Rechnungstabelle und die Kundentabelle einer Firma (Abb. 4.6):

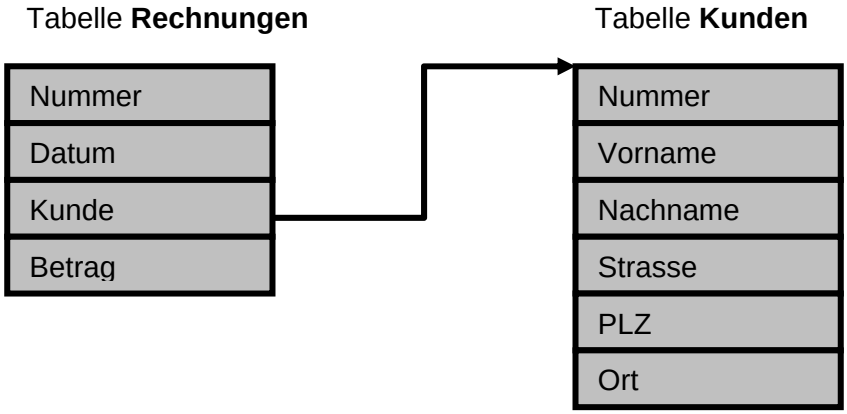


Abb. 4.6: Einsatz eines Fremdschlüssels bei einer Master-Detail-Verknüpfung (nach EBNER, 1999)

In der Rechnungstabelle finden sich Rechnungsnummer, Datum und Betrag. Vom Kunden wird nicht die komplette Adresse gespeichert, sondern nur die Kundennummer - mit dieser kann dann in der Kundentabelle die Adresse ermittelt werden.

Die Kundennummer, also das Atrribut Kunde ist ist hier folglich ein *Foreign Key* (Fremdschlüssel). Dieser verweist auf den Primärschlüssel einer anderen Tabelle und stellt so eine Verbindung zwischen zwei Tabellen her.
Unter *Normalisierung* versteht man die schrittweise Optimierung der Datenbank durch Veränderung der Tabellendefinition. Dazu werden mehrere Normalformen definiert, die Bedingungen an den Aufbau der Datenbank stellen. Nähere Angaben zur Normalisierung findet man in Anhang F.

4.2.2.4 Das Entity-Relationship-Modell

Um das Erstellen der Tabellen zu erleichtern und zu systematisieren, wurde das *Entity-Relationship-Modell* entworfen. Das Entity-Relationship-Modell gliedert die in der Datenbank abzubildende Realität in *Entities* (Objekte, die Tabellen entsprechen) und *Relationships* (Beziehungen zwischen den Objekten). Das erweiterte Entity-Relationship-Modell kennt dabei verschiedene Arten von Beziehungen, die zwischen Tabellen möglich sind. Von einer Tabelle kann dann eine der folgenden *Halbrelationen* zu einer anderen Tabelle ausgehen: 0/1, 1, 1/*n*, 0/1/*n* . Eine Relation besteht aus zwei Halbrelationen. Mögliche Relationen sind dann z.B.:

- 1:1 Eins-zu-Eins-Beziehung (z.B. ein Land mit einer Hauptsstadt)
- 1:0/1/*n* Eins-zu-Viele-Beziehung (z.B. ein Land mit vielen Städten)
- 0/1/*n*:0/1/*m* Viele-zu-Viele-Beziehung (z.B. Flüsse, die durch mehrere Städte fliessen, und Städte, durch die mehrere Flüsse fliessen)

Wird die Halbrelation 0/1 bzw. 0/1/*n* verwendet, so darf die Menge auch leer sein, wird die Halbrelation 1 bzw. 1/*n* verwendet, so muss die Menge mindestens einen Eintrag enthalten.

Diesen Kombinationsmöglichkeiten sind jeweils Lösungsvorschriften für den Aufbau einer Datenbank zugeordnet. So gibt es z.B. bei einer 1:1-Beziehung keinen Grund, verschiedene Tabellen zu verwenden, Hauptstädte sind einfach verschiedene Spalten derselben Tabelle. 1:0/1/*n*-Beziehungen führen zu Master-Detail-Tabellen; Für 0/1/*n*:0/1/*m*-Beziehungen wird auf *intermediate tables* zurückgegriffen, also auf eine Zwischentabelle, deren eine Spalte auf die erste Tabelle und deren zweite Spalte auf die zweite Tabelle verweist.

4.2.2.5 Spezielle Eigenschaften der Datenbank von Smallworld

Die Datenbank von *Smallworld* ist in die objektorientierte Struktur der *Smallworld*-eigenen Programmiersprache *Magik* eingebettet. Die einzelnen Tabellen stellen eigene Objektklassen dar, die Datensätze sind die Objekte dieser Klassen. Ihre Eigenschaften ergeben sich aus den Spalteneinträgen (Attributen) des Datensatzes. In *Smallworld* werden die Eigenschaften einer Datenbanktabelle *Felder* genannt.

Eine spezielle Fähigkeit der *Smallworld*-Datenbank besteht darin, unterschiedliche *Alternativen* (Versionen) zu verwalten. Dies erlaubt es insbesondere in grösseren Projekten mit mehreren Mitarbeitern, dass diese gleichzeitig jeweils ihre Alternative weiterentwickeln und später zusammenführen (*merge*). Ein *Checkpoint*-Mechanismus ermöglicht es, den Zustand der Datenbank auf einen früheren Kontrollpunkt zurückzusetzen (*roll-back*), falls z.B. Änderungen verworfen werden sollen.

4.2.3 Die Programmiersprache

4.2.3.1 Das Objektorientierte Paradigma

Dem lange Zeit dominierenden *imperativen* Programmierstil haftet das grundsätzliche Makel an, dass globale Variablen potentiell in jedem Teil des Programms inspiziert werden können. Grosse Programme ohne jede Disziplin im Zugriff auf globale Variablen werden leicht unübersichlich und somit zwangsläufig fehlerträchtig. Das liegt daran, dass kein Modul (Komponente), das globale Variablen benutzt, unabhängig von anderen Modulen entwickelt und verstanden werden kann, die diese Variablen auch benutzen (WATT, 1996). Es gibt keine ausreichende Trennung zwischen einzelnen Komponenten, so dass das Programm als ganzes eine amorphe Struktur besitzt und ab einer gewissen Grösse die Einflüsse der einzelnen Programmteile aufeinander kaum noch nachvollziehbar sind.
Dieses Problem wurde Anfang der 70er Jahren von D. PARNAS (PARNAS, 1972) erkannt, der die Disziplin der *Schnittstellen* (*information hiding*) als Ausweg empfahl. Dieses Konzept wurde im Laufe der Zeit erweitert und man spricht heute vom *objektorientierten Paradigma*.

Als modernes Konzept der Programmentwicklung basiert die objektorientierte Programmierung (*object-oriented programming, OOP*) auf Objekten statt auf Aktionen, mehr auf Daten als auf logischen Abläufen.
Historisch wurde ein Programm als eine logische Prozedur angesehen, die eine Menge von Eingabedaten entgegennimmt, diese verarbeitet und eine Ergebnismenge zurückliefert. Der Schwerpunkt der Entwicklung wurde darin gesehen, den logischen Ablauf eines Programms zu beschreiben, weniger in der Definition der Datenstrukturen. Die objektorientierte Programmierung hat sich die Sichtweise zu eigen gemacht dass das, was uns interessiert eher die *Daten* sind als die Logik, mit deren Hilfe diese Daten manipuliert werden. Beispiele von Objekten reichen von Gebäude (beschrieben durch Grundfläche, Zahl der Zimmer, Besitzer) über Personen (beschrieben durch Name, Adresse, usw.) bis hin zu kleinen graphischen Elementen auf der Bildschirmoberfläche (Schalter, Rollbalken, Fenster) eines Computers.
Der erste Schritt in der *OOP* besteht darin, alle Objekte zu identifizieren und zu benennen, die manipuliert werden sollen, und deren Verbindung zu anderen Objekten zu erfassen. Dieser

Schritt wird oft als *data modelling* bezeichnet. Nachdem ein Objekt erfasst wurde, wird es als *Klasse* von Objekten generalisiert (man denke hier an PLATO'S idealen Stuhl, der für alle Stühle steht). Nun wird die Art der Daten definiert, die diese Klasse enthält und es werden alle logischen Aktionen angegeben, die diese Klasse ausführen kann. Die logischen Aktionen werden als *Methoden* bezeichnet. Eine reale *Instanz* (Vertreter, Inkarnation) einer Klasse wird folglich *Objekt* genannt. Dieses Objekt kann über wohldefinierte Schnittstellen, die Nachrichten (*messages*) genannt werden, mit anderen Objekten und mit dem Benutzter kommunizieren. Eine Nachricht hat beim Empfängerobjekt die Ausführung einer Methode zur Folge.

Die Konzepte und Regeln der objektorientierten Programmierung bieten die folgenden wichtigen Vorteile (WHATIS, 2000):

- Das Konzept der Datenklassen erlaubt die Definition von Unterklassen (*subclass*), die Daten von der Oberklasse (Superklasse) erben und erweitern. Dieses Prinzip wird Vererbung (*inheritance*) genannt.
- Da ein Objekt prinzipiell nur Zugriff auf die eigenen Daten hat und mit anderen Objekten nur über Nachrichten kommunizieren kann, ist ein versehentliches Ändern der Daten anderer Objekte nicht möglich. Dieses Prinzip des *data hiding* führt zu grösseren Systemsicherheit und -stabilität.
- Die Definition einer Klasse ist nicht nur in dem Programm wiederverwendbar, in dem sie erstellt wurde, sondern auch durch andere objektorientierte Programme. Das *data hiding* veranlasst somit nicht nur eine grössere Sicherheit sondern auch eine möglichst lose Kopplung zwischen einzelnen Komponenten (den Klassen), wodurch die Wiederverwendung, der Austausch einer Komponente durch eine andere, die verteilte Anwendung in einem Netzwerk und, soziologisch gesprochen, eine dezentrale Organisation gefördert wird.

Eine der ersten objekt-orientierten Programmiersprachen war *Smalltalk* (GOLDBERG & ROBSON, 1983). In letzter Zeit sind vor allem *C++* (STROUSTRUP, 1986) und *Java* (SUN, 2000) sehr populär geworden.

4.2.3.2 Spezielle Eigenschaften der Programmiersprache Magik

Das Konzept des GIS *Smallworld* sieht *Magik* als eine interaktive Programmiersprache vor, die alleinig alle Aufgaben der Implementierung eines GIS als grosses interaktives System übernimmt. *Magik* bildet die Entwicklungsumgebung sowohl für das grundlegende GIS-System als auch für GIS-Benutzeranwendungen wie *LIWIS* und *OGRUT*. Über *Magik* ist das Datenbankverwaltungssystem implementiert, und mit *Magik* werden graphische Oberflächen erstellt.

4.2.3.2.1 Externe Programme

Smallworld ist in der Lage, in anderen Sprachen geschriebene Programme, die als ausführbare Dateien vorliegen, über die Systemumgebung auszuführen. Mit solchen Programmen können (zumindest unter Windows) Daten nur über externe Dateien ausgetauscht werden.

4.2.3.2.2 Alien co-processor

Über sogenannte *alien co-processors* (ACP) lassen sich externe Programme aber auch direkt in *Smallworld* integrieren. Dazu ist es nötig, dass die Programme im Quellcode vorliegen, da einige Änderungen vorgenommen werden müssen. Im wesentlichen betrifft das die Art, wie Daten entgegen genommen und zurückgeliefert werden, die eigentliche Datenverarbeitung im externen Programm ist nicht betroffen. Anders als beim Ausführen externer Programme über

die Systemumgebung findet die Datenübermittlung zwischen *Smallworld* und dem über ACP angebundenen Programm direkt über Funktionsaufrufe statt.

4.2.3.2.3 Multithreading

Threads sind vergleichbar mit Prozessen, die einem ausführbaren Programm entsprechen, sie sind jedoch leichtgewichtiger. *Multithreading* ist eine Technologie die es erlaubt, mehrere Operationen parallel auszuführen. *Magik* ist eine Programmiersprache, die es ermöglicht, parallele Threads zu erzeugen und so unterschiedliche Berechnungen gleichzeitig durchzuführen. Von dieser Fähigkeit wird auch bei der Umsetzung in dieser Diplomarbeit Gebrauch gemacht. Prinzipiell erlaubt Multithreading eine Beschleunigung der Berechnung von Algorithmen, die *parallelisierbar* sind. Voraussetzung für einen sinnvollen Einsatz von Threads ist auch entsprechend leistungsfähige Hardware mit mehreren Prozessoren. Aber auch wenn nur ein Prozessor zur Verfügung steht, können Threads sinnvoll sein, da sie z.B. die Aufteilung eines Programmlaufs in Hintergrundthreads niedriger Priorität und einen Vordergrundthread hoher Priorität erlauben. So ist es möglich, den Befehl zum Verarbeiten einer bestimmten Datenmenge zu starten und parallel weiterzuarbeiten, z.B. um andere Daten abzurufen (SMALLWORLD SYSTEMS, 1999).

4.2.4 Das CASE-Tool

Heute werden in der Programmentwicklung vielfach Werzeuge eingesetzt, die die Erstellung von Anwendungen erleichtern. Solche Werkzeuge nennen sich *CASE-Tools* (*Computer-Aided Software Engineering*). CASE-Tools sind definierte Programmierregeln um Entwicklungsprinzipien, -methoden, -techniken und -konzepte umzusetzen. Diese Werkzeuge unterstützten die Entwicklung in einer definierten Phase eines Softwareprojektes, indem sie manuelle Handlungen durch strukturiertes Prototyping automatisieren. Diese Technik verringert Entwicklungszeit und sichert die Konsistenz des erzeugten Programmcodes (ESRI, 1999).
CASE basiert auf der Verwendung von computerunterstützten Methoden, um die Entwicklung von Software zu organisieren und zu kontrollieren. Insbesondere bei grossen, komplexen Projekten in denen eine Vielzahl von Software-Komponenten und Personen zum Einsatz kommen macht sich diese Technik bezahlt. Die Verwendung von CASE erlaubt es allen am Entwicklungsprozess beteiligten Personen vom Programmierer bis zum Manager, eine gemeinsame Sicht auf den Entwicklungsstand des Projektes zu erhalten. CASE unterstützt einen disziplinierten Entwicklungsprozess, der durch Kontrollpunkte (*checkpoints*) und Versionsverwaltung abgesichert ist. Der Fortgang eines Projektes kann graphisch dargestellt werden.
Die Entwicklung von CASE begann in den 70er Jahren, als versucht wurde, Methoden bei der Hardwareproduktion auf die Softwareproduktion zu übertragen, die als ungenügend entwickelte Disziplin angesehen wurde. Einige CASE-Tools unterstützten das Konzept der *strukturierten Programmierung* und ähnlich organisierte Entwicklungsmethoden. In letzter Zeit wurde es zur Aufgabe der CASE-Tools gemacht, die visuelle Programmierung und die *objektorientierte Programmierung* zu ermöglichen.
Zusammenfassend lässt sich sagen, dass ein CASE-Tool als Werkzeug angesehen wird, das in Verbindung mit anderen Werkzeugen die Qualität eines Entwicklungsprozesses sicherstellt, wie er z.B. im ISO 9000-Standard beschrieben ist (ISO, 1987). Da der Entwicklungsprozess Testen und Änderung der Anforderungen mit umfasst, können die Kosten für die Wartung eines Softwareproduktes über dessen Lebenzeit gerechnet deutlich gesenkt werden. Eine organisierte Vorgehensweise bei der Entwicklung unterstützt die Wiederverwendung von Programmcode, verringert die Kosten und erhöht die Qualität (WHATIS, 2000).

4.3 Fazit

Die *Gerinnehydraulik* bietet eine Reihe von mathematischen Lösungen, die in ihrer Komplexität sehr unterschiedlich sind. Je genauer man die Fliessverhältnisse erfassen möchte, desto höher ist der Aufwand. Dies gilt sowohl für den stationären als auch für den instationären Fall. Während bei zeitinvarianten Berechnungen im wesentlichen die MANNING-STRICKLER-Gleichung und die DARCY-WEISBACH-Gleichung zur Verfügung stehen, hat man im instationären Wellenablauf die Wahl zwischen einfachen Methoden wie dem MUSKINGUM-Verfahren und aufwendigen Gleichungssystemen wie den SAINT-VENANT-Gleichungen. Bei der Wahl eines Verfahrens ist daher genau abzuwägen, in welchem Zusammenhang und mit welcher Genauigkeitsanforderung die Berechnungen durchgeführt werden sollen. Bei der im Verlauf dieser Arbeit implementierten stationären, ungleichförmigen Wasserspiegelberechnung muss festgestellt werden, dass nur wenig Literatur den Fall des superkritischen (schiessenden) Abflusses berücksichtigt. Dies hat die Wahl des verwendeten Modells eingeschränkt. Ein weiteres Problem ist die Berechnung eines gemischten Abflussregimes. Im strömenden Fall muss die Iteration stromaufwärts erfolgen, im schiessenden Fall stromabwärts. Dadurch entstehen Lücken in der Berechnung, die durch geeignete Verfahren überbrückt werden müssen.

Geographische Informationsysteme haben heute das Potential, auf integrative Weise den gesamten Datenvorrat eine Projektes zu verwalten, zu präsentieren und zu bearbeiten. Die Steuerung externer Programme vom GIS aus stellt grundsätzlich kein Problem dar. Durch einheitliche Datenstrukturen und Bearbeitungsmechanismen wird ein hohes Mass an Konsistenz und Verfügbarkeit erreicht. Das Erstellen von Anwendungen, die auf eine Vielzahl unterschiedlicher Daten zugreifen und umfangreiche Ergebnisse produzieren wird durch das zentrale Management und durch mächtige Programmierwerkzeuge sehr vereinfacht. Zur Definition neuer Datenstrukturen können CASE-Tools eine grosse Hilfe leisten, indem sie Fehler vermeiden helfen und die Übersichtlichekeit über den Stand des Projektes steigern.
Smallworld hat gegenüber anderen GIS-Systemen einige Vorteile aufzuweisen, die es als integratives Managementssystem für Daten und Werzeuge geeignet machen. Alle nötigen Vorrausetzungen sind vorhanden, einige zusätzliche Fähigkeiten erleichtern die Arbeit bzw. machen sie produktiver. Erwähnt sei insbesondere noch einmal die vollwertige, objektorientierte Programmiersprache, die über die Skriptsprachen anderer Systeme hinausgeht. Auch das Konzept der Versionsverwaltung ist ein besonderer Schwerpunkt von *Smallworld*.

5 *UWSP*

UWSP ist ein selbstentwickeltes Programm zur Berechnung stationärer, ungleichförmiger Wasserspiegellinien. Ursprünglich war vorgesehen, für die Berechnung der Wasserspiegellagen das Programm *Hydrologic Engineering Center's River Analysis System* (*HEC-RAS*, in: BRUNNER, 1998) einzusetzen. Dieses hat die Fähigkeit, eindimensionale stationäre ungleichförmige Wasserspiegelberechnungen durchzuführen. Das *HEC-RAS* Modellierungssystem wurde als Teil des ingenieurhydraulischen Softwareprojekts *NextGen* des *US Hydraulic Engineers Center* entwickelt. Dieses Projekt umfasst verschiedene Aspekte der Ingenieurshydraulik, wie Niederschlag-Abfluss-Analyse, Gerinnehydraulik, Speichersimulation, Flutschadensanalyse und Echtzeit-Abflussvorhersage. *HEC-RAS* verfügt über eine graphische Benutzeroberfläche, separaten Komponenten für die Abflussanalyse, ein Datenmanagement, graphische Darstellungs- und Berichtmöglichkeiten. *HEC-RAS* kann eindimensionale hydraulische Berechnungen für ein ganzes Netzwerk natürlicher und künstlicher Gerinne ausführen.

Der Einsatz von *HEC-RAS* wurde verworfen als sich herausstellte, dass es nicht in der Lage ist, Zeitreihen im gewünschten Umfang zu berechnen. Ein Batch-Betrieb, bei dem die Eingabedatein automatisch generiert, das Programm dann automatisch gestartet und die Ausgabedateien wiederum zurückgelesen worden wären, liess sich nicht realisieren, da die Ausgabedateien von *HEC-RAS* in einem undokumentierten Binärformat erzeugt werden.

Neben *HEC-RAS* wurde noch andere Literatur zur Gerinnehydraulik auf die Eignung für eine direkte praktische Umsetzung untersucht (z.B. NAUDASCHER 1987). In BRETSCHNEIDER & SCHULZ (1982) wurde von Seiten des DVWK (*Deutsche Vereinigung für Wasserwirtschaft, Abwasser und Abfall*) ein erster Schritt auf dem Weg zu einem einheitlichen Berechnungsverfahren unternommen. Erwähnenswert ist vor allem die Arbeit von ROUVÉ (1987), in der die Fliesswiderstände in Gerinnen mit durchströmter Ufer- und Vorlandvegetation untersucht werden. In DVWK (1991) findet die Berechnung einer *Trennflächenrauhigkeit* Eingang. Dafür werden zwei Verfahren, nach MERTENS (1989) und nach PASCHE (1984) angegeben. Im Gegensatz zu *HEC-RAS* wird hier die Abflussberechnung nicht nach der Gleichung von MANNING-STRICKLER sondern nach dem Fliessgesetz von DARCY-WEISBACH empfohlen. Eine ähnliche Form der in *UWSP* verwendeten Energiegleichung ist in NAUDASCHER (1956) zu finden. Dort wird ausserdem gezeigt, dass diese Formel auch bei Fliessgewässern mit stark veränderlichen Rauhigkeitsbeiwerten über die Profilbreite ausreichend genau ist, wenn der Gewässerquerschnitt in Teilbereiche eingeteilt wird und der Wasserspiegel horizontal ist. Die in DVWK (1991) angegebene Methode zur Berechnung der Wasserspiegellinien kam nicht zum Einsatz, da lediglich die Berechung des *strömenden* Abflussregimes beschrieben wurde.

5.1 Hydraulik des Programms UWSP

Aufgrund der Einschränkungen von *HEC-RAS* wurde die Entscheidung getroffen, ein eigenständiges Wasserspiegelprogramm zu implementieren, das sich auf die in *HEC-RAS* verwendete Hydraulik stützt. Dabei wurde *HEC-RAS* nicht in vollem Umfang nachgebaut, sondern lediglich der für die vorliegende Arbeit relevante Kern der Funktionalität von *HEC-RAS* nachgebildet.

Als Ergebnis dieser Arbeit entstand ein neues Wasserspiegelprogramm, *UWSP* (*Ungleichförmiges Wasserspiegelprogramm*), das im folgenden beschrieben wird. Die grundlegenden Begriffe und Gleichungen werden dargestellt und Lösungsmuster für verschiedene

Gleichungen werden gezeigt. Es wird diskutiert, wie man die Gleichungen anwenden sollte und wo die Grenzen ihrer Anwendung liegen.

Das in dieser Arbeit erstellte Wasserspiegelprogramm *UWSP* greift im wesentlichen auf die Algorithmen und Formeln zurück, die *HEC-RAS* verwendet. An einigen Stellen wird von der Vorgehensweise bei *HEC-RAS* abgewichen. Solche Abweichungen betreffen im wesentlichen Vereinfachungen und Funktionalitäten, die für diese Arbeit keine Bedeutung haben (z.B. Einfluss von Eisstau auf den Wasserstand). Die teilweise einfachere Fragestellung dieser Arbeit wird allein schon dadurch nötig, dass *UWSP* automatisch von *WAQIS* aus parametrisiert und aufgerufen werden soll.
Weil zur Zeit noch keine umfangreichen Daten über Sonderprofile (Brückenprofile, Wehre, Schütze) vorliegen, wird für den Augenblick auch davon abgesehen, deren spezielle hydraulische Eigenschaften zu berücksichtigen. Bei der Implementation des Programms stand aber durchaus das Ziel vor Augen, eine spätere Erweiterung und Ergänzung soweit wie möglich zu unterstützen. Unter anderem deshalb wurde als Programmiersprache *C++* gewählt, da diese objektorientierte Sprache die Modularität und Erweiterbarkeit fördert. So ist es z.B. keine grosse Schwierigkeit, zusätzlich zu den gegenwärtig vorhandenen einfachen Querprofilen (einfache Gerinnequerschnitte) auch weitere, davon abgeleitete Profilklassen zu zu entwickeln, die lediglich als Unterklassen der Basisklasse `CrossSection` definiert werden müssten.

5.1.1 Stationäre Wasserspiegellinienberechnung

UWSP ist zur Zeit in der Lage, eindimensionale Wasserspiegellinienberechnungen für gleichförmigen Abfluss (*steady flow*) in natürlichen und künstlichen Gerinnen durchzuführen. Subkritischer (strömender), kritischer und überkritischer (superkritischer, schiessender) Abfluss kann berechnet werden. *UWSP* berechnet ein gemischtes Abflussregime und entscheidet aufgrund der in *HEC-RAS* angegebenen Kriterien, welches Abflussregime wirksam ist (*kontrolliert*).

Grundlage der Berechnung ist im Kapitel 4.1.2 diskutierte BERNOULLI-Gleichung. Diese Gleichung wurde in der Form

$$z + \frac{p}{\rho \cdot g} + \frac{v^2}{2 \cdot g} = \text{konstant} . \qquad \textit{Gl. 5.1}$$

angegeben. Gibt man der Druckhöhe $\frac{p}{\rho \cdot g}$ den Namen y, fügt ausserdem eine Konstante α für die Gewichtung der über den Querschnitt eines Querprofils variierenden Geschwindigkeiten hinzu, so erhält man

$$z + y + \frac{\alpha \cdot v^2}{2 \cdot g} = \text{konstant} . \qquad \textit{Gl. 5.2}$$

Wasserspiegelberechnungen werden von einem Querprofil (*cross section*) zum nächsten vorgenommen, indem die Energiegleichung iterativ gelöst wird. Man vergleicht also die Energiebilanz zwischen zwei Querprofilen, die in der Summe gleich sein muss. Abgezgen werden muss jedoch ein Energiehöhenverlust h_e durch Reibung auf der Fliesstrecke zwischen den beiden betrachteten Querprofilen. Die Gleichung die geeignet ist, die Energiebilanz zwischen zwei aufeinanderfolgenden Querprofilen zu beschreiben und von *UWSP* verwendete wird lautet dann:

$$y_2 + z_2 + \frac{\alpha_2 \cdot v_2^2}{2 \cdot g} = y_1 + z_1 + \frac{\alpha_1 \cdot v_1^2}{2 \cdot g} + h_e \qquad \textit{Gl. 5.3}$$

Mit:
- y_1, y_2 Wassertiefe an den Querprofilen
- z_1, z_2 Geländehöhe der tiefsten Punkte des Hauptgerinnes
- v_1, v_2 mittlere Geschwindigkeiten (Gesamtabfluss / gesamter Fliessquerschnitt)
- α_1 , α_2 Geschwindigkeitsgewichtungs-Koeffizienten
- g Erdbeschleunigung
- h_e Energiehöhenverlust

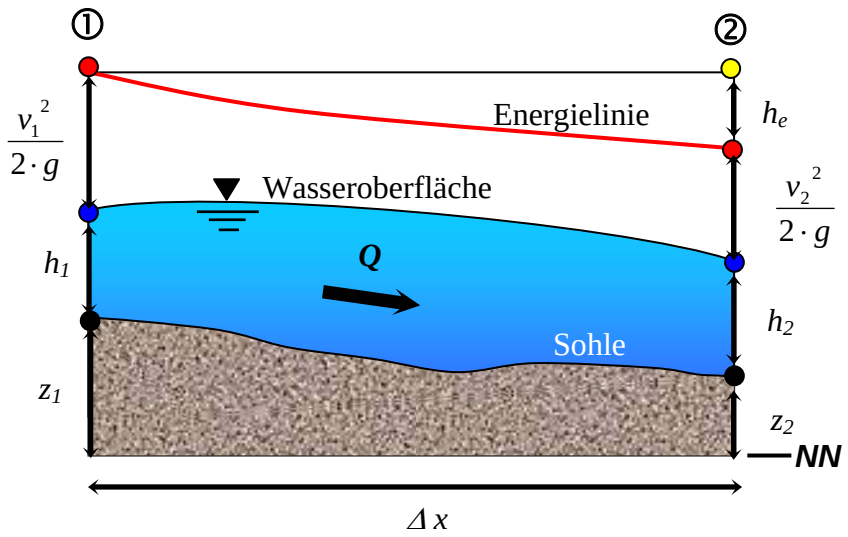


Abb. 5.1: Die Terme der Energiegleichung (nach DVWK, 1991)

Die Abb. 5.1 zeigt die einzelnen Terme der Energiegleichung für zwei aufeinanderfolgende Querprofile.
Der Energiehöhenverlust h_e (*energy head loss*) zwischen zwei Querprofilen setzt sich zusammen aus Reibungsverlusten (*friction losses*) einerseits und örtlichen Kontraktions- und Expansionsverlusten andererseits. Die Gleichung für den Energiehöhenverlust ist:

$$h_e = L \cdot \overline{S}_f + C \cdot \left| \frac{\alpha_2 \cdot v_2^2}{2 \cdot g} - \frac{\alpha_1 \cdot v_1^2}{2 \cdot g} \right| \qquad \textit{Gl. 5.4}$$

Mit:
- L Abfluss-gewichtete Länge des Flussabschnittes
- S_f Repräsentative Energieliniengefälle zwischen zwei Querprofilen
- C Expansions- oder Kontraktionsverlust- Koeffizient

Die Abfluss-gewichtete Länge des Flussabschnittes L wird wie folgt berechnet:

$$L = \frac{L_{lob} \cdot \overline{Q}_{lob} + L_{ch} \cdot \overline{Q}_{ch} + L_{rob} \cdot \overline{Q}_{rob}}{\overline{Q}_{lob} + \overline{Q}_{ch} + \overline{Q}_{rob}} \qquad \textit{Gl. 5.5}$$

Mit: L_{lob} , L_{ch} , L_{rob} Längen der Flussabschnitte zwischen zwei Querprofilen, jeweils für den Abfluss auf dem linken Vorland (*overbank*), Haupt gerinne (*main channel*) und rechtes Vorland

Q_{lob} , Q_{ch} , Q_{rob} Arithmetisches Mittel des Abflusses durch Querprofile, jeweils des linken Vorlandes, des Hauptgerinnes und des rechten Vorlandes

5.1.1.1 Unterteilung eines Querprofils zur Berechnung der Transportleistung

Die Bestimmung der Transportleistung (*conveyance*) und des Geschwindkeits-Gewichtungs-koeffizienten α für ein Querprofil macht es nötig, dass der Wasserstrom in Abschnitte unterteilt wird, in dem die Geschwindigkeit gleichmässig verteilt ist. Der Ansatz von *UWSP* besteht darin, die Strömungsabschnitte in Ufersegmente (*overbank*) aufzuteilen, wobei die Änderung des STRICKLER-Beiwerts k_{St} zwischen zwei Stationen als Trennungsstellen verwendet werden (s. Abb. 5.2).

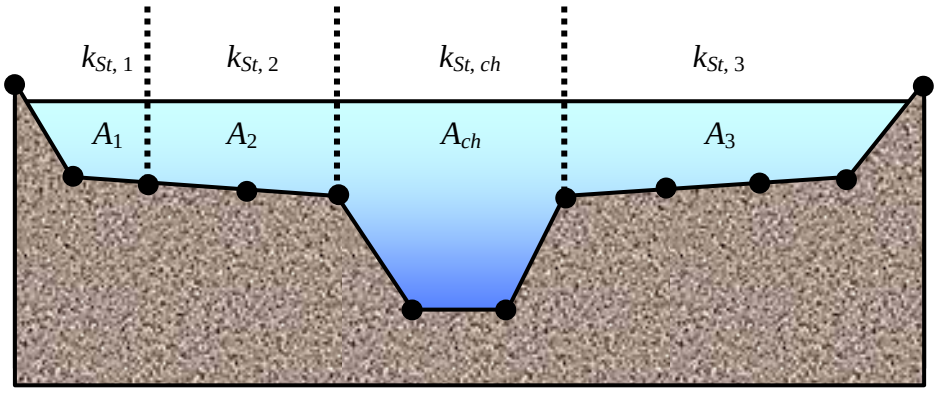
Die Transportleistung wird für jeden Unterabschnitt mit der MANNING-STRICKLER-Gleichung berechnet:

$$Q = K \cdot \sqrt{S_f} \qquad \textit{Gl. 5.6}$$

$$K = k_{St} \cdot A \cdot R^{2/3} \qquad \textit{Gl. 5.7}$$

Mit: K Transportleistung für den Unterabschnitt
k_{St} STRICKLER-Rauhigkeitsbeiwert für den Unterabschnitt
A Durchflussquerschnitt für den Unterabschnitt
R Hydraulischer Radius für den Unterabschnitt

Das Programm summiert alle Transportleistungen K_i in den Uferabschnitten auf, um die Transportleistung für die linke und die rechte Uferzone zu erhalten. Die Transportleistung des Flusschlauches wird als einzelnes Element berechnet. Die Gesamttransportleistung für das Querprofil erhält man als Summe der Transportleistungen der drei Unterabschnitte (linkes Vorland, Flusschlauch, rechtes Vorland).



$K_{lob} = K_1 + K_2$ K_{ch} $K_{rob} = K_3$

Abb. 5.2: Unterteilung von UWSP zur Berechnung der Transportleistung (nach BRUNNER, 1998).

Eine alternative Methode, die in *HEC-RAS* zur Verfügung steht und von *UWSP* nicht genutzt wird, berechnet die Transportleistung zwischen allen Koordinatenpunkten im Vorland. Die Transportleistung wird dann wieder summiert um die Gesamt-Transportleistung für das linke und das rechte Vorland zu erhalten. Die beiden Berechnungsmethoden führen immer dann zu unterschiedlichen Ergebnissen, wenn Teile des Vorlandes Abschnitte mit steilen Hängen enthalten. Die erste Methode liefert eine niedrigere Transportleistung bei gegebener Wasserspiegelhöhe. Um die Signifikanz im Unterschied des Ergebnisses aus den beiden Berechnungsarten zu testen, wurden Vergleiche durchgeführt (HEC, 1986). Die Ergebnisse dieser Untersuchung zeigen nicht, welche Berechnungsmethode die genauere ist, sie zeigen lediglich die Unterschiede. BRUNNER (1998) nimmt jedoch an, dass die von *UWSP* verwendete Methode der MANNING-STRICKLER-Gleichung dem Konzept separater Flusselemente angemessener ist.

5.1.1.2 Berechnung der mittleren Geschwindigkeitshöhe

Da *UWSP* ein Programm zur Berechnung *eindimensionaler* Wasserspiegellinien ist, wird nur eine einzige Wasserstandshöhe und somit nur eine einzige, mittlere Energie an jedem Querprofil berechnet.
Für einen gegebenen Wasserstand wird die mittlere kinetische Energie berechnet, indem die kinetische Energie aus den drei Abschnitten eines Querprofils (linkes Vorland, Flusschlauch, rechtes Vorland) mit dem jeweiligen Abfluss gewichtet wird. Abb. 5.3 zeigt, wie die mittlere kinetische Energie für ein Querprofil mit einem Flusschlauch und einem rechten Vorland (kein linkes Vorland) berechnet würde.

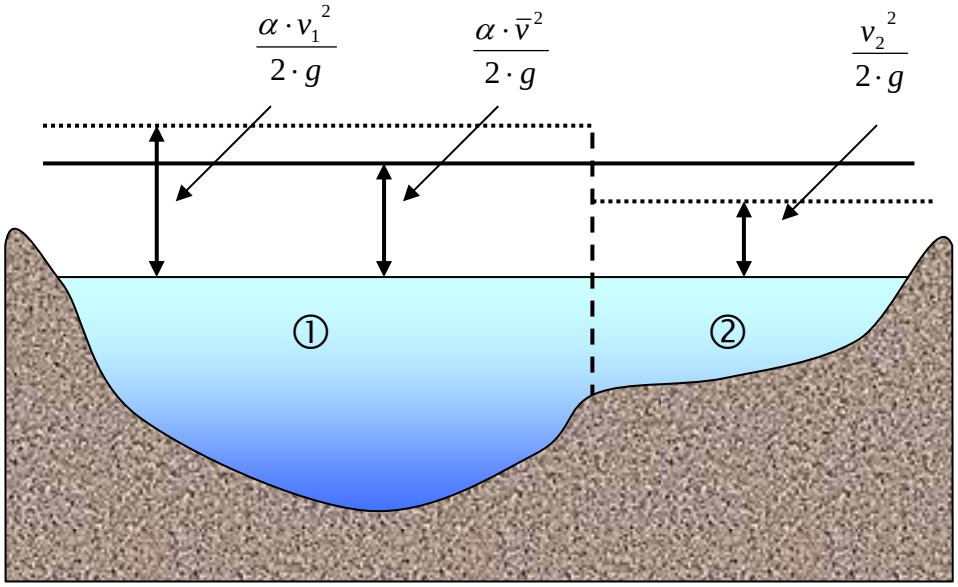


Abb. 5.3: Beispiel, wie die mittlere kinetische Energie berechnet wird (nach BRUNNER, 1998).

v_1 mittlere Geschwindigkeit für Unterbereich 1
v_2 mittlere Geschwindigkeit für Unterbereich 2
α Geschwindigkeits-Gewichtungskoeffizient

Um die mittlere kinetische Energie zu bestimmen, ist es nötig, den Koeffizienten zur Gewichtung der Geschwindigkeitshöhe α zu bestimmen. α berechnet sich wie folgt:

Mittlere kinetische Energiehöhe = Abflussgewichtete Geschwindigkeitshöhe

$$\alpha \cdot \frac{\bar{v}^2}{2 \cdot g} = \frac{Q_1 \cdot \left(\frac{v_1^2}{2 \cdot g} \right) + Q_2 \cdot \left(\frac{v_2^2}{2 \cdot g} \right)}{Q_1 + Q_2} \qquad \textit{Gl. 5.8}$$

Allgemein:

$$\alpha = \frac{\left(Q_1 \cdot v_1^2 + Q_2 \cdot v_2^2 + \ldots + Q_n \cdot v_n^2 \right)}{Q \cdot \bar{v}^2} = \frac{\sum_{i=1}^{n} Q_i \cdot v_i^{\,2}}{Q \cdot \bar{v}^2} \qquad \textit{Gl. 5.9}$$

Der Geschwindigkeitskoeffizient α wird aus der Transportleistung in den drei Abflusselementen linkes Vorland, Flusschlauch und rechtes Vorland berechnet. Dies kann auch durch Transportleistung K und Fläche A ausgedrückt werden wie in der folgenden Gleichung:

$$\alpha = \frac{A_t^{\,2} \cdot \left(\frac{K_{lob}^{\;3}}{A_{lob}^{\;2}} + \frac{K_{ch}^{\;3}}{A_{ch}^{\;2}} + \frac{K_{rob}^{\;3}}{A_{rob}^{\;2}} \right)}{K_t^{\;3}} \qquad \textit{Gl. 5.10}$$

Mit:	A_t	Gesamtdurchflussfläche des Querprofils
	A_{lob}, A_{ch}, A_{rob}	Durchflussfläche des linken Vorlandes, Flusschlauches und rechten Vorlandes
	K_t	Gesamttransportleistung des Querprofils
	K_{lob}, K_{ch}, K_{rob}	Transportleistung des linken Vorlandes, Flusschlauches und rechten Vorlandes

5.1.1.3 Berechnung der Reibungsverluste

Die Reibungsverluste werden in *UWSP* als Produkt von $\bar{S}_f$ und L berechnet (Gl. 5.4), wobei $\bar{S}_f$ das *repräsentative* Energieliniengefälle in einem Abschnitt ist und L durch Gl. 5.5 gegeben ist. Das Energieliniengefälle (Neigung der Energielinie) wird für jedes Querprofil mit der MANNING-STRICKLER-Formel wie folgt berechnet:

$$S_f = \left(\frac{Q}{K} \right)^2 \qquad \textit{Gl. 5.11}$$

Für das repräsentative Energieliniengefälle $\bar{S}_f$ wird die folgende Gleichungen verwendet:

$$\bar{S}_f = \left(\frac{Q_1 + Q_2}{K_1 + K_2} \right)^2 \qquad \textit{Gl. 5.12}$$

5.1.1.4 Berechnung der Kontraktions- und Expansionverluste

Kontraktions- und Expansionsverluste werden in *UWSP* mit der folgenden Gleichung berechnet.

$$h_0 = C \cdot \left| \frac{\alpha_1 \cdot v_1^{\,2}}{2 \cdot g} - \frac{\alpha_2 \cdot v_2^{\,2}}{2 \cdot g} \right| \qquad \textit{Gl. 5.13}$$

Mit:	C	Kontraktions- oder Expansionskoeffizient

Das Programm geht davon aus an, dass Kontraktion immer dann stattfindet, wenn die Geschwindigkeitshöhe stromabwärts grösser als die Geschwindigkeitshöhe stromaufwärts ist. Umgekehrt geht dass Programm davon aus, dass Expansion eintritt, wenn die Geschwindigkeitshöhe stromaufwärts höher als stromabwärts ist. Typische Werte für den Koeffizienten C

findet man in BRUNNER (1998). Typische Werte für Kontraktions- und Expansionskoeffizienten werden in Tab. 5.1 dargestellt.

Tab. 5.1: Kontraktions- und Expansionskoeffizient C für subkritischen Abfluss (aus BRUNNER, 1998) .

	Kontraktion	**Expansion**
Kein Übergangsverlust	0.0	0.0
Graduelle Übergänge	0.1	0.3
Typische Brückenprofile	0.3	0.5
Abrupte Übergänge	0.6	0.8

Der maximale Wert für den Kontraktions- und Expansionskoeffizienten ist 1.0 . Im Allgemeinen sollten die empirischen Kontraktions- und Expansionskoeffizienten bei strömendem Abfluss kleiner sein. Bei schiessendem Abfluss sind die Geschwindigkeitshöhen viel grösser, und kleine Änderungen der Wassertiefe können zu starken Änderungen der Geschwindigkeitshöhe führen. Verwendet man hier Kontraktions- und Expansionskoeffizienten, die für strömenden Abfluss typisch sind, kann dies zu einer Überschätzung der Energieverluste und Oszillationen der berechneten Wasserspiegelhöhe führen (BRUNNER, 1998).

5.1.1.5 Berechnungsvorgang

Der gesuchte Wasserstand an einem Querprofil wird iterativ mit Gl. 5.3 und Gl. 5.4 berechnet. Der Berechnungsablauf ist folgender (s. Abb. 5.4):

1. Schätze einen Wasserstand WS_2 am oberstromigen Querprofil (oder am unterstromigen Querprofil, wenn ein überkritisches Regime berechnet wird).
2. Berechne auf Basis des angenommenen Wasserstandes die zugehörige Gesamttransportleistung und die Geschwindigkeitshöhe.
3. Berechne mit den Ergebnissen aus Schritt 2 den Wert von S_f und Löse Gl. 5.4 für h_e.
4. Berechne mit den Ergebnissen aus Schritt 2 und 3 die Gl. 5.3 für den Wasserstand WS_2.
5. Vergleiche den berechneten Wert für WS_2 mit dem geschätzten Wert aus Schritt 1. Wiederhole die Schritte 1 bis 5, bis die Werte eine geforderten Fehlerschwelle unterschreiten. *UWSP* verwendet hier 3 *mm*.

Das Kriterium, nach dem Wasserspiegelhöhen in der iterativen Prozedur geschätzt werden, ändert sich von Durchlauf zu Durchlauf. Beim *ersten* Durchlauf wird der Wasserstand des vorhergehenden Querprofils auf das aktuelle Querprofil projeziert. Beim zweiten Durchlauf wird der Wasserstand auf den geschätzten Wasserstand plus 70 % des Fehlers beim ersten Durchlauf (berechneter Wasserstand WS_{comp} - geschätzter Wasserstand WS_{assum}) gesetzt, also

$$WS_{comp}[2] := WS_{assum}[1] + 0.7 \cdot (WS_{comp}[1] - WS_{assum}[1]) = 0.3 \cdot WS_{assum}[1] + 0.7 \cdot WS_{comp}[1] \qquad \textit{Gl. 5.14}$$

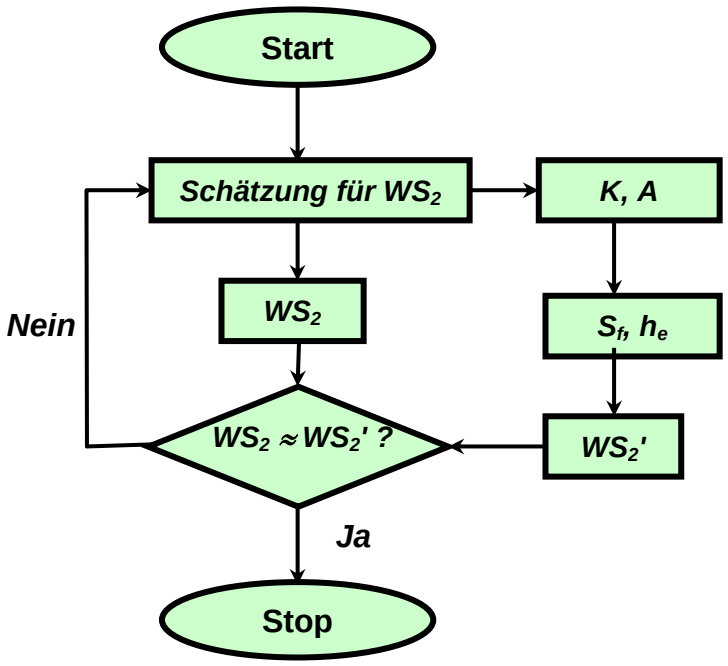


Abb. 5.4: Struktogramm der von UWSP durchgeführten iterativen Berechnung des Wasserstandes an einem Querprofil. $WS_2 \approx WS_2'$ ist erfüllt, wenn $| WS_2 - WS_2' | < \varepsilon$ ist, wobei ε die gewünschte Genauigkeit der Berechnung ist.

Ab dem dritten Durchlauf wird bei *UWSP* die Sekantenmethode verwendet, die die Änderungsrate der Differenz zwischen berechneten und geschätzten Wasserständen aus den jeweils zwei vorangegangenen Durchläufen (*i*-1 und *i*-2) berechnet. Die Gleichung für die Sekantenmethode lautet:

$$WS_{assum}[i] = WS_{assum}[i-2] - \frac{err[i-2] \cdot err_{assum}}{err_{diff}} \qquad \text{Gl. 5.15}$$

Mit:

$WS_{assum}[i]$	der geschätzte Wasserstand
$WS_{assum}[i\text{-}1]$	Der im letzten Iterationsschritt geschätzte Wasserstand
$WS_{assum}[i\text{-}2]$	Der im vorletzten Iterationsschritt geschätzte Wasserstand
$err[i\text{-}2]$	Der Fehler vom vorletzten Iterationsschritt (berechneter Wasserstand WS_{comp} minus geschätzter Wasserstand WS_{assum} vom vorletzten Iterationsschritt, also $err[i\text{-}2] = WS_{comp}[i\text{-}2] - WS_{assum}[i\text{-}2]$)
err_{assum}	Die Differenz zwischen geschätztem Wasserstand aus den zwei vorhergehenden Iterationen, also $err_{assum} = WS[i\text{-}2] - WS[i\text{-}1]$
err_{diff}	Der geschätzte Wasserstand minus der berechnete Wasserstand vom vorhergehenden Durchlauf (*i*-1), plus der Fehler vom vorletzten Durchlauf ($err[i\text{-}2]$), also $err_{diff} = WS_{assum}[i\text{-}1] - WS_{comp}[i\text{-}1] + err[i\text{-}2]$

Die Änderung zwischen zwei Durchläufen wird dabei auf eine Tiefe von maximal ± 50% der geschätzten Tiefe aus dem vorherigen Iterationsschritt beschränkt. Es gibt Fälle, in denen die Sekantenmethode versagt, wenn der Wert von err_{diff} zu klein wird. Wird err_{diff} kleiner als $1.0 \cdot 10^{-2}$, wird die Sekantenmethode von *UWSP* nicht verwendet. Das Programm berechnet dann einen neuen Schätzwert, indem es den Mittelwert von geschätztem und berechnetem Wasserstand der vorherigen Iteration bildet.
Die Iteration wird auf einem maximale Schrittzahl (standardmässig 20 Schritte) beschränkt. Während das Programm iteriert, hält es den Wasserstand fest, der bei den zurückliegenden Iterationsschritten den kleinsten Fehler hatte. Dieser Wasserstand wird der *minimale Fehler-Wasserstand* genannt. Ist die Maximalzahl der Iterationsschritte erreicht, bevor ein

ausgeglichener Wasserstand gefunden wurde, berechnet das Programm die kritische Tiefe. Es prüft dann, ob der Fehler, der mit dem minimalen Fehlerwasserstand verbunden ist, innerhalb einer definierten Toleranzgrenze liegt (standardmässig ist dies 0.1 *m*). Hat der minimale Fehler-Wasserstand einen Fehler kleiner als diese Toleranzgrenze, und liegt der Wasserspiegel auf der korrekten Seite der kritischen Tiefe, dann verwendet das Programm diesen Wasserstand als Ergebnis. Hat der minimale Fehler-Wasserstand einen grösseren Fehler als die vordefinierte Toleranzgrenze, oder liegt er auf der falschen Seite der kritischen Tiefe, so verwendet das Programm die kritische Tiefe. Die Begründung für diese Vorgehensweise ist, dass der minimale Fehler-Wasserstand wahrscheinlich eine bessere Lösung ist als die kritische Tiefe, solange die obigen Kriterien erfüllt sind.
Sowohl der minimale Fehler-Wasserstand als auch die kritische Tiefe werden in dieser Situation nur verwendet, um dem Programm die Fortsetzung der Rechnung zu ermöglichen. Keine dieser beiden Wert wird als gültige Lösung angesehen, daher werden Warnungen ausgegeben. Im Allgemeinen ist die Ursache für ein Unvermögen des Programms, den Wasserstand auszugleichen darin zu suchen, dass eine ungenügende Zahl an Querprofilen gegeben ist (zu weit auseinanderliegende Querprofile). Eine andere Möglichkeit ist, das das Programm versucht, eine unterkritische Lösung zu finden, obwohl der Abfluss an dieser Stelle tatsächlich überkritisch (schiessend) ist.
Konvergiert die Iteration für ein gegebenes Querprofil, wird sichergestellt, dass der Wasserstand sich auf der richtigen Seite der kritischen Tiefe befindet (also z.B. über der kritischen Tiefe, wenn ein unterkritischer Abfluss berechnet wurde). Befindet sich der berechnete Wasserstand auf der falschen Seite der kritischen Tiefe, wird die kritische Tiefe als Lösung verwendet und eine Warnung wird ausgegeben.

5.1.1.6 Berechnung der kritischen Tiefe

Die Gesamtenergiehöhe für ein Querprofil ist gegeben durch

$$H = WS + \frac{\alpha \cdot v^2}{2 \cdot g} \qquad \textit{Gl. 5.16}$$

Mit: H Gesamtenergiehöhe
WS Wasserspiegelhöhe
$\frac{\alpha \cdot v^2}{2 \cdot g}$ Geschwindigkeitshöhe

Die kritische Wasserspiegelhöhe ist die Höhe, für die die Gesamtenergiehöhe *H* ein Minimum ist (s. Abb. 5.5). Die kritische Höhe wird in einer iterativen Prozedur berechnet, indem Werte für *WS* angenommen werden und dann korrespondierende Werte für *H* mit Gleichung Gl. 5.16 bestimmt werden. Die Iteration läuft so lange, bis ein Minimum für *H* gefunden wurde.

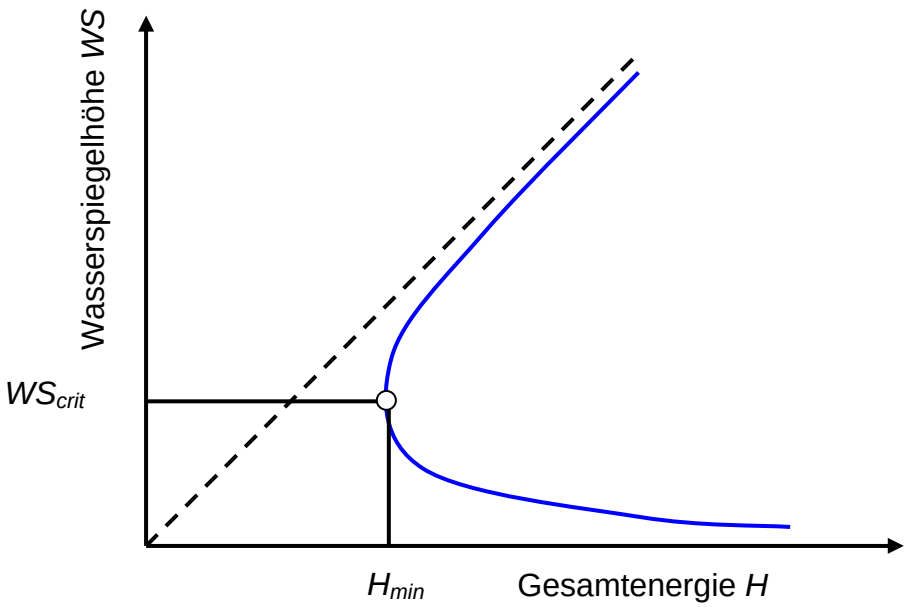


Abb. 5.5: In diesem Diagramm ist die Energie gegen die Wasserspiegelhöhe aufgetragen. Die kritische Tiefe WS_{crit} ist die Tiefe mit der minimalen Energie, ausgedrückt als Energiehöhe H (nach BRUNNER, 1998).

HEC-RAS verfügt über zwei Methoden zur Berechnung der kritischen Tiefe: eine parabolische Methode und eine Sekantenmethode. Die parabolische Methode ist schneller, sie kann aber nur ein Energieminimum ermitteln. Für die meisten Querprofile existiert nur ein Energieminimum auf der Gesamtenergiekurve, daher wird die parabolische Methode bei *HEC-RAS* standardmässig verwendet. Konvergiert die parabolische Methode nicht, so wird automatisch die Sekantenmethode versucht.
UWSP verwendet für Iterationen eine eigene Methode, den Schätzwert für den jeweils nächsten Iterationsschritt zu bestimmen. Diese Methode wurde gewählt, da sie erstens garantiert, dass das Programm konvergiert, sofern die Lösung eindeutig ist (Sollte der oben geschilderte Fall eintreten, dass es mehrere Lösungen gibt, so wäre eine Iterationsmethode von *HEC-RAS* zu verwenden). Zweiten erfolgt die Konvergenz sehr rasch.
Der Algoritmus basiert auf einem *binären* Prinzip. Es werden neben der Variablen für den nächsten Schätzwert *WS*[*i*] zwei weitere Variablen *low* und *high* verwendet, die als Intervallgrenzen für den betrachteten Konvergenzraum dienen. Der gesuchte Konvergenzwert befindet sich bei jedem Iterationsschritt zwischen diesen beiden Grenzen. Als nächster Schätzwert wird dann der Mittelwert von *low* und *high* verwendet. Dieser stellt nun entweder eine Überschätzung oder eine Unterschätzung dar, wobei das *Ausmass* des Fehlers nicht interessiert, lediglich das *Vorzeichen* (die Richtung). Je nach eingetretenem Fall wird dieser Schätzwert nun entweder der Variablen *low* oder der Variablen *high* zugewiesen. Dadurch, dass *WS* zuvor als Mittelwert gewählt wurde, wird das Intervall durch *WS* halbiert. So folgt aus der Zuweisung an *low* oder *high*, dass sich die Intervallgrenzen mit jedem Iterationsschritt aufeinander zubewegen und dabei sich das Intervall um die Hälfte verkleinern (daher *binäre* Methode). Würde man z.B. mit Wasserstandsgrenzen für *low* und *high* von 1 *m* bzw. 10 *m* beginnen (Intervallbereich 9 *m*), so wäre bereits nach 10 Iterationsschritten das Intervall auf einen Bereich von 1 *cm* geschrumpft.

5.1.1.7 Impulsgleichung

In *UWSP* wird anders als bei *HEC-RAS* die Impulsmethode generell für jedes Querprofil berechnet und geprüft, ob eine Lösung existiert. Im den meisten Fällen gibt es nur entweder

eine Lösung der BERNOULLI-Gleichung *oder* eine Lösung der Impulsgleichung. Haben beide Gleichungen Lösungen, so wird die Lösung mit der höheren Energie verwendet. *UWSP* verwendet die Impulsgleichung in der Form Gl. 5.18, die hier noch einmal wiedergegeben ist:

$$\frac{Q_2^{\,2} \cdot \beta_2}{g \cdot A_2} + A_2 \cdot \bar{y}_2 + \left(\frac{A_1 + A_2}{2}\right) \cdot L \cdot S_0 - \left(\frac{A_1 + A_2}{2}\right) \cdot L \cdot \bar{S}_f = \frac{Q_1^{\,2} \cdot \beta_1}{g \cdot A_1} + A_1 \cdot \bar{y}_1 \qquad \textit{Gl. 5.17}$$

5.1.2 Berechnung eines gemischten Abflussregimes

UWSP berechnet grundsätzlich gemischte Abflussregime. Die *spezifische Kraft-Gleichung* wird verwendet um zu ermitteln, welches Abflussregime kontrolliert. Die Gleichung für die spezifische Kraft wird aus der Impulsgleichung abgeleitet. Wird die Impulsgleichung auf einen sehr kleinen Flussabschnitt angewendet, dann ist die Reibungskraft und die Kraft aus dem Gewicht des Wassers sehr gering, und kann somit vernachlässigt werden. Die Impulsgleichung reduziert sich dann auf die folgende Gleichung:

$$\frac{Q_1^{\,2} \cdot \beta_1}{g \cdot A_1} + A_1 \cdot \bar{y}_1 = \frac{Q_2^{\,2} \cdot \beta_2}{g \cdot A_2} + A_2 \cdot \bar{y}_2 \qquad \textit{Gl. 5.18}$$

Mit:
- Q Abfluss an jedem Querprofil
- β Impulskoeffizient (vergleichbar mit dem Koeffizienten α)
- A Gesamter Abflussqerschnitt
- $\bar{y}$ Tiefe von der Wasseroberfläche bis zum Zentroid der Fläche
- g Gravitationsbeschleunigung

Die beiden Seiten der Gleichung sind analog und können für jedes Gerinnequerprofil als eine allgemeine Funktion ausgedrückt werden:

$$SF = \frac{Q^2 \cdot \beta}{g \cdot A} + A \cdot \bar{y} \qquad \textit{Gl. 5.19}$$

Die generalisierte Funktion besteht aus zwei Termen. Der erste Term ist der Impuls des Abflussvolumens durch den Querprofil-Querschnitt je Zeiteinheit. Dieser Teil der Gleichung wird als dynamische Komponente angesehen. Der zweite Term repräsentiert das Moment der statischen Komponente, welche die Kraft darstellt, die durch den hydrostatischen Druck des Wassers ausgeübt wird. Die Summe *SF* der beiden Terme wird spezifische Kraft genannt (CHOW, 1959). Wenn die spezifische Kraft-Gleichung auf natürliche Gerinne angewandt wird, wird sie folgendermassen geschrieben:

$$SF = \frac{Q^2 \cdot \beta}{g \cdot A_m} + A_t \cdot \bar{y} \qquad \textit{Gl. 5.20}$$

mit:
- A_m Durchflussfläche, in der Bewegung stattfindet
- A_t Gesamte Durchflussfläche, einschliesslich Totwasserbereiche

Die gemischte Abflussregimeberechnung wird folgendermassen durchgeführt:

1. Zuerst wird ein subkritisches Abflussprofil berechnet, beginnend bei einem bekannten, stromabwärts gelegenen Querprofil.
2. Danach beginnt das Programm eine superkritische Profilberechnung, stromaufwärts beginnend. Existiert sowohl eine subkritische als auch eine superkritische Lösung für den Wasserstand, dann prüft das Programm, für welche der beiden Lösungen die *spezifische Kraft grösser* ist. Ist die spezifische Kraft für das superkritische Abflussregime grösser, wird angenommen, das dieses Regime kontrolliert. Hat die Lösung des strömenden Falls

die grössere spezifische Kraft, so wird das subkritische Regime als kontrollierend angesehen.

3. Die Berechnung für das superkritische Abflussregime wird stromabwärts fortgesetzt. Wechselt zwischen zwei Querprofilen das Abflussregime, so wird das Auftreten eines hydraulischen Sprungs (Wechselsprung) angenommen.

5.1.3 Anwendungsgrenzen

Die folgenden Bedingungen werden in den Formeln der aktuellen Version als gegeben vorausgesetzt:

1. Der Abfluss ist stationär.
2. Der Abfluss ist eindimensional. (z.B. Geschwindigkeitskomponenten in eine andere Richtung als in die Hauptfliessrichtung des Gerinnes werden nicht betrachtet).
3. Das Längsgefälle der Gerinne ist schwach, unter 1:10.

Der Abfluss wird als stationär vorausgesetzt, da keine zeitabhängigen Terme in der Energiegleichung enthalten sind. Der Abfluss wird als eindimensional vorausgesetzt,da sich Gl. 5.16 darauf stützt, dass die gesamte Energiehöhe an allen Punkten eines Querprofils gleich ist. Ein geringes Längsgefälle ist Bedingung, da der hydraulische Druck in Gl. 5.3 durch die vertikal gemessene Wassertiefe repräsentiert wird. Das Programm besitzt im Gegensatz zu *HEC-RAS* nicht die Fähigkeit, mit beweglichen Grenzen(z.B. Sedimenttransport) umzugehen und verlangt, dass die Energieverluste durch die Terme in Gl. 5.4 beschrieben werden können.

5.2 Realisierung des Programms UWSP

UWSP lässt sich in zwei verschiedenen Modi betreiben. Die eine Möglichkeit ist, das Programm *interaktiv* zu bedienen, die andere, das Programm mit *Kommandozeilenparametern* aufzurufen.

Abb. 5.6: Interaktive Dateneingabe beim Programm UWSP.

Im ersten Fall startet das Programm mit einer graphischen Benutzeroberfläche, auf der vor Beginn der Berechnung die nötigen Angaben gemacht werden müssen (Abb. 5.6). Im zweiten Fall werden diese Angaben durch die Parameter der Kommandozeile mitgeteilt. Die graphische Oberfläche wird dann zwar ebenfalls gestartet, jedoch wird sofort mit der Berechnung begonnen. Dieser Modus eignet sich für die automatische Aktivierung von *Smallworld* aus.
Die nötigen Daten erschliessen sich aus den zur Verfügung stehenden Einstellungsmöglichkeiten. Für die einzelnen Bereiche sind im Menu Buttons vorhanden, bei deren Betätigung sich jeweils ein Fenster mit den einzelnen Einstellungsmöglichkeiten öffnen. Diese Buttons sind:

- Geometriedaten
- Abflussdaten
- Datenausgabe

Über die entsprechenden Dialogfenster müssen Dateinamen angegeben werden, in denen die Eingabedaten abgelegt sind (Buttons Geometriedaten und Abflussdaten) bzw. in welche die Ausgabedaten geschrieben werden sollen (Button Datenausgabe). Das Format für korrekte Eingabedaten wird vom *OGRUT*-Menu in *Smallworld* erzeugt, wenn man die Option `unstrukturierte Daten` bei der Geometrieausgabe bzw. die Option `strukturierte Daten` bei der Abflussausgabe wählt. Bei Geometridaten kann *UWSP* gegenwärtig noch keine strukturierten Daten verarbeiten.
Die *Geometridaten* umfassen Kilometrierung und die Koordinaten der Punkte eines Querprofils, Angaben über den Abschnitt (linkes Vorland, Flusschlauch, rechtes Vorland) und STRICKLER-Beiwerte. Die *Abflussdaten* sind Zeitreihen des Abflusses. Als Ergebnis erzeugt *UWSP* eine Ausgabedatei mit den berechneten Wasserspiegellinien für die in der Abflussdatei angegeben Zeitreihenwerte des Abflusses. Optional kann auch (z.B. zu Kalibrierungszwecken) ein einzelner Abflusswert angegeben werden, so dass keine ganze Zeitreihe gerechnet wird.

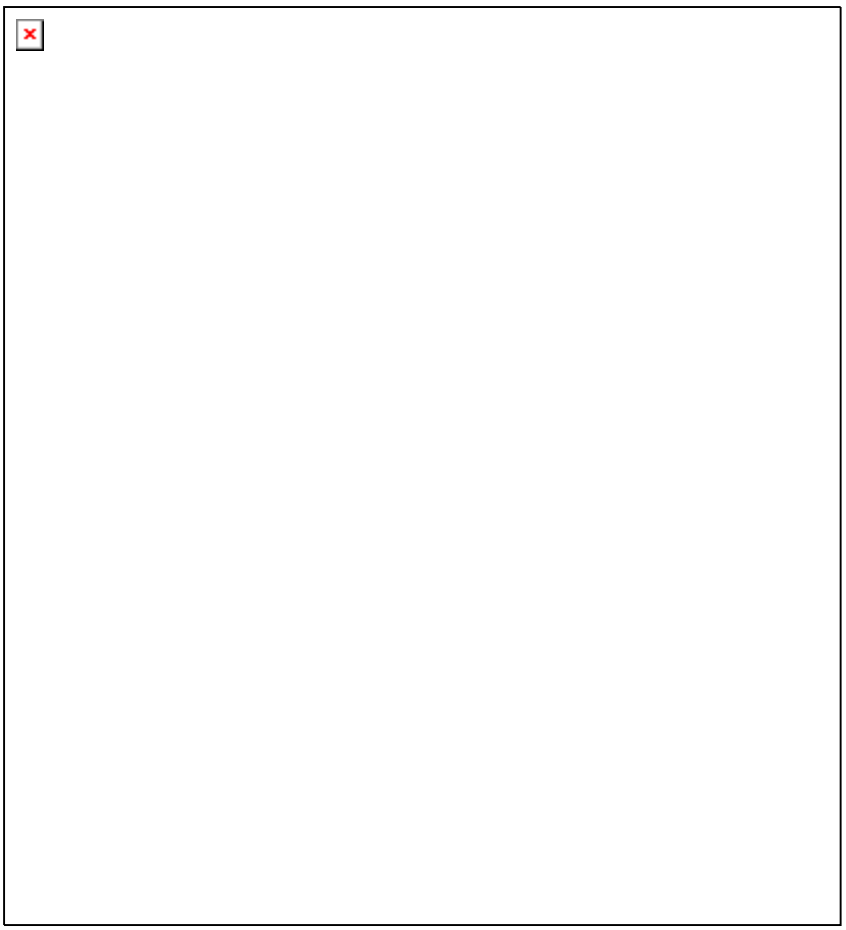


Abb. 5.7: UWSP bei der Berechnung von Wasserspiegellinien.

Wird *UWSP* mit *Kommandozeilenparametern* gestartet, wie dies auch *Smallworld* macht um *UWSP* automatisch auszuführen, so sind ebenfalls die Pfade zu den Quell- und Zieldateien anzugeben, ausserdem der Name des zu berechnenden Flusses. Dieser Name wird verwendet, um die gesuchten Dateien in den angegebenen Pfaden zu finden. *UWSP* geht also davon aus, dass die einzelnen Dateien den gleichen Namen haben und sich in verschiedenen Verzeichnissen befinden. Diese statische Struktur wurde gewählt, da sie sich sehr gut mit der gewünschten Automatisierbarkeit des *UWSP*-Aufrufs vereinbaren lässt. Ein Kommandozeilenaufruf würde dann z.B. so aussehen:

```
wsp_project   Brugga.txt
        \\eine\smallworld\liwis\data\input\profile\
        \\eine\smallworld\liwis\data\input\abfluss\
        \\eine\smallworld\liwis\data\output\wasserstand\
```

Sind alle erforderlichen Angaben gemacht, so startet das Programm entweder sofort (bei Eingaben über die Kommandozeile) oder durch Betätigung des Buttons "Berechnen" (s. Abb. 5.7).
Die Wasserspiegellagenberechnung findet dann nach dem im theoretischen Teil beschriebenen, an *HEC-RAS* angelehnten Verfahren statt. Die Abb. 5.8 zeigt in knapper Form den Ablauf der Berechnung in *UWSP*.

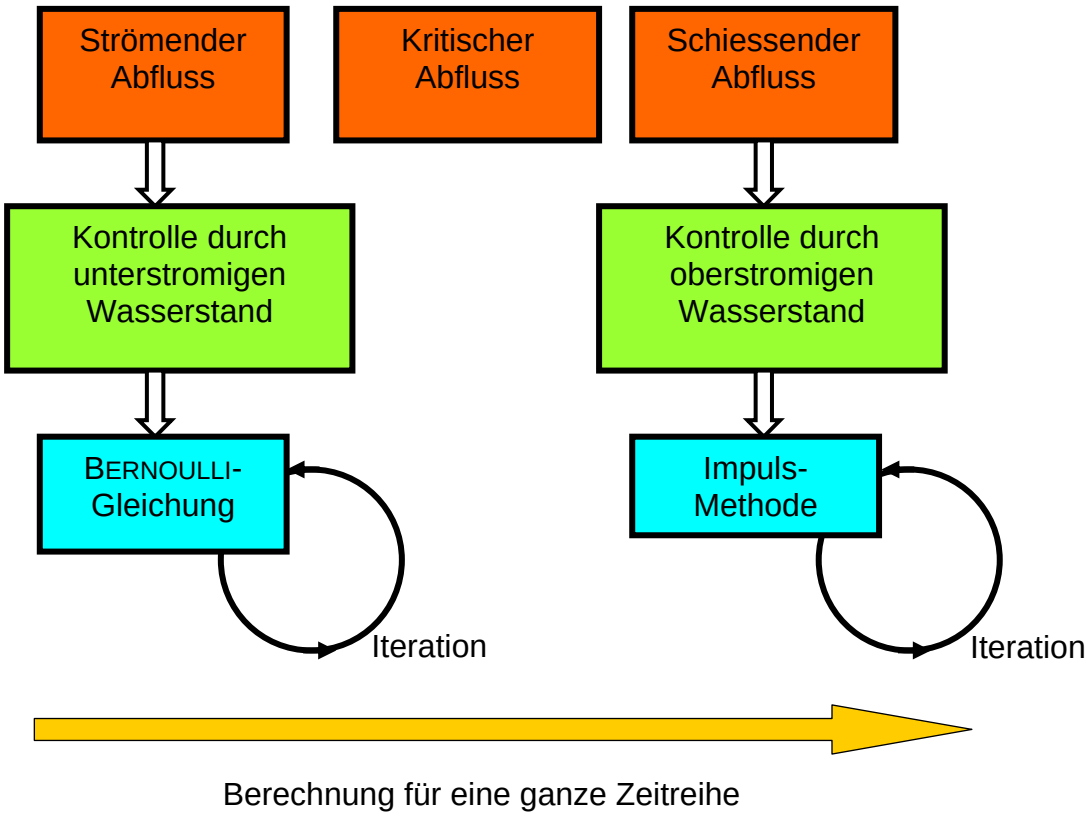


Abb. 5.8: Vorgehensweise von UWSP (schematisch).

Das Programm *UWSP* wurde in *C++* geschrieben, da diese Sprache eine objektorientierte Strukturierung des Programms erlaubt. Als Compiler wurde *Borland C++* 3.0 verwendet. Das Programm besteht aus einer Anzahl von cpp-Dateien und Header-Dateien, die im Normalfall eine Klasse enthalten. Die Projektdatei wsp_project.cpp ist die Hauptdatei. In verschiedenen anderen Dateine werden die einzelnen Klassen gespeichert. Es gibt folgende Klassen:

- TForm1, TForm2, TForm3, TForm4: Dies sind Formularklassen, also sichtbare Fenster mit Schaltflächen. Diese sind den *C++* -Dateien wsp.ccp, geometrie.cpp, abfluss.cpp und output.cpp zugeordnet.
- Reach: Diese Klasse entspricht einen Fluss oder Flussabschnitt, der eine Anzahl Querprofile enthält.
- Station: Diese Klasse repräsentiert einen einzelnen Querprofil-Punkt. Sie nimmt lediglich Koordinaten-Daten auf und besitzt keine besonderen Methoden.

- `CrossSection`: Diese Klasse definiert ein Querprofil. Jedes Querprofil enthält Verweise auf die ihm zugeordneten Stationen (Klasse `Station`). Jedem Querprofil ist weiterhin ein Array mit Elementen der Klasse `FlowData` zugeordnet. Das Array enthält Elemente für die Fälle Normalabfluss, subkritischer Abfluss, kritischer Abfluss und superkritischer Abfluss. Weiter ist jedem Querprofil ein Verweis auf eine Ganglinie der Klasse `Ganglinie` zugeordnet.
- `FlowData`: Diese Klasse enthält jeweils die berechneten Daten (Geschwindigkeit, Energiehöhenverlust, Durchflussfläche, Transportleistung, Wasserstand, usw.) für ein Abflussregime (Normalabfluss, subkritischer Abfluss, kritischer Abfluss oder superkritischer Abfluss) an einem Querprofil zum aktuell betrachteten Zeitpunkt (ein Zeitreihenwert). Nachdem alle Fälle berechnet sind, wird einer dieser Werte gewählt (die Vorgehensweise bei der Auswahl wurde weiter oben beschrieben) und in den aktuellen Zeitpunkt der Ganglinie übertragen. Es wird also ein neuer Eintrag in der Ganglinie erzeugt.
- `Ganglinie`: Diese Klasse enthält eine Liste von Werten, den einzelnen Zeitreihenwerten der berechneten Wasserstände. Die Element der Liste sind vom Standardtyp `float`. Jedes Querprofil enthält einen Verweis auf eine Ganglinie.
- `Scanner`: Diese Klasse dient zum Einlesen der Eingabedaten aus Dateien. Der Scanner zerlegt den Zeichenstrom in einen Symbolstrom. Die einzelnen Symbole werden dann an die aufrufenden Funktionen zurückgegeben. Eine analoge Klasse mit derselben Aufgabe findet sich auch im *OGRUT*-Modul von *Smallworld*.

Die Felder und Methoden der einzelnen Klassen sind im Anhang A aufgeführt. Die Klassen und deren strukturelle Verbindungen sind in Abb. 5.9 dargestellt.

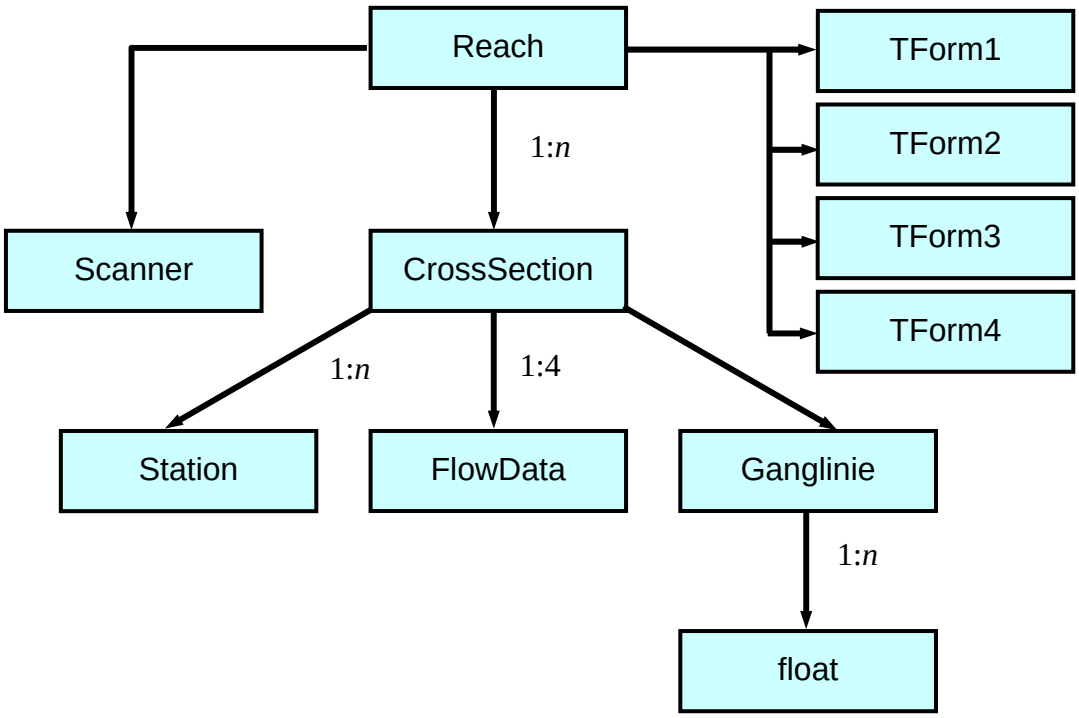


Abb. 5.9: Struktur des Programms UWSP. Dargestellt sind die Verbindungen der einzelnen Klassen untereinander über Listenbeziehungen (1:n) bzw. Arrays (1:4).

5.3 Fazit

Bei der Gerinnehydraulik existieren für die unterschiedlichsten Anwendungsfälle Lösungen, wobei der Aufwand den Anforderungen entspricht. Zur Berechung stationärer, ungleichförmiger, eindimensionaler Verhältnisse gibt es eine Reihe von Beschreibungen, wie ein entsprechendes Programm zu implementieren ist. Die Verfahren beruhen auf MANNING-STRICKER, neuere Verfahren ziehen i.a. die DARCY-WEISBACH-Gleichung vor.

Als Standardprogramm bot sich zunächst *HEC-RAS* an. Allerdings stellte sich wie erwähnt heraus, dass *HEC-RAS* nicht in der Lage ist, eine zentrale Anforderungen zu erfüllen, die Berechnung längere Zeitreihen. Ausserdem liegen die Ausgabedaten in einem undokumentierten Binärformat vor. Andere Beschreibungen von Algorithmen für Hydraulikprogramme liessen den Fall von schiessendem Abfluss bzw. gemischtem Abflussregime weitgehend unbeachtet (z.B. DVWK, 1991).

Die Entscheidung, daraufhin ein eigenständiges Programm (*UWSP*) zu implementieren, hat sich als gangbar herausgestellt. Das Programm wurde insbesondere mit Blick auf die nahtlose Integration in *OGRUT* entwickelt. Es greift auf eine definierte Schnittstelle zu und liefert die Ergebnisse genau in dem Format zurück, die für die Rücknahme in die Datenbank optimal sind. Auch in Hinblick auf Weiterentwicklungen und Experimente bietet das Programm einigen Spielraum, da der Quellcode zur Verfügung steht. Der objektorientierte Aufbau ermöglicht es, einzelne Komponente einfach und konsistent hinzuzufügen oder zu ersetzen.

Ein hydraulisches Problem, das noch nicht optimal gelöst ist besteht in der Tatsache, dass bei strömendem Abfluss die Rechnungsrichtung stromaufwärts erfolgen muss, bei schiessendem Abfluss dagegen stromabwärts. Hierdurch entsteht eine Lücke zwischen den untersten und den obersten Querprofilen. In dieser Lücke befinden sich Querprofile, für die jeweils die Daten des vorhergehenden bzw. des nachfolgenden Querprofils noch nicht berechnet sind, allerdings verlangt werden bevor das Profil selbst gerechnet werden kann. Diese Lücke wurde durch ein an *HEC-RAS* angelehntes Verfahren geschlossen (s. Kapitel 5.1.2). Dennoch wäre hier ein zweistufiges Vorgehen in Erwägung zu ziehen, bei dem die Iterations-Rechnung solange wiederholt wird, bis die Wasserstände an allen Querprofilen, und nicht nur an zwei aufeinanderfolgenden Querprofilen stabil werden. Der rechnerische Aufwand hierfür wäre allerdings beträchtlich.

6 OGRUT

OGRUT ist ein Oberflächen- und Grundwasser-Teilinformationssystem für *Smallworld* GIS.

6.1 Vorgehensweise und Umsetzung

Die Entwicklung neuer GIS-Module in *Smallworld* verläuft nach einem Schema, dessen einzelne Schritte in einer gewissen zeitlichen Reihenfolge erfolgen. Abweichungen sind möglich, dennoch kann die im folgenden skizzierte Vorgehensweise als Richtlinie dienen.

Der erste Schritt ist die *Definition eines Datenmodells*. Dies wird grösstenteils mit dem CASE-Tool durchgeführt. Es erlaubt wie die einfache, graphische Erstellung von Klassen-Tabellen (s. Abb. 6.1). Man gibt die einzelnen Spalten (Attribute) einer Tabelle mit deren jeweiligem Typ an. Einige Attribute stellen mittels Schlüsseln Verbindungen zu anderen Tabellen her, dies sind die Relationships. Die Verbindungen der Tabellen untereinander werden bildlich dargestellt durch Linien und Symbole an den Enden der Linien. Diese zeigen an, welcher Art die Verbindung ist (z.B. 1:1, 1:*n*, *n*:*m*).

Abb. 6.1: Ausschnitt aus dem mit dem CASE-Tool definierten Datenmodell.

Der zweite Schritt betrifft die *Programmierung der Funktionalität* der einzelnen Klassen. Dies umfasst die Erstellung der Methoden, die die jeweiligen Klasseninstanzen (Objekte) ausführen können. Neben den üblichen Programmelementen (Zuweisung, Sequenz, Alternative, Iteration) stehen in der *Smallworld*-Programmiersprache *Magik* zusätzliche Funktionen für den Zugriff auf Datenbankelemente und Objekte des GIS zur Verfügung.

Zuletzt gilt es, die neue Funktionalität in einer leicht bedienbaren *graphischen Oberfläche* zu integrieren. Diese muss es auch Benutzern ohne tiefergehende Programmierkenntnis ermöglichen, die neuen Module zu nutzen.

OGRUT basiert auf einer Reihe bestehender *Smallworld*-Module, denen weitere Funktionalität hinzugefügt wurde. Die Abb. 6.2 zeigt, dass es sich dabei im wesentlichen um die Module *LIWIS OFG*, *LIWIS ISO*, *BOMET* und *LIWIS AGRAR* handelt. Der mit *DGM-Modul* bezeichnete Block verwendet grundlegende geometrische Objekte von *Smallworld GIS*. Während bei *BOMET* und *LIWIS AGRAR* die bestehenden Fähigkeiten ausreichen und nicht erweitert werden, bieten *LIWIS OFG* und *LIWIS ISO* lediglich rudimentäre Datenstrukturen ohne weitergehende Funktionalität. Ein Hauptteil der Arbeit war daher die Erweiterung dieser beiden Module, die den Kern des *OGRUT* ausmachen (*OGRUT OFG* und *OGRUT ISO*).
Weitere Arbeiten betreffen das digitale Geländemodell (*OGRUT DGM*) und die Anbindung an die Katasterflächen und Bodendaten (*OGRUT AGR*). Im folgenden wird der Aufbau dieser Module beschrieben und diskutiert.

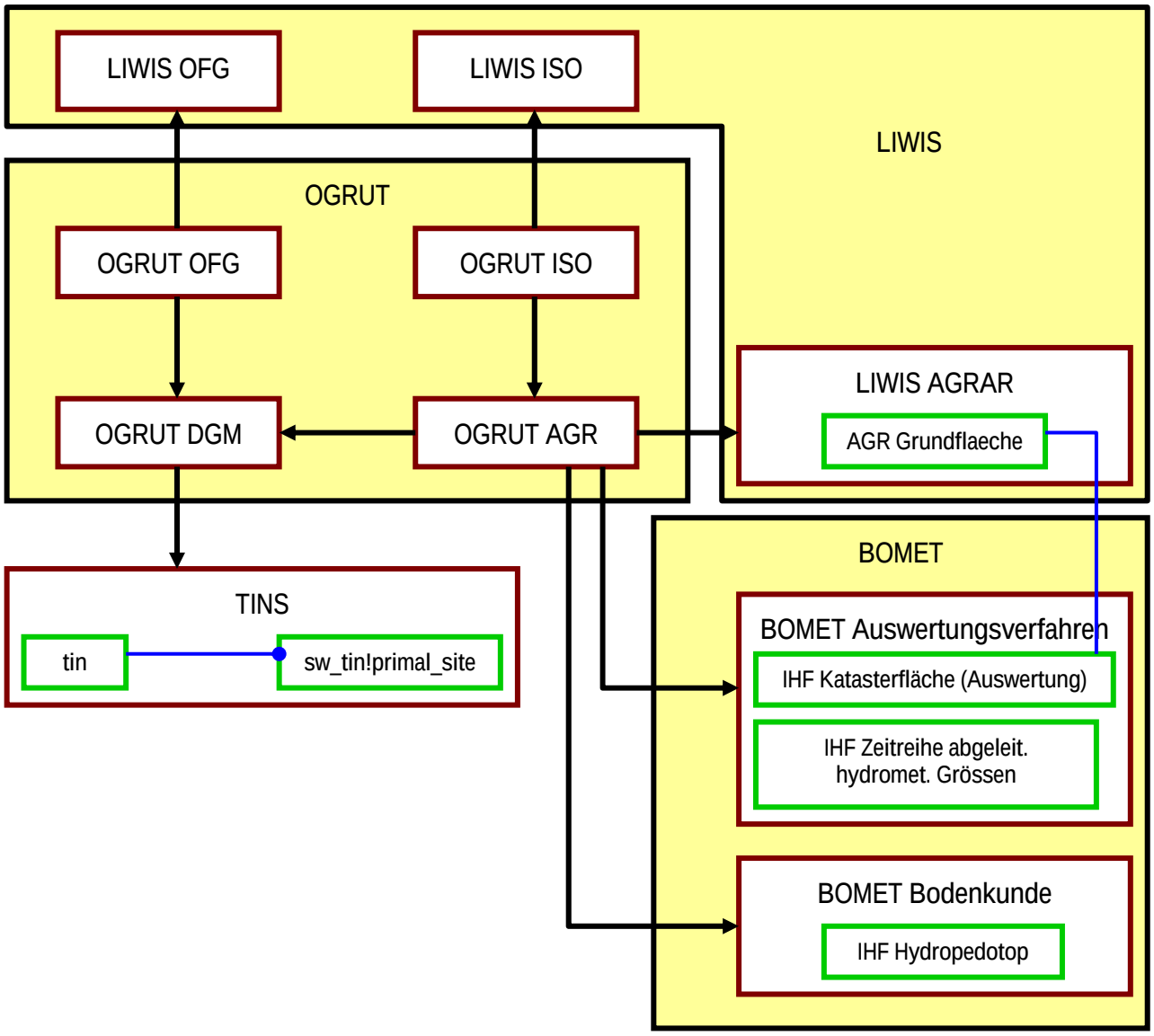


Abb. 6.2: Diagramm der gegenseitigen Abhängigkeiten zwischen den OGRUT-Teilmodulen und anderen Smallworld-Modulen.

6.2 Vorhandene Module und Fachschalen für Smallworld GIS

Smallworld besteht im Kern aus Datenbankobjekten (Tabellen, Relationen), einfachen Datenstrukturen (Felder, Listen, Mengen) und einfachen Funktionen (Mengenoperationen, algebraische Operationen, Suchfunktionen, Sortierfunktionen). Diese Elemente bilden die Grundlage, auf deren Basis komplexere GIS-Module erstellt werden. Eine Reihe solcher Module liegen bei *Smallworld* schon vor. Sie werden *Fachschalen* genannt und implementieren die Datenstrukturen für einen speziellen Anwendungsbereich (*Thema*). Hierzu gehören z.B. die Fachschalen Gas, Wasser und Verkehr. Verschiedene Firmen fügen diesen Fachschalen weitere hinzu oder erweitern bestehende Fachschalen für spezielle Anwendungsfälle. Hierzu gehört das beschriebene *wasserwirtschaftliche Informationssystem LIWIS* der Firma *Lahmeyer International* und das vom Insitut für Hydrologie erstellte Modul *BOMET*. Auch das Ergebnis dieser Arbeit, das Modul *OGRUT*, zählt dazu.

6.2.1 LIWIS

Die Firma *Lahmeyer International* hat für *Smallworld* ein wasserwirtschafliches Informationssystem (*LIWIS*) erstellt, das aus mehreren Teilmodulen der Wasserwirtschft besteht. Hierzu gehören unter anderem Komponente für die Themen Grundwasser (mit einem Finite-Elemente-Modul), Oberflächengewässer, landwirtschaftliche Flächen und Isolinien. Von Teilen dieser Anwendungen, insbesondere dem Modul für Oberflächengewässer und dem Isolinienmodul wird in der vorliegenden Arbeit Gebrauch gemacht.

Das Agrarmodul von *LIWIS* wurde für Grund- und Bewirtschaftungsdaten, Landwirtschaftliches Monitoring, N_{min}-Untersuchungen, Bodenuntersuchungen und Anbauberatung entwickelt.
Vom *LIWIS*-Agrarmodul verwendet *OGRUT* die Datenstruktur für Agrar-Grundflächen (Katasterflächen).

Das Oberflächengewässermodul von *LIWIS* stellt eine Anzahl grundlegender Objektklassen im Umfeld natürlicher und künstlicher Gerinne bereit. Zunächst definiert man eine Anzahl von Gerinnen, dann können weitergehende Objektklassen wie Oberflächengewässerpegel oder Talsperren bearbeitet werden. Die in diesem Modul enthaltenen Klassen sind unter anderem Einleitungen, Bauwerke, Segmente, Profile, Gewässergüte, Oberflächengewässerpegel, Talsperren und Hochbehälter. Allerdings werden keinerlei Auswertungs- und Berechnungsfunktionen bereitgestellt.
OGRUT nutzt diese Datenstrukturen und erweitert sie um Klassen z.B. für digitale Geländemodelle und Zeitreihen, ausserdem um Auswertungsfunktionen.

Das Isolininemodul von *LIWIS* verfügt über die Basiselemente `iso_node` und `iso_element`, also Iso-Knoten und Iso-Dreiecke, die Flächen zwischen den verbundenen Knoten. Im Gegensatz zu dem direkt von *Smallworld* bereitgestellten geometrischen Element `tin` kann das Isolinienmodul keine selbständige Triangulation durchführen.
OGRUT nutzt diese Datenstrukturen und erweitert sie unter anderem um Datenstrukturen für Zeitreihen und um Auswertungsfunktionen.

6.2.2 BOMET

Das *Bodenkundlich-Meteorologische Teilinformationssystem BOMET* (EBERLE, 1999) besteht zum einem aus einem *Bodenmodul*. Die Basis ist hierbei eine bodenkundliche Datenbank, in der sämtliche für den Trinkwasserschutz relevanten Bodendaten verfügbar gemacht werden.
Das zweite Teilmodul ist das *Meteorologisches Modul*. Hier werden meteorologische Daten von Klimastationen verwaltet, die neben den Bodendaten von den Auswertungswerkzeugen benötigt werden (WAQIS, 2000).
Die vorliegende Arbeit stützt sich an einigen Stellen auf *BOMET*. So liefert *BOMET* bereits die Funktionen und Daten für die direkte Grundwasserneubildung aus Niederschlag. Die Hydropedotopflächen aus *BOMET* werden für den Durchlässigkeitsbeiwert der Böden im Zartener Becken benötigt. Die Verwaltung von Zeitreihen an Katasterflächen geschieht mit Hilfe der von *BOMET* definierten Klasse `IHF Katasterfläche Auswertung`.

6.3 Oberflächengewässermodul OGRUT OFG

Das Oberflächengewässermodul *OGRUT OFG* basiert auf *LIWIS OFG*. Es dient zum Verwalten von Geometriedaten von Gerinnen, Pegeldaten von Abflusspegeln und Zeitreihen (Abfluss, Wasserstand) an einzelnen Querprofilen. Ausserdem sind jedem Fluss digitale Höhenmodelle zugeordnet, die der Fluss bei gegebenem Wasserstand überfluten kann.

6.3.1 Realisierung des Teilmoduls *OGRUT OFG*

Zunächst ist eine Anzahl an Oberflächengewässern und deren Geometriedaten zu definieren. Danach können Pegel an den Oberflächengewässern ausgewiesen werden. An den Pegeln sind Ganglinien des Abflusses anzugeben. Dies kann durch direkte Eingabe oder durch Einlesen aus einer Datei vollzogen werden. Sind die Zeitreihen für einzelne Pegel nicht kongruent (z.B. verschieden lang), so sind fehlende Daten z.B. durch Korrelation zu ergänzen.
Jedem Fluss ist ein digitales Höhenmodell zugeordnet, das aus einzelnen digitalen Geländepunkten besteht. Dies kann ebenfalls entweder durch Eingabe von Hand oder durch Einlesen aus einer Datei erfolgen.
Der Wasserstand an den Querprofilen wird dann aus den Abflussganglinien an den Pegeln berechnet. Die Berechnung erfolgt im externen Programm *UWSP*. Dieses Programm legt dann die Ergebnisse (Zeitreihen des Wasserstandes an den einzelnen Querprofilen) in einer Datei ab. Danach wird die Ausgabe von *UWSP* wieder zurückgelesen. Daraus wiederum wird die Überflutung des mit dem Fluss assoziierten digitalen Geländemodells berechnet. Die Methodik der Berechnung der Wasserstandsganglinien an den Querprofilen wurde im zusammen mit der Beschreibung von *UWSP* in Kap. 5.1 erläutert. Die Methodik der Berechnung der Überflutung des digitalen Geländemodells wird in Kap. 6.4.1.5 im Zusammenhang mit dem Modul *OGRUT DGM* erklärt.

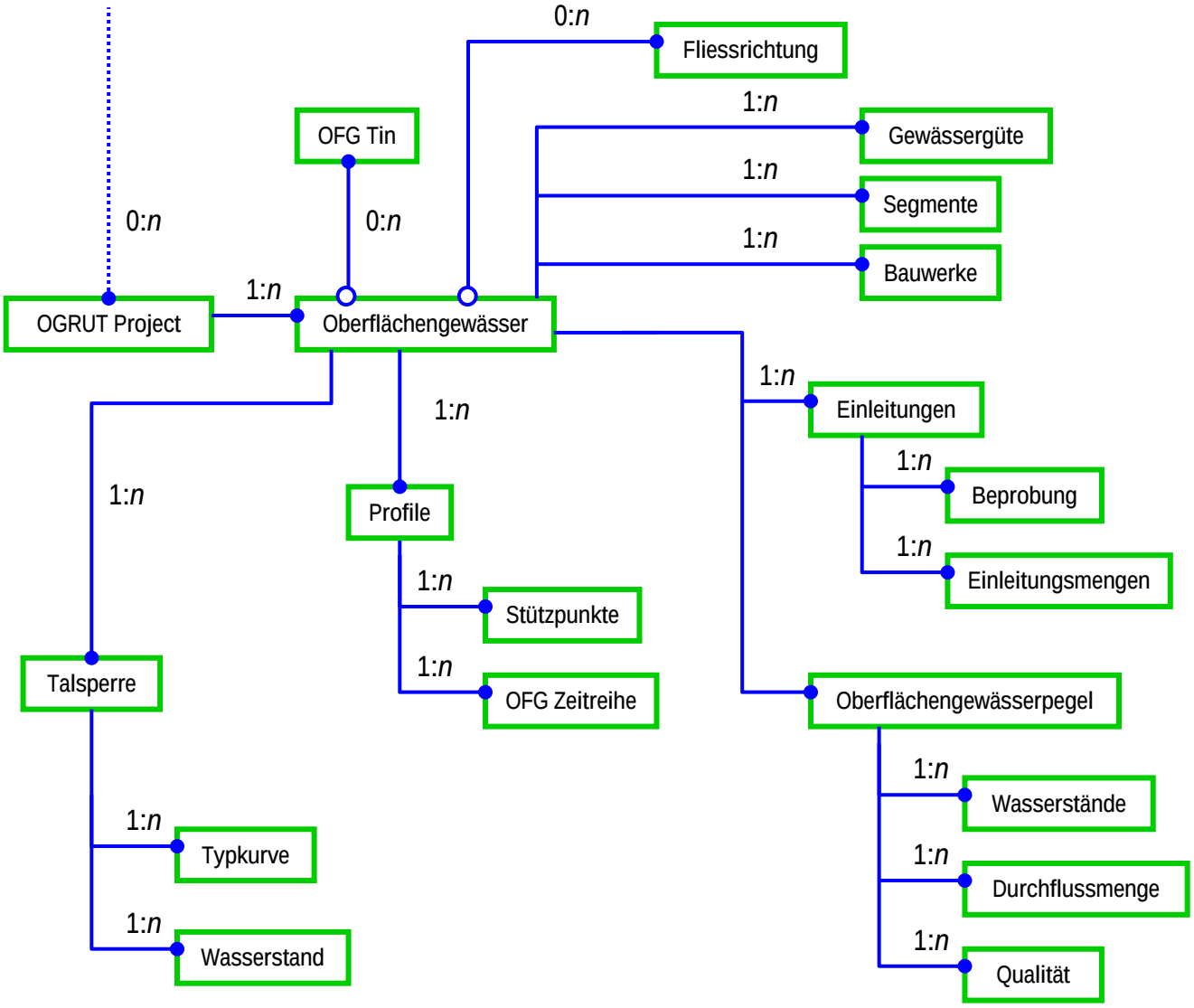


Abb. 6.3: Oberflächengewässer-Datenmodell des OGRUT.

6.3.2 Kalibirierung der Wasserstände

Um die Ergebnisse des Wasserspiegel-Programms *UWSP* zu überprüfen, wurden die Gerinnegeometriedaten für den Krummbach und ein Abflussereignis von 15 m^3s^{-1} über den STRICKLER-Beiwert an die Ergebnisse von WASSERWIRTSCHAFTSAMT FREIBURG (1988) kalibriert. Dort wurden unter anderem für den Krummbach Bemessungsganglinien und die Ausweisung von Überflutungsflächen berechnet. Das dabei eingesetzte Gerinnehydraulikprogramm war *REHM/FLUSS* 2, Version 86.05 .

Die Kalibirierung von *UWSP* an *REHM/FLUSS* führte für die Mehrzahl der Querprofile ohne grossen Anpassungsaufwand zu einer guten Übereinstimmung.
Schwierigkeiten ergaben sich jedoch für einige Querprofile, die sch auch durch unrealisitische Anpassung der STRICKLER-Beiwerte nicht in befriedigender Weise an die Ergebnisse von BELLER (in WASSERWIRTSCHAFTSAMT FREIBURG, 1988) anpassen liessen. Die Ursache hierfür waren bei genauerer Betrachtung jedoch nicht unrealistische Ergebnisse von *UWSP* sondern eine geringere Zahl an zur Verfügung stehenden Querprofilen, als dies bei BELLER der Fall war. BELLER verwendet in seiner Wasserspiegelberechnung ungefähr ein Dutzend Querprofle, von denen die Daten für unsere Berechnung nicht vorlagen. Hierzu gehörten insbesondere Brückenprofile, die durch ihre hohen Expansions- und Kontraktionsverluste einen starken Einfluss auf den Verlauf des Wasserspiegelprofils ausüben. Wir führten *UWSP* daraufhin mit weiteren, simulierten Brückenquerprofilen aus. Da aus den Daten vom WASSERWIRTSCHAFTSAMT FREIBURG nur die Existenz solcher Bückenprofile, nicht aber deren Geometrie bekannt war, wurden testweise geschätzte Geometrien verwendet.

Das Ergebnis war, dass *UWSP* sehr sensitiv auf solche weiteren, relativ engen Brückenquerprofile reagiert. Die Schwankungen bei kleinen Variationen der Querprofilgeometrie waren sehr stark, und eine Kalibrierung an die Ergebnisse von *REHM/FLUSS* wäre auf diesem Wege kein Problem mehr gewesen. Da die Gerinnegeometrie jedoch völlig spekulativ war, verwarfen wir diese Option und stellen lediglich fest, dass *UWSP* im Rahmen der verfügbaren Daten befriedigende Ergebnisse erzeugte, die mit denen von *REHM/FLUSS* vergleichbar waren. Ein wirklicher Vergleich der Ergebnisse ist jedch nicht möglich, da die Datengrundlage bei den Geometriedaten zu unterschiedlich ist.

Daher werden in Abb. 6.4 lediglich die auf den STRICKLER-Beiwert kalibrierten Wasserstände mit den Ergebnissen vom WASSERWIRTSCHAFTSAMT FREIBURG verglichen. Es ist noch einmal zu betonen, dass bei den Ergebnissen wegen der unterschiedlichen Datengrundlage unterschiedliche Ergebnisse zu erwarten waren, insbesondere da die fehlenden Profile einen starken Einfluss ausüben. Es wurde daher z.T. nötig, unrealistische STRICKLER-Beiwerte zu verwenden, um eine befriedigende Anpassung zu ereichen, so dass der k_{St}-Wert an den entsprechenden Querprofilen nicht mehr als physikalischer Parameter sondern nur noch als Anpassungsparameter anzusehen ist.

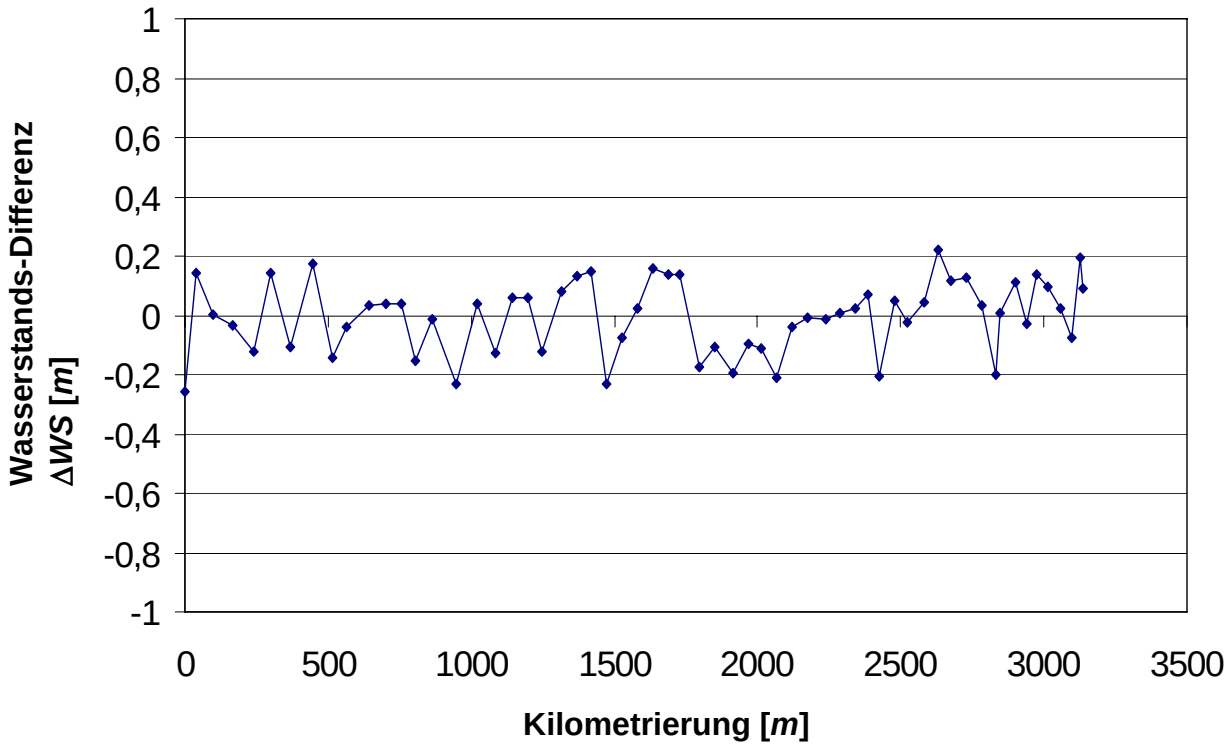


Abb. 6.4: Kalibrierung des Wasserstandes über den k_{St}*-Wert. Krummbach bei einem Durchfluss von 15* m^3s^{-1} *. Auf der* x*-Achse die Kilometrierung in* m*, auf der* y*-Achse der Unterschied der ermittelten Wasserstände für die einzelnen Querprofile. Aufgetragen sind die Abweichungen zwischen den Ergebnissen von WASSERWIRTSCHAFTSAMT FREIBURG (1988) auf der* x*-Geraden (*y *= 0) und den Werten aus* UWSP *(Kurve). Es wurde bei dieser Kalibirierung auf die Verwendung künstlicher Brückenprofile verzichtet.*

6.4 Digitales Geländemodell OGRUT DGM

Das digitale Geländemodell nimmt die Geländedaten der überflutungsgefährdeten Gebiete auf. Jedem Gerinne kann ein Geländemodell zugeordnet werden. Neben den Geländedaten werden Zeitreihen der Überflutung für die einzelnen Geländepunkte verwaltet. Das hierfür erstellte Datenmodell ist in Abb. 6.5 dargestellt.

Auf Basis von Durchlässigkeitsbeiwerten aus den Hydropeotopen, in denen die jeweiligen digitalen Geländepunkte liegen, wird dann direkt für die Geländepunkte eine Grundwasser-neubildungs-Zeitreihe berechnet. Es stellt sich die Frage nach der dafür geeigneten Formel. Dies wird im Kapitel 6.4.1 erörtert. Die konkrete Umsetzung des *Smallworld*-Teilmoduls *OGRUT DGM* ist Bestandteil des Kapitels 6.4.2.

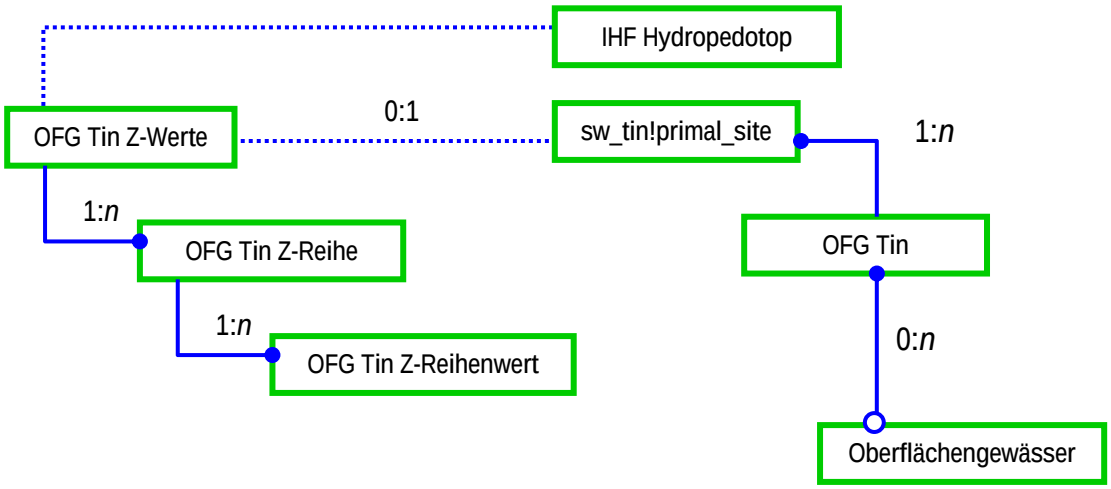


Abb. 6.5: DGM-Datenmodell des OGRUT. Zwischen `OFG Tin Z Werte` *und* `sw_tin!primal_site` *besteht keine Relation. Es werden lediglich die Schlüsselfelder der site in* `OFG Tin Z Werte` *gespeichert.*

6.4.1 Berechnung der Grundwasserneubildung

In dieser Arbeit werden zwei Fälle der Grundwasserneubildung betrachtet. Einerseits die *direkte Grundwasserneubildung* aus Niederschlag. Diese wird schon vom Teilmodul *BOMET* auf Basis von Katasterflächen bereitgestellt. Andererseits die *indirekte Grundwasserneubildung* aus Überflutungsflächen, deren Berechnung Teil dieser Arbeit ist.

6.4.1.1 Infiltration

In beiden Fällen erfolgt die Grundwasserneubildung durch Infiltration. Der Infiltrationsprozess kann nach DYCK&PESCHKE (1995) wie folgt beschrieben werden:
An der unteren Grenze einer dünnen, gesättigten Schicht (*Sättigungszone*) entsteht ein steiler Gradient des *Matrixpotentials* Ψ_M (*Feuchtefront*). Diese Feuchtefront wandert bei anhaltenden Niederschlägen in die Tiefe. Oberhalb der Feuchtefront entsteht eine *Transportzone*. Der Bodenspeicher wird mit fortschreitender Feuchtefront aufgefüllt, bis er zur Transportzone geworden ist (s Abb. 6.6). Sukzessive Auffeuchtung bei fortschreitender Infiltration verringert das Matrixpotential Ψ_M und damit die Infiltrationsrate. Erreicht Ψ_M vernachlässigbar kleine Werte, so stellt sich die Infiltration auf den konstanten Wert der hydraulischen Leitfähigkeit ein. Nach Beendigung der Infiltration wird der Bodenspeicher durch Perkolation und Evapotranspiration wieder entleert.

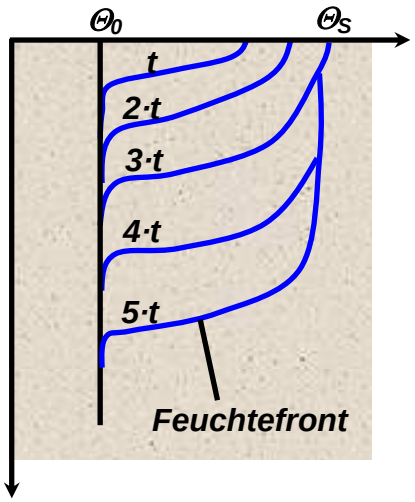


Abb. 6.6: Fortschreiten der Feuchtefront bei der Infiltration (aus DYCK&PESCHKE, 1995).

Für die hohen *Anfangsinfiltrationsraten* f_0 ist das Matrixpotential verantwortlich. Das Infiltrationsvermögen nimmt bei zunehmender Sättigung ab, da sich das Matrixpotential verringert. Zwar nimmt die *hydraulische Leitfähigkeit* $k(\Theta)$ zu, aber die Abnahme des Gradienten von Ψ_M wiegt schwerer. Es kommt zu einer asymptotische Annäherung an die *Endinfiltrationsrate* f_∞ , für die nur noch die Schwerkraft verantwortlich ist.

Die Grundlage für Infiltrationsmodelle ist die DARCY-Gleichung. Die Infiltrationsmodelle setzen daher im allgemeinen homogene und isotrope Böden, laminare Fliessverhälnisse (keine Makroporen) und eine gleichmässige Wasserverteilung der Böden in der Horizontalebene voraus.

6.4.1.2 Dynamik des Wassers im Untergrund

Durch DARCY (1856) wurde erkannt, dass die Bewegung des Grundwassers in einem weiten Geschwindigkeitsbereich einer einfachen Gesetzmässigkeit unterliegt. Danach ist die *Filtergeschwindigkeit* v_f dem Gefälle der Strandrohrspiegelhöhe proportional:

$$v_f = k_f \cdot \frac{\Delta h}{\Delta x} \qquad \textit{(Gl. 6.1)}$$

In dieser Form ist das DARCY-Gesetz auf die gesättigte, eindimensionale Strömung beschränkt. Es lässt sich aber auch auf den Fall einer räumlichen Strömung sowohl im gesättigten als auch im ungesättigten Boden verallgemeinern. Die Formel für diesen Fall wurde erstmals von BUCKINGHAM (1907) beschrieben. Die folgende DARCY-BUCKINGHAM-Gleichung verwendet daher eine vom Wassergehalt abhängige hydraulische Leitfähigkeit $k(\Theta)$:

$$\vec{v}_f = -k(\Theta) \cdot \operatorname{grad} \Psi \qquad \textit{(Gl. 6.2)}$$

Das Gefälle der Standrohrspiegelhöhe wird durch das gesamte Potential Ψ ersetzt, das auf die Wasserströmung in alle Koordinatenrichtungen wirkt. Für das Gesamtpotential gilt im wesentlichen

$$\Psi = \begin{cases} z + \dfrac{p}{\rho \cdot g} & \text{im gesättigten Boden} \\ z + \Psi_M & \text{im ungesättigten Boden} \end{cases}$$

Dabei bezeichnet Ψ_M das feuchteabhängige Matrixpotential.

Als weitere Gleichung, die bei Massenströmungen Gültigkeit hat, kommt die Kontinuitätsgleichung zum Einsatz. Sie beschreibt das Gesetz der Erhaltung der Masse. Danach ist die Änderung der Wassermasse in einem gegebenen Volumen gleich der Bilanz aus ein- und ausfliessendem Wasser. Die Gleichung lautet

$$\operatorname{div} \vec{v}_f = \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z} = \frac{\partial \Theta}{\partial t} \qquad \textit{(Gl. 6.3)}$$

Setzt man die DARCY-Gleichung in die Kontinuitätsgleichung ein, so erhält man die folgende Differentialgleichung:

$$\operatorname{div}(k(\Theta)\cdot \operatorname{grad}\Psi) = \frac{\partial\Theta}{\partial t} \qquad \textit{(Gl. 6.4)}$$

Die gesättigte hydraulische Leitfähigkeit k_f liegt nach DYCK&PESCHKE (1995) bei $2.3\cdot10^{-7}$ ms^{-1} in Böden mit äusserst geringer und $6.9\cdot10^{-5}$ ms^{-1} bei solchen mit äusserst hoher Durchlässigkeit (s. Tab. 6.1).

Tab. 6.1: Klassifizierung der gesättigten hydraulischen Leitfähigkeit.

Klasse	k_f [$m\ s^{-1}$]	Charakterisierung
I	$< 2.3\cdot10^{-7}$	äusserst gering
II	$< 6.9\cdot10^{-7}$	sehr gering
III	$< 2.3\cdot10^{-6}$	gering
IV	$< 6.9\cdot10^{-6}$	mittel
V	$< 2.3\cdot10^{-5}$	hoch
VI	$< 6.9\cdot10^{-5}$	sehr hoch
VII	$> 6.9\cdot10^{-5}$	äusserst hoch

6.4.1.3 Grundwasserneubildung

Unter *Grundwasserneubildung* versteht man die Sickerwassermenge, die die Grundwasseroberfläche erreicht. Der Grundwasserneubildung ist demnach der Prozess der Infiltration, der Bodenfeuchtespeicherung und der Abflussbildung in der ungesättigten Zone und des Entzugs der Bodenfeuchte infolge Evapotranspiration vorgeschaltet DYCK&PESCHKE (1995).

Sowohl die direkte Grundwasserneubildung aus flächenhaftem Niederschlag als auch die indirekte Grundwasserneubildung werden in *Smallworld* berechnet. Für beide Fälle muss ein Infiltrationsmodell gewählt werden. Für die direkte Grundwasserneubildung wurde bereits in *BOMET* ein Verfahren implementiert. Für die indirekte Grundwasserneubildung aus Überflutungsflächen war noch ein geeignetes Verfahren zu finden.

6.4.1.4 Direkte Grundwasserneubildung

Die *direkte Grundwasserneubildung* ist bereits in *BOMET* (EBERLE, 1999) enthalten. Sie wird über die Simulation des Bodenwasserhaushalts nach HAUDE-RENGER berechnet, die Zeitreihen der *aktuellen Evapotranspiration*, des *Bodenwassergehalts* und der *direkten Grundwasserneubildung* liefert. Die Auswahl dieses Verfahrens für das *WAQIS* wurde in der Arbeit von ALBERS (1998) getroffen und diskutiert.

Das HAUDE-RENGER-Verfahren (RENGER et al., 1977) betrachtet den Bodenraum bis in 1 *m* Tiefe als einen einzigen Bodenwasserspeicher. Sein Input ist der Niederschlag, Verdunstung und Grundwasserneubildung dezimieren den Bodenwassergehalt im Speicher. Vom Prinzip ist das Verfahren nach HAUDE-RENGER ein stark vereinfachter Kapazitätsansatz.

Für einen Zeitpunkt t ergibt sich der Bodenwassergehalt $\Theta(t)$ (in $mm\ m^{-1}$) aus der folgenden Gleichung:

$$\Theta(t_i) = \Theta(t_{i-1}) + P(t_i) - ET_R(t_i) \qquad \text{(Gl. 6.5)}$$

$P(t)$ ist der Niederschlag und $ET_R(t)$ die reale Evapotranspiration zum Zeitpunkt t.
Überschreitet der Bodenwassergehalt die Feldkapazität, so wird das übrschüssige Bodenwasser aus dem Bodenraum nach unten abgegeben. Unter der Annahme, dass dieses Wasser nicht mehr von den Pflanzenwurzeln erreichbar ist, stellt es die zu berechnende Grundwasserneubildung $GWNB(t)$ dar:

$$GWNB(t) = \begin{cases} \Theta(t) - FK & \text{wenn } \Theta(t) > FK \\ 0 & \text{wenn } \Theta(t) \leq FK \end{cases} \qquad \text{Gl. 6.6}$$

Im ersten Fall wird $\Theta(t)$ anschliessend auf die Feldkapazität FK gesetzt.

Die reale Evapotranspiration aus (Gl. 6.5) berechnet sich folgendermassen:

$$ET_R = ET_P \cdot f(\Theta_{nFK})$$

mit

$$f(\Theta_{nFK}) = \begin{cases} 1 & \text{wenn } \Theta_{nFK}(t) \geq 70\% \\ 0.2 + 2 \cdot \frac{\Theta_{nFK}}{100} - 1.2 \cdot \left(\frac{\Theta_{nFK}}{100}\right)^2 & \text{wenn } \Theta_{nFK}(t) < 70\% \end{cases} \qquad \text{Gl. 6.7}$$

Θ_{nFK} stellt den pflanzenverfügbaren Bodenwassergehalt (Θ - PWP) dar, als prozentualen Anteil an der nFK. Die potentielle Evapotranspiration ET_p wird nach HAUDE berechnet (EBERLE, 1999).

6.4.1.5 Indirekte Grundwasserneubildung

Es wurden einige Infiltrationsmodelle auf ihre Eignung für den Einsatz in dieser Arbeit untersucht. Wichtig war die Automatisierbarkeit und die besondere Eignung für den Fall stark überstauter Flächen.
Eine Bilanzgleichung, die die Grundwasserneubildung aus der Evapotranspiration und dem Input herleitet ist bei überstautem Gelände nicht verwendbar. Die Evapotranspiration ist vernachlässigbar, und die Infiltrationsleistung ist nach Sättigung des Bodens nur von der Leitfähigkeit des Bodens abhängig, nicht von dem verfügbaren Input. Denn dieser übersteigt bei Überstau die Infiltrationsleistung des Bodens.
Bei der indirekten Grundwasserneubildung aus Überflutungsflächen müsste man, um korrekt vorzugehen, zuerst den Zeitraum berücksichtigen der nötig ist, um den Bodenwasserspeicher aufzufüllen. Erst danach könnte man bei weiter versickerndem Wasser in vollem Umfang mit Grundwasserneubildung rechnen. Wir stellen die Aufgabe jedoch etwas anders und vereinfachen dadurch die Berechnung: Wir gehen jedoch davon aus, dass der Boden schon voll aufgesättigt ist und das infiltrierte Wasser somit nur noch dem Gravitationspotential und dem Druckpotential (gesättigte Verhältnisse) unterliegt. Diese Annahmen erscheinen uns vertretbar, da wir im Falle einer Überflutung von einer raschen Aufsättigung des oberen Bodens ausgehen können. Wenn wir diese Bedingungen aufstellen, erhalten wir vereinfachte Ableitungen aus den verfügbaren Formeln zu Infiltration.

Von KOSTIAKOV (1932) stammt ein Infiltrationsmodell, das jedoch für bewässerte Gebiete entwickelt wurde. Die Formel lautet

$$f(t) = a \cdot t^{-b} \qquad \text{(Gl. 6.8)}$$

Mit $f(t)$ maximale Infiltrationsrate zur Zeit t [$mm\ h^{-1}$]
a, b Konstanten
t Zeit seit Bewässerungsbeginn

Die Konstanten a und b sind abhängig von der Bodenart und bedürfen einer Eichung. Die Formel lässt daher eine einfache Automatisierung nicht zu. Zudem liefert die Formel für $t \to \infty$ eine gegen 0 gehende Infiltrationsrate. Dies widerspricht der Tatsache, dass die Infiltrationsrate sich einem Grenzwert > 0 annähert. Für $t \to 0$ wird $f \to \infty$, was ebenfalls problematisch ist.

Das Modell von HORTON (1940) arbeitet mit einem Anfangsinfiltrationswert f_0 der zu Beginn des Infiltrationsprozesses gilt. Mit zunehmender Dauer verringert sich dessen Auswirkung und es stellt sich ein Endinfiltrationswert f_∞ ein. Das Modell beschreibt eine exponentielle Abnahme der Infiltrationsrate (Rückgangsterm) bis zur End-Infiltrationsrate bei Sättigung.

$$f(t) = f_\infty + \left(f_0 - f_\infty\right) \cdot e^{-k \cdot t} \qquad \textit{(Gl. 6.9)}$$

Mit $f(t)$ maximale Infiltrationsrate zur Zeit t [$mm\ h^{-1}$]
f_∞ maximale Infiltrationsrate bei Sättigung ($t \to \infty$)
f_0 maximale Infiltrationsrate zu Beginn ($t = 0$)
k Rückgangskonstante

Die Parameter für das Modell werden im Allgemeinen abgeschätzt. Für unbewachsenen feinsandiger Ton bzw. für Grasboden gelten die Werte in Tab. 6.2.

Tab. 6.2: Werte für f_0, f_∞ und k im Infiltrationsmodell nach HORTION.

	Ton	**Grasboden**
$\boldsymbol{f_0}$ [$mm\ h^{-1}$]	210	900
$\boldsymbol{f_\infty}$ [$mm\ h^{-1}$]	2	290
$\boldsymbol{k}$ [min^{-1}]	0.8	2

Aus dem Modell von HORTON unter der Annahme, dass die Sättigungs-Infiltrationsrate gilt, also praktisch der Zustand $t \to \infty$ erreicht ist, folgt:

$$\lim_{t \to \infty} f(t) = \lim_{t \to \infty}\left(f_\infty + \left(f_0 - f_\infty\right) \cdot e^{-k \cdot t}\right) = f_\infty \qquad \textit{Gl. 6.10}$$

Von HOLTAN (1961) stammt ein Modellkonzept, dem wie bei der Formel von GREEN&AMPT (1911) die Vorstellung der Bodenzone als Feuchtigkeitsspeicher zugrundeliegt. 5 Parameter müssen durch Eichung empirisch ermittelt werden. Die Infiltration wird als Funktion des aktuell verfügbaren Speicherraumes angegeben.

$$f(t) = f_\infty + a \cdot \left(\Theta_\infty - \Theta(t)\right)^n \qquad \textit{Gl. 6.11}$$

Mit $f(t)$ maximale Infiltrationsrate zur Zeit t [$mm\ h^{-1}$]
f_∞ maximale Infiltrationsrate bei Sättigung ($t \to \infty$)
a Boden- und Vegetationsparameter

Θ_∞ maximal speicherbare Wassermenge
$\Theta(t)$ Inhalt des Speichers der oberen Bodenzone
n von der Bodenart abhängige Konstante

Als Speicher der oberen Bodenzone wird z.B. der *A*-Horizont, also etwa die oberen 5 bis 10 *cm* verwendet. Für die Konstante *n* wird z.B. der Wert 1.4 verwendet.
Wenn die Sättigungs-Infiltrationsrate erreicht ist, folgt aus dem Modell von HOLTAN das gleiche Ergebnis wie bei HORTON: Aus $\Theta := \Theta_\infty$, also bei erreichter Sättigungsfeuchte des Bodens, ergibt sich:

$$f(t,\Theta=\Theta_\infty) = f_\infty + a\cdot\left(\Theta_\infty - \Theta(t)\right)^n = f_\infty + a\cdot\left(\Theta_\infty - \Theta_\infty\right)^n = f_\infty \qquad \text{Gl. 6.12}$$

PHILIP (1957) führt den Ausdruck *Sorptivität S* ein. Es handelt sich physikalisch gesehen um die kumulative Menge des infltrierten Wassers zwischen der Zeit $t = 1$ und der Zeit, bei der die Infiltrationsrate auf die Hälfte des Wertes von *S* gesunken ist. Die Näherungsformel von PHILIP (1960) lautet:

$$f(t) = f_\infty + \frac{1}{2}\cdot S\cdot\frac{1}{\sqrt{t}} \qquad \text{Gl. 6.13}$$

Mit $f(t)$ maximale Infiltrationsrate zur Zeit t [$mm\ h^{-1}$]
f_∞ maximale Infiltrationsrate bei Sättigung ($t \to \infty$)
S Wasseraufnahmefähigkeit des Bodens (Sorptivität)
t Infiltrationsdauer

S muss in Versuchen aus der Saugspannung bestimmt werden. Das Problem ist der unrealistische Verlauf am Anfang, also für $t \to 0$ da hier $f \to \infty$ geht.
Das gleiche Bild wie bei den vorherigen Modellen ergibt sich, wenn man das Modell von PHILIP bei Sättigungs-Infiltrationsrate betrachtet. Ist der Boden gesättigt, so ist die Sorptivität des Bodens $S = 0$. Dann folgt auch hier:

$$f(t,S=0) = f_\infty + \frac{1}{2}\cdot S\cdot\frac{1}{\sqrt{t}} = f_\infty \qquad \text{Gl. 6.14}$$

Was hat man gewonnen? Man erkennt, dass die Endinfiltrationsrate f_∞ , die direkt mit der gesättigten hydraulischen Leitfähigkeit zusammenhängt, der alleinige Term in allen Formeln ist. Befriedigend ist dieses Ergebnis nicht, insofern als keine Berücksichtigung der Mächtigkeit des Bodens stattfindet und auch die Überstauhöhe, die ein Mass für den Grad der Überschwemmung ist, keinen Eingang findet. Man muss auch berücksichtigen, dass alle bisher betrachteten Formel empirisch sind. Es muss daher damit gerechnet werden, dass sie zur Beschreibung überfluteter Böden versagen, da sie eventuell Parameter vernachlässigen, die nur in diesem Fall bedeutsam werden.

Betrachten wir daher physikalisch begründete Modelle, wie die Formel von GREEN&AMPT (1911). Diese Formel wird aus dem DARCY-Gesetz bzw. der DARCY-BUCKINGHAM-Gleichung abgeleitet. Das Modell approximiert den Infiltrationsprozess durch ein Stufenprofil mit vollständiger Wassersättigung in der Sättigungszone *und* der Transportzone. Nur der wassergesättigte Teil des Bodens wird betrachtet (k_f konstant). Das Prinzip dieser Formel beruht auf der Gradientenmethode, die durch das DARCY-Gesetz erfasst wird. Zu Beginn wirkt eine hohe Saugspannung (Matrixpotential), die mit der Zeit schwächer wird.

$$f(t) = k_f \cdot \left(\frac{\Psi_M(\Theta) + z_0 + z_f}{z_f(t)} \right) = k_f \cdot \left(\frac{\Psi_M(\Theta) + z_0}{z_f(t)} + 1 \right) \qquad \text{Gl. 6.15}$$

Mit $f(t)$ maximale Infiltrationsrate zur Zeit t [$mm\text{-}h^{-1}$]
k_f gesättigte (hydraulische) Leitfähigkeit [$mm\ min^{-1}$]
Ψ_M mittlere Saugspannung (Betrag des effektiven Matrixpotentials an der Feuchtefront)
z_0 Dicke des Wasserfilms (Höhe der überstauenden Wasserschicht)
$z_f(t)$ Tiefe der Feuchtefront zum Zeitpunkt t

Nun betrachten wird die der Gleichung von GREEN&AMPT zugrundeliegende Gleichung, die DARCY-BUCKINGHAM-Gleichung:

$$\vec{v}_f = -k(\Theta) \cdot \operatorname{grad} \Psi \qquad \text{Gl. 6.16}$$

Berücksichtigen wir nun ausschliesslich den senkrechten Wasserfluss, der bei der Infiltration wirksam ist. Wenn wir eine in der horizontalen Ebene homogene Infiltration voraussetzen, ist dies auch die alleinige Richtung der Wasserbewegung. Die Formel vereinfacht sich somit zu

$$v_{f,z} = -k(\Theta) \cdot \frac{\partial \Psi}{\partial z} \qquad \text{Gl. 6.17}$$

Wir hatten vorausgesetzt, dass bereits Sättigung eingetreten ist, so dass die hydraulische Leitfähigkeit in die *gesättigte* hydraulische Leitfähigkeit übergegangen ist. Dann erhält man

$$v_{f,z} = k_f \cdot \frac{\partial \Psi}{\partial z} \qquad \text{Gl. 6.18}$$

Nun setzt man die Potentiale ein, die im gesättigten Boden wirken. Dies sind Gravitationspotential und Druckpotential, jeweils ausgedrückt als Potentialhöhe:

$$\Psi = z + \frac{p}{\rho \cdot g} \qquad \text{Gl. 6.19}$$

Man erhält also

$$v_{f,z} = k_f \cdot \frac{\partial \left(\frac{p}{\rho \cdot g} + z \right)}{\partial z} = k_f \cdot \frac{\partial (h + z)}{\partial z} \qquad \text{Gl. 6.20}$$

Wobei h die Höhe der überstauten Wasserschicht darstellt. Da wir die Bodenoberfläche als Nullpunkt unseres Gravitationspotentials betrachten, fällt das Glied z weg. Setzt man nun für die Filtergeschwindigkeit die Formel für den Wasserfluss Q durch die Querschnittsfläche A des Bodens ein,

$$v_f = \frac{Q}{A} \qquad \text{Gl. 6.21}$$

so erhält man

$$Q(t) = A \cdot k_f \cdot \frac{\Delta h}{\Delta z} \qquad \text{Gl. 6.22}$$

Über einen Zeitraum Δt ergibt sich dann die Infiltrationsmenge

$$Q(\Delta t) = \int_{t_0}^{t} A \cdot k_f \cdot \frac{\Delta h}{\Delta z} dt = A \cdot k_f \cdot \frac{\Delta h}{\Delta z} \cdot \Delta t \qquad \textit{Gl. 6.23}$$

Betrachten wir nun die Infiltrationsleistung für verschiedene Durchlässigkeitsbeiwerte. Wir setzen, um eine grobe Abschätzung zu erhalten, für $\Delta h := 1\ m$, für die Bodenmächtigkeit $\Delta z := 1\ m$, für den betrachteten Bodenausschnitt die Einheitsfläche 1 m^2. *F*ür den k_f-Wert setzen wir jeweils die Ober- und die Untergrenze aus Tab. 6.1 ein. Zusätzlich ist dort die Infiltration für einen k_f-Wert von $1.15 \cdot 10^{-5}\ m\ s^{-1}$ angegeben. Dieser entspricht der höchsten Leitfähigkeit von Böden aus dem Untersuchungsgebiet. Dann erhalten wir die Ergebnisse in Tab. 6.3:

Tab. 6.3: Infiltrationsleistung aus überstauten Flächen für verschiedene k_f-Werte

k_f [$m\ s^{-1}$]	Infiltration [$m^3\ m^{-2}\ s^{-1}$]	Infiltration [$m^3\ m^{-2}\ d^{-1}$]
$2.3 \cdot 10^{-7}$	$2.3 \cdot 10^{-7}$	0.02
$6.9 \cdot 10^{-5}$	$6.9 \cdot 10^{-5}$	5.96
$1.15 \cdot 10^{-5}$	$1.15 \cdot 10^{-5}$	0.99

Vergleichen wir diese Werte der Infiltrationsleistung aus flächenhafter Überflutung mit der Anreicherungsleistung für verschiedene Methoden zum Zwecke der künstlichen Grundwasseranreicherung in BÖHM. et al (1999). Dort werden die Zahlen in Tab. 6.4 genannt:

*Tab. 6.4: Anreicherungsleistung für verschiedene künstliche Anreicherungsmethoden (*aus *BÖHM et al, 1999).*

Art der Anreicherung	Mittlere Anreicherungsleistung [$m^3\ m^{-2}\ d^{-1}$]
Beregnung	0.2 .. 1.0
Flächenhafte Überflutung	0.2 .. 1.0
Versickerungsgräben, Becken, Teiche	1.0 .. 4.0, in Sonderfällen 0.15 .. 12.0

Es zeigt sich, dass die berechneten Infiltrationsleistungen - abgesehen von extrem guten und extrem schlechten Leitfähigkeitsbedingungen - mit den hier aufgeführten Werten zur Anreicherungsleistung aus künstlicher, flächenhafter Überflutung prinzipiell übereinstimmen.

6.4.2 Realisierung des Teilmoduls *OGRUT DGM*

Das digitale Geländemodell basiert auf der im Anhang beschriebenen Klasse `OFG Tin` (s. Tab. B.9). Die Zeitreihenwerte für die Wasserstände überfluteter *sites* (digitaler Geländepunkte) werden nicht direkt als Relation mit der Klasse `sw_tin!primal_site` verbunden, da dies eine spezielle Geometrieklasse ist. Stattdessen werden die Schlüsselfelder von sites in der Klasse `OFG Tin Z-Werte` gespeichert. `OFG Tin Z-Werte` enthält eine Relation zur Klasse `OFG Tin Z-Reihe`, und diese wiederum enthält eine Relation zur Klasse `OFG Tin Z-Reihenwert`, in der einzelne Zeitreihenwerte gespeichert werden. Der Grund für

diese dreistufige Aufteilung ist, dass zu jedem mit einer site assoziierten `OFG Tin Z-Werte`-Objekt nicht nur eine Zeitreihe verwaltet werden können soll, sondern mehrere.
Konkret werden in *OGRUT* zwei Zeitreihen verwaltet, eine für den Wasserstand und eine für die Grundwasserneubildung. Diese beiden Zeitreihen werden dann jeweils durch ein `ofg_tin_z_reihe`-Objekt dargestellt. Die einzelnen Zeitreihenwerte werden dann in Objekten der Klasse `ofg_tin_z_reihenwert` verwaltet (s. Abb. 6.6).

Ein Tin lässt sich triangulieren. Allgemein versteht man unter einer *Triangulation* von Punkten in der Ebene eine maximale Menge sich nicht kreuzender Liniensegmente mit vorgegebenen Endpunkten. Als Ergebnis erhält man eine Menge von Kanten, die die einzelnen Punkte miteinander verbinden und so eine Nachbarschaftsbeziehung zwischen Punkten herstellen.
Eine besonders günstige Triangulation mit wenig spitzen Winkeln ist die DELAUNY-Triangulation (z.B. beschrieben in LEE&SCHACHTER, 1980).

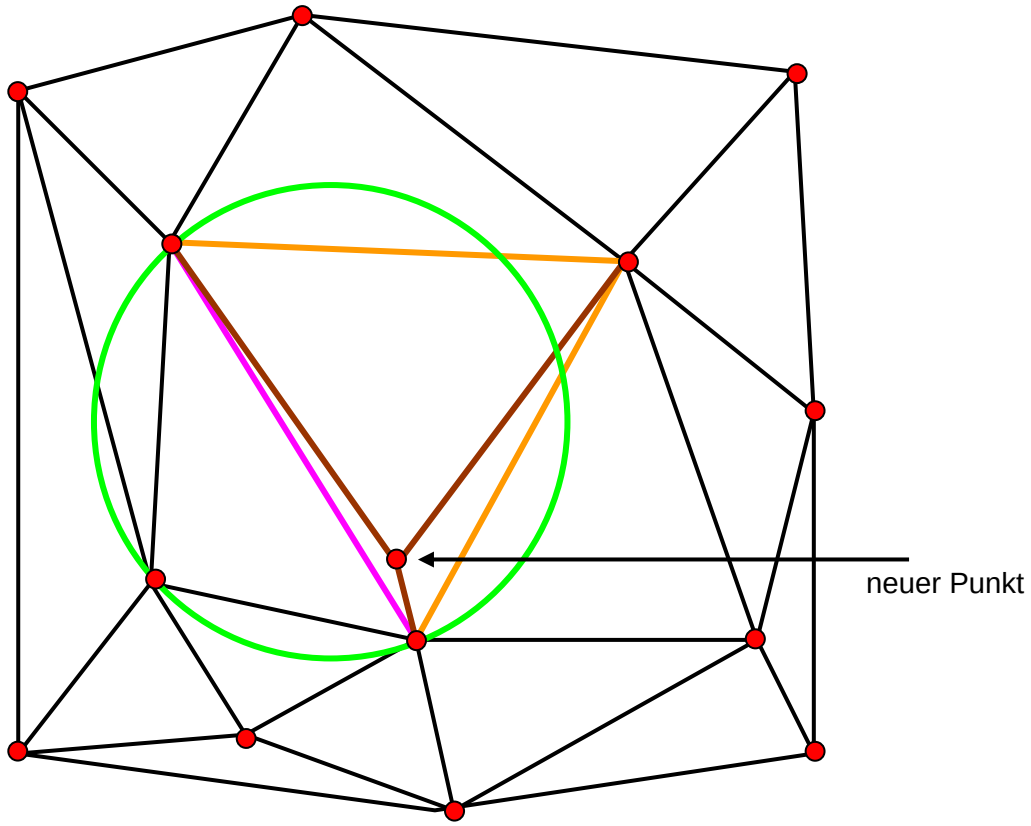


Abb. 6.7: Die DELAUNY-Triangulation erfolgt durch Prüfung der Lage eines neuen Punktes bezüglich der Umkreise des Dreiecks, mit dessen Punkten dieser neue Punkt verbunden wird. Liegt der Punkt innerhalb des Umkreises, so findet ein edge-flip statt (nach HAGEN, 1999).

Wenn man zu einem *DELAUNY-Dreieck* den Umkreis bildet, also den eindeutig bestimmten Kreis durch die drei Eckpunkte, so kann dieser Umkreis keinen anderen Punkt enthalten. Das hat etwas damit zu tun, daß der Kreismittelpunkt ein Knoten im VORONOI-Diagramm ist. Durch diese Eigenschaft lässt sich die *DELAUNY-Triangulation* charakterisieren
Mit der Umkreismethode lässt sich die DELAUNY-Triangulation inkrementell konstruieren. Man startet mit einem Dreieck und fügt dann die Punkte nacheinander hinzu. Wenn ein neuer Punkt zu einer bestehenden Triangulation hinzukommt, verbindet man ihn mit den Eckpunkten des Dreiecks, in dem er liegt.

Dann nimmt man nacheinander die Nachdreiecke und testet, ob ihr Umkreis den neuen Punkt enthält (s. Abb. 6.2). Ist das nicht der Fall, so kann das Dreieck bestehen bleiben. Wenn aber der neue Punkt im Umkreis enthalten ist, so wird ein *edge flip* ausgeführt. Hierdurch entstehen zwei neue Nachbardreiecke, die auch wieder zu testen sind - und so weiter. Möglicherweise müssen dabei sehr viele Kanten geflippt werden (HAGEN, 1999).

Das geometrische Attribut `tin` in *Smallworld* erlaubt die DELAUNY-Triangulation mit der Methode `tin.triangulate()`. Diese Triangulation wird in *OGRUT* benötigt, um die Ausbreitung der Überflutung über das digitale Geländemodell zu berechnen. In Abb. 6.8 ist beispielhaft das digitale Geländemodell für den Krummbach dargestellt, wie es in *OGRUT* verwendet wird. Digitalisiert wurden die Höhenlinien, um das Gerinne herum wurde eine Verdichtung durch Interpolation vorgenommen.

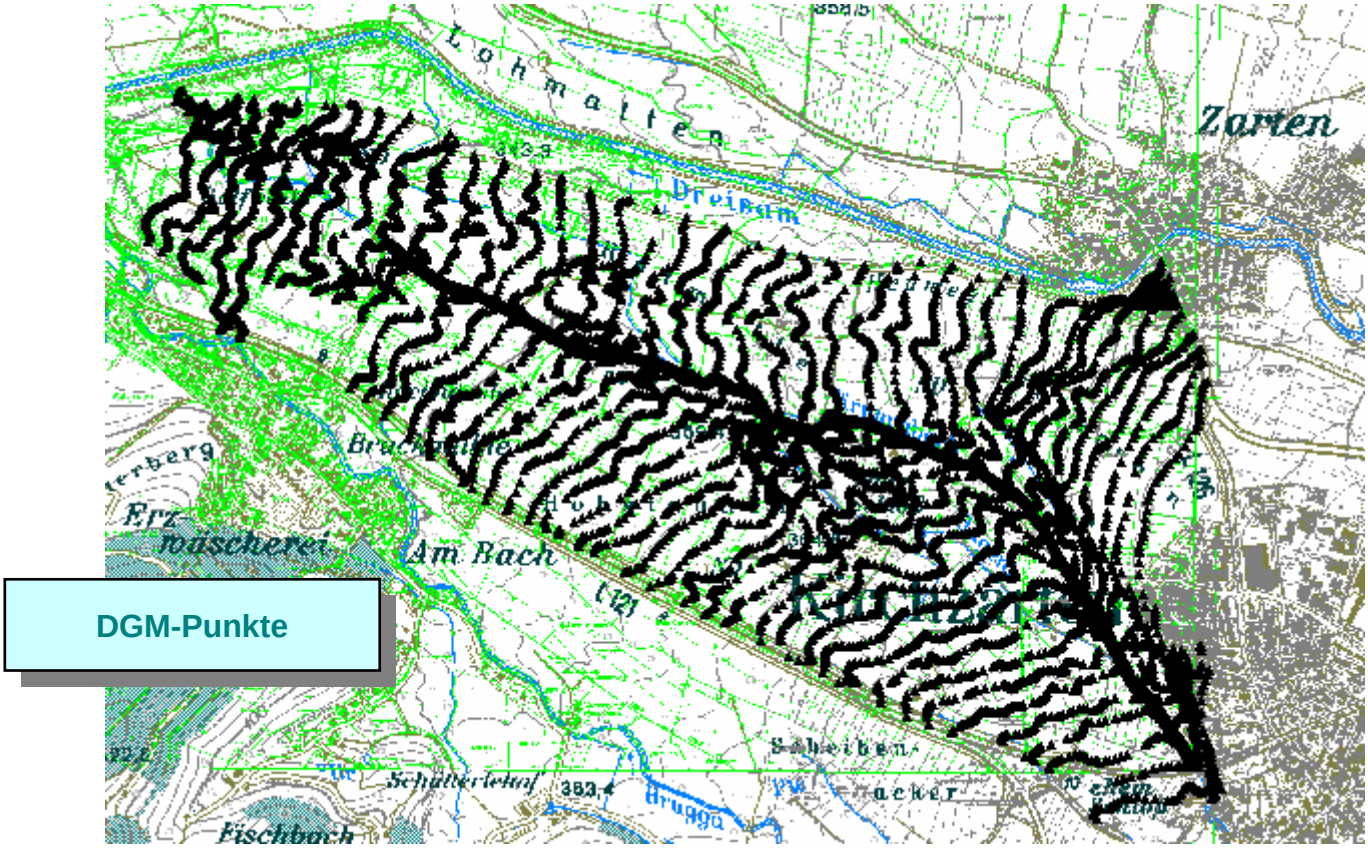


Abb. 6.8: Das digitale Geländemodell in OGRUT, Beispiel Krummbach. Die einzelnen Linien sind tatsächlich digitale Geländepunkte.

Nachdem das einem Gerinne zugeordnete digitale Geländemodell trianguliert ist, existiert zu jedem Geländepunkt eine Anzahl natürlicher Nachbarn, die sich mit der Methode

```
tin.natural_neighbours(site)
```

ermitteln lassen.
Dies ermöglicht nun, eine sinnvolle Vorgehensweise für die Überflutung des DGM zu definieren. Der Algorithmus besteht aus zwei Schritten:

- Zu Beginn sind nur die Wasserstände an den einzelnen Querprofilen des Gerinnes bekannt. Es ist die Überflutung, ausgehend von einem Gerinnequerprofil, für die dem Querprofil nächstgelegenen digitalen Geländepunkte (sites) zu berechnen (s. Abb. 6.9).

- Danach ist das Fortschreiten der Überflutung von einer site zu ihren natürlichen Nachbarn (bezüglich der Triangulation) zu ermitteln.

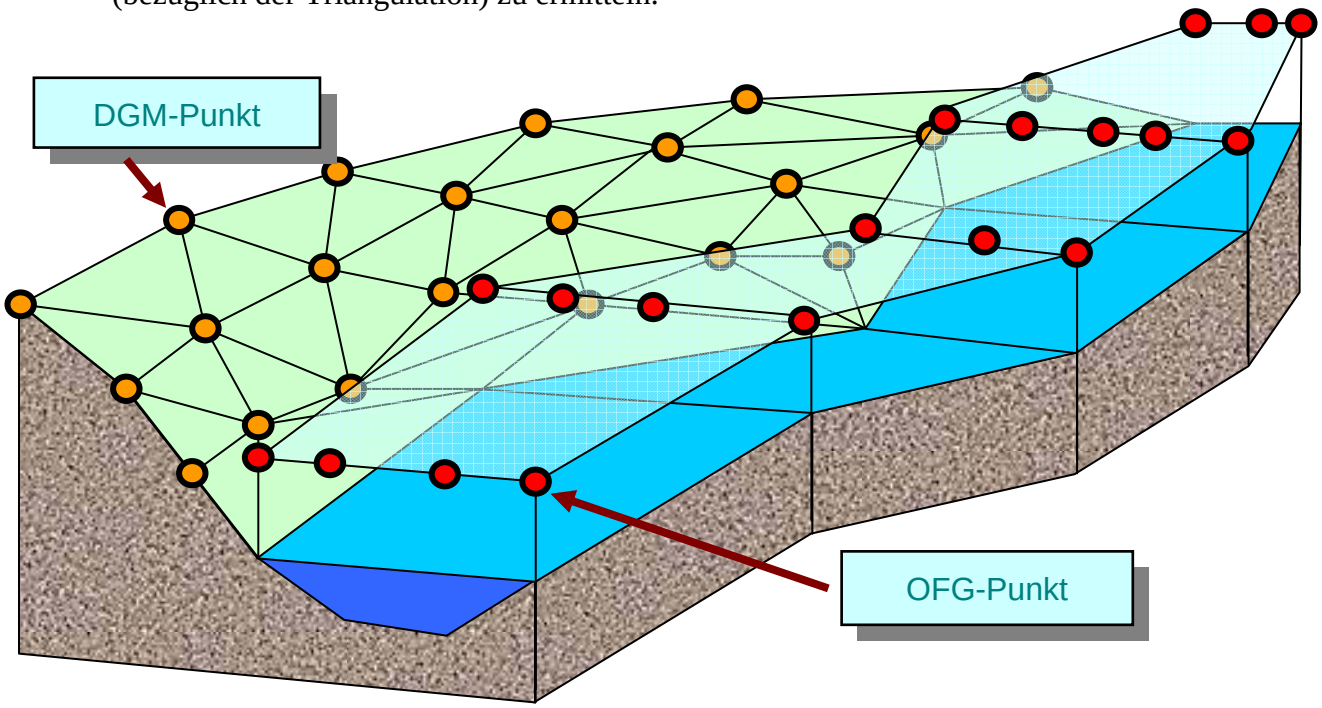


Abb. 6.9: Überflutung des digitalen Geländemodells durch einen Fluss. OFG-Punkte bezeichnen Stützpunkte an Querprofilen des Flusses, DGM-Punkte sind Punkte des digitalen Geländemodells.

Aus Abb. 6.9 ist erkennbar, dass ein Querprofil als Linie eine gewisse räumliche Erstreckung hat. Um die Überflutung des DGM möglichst bei den Geländepunkten zu beginnen, die dem Vorfluter am nächsten liegen, wurde aus den Koordinaten der beiden Randpunkte des DGM der (arithmetische) Mittelwert gebildet. Es entsteht ein für das Querprofil repräsentativer Punkt (Abb. 6.10). Zu diesem Punkt wird mit

```
coord << Coordinate.new(rw, hw)
```

ein Koordinatenobjekt `coord` erzeugt. Mit der Methode

```
site << tin.site_nearest(coord)
```

wird der diesem Koordinatenobjekt nächstgelegene digitale Geländepunkt `site` ermittelt.

Dieser Vorgang erfordert für jeden Zeitreihenschritt und für jedes Querprofil eine einzige Befehlssequenz. Der weitere Ablauf des Programms läuft dagegen in einer Iteration ab.
Alle im ersten Schritt ermittelten Geländepunkte werden in die Menge `current_sites_set` aufgenommen. Diese Menge wird nun durchlaufen. Wird ein darin enthaltener Geländepunkt überflutet, dann werden seine natürlichen Nachbarn in eine Menge `next_sites_set` gespeichert. Der betrachtete Punkt wird in eine Menge bereits bearbeiteter Punkte (`all_sites_set`) aufgenommen. Nachdem die Menge `current_sites_set` komplett abgearbeitet ist, wird die Liste `next_sites_set` daraufhin geprüft, ob sie leer ist. Ist dies der Fall, so ist die Überflutung für diesen Zeitreihenschritt beendet, andernfalls wird `next_sites_set` als Menge `current_sites_set` für den nächsten Durchlauf verwendet:

```
current_sites_set << next_sites_set
```

Die natürlichen Nachbarn aller überfluteten Punkte des vorhergehenden Durchlaufs sind also die auf Überflutung zu überprüfenden Punkte des nächsten Durchlaufs.
Die Menge der überfluteten Punkte kann am Anfang gross sein, wird aber nach einiger Zeit abnehmen, da das Gelände vom Flussufer weg ansteigt. Dann bricht die Schleife ab.

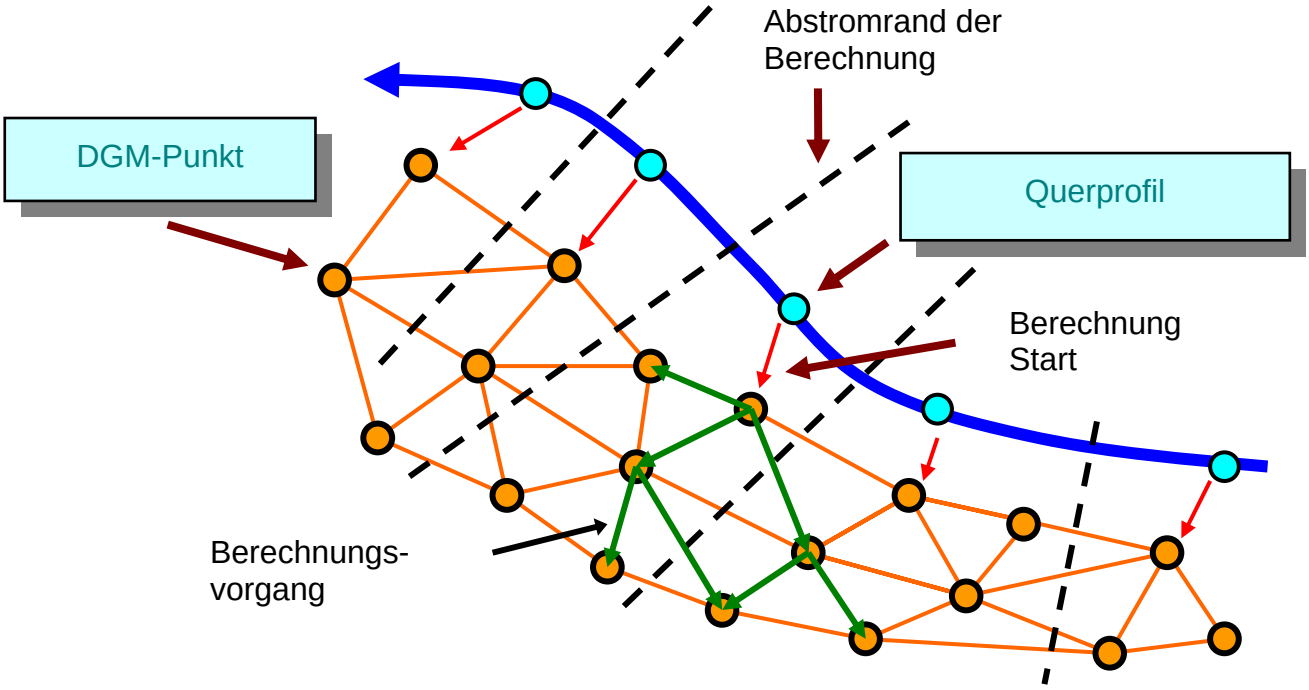


Abb. 6.10: Darstellung des Algorithmus zur Überflutungsberechnung des digitalen Geländemodells. Die einzelnen Querprofile dienen als Ausgangspunkte des Graphen, entlang dessen Kanten sich die Überflutung ausbreitet.

Problematisch ist jedoch, dass die Geländehöhe talabwärts abnimmt. Würde also keine Grenze der Überflutungsausbreitung in Abstromrichtung gesetzt, dann würde Querprofil möglicherweise das ganze Tal stromabwärts "ertränken". Dies wird vermieden, indem für jedes Querprofil ein Abstromrand der Berechnung festgelegt wird. Es wird festgelegt, dass die Überflutung von einem Querprofil stromabwärts nur bis zur Mittellinie zwischen dem Schwerpunkt des betrachteten Querprofils und dem Schwerpunkt des nächsttieferen Querprofils ausgehen kann, jedoch beliebig weit in alle anderen Richtungen (s. Abb. 6.10).
Es hat sich herausgestellt, dass diese Bedingung in mindestens einem Fall nicht ausreicht, um physikalisch unsinnige Überflutung zu vermeiden. Dieser Fall tritt ein, wenn ein Querprofil und das nächsttiefere an einer scharfen Kurve im Gerinnebett liegen (s. Abb. 6.11). Dieser Fall wird vom gegenwärtigen Algorithmus noch nicht berücksichtigt, da noch keine Lösung gefunden wurde.

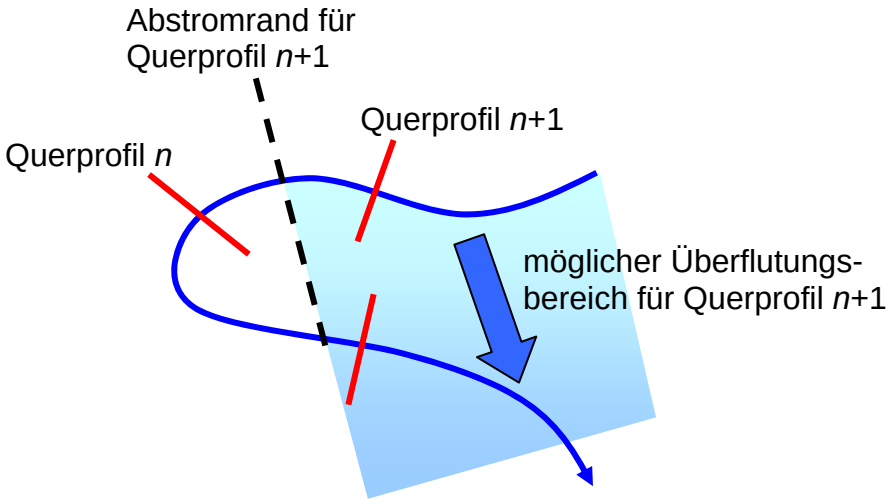


Abb. 6.11: Spezialfall, den der gegenwärtige Überflutungsalgorithmus noch nicht berücksichtigt: Bei scharfen Kurven kann die Bedingung, das nächsttiefere Querprofil als Abstromrand der Überflutungsberechnung zu verwenden, unzureichend sein.

Die Überflutungsberechnung wird mit der Methode

```
ofg_dar.berechne_h_tin(tin_id, from_date, to_date,
                       delete_old?, display_tin, _optional items)
```

gestartet. Die Methode besitzt folgenden prinzipiellen Ablauf: Zunächst wird das Tin trianguliert, falls dies noch nicht geschehen ist. Dann wird der Schwerpunkt des Querprofils berechnet. Danach wird für jeden Zeitschritt die Methode `berechne_h_tin_step()` aufgerufen, die die eigentliche Überflutung für einen Zeitreihenschritt ermittelt.

```
_method berechne_h_tin(...)
    _if _not the_tin.triangulated? _then
            the_tin.triangulate()
    _endif

    profil_rope << rope.new()
    _for profil _over ∀ Querprofile _loop
            min_rw, max_rw << profil.lw_stuetzpunkte.an_element().rw
            min_hw, max_hw << profil.lw_stuetzpunkte.an_element().hw
            _for stuetzpunkt _over ∀ Stützpunkte _loop
                    min_rw << min(stuetzpunkt.rw, min_rw)
                    min_hw << min(stuetzpunkt.hw, min_hw)
                    max_rw << max(stuetzpunkt.rw, max_rw)
                    max_hw << max(stuetzpunkt.hw, max_hw)
            _endloop
            the_rw << (min_rw + max_rw) / 2
            the_hw << (min_hw + max_hw) / 2
            profil_rope[i] << Coordinate.new(the_rw,the_hw)
    _endloop

    _for ∀ Zeitschritte _loop
            berechne_h_tin_step(...)
    _endloop
_endmethod
```

Die Methode

```
ofg_dar.berechne_h_tin_step(the_tin, i, ufer_table, date, profil_rope,
        display_tin, _optional items)
```

erstellt zuerst die Mengen für die schon bearbeiteten sites und die im aktuellen Schleifendurchlauf betrachteten sites. Ausserdem wird in `site_table` für jede untersuchte site das Querprofil, von dem aus die Überflutung erfolgte und das nächste Querprofil stromabwärts festgehalten.

```
_method berechne_h_tin_step(...)
    all_sites_set << equality_set.new()
    current_sites_set << equality_set.new()
    site_table << hash_table.new()
```

Als erstes wird dann zu jedem Querprofil für die näheste site der Wasserstand berechnet bzw. erhöht, falls schon ein niedriger Wasserstand von einem anderen Querprofil aus ermittelt wurde. Dies ist die Anfangssite, von der aus sich die Überflutung auf andere sites ausbreiten kann. Die site wird ausserdem in die Tabelle `site_table` aufgenommen, da sie weiter betrachtet werden muss - es können von ihr aus eventuell weitere sites überflutet werden. Es werden für jede site die Koordinaten des überflutenden Querprofils und die Koordinaten des vorhergehenden Querprofils gespeichert. Dies ist erforderlich, um wie oben beschrieben entscheiden zu können, ob die Überflutung von dieser site in einer bestimmten Richtung fortschreiten darf oder nicht. Wird ein Überflutungswert für eine site gesetzt oder erhöht, so geschieht dies in der Methode `ofg_tin.update_h(...)`. Sie wird weiter unten beschrieben.

```
prev_sec << v.collections[:lw_profil].at(_self.ofg_name, 1)
_for sec _over ∀ Profile _loop
        zeitreihen_wert << sec.ofg_zeitreihe.at(_self.ofg_name, sec.profil_nr, date)
        h << zeitreihen_wert.h
        coord << profil_rope[sec.profil_nr]
        _if sec.profil_nr = 1  _then
                prev_coord << coord
        _else
                prev_coord << profil_rope[sec.profil_nr -1]
        _endif
        site << the_tin.site_nearest(coord)
        _if the_tin.update_h(...) _then
                current_sites_set.add(site)
                site_table[site.id] << {coord, prev_coord}
        _endif
        prev_sec << sec
_endloop
```

Aus der vorherigen Schleife wurden einige sites ermittelt, die vom Fluss aus überflutet werden. Diese werden jetzt weiter betrachtet, da von ihnen aus weitere sites überflutet werden können. Es wird hier also nicht mehr die Überflutung Querprofil → site berechnet, sondern die Überflutung site x → site y.

Die äussere Schleife wird solange durchlaufen, wie im jeweils vorhergehenden Durchlauf weitere überflutete sites hinzukommen. Die jeweils neu überfluteten sites werden im nächsten Durchlauf daraufhin überprüft, ob von ihnen aus wiederum Überflutung ausgeht, usw. Alle überfluteten sites werden in `all_sites_set` aufgenommen. Die jeweils gerade betrachteten sites liegen in `current_sites_set`.

```
    _loop
        _if current_sites_set.size = 0 _then _leave _endif
        all_sites_set.add_all(current_sites_set)
        next_sites_set << equality_set.new()
        _for site _over current_sites_set.elements() _loop
            h << v.collections[:ofg_tin_z_reihenwert].at(
                date,"h", site).wert
            _for neighbour_site _over the_tin.natural_neighbours(site) _loop
                profil_vec << site_table[site.id]
                neighbour_coord << coordinate.new(neighbour_site.x,
                                                  neighbour_site.y)
```

Überflutung findet wie gefordert nur statt, wenn sich die betrachtete site näher an dem Querprofil, von dem die Überflutung ausgeht als am unterstromigen Querprofil befindet.

```
                if_ (neighbour_coord-profil_vec[2]).abs <
                    (neighbour_coord-profil_vec[1]).abs _orif
                    all_sites_set.includes?(neighbour_site) _then
                    _continue
                _endif
                _if the_tin.update_h(...) _then
                    next_sites_set.add(neighbour_site)
                    site_table[neighbour_site.id] << profil_vec
                _endif
            _endloop
        _endloop
        current_sites_set << next_sites_set
    _endloop
_endmethod
```

In diesem Zusammenhang muss noch die von `berechne_h_tin_step()` aufgerufene Methode

```
    ofg_tin.update_h(h, date, site, display_tin)
```

vorgestellt werden.

Sie ermittelt, ob eine Überflutung der site bei dem gegebenen Wasserstand eintritt. Ist dies der Fall, so wird für die angegebene site im Tin ein Objekt `ofg_tin_z_werte` erzeugt, an das ein Objekt der Klasse `ofg_tin_z_reihe` angehängt wird. Dieses enthält nun eine Anzahl von Zeitreihenwerten vom Typ `ofg_tin_z_reihenwert`. Dieses Objekt nimmt den für den gegebenen Zeitschritt angegebenen Wasserstand `h` auf. Bestehen die gegebenen Objekte schon, wird der Wasserstand betrachtet und modifiziert, wenn der neue Wasserstand höher ist. Wurde das `ofg_tin_z_reihenwert` -Objekt modifiziert, so wird auch die Grundwasser-neubildung neu berechnet.
Die Methode ist folgendermassen definiert:

```
_method ofg_tin.update_h(...)
    _if h <= (site.z / 1000.0)
    _then
        _return _false
    _endif
    z_werte << ofg_tin_z_werte.get_or_create(site)
    z_reihe << ofg_tin_z_reihe.get_or_create("h",site)
```

```
        z_reihen_wert << ofg_tin_z_reihenwert.get_or_create(date,"h",site)
        _if h <= z_reihen_wert.wert _then _return _false
        _endif
        z_reihen_wert.wert << h

        z_reihe << ofg_tin_z_reihe.get_or_create("gwnb", site)
        z_reihen_wert << ofg_tin_z_reihenwert.get_or_create(date, "gwnb", site)
        maechtigkeit << katasterflaeche.agrar_grundflaeche.gruendigkeit
        z_reihen_wert.wert << _self.calculate_gwnb(h-site.z/1000.0, z_werte. kf_wert, maechtigkeit)
        _return _true
_endmethod
```

Die Berechnung der Grundwasserneubildung findet in der Methode

```
        ofg_tin.calculate_gwnb(...)
```

statt. Die verwendete Formel ist

$$GWNB = \frac{k_f \cdot h \cdot \Delta t}{\Delta z} \qquad \textit{Gl. 6.24}$$

Mit $GWNB$ Grundwasserneubildung
k_f gesättigter hydraulischer Durchlässigkeitsbeiwert
h Überstauhöhe
Δt Zeitschritt
Δz Mächtigkeit des Bodens

Danach lautet die Methode:

```
_method ofg_tin.calculate_gwnb(...)
        step << as_time_interval(_self.ofg_dar.time_step)
        value << kf_wert*h*step.seconds/maechtigkeit
        _return value
_endmethod
```

Der Überflutungsgrad, d.h. der Wasserstand an den einzelnen Geländepunkten, wird nicht als Überstauhöhe *h* verwendet, da bei Überflutung nicht immer ein horizontaler Wasserspiegel über ein weites Gebiet ausgebildet wird. Dies wäre nur der Fall, wenn z.B. eine kleinere Mulde volläuft, nicht aber bei einem weitläufigen Tiefland, wo das Wasser bei Überschwemmung versickert, bevor es bis zum Wasserstand im Gerinne angestiegen ist. Der Überstau flacht sich hier also entsprechend der zunehmenden Entfernung von der Quelle immer mehr ab.
Aufgrund dieser schwer zu quantifizierenden Effekte wird der Wasserstand im Gerinne nicht als Überstauhöhe im Überflutungsbereich genommen, stattdessen wird mit einer fixen Überstauhöhe (z.B. 0.5 *m*) gerechnet.

6.5 Katasterflächen- und Agrar-Modul OGRUT AGR

Nachdem die Überflutung des digitalen Geländemodells wie im vorhergehenden Abschnitt beschrieben berechnet wurde, gilt es nun, zunächst die Zuordnung einzelner digitaler Geländepunkte zu den Katasterflächen zu ermitteln, in denen diese Geländepunkte liegen. Dann lässt sich die Überflutung der Katasterflächen aus der Überflutung des digitalen Geländemodells berechnen. An dieser Stelle findet eine Übertragung der Überflutung vom Punkt auf die Fläche statt. Danach kann aus der Überflutung die Grundwasserneubildung für eine Katasterfläche ermittelt werden.

Die Zuordnung site ↔ Katasterfläche kann statisch erfolgen, ist also nur dann nötig, wenn sich das digitale Geländemodell oder die Katastereinteilung ändert. Die Überflutungsdaten müssen jedoch für jeden Zeitreihenschritt separat berechnet werden.

Die im Zusammenhang mit der statischen Zuordnung und der dynamischen Berechnung erforderlichen Datenstrukturen sind im Teilmodul *OGRUT AGR* definiert bzw. werden von dort aus bedient (s. Abb. 6.5).

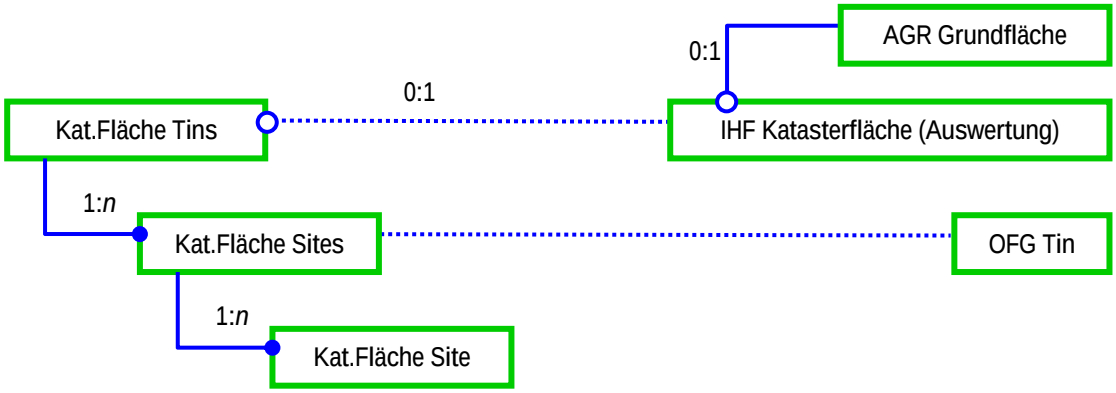


Abb. 6.12: Agrar-Datenmodell des OGRUT.

6.5.1 Realisierung des Teilmoduls *OGRUT AGR*

Bei der Berechnung der Überflutung von Katasterflächen durch Punkte des digitalen Geländemodells findet wie erwähnt eine Übertragung vom Punkt auf die Fläche statt. Das hierbei verwendete Gewichtungsverfahren kann prinzipiell flexibel gewählt werden. Gegenwärtig wird lediglich ein einfaches arithmetisches Mittel aus der Grundwasserneubildung an einzelnen digitalen Geländepunkten gewählt, ohne Gewichtung der flächenmässigen Bedeutung (z.B. THIESSEN-Polygone). Diese Methode erscheint ausreichend, da innerhalb einer einzelnen Katasterfläche in der Regel keine allzu grossen Variabilitäten bodenkundlicher Merkmale zu erwarten sind.

Für jeden digitalen Geländepunkt wird dessen Lage in einem Hydropedotop ermittelt, da sich hieraus der gesättigte hydraulische Durchlässigkeitsbeiwert k_f des Bodens ergibt.

6.6 Finites Elemente-Modul *OGRUT ISO*

Das Iso-Modell wird von *FEFLOW* zur Simulation der Grundwasserstände verwendet. *FEFLOW* verlangt daher die Eingabe von Randbedingungen auf Basis dieses Finite Elemente-Netzes. Entsprechend wird in *OGRUT* ein solches Iso-Modell implementiert. Ein Auschnitt aus diesem Netz ist in Abb. 6.13 dargestellt.

Abb. 6.13: Ausschnitt aus dem Finite Elemente-Netz im Bereich des Krummbachs. Man erkennt, dass das Netz im Bereich von Entnahmebrunnen verdichtet ist.

FEFLOW bietet die Möglichkeit, über einen *Interface Manager* (IFM) Daten mit anderen Programmen über externe Dateien auszutauschen. Über dieses Werkzeug lassen sich z.B. gezielt Momentaufnahmen bestimmter Grundwasserparameter für einen einzelnen Zeitschritt machen. In JUNGHANS (1998) und BOLD (2000) wird *FEFLOW* für ein Grundwassermodell des Zartener Beckens verwendet. Die Randbedingungen der Simulation werden aus *OGRUT* geliefert, die Ergebnisse über den Interface Manager an *OGRUT* zurückgeliefert. Zur Simulation verwendet *FEFLOW* ein Netz finiter Elemente über mehrere Grundwasserstockwerke.

Das Datenmodell eines finites Elemente-Netzes basiert im wesentlichen auf Knoten und Elementen. Dabei wird jedes Element durch eine Dreiecksfläche dargestellt, deren Ecken durch drei Knoten gebildet wird. Jeder Knoten kann seinerseits Eckpunkt einer variablen Zahl von Dreiecksflächen sein. Eine Übersicht über das Datenmodell ist in Abb. 6.14 dargestellt.

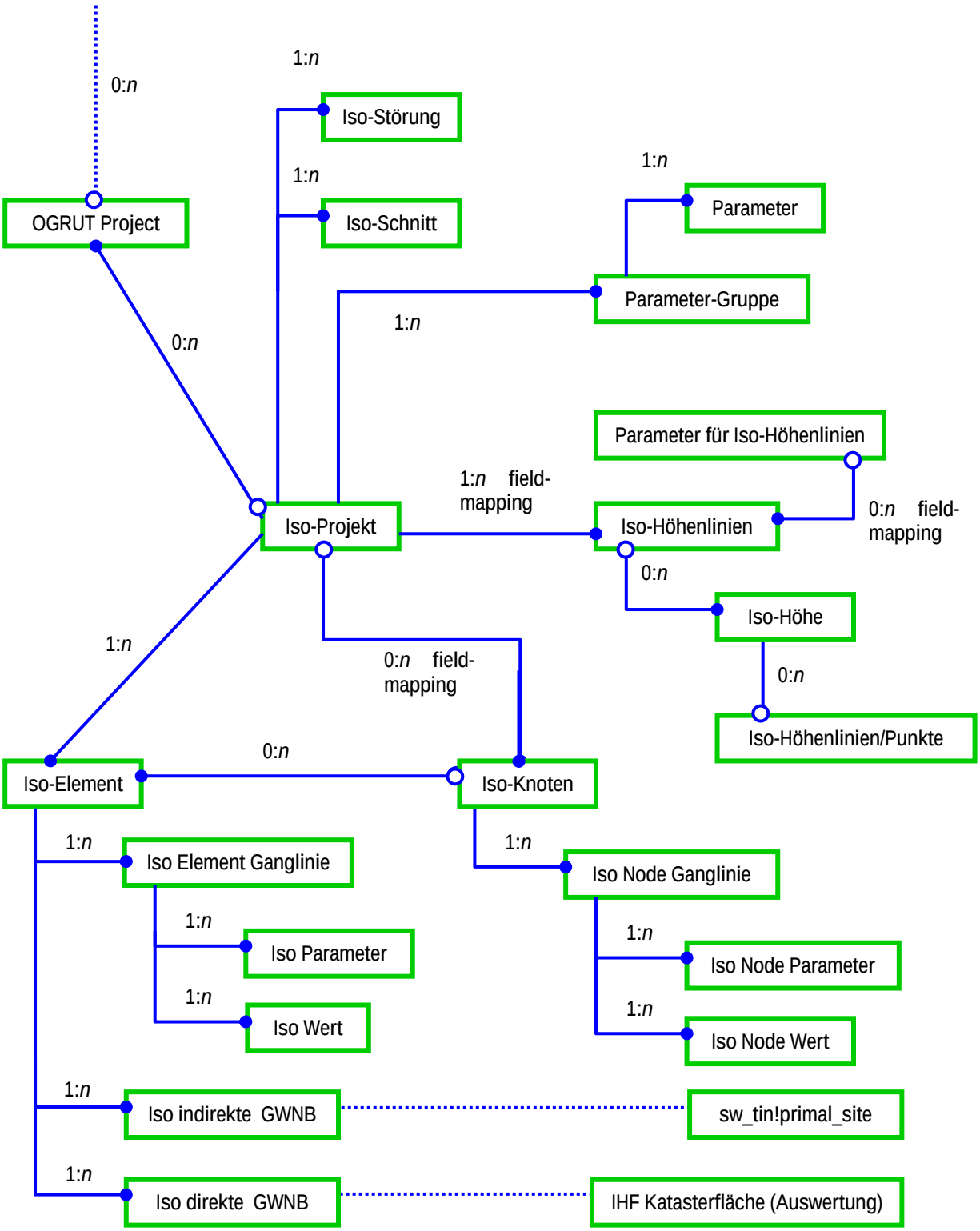


Abb. 6.14: Finite Elemente-Datenmodell des OGRUT.

6.6.1 Realisierung des Teilmoduls *OGRUT ISO*

Die Zeitreihenwerte der Grundwasserneubildung an den Katasterflächen werden auf einzelne Iso-Element übertragen. Dabei wird der Wert jeder Katasterfläche gemäss ihrem Flächenanteil am finiten Element gewichtet. *FEFLOW* erwartet die Randbedingungen der Grundwasserneubildung an Iso-Elementen. Die Grundwasserstände werden von *FEFLOW*

jedoch für Iso-Knoten zurückgeliedert (s. Abb. 6.15). Daher wird sowohl für Iso-Elemente als auch Iso-Knoten eine Datenstruktur zur Verwaltung von Zeitreihen definiert.

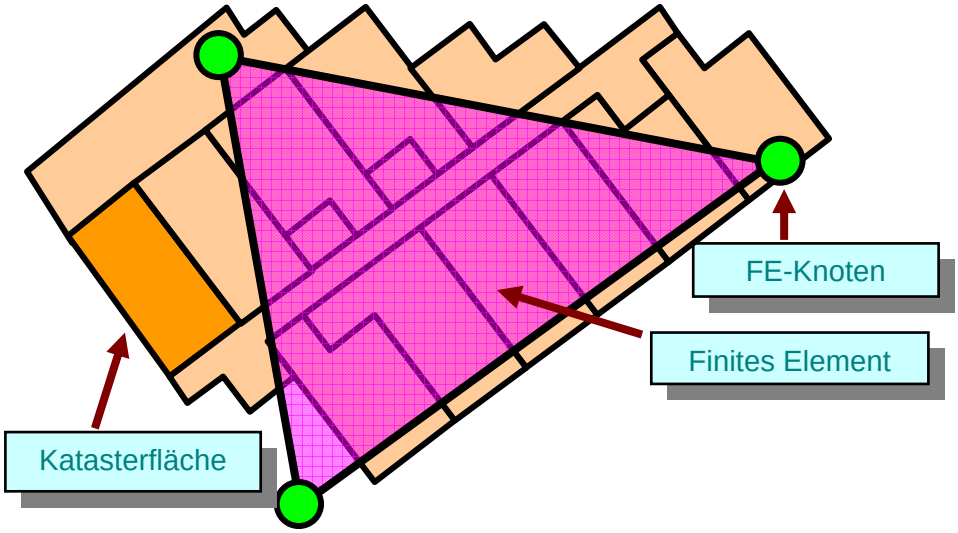


Abb. 6.15: Die Übertragung der Grundwasserneubildung von Katasterflächen auf Iso-Elemente geschieht durch Gewichtung der Flächenanteile der einzelnen Katasterflächen an einem Iso-Element. Die Zeitreihen der Grundwasserneubildung werden an die Iso-Elemente angehängt. Die Grundwasserstände, die FEFLOW zurückliefert, werden dagegen an Iso-Knoten angehängt.

Berechnet man die indirekte Grundwasserneubildung für die einzelnen Isoflächen direkt aus den Zeitreihenwerten an den digitalen Geländepunkten, die in der Isofläche liegen, so erhält man nur einen Teil der Daten. Die Zeitreihen für die direkte Grundwasserneubildung aus Niederschlag werden von *BOMET* nämlich an den Katasterflächen verwaltet. Daher wird normalerweise wie folgt vorgegangen:

- Durch entsprechende *BOMET*-Funktionen wird der Niederschlag (auf Basis von Katasterflächen) berechnet.
- Die indirekte Grundwasserneubildung aus flächenhafter Überflutung wird durch Funktionen, berechnet, die in der Benutzeroberfläche von *OGRUT* bereitgestellt werden. Dann wird die indirekte Grundwasserneubildung als weitere Zeitreihe für Katasterflächen berechnet.
- Danach wird die *gesamte* Grundwasserneubildung für eine Katasterfläche berechnet. Dies geschieht durch eine Methode, die die Grundwasserneubildung aus Niedererschlag mit dem HAUDE-RENGER-Verfahren berechnet und die Zeitreihe für die indirekte Grundwasserneubildung hinzuaddiert. Die entsprechende Methode wird im *BOMET*-Menu aufgerufen.
- Dann wird die Grundwasserneubildung für die Isoelemente aus der Grundwasser-neubildung der Katasterflächen übertragen.

Auf diese Art erhält man die gesamte Grundwasserneubildung für die einzelnen Isoflächen.

6.7 Die graphische Benutzeroberfläche

Die graphische Benutzeroberfläche ermöglicht es dem Benutzer, die zentralen Funktionen von *OGRUT* von einer einheitlichen Schnittstelle aus zu erreichen. Ausserdem wird durch verschiedene Symbole und Statusanzeigen unterstützt, dass die einzelnen Arbeitsschritte in

der richtigen Reihenfolge durchgeführt werden. Nur wenn man spezielle Fähigkeiten von *OGRUT* nutzen möchte, muss man auf die direkte Ausführung der Funktionen an der Kommandozeile zurückgreifen.
Die Struktur des Datenmodells der graphischen Oberfläche ist in Abb. 6.16 zu sehen:

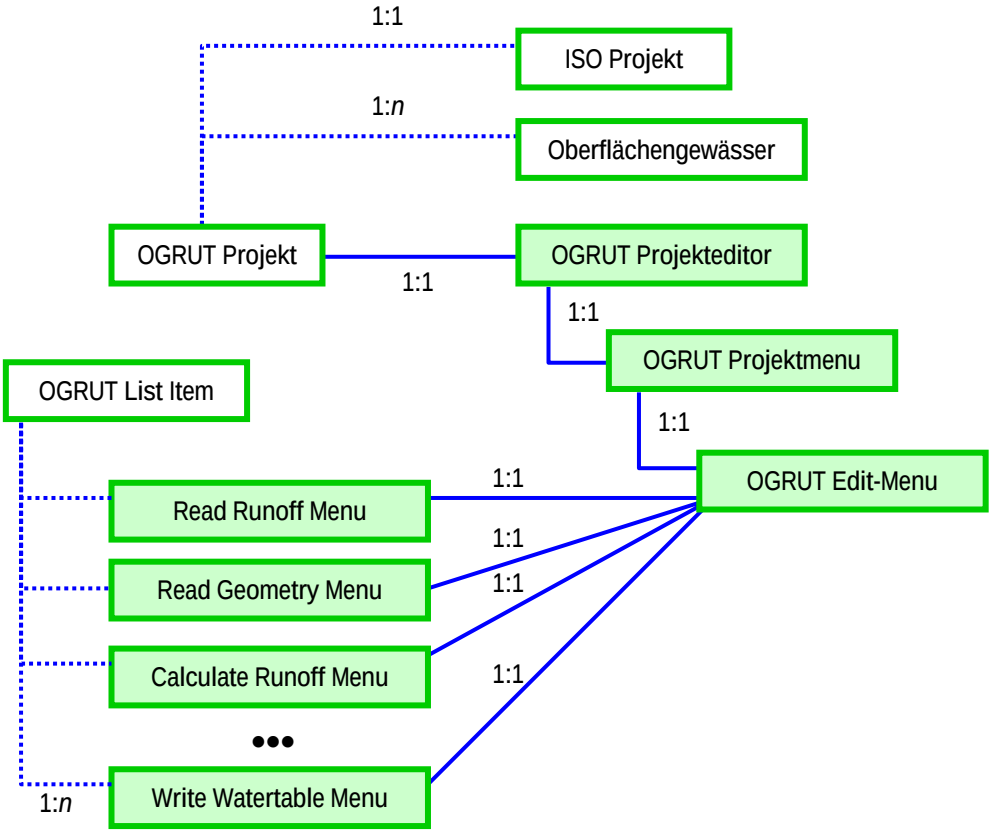


Abb. 6.16: Datenmodell der graphischen Benutzeroberfläche von OGRUT.

Das gesamte Projekt wird in einer Instanz der Klasse OGRUT-Projekt verwaltet. In jedem Projekt können mehrer Gerinne (Klasse Oberflächengewässer) und ein Finites Elemente-Modell (Klasse ISO Projekt) enthalten sein. OGRUT-Projekte können wie jede Datenbankklasse in *Smallworld* durch einen Standardeditor bearbeitet werden (OGRUT Projekteditor). Dieser wurde jedoch so ergänzt, das von ihm aus ein weiterer Editor zur Bearbeitung des aktuellen OGRUT-Projektes geöffnet werden kann. Hier kann festgelegt werden, welche Oberflächengewässer und welches Iso-Modell im Projekt enthalten sein sollen. Ausserdem kann man von hier aus einen weiteren Editor aufrufen, das OGRUT Edit-Menu. Von hier aus sind alle wesentlichen Funktionen abrufbar, mit denen Daten eingelesen, berechnet, gespeichert und Anwendungen gestartet werden können. Die einzelnen Klassen und Funktionen sind im Anhang beschrieben.

6.7.1 Bedienelemente der Benuzteroberfläche

Die Benutzeroberfläche von OGRUT hat zum Zweck, alle wesentlichen Aufgaben und Funktionen an zentraler Stelle zu vereinen. Ausserdem sollen informative Statusanzeigen dafür sorgen, dass sofort erkennbar ist, welche Funktionen für welche Elemente des aktuellen Projektes wie weit fortgeschritten sind.

Am Anfang wird ein bestehendes OGRUT-Projekt ausgewählt oder ein neues Projekt erzeugt. Dies geschieht vom OGRUT-Projekt-Fenster aus (s. Abb. 6.17). Durch Betätigen des Buttons

öffnet man ein weiteres Fenster, in dem das Projekt zusammengestellt werden kann (s. Abb. 6.18).

Abb. 6.17: Das OGRUT Projekt-Fenster ist ein modifizierter Standardeditor für diese Klasse.

Abb. 6.18: Das OGRUT-Projekt-Menu dient zum einfachen Auswählen der Bestandteile des Projektes.

In diesem Fenster gibt es zwei Listen mit Flüssen und eine Liste mit Iso-Projekten. In der linken Liste sind alle Flüsse enthalten, die in der Datenbank vorhanden sind. In der rechten Liste sind die Flüsse aufgeführt, die dem aktuellen OGRUT-Projekt zugerechnet werden. Um einen Fluss dem Projekt hinzuzufügen, muss er einfach in der linken Liste markiert werden. Durch Betätigen des Buttons wird er in die rechte Liste und damit in das aktuelle OGRUT-Projekt aufgenommen. Umgekehrt kann ein Fluss aus dem Projekt entfernt werden, indem er in der rechten Liste markiert wird und danach der Button betätigt wird.

In jedem OGRUT-Projekt kann immer nur ein Iso-Projekt enthalten sein. Dieses Iso-Projekt wird rechts neben der Liste der verfügbaren Iso-Projekte angezeigt. Es wird geändert, indem das gewünschte neue Iso-Projekt in der Iso-Projekt-Liste markiert wird und dann der Button betätigt wird.

Durch den Schalter wird das Hauptfenster geöffnet, von dem aus alle wesentlichen Funktionen von *OGRUT* erreichbar sind (s. Abb. 6.19).

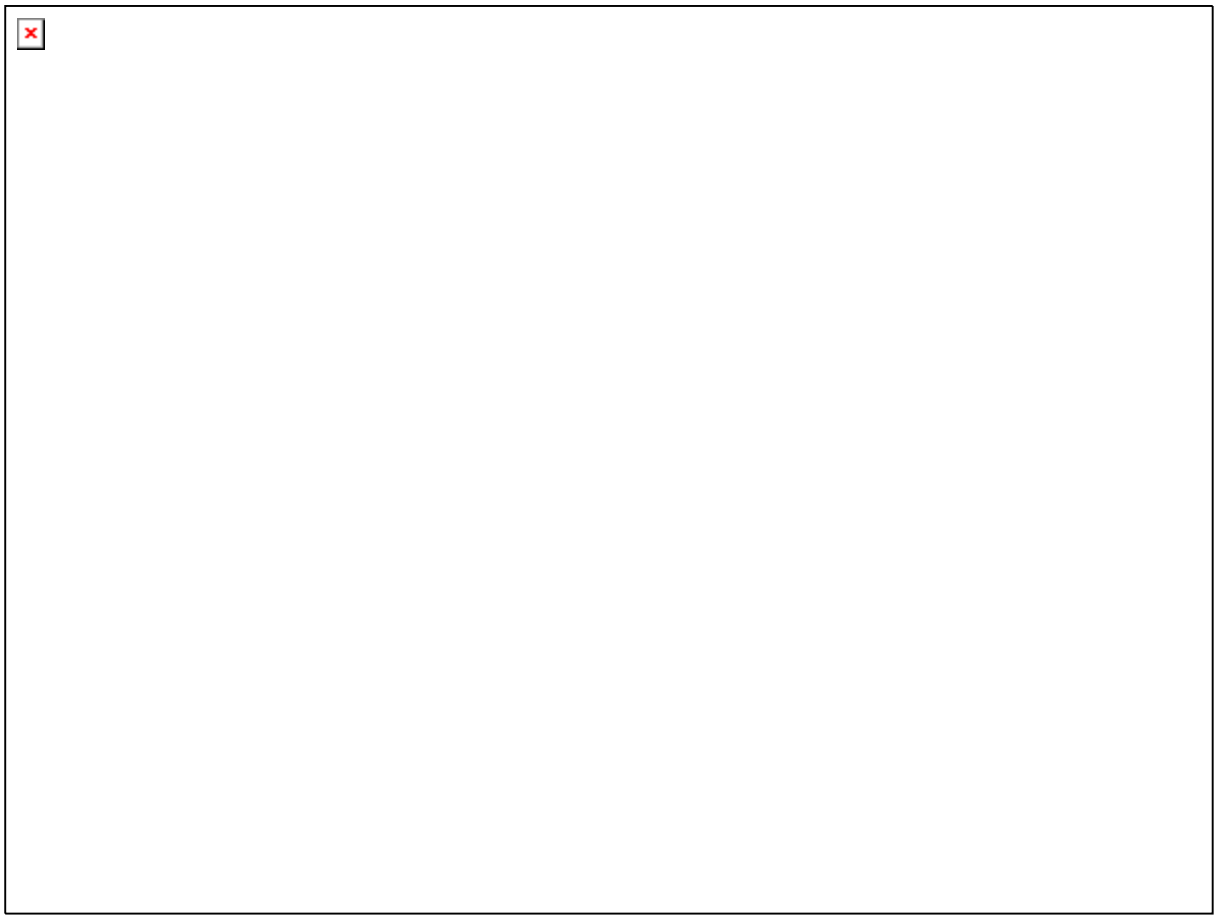


Abb. 6.19: Das OGRUT-Editorfenster wird vom OGRUT-Projektdialog aus geöffnet. Es vermittelt einen Überblick über den Zustand des Projektes und erlaubt die Ausführung der einzelnen Rechenschritte und die Eingabe bzw. Ausgabe von Daten.

Die einzelnen Funktionen sind nach Kategorien gruppiert. Die erste Gruppe betrifft das Einlesen von Daten aus externen Dateien, die zweite Gruppe berechnet statische Zuordnungen. Auf der rechten Seite des Fensters finden sich die Gruppen für die Berechnung von Zeitreihendaten und das Auslesen von Daten in externe Dateien. Jede Funktion ist durch einen Schalter erreichbar.
Die Funktionen können nicht in beliebiger Reihenfolge ausgeführt werden. Dies wird durch das OGRUT-Edit-Menu-Fenster angezeigt, indem hinter jeder Funktion ein Symbol mit drei möglichen Zuständen angezeigt wird. Diese Zustände geben Auskunft über den Status der entsprechenden Funktion:

- Die Funktion kann noch nicht auf alle Elemente im Projekt angewendet werden, da für mindestens ein Element die nötigen Vorarbeiten noch nicht erledigt wurden.
- Die Funktion wurde noch nicht auf alle Elemente im Projekt angewendet. Sie jedoch auf alle Elemente anwendbar, da die Voraussetzungen zur Anwendung erfüllt sind. D.h., alle Vorarbeiten wurden für alle Elemente durchgeführt.
- Die Funktion wurde bereits auf alle im Projekt enthaltene Elemente angewendet und ist somit abgeschlossen. Sie kann dennoch erneut aufgerufen werden. Bei Funktionen, die die Ausgabe von Ergebnisdaten in eine externe Datei zum Zweck haben, wird jedoch nicht überprüft, ob schon entsprechende Dateien existieren.

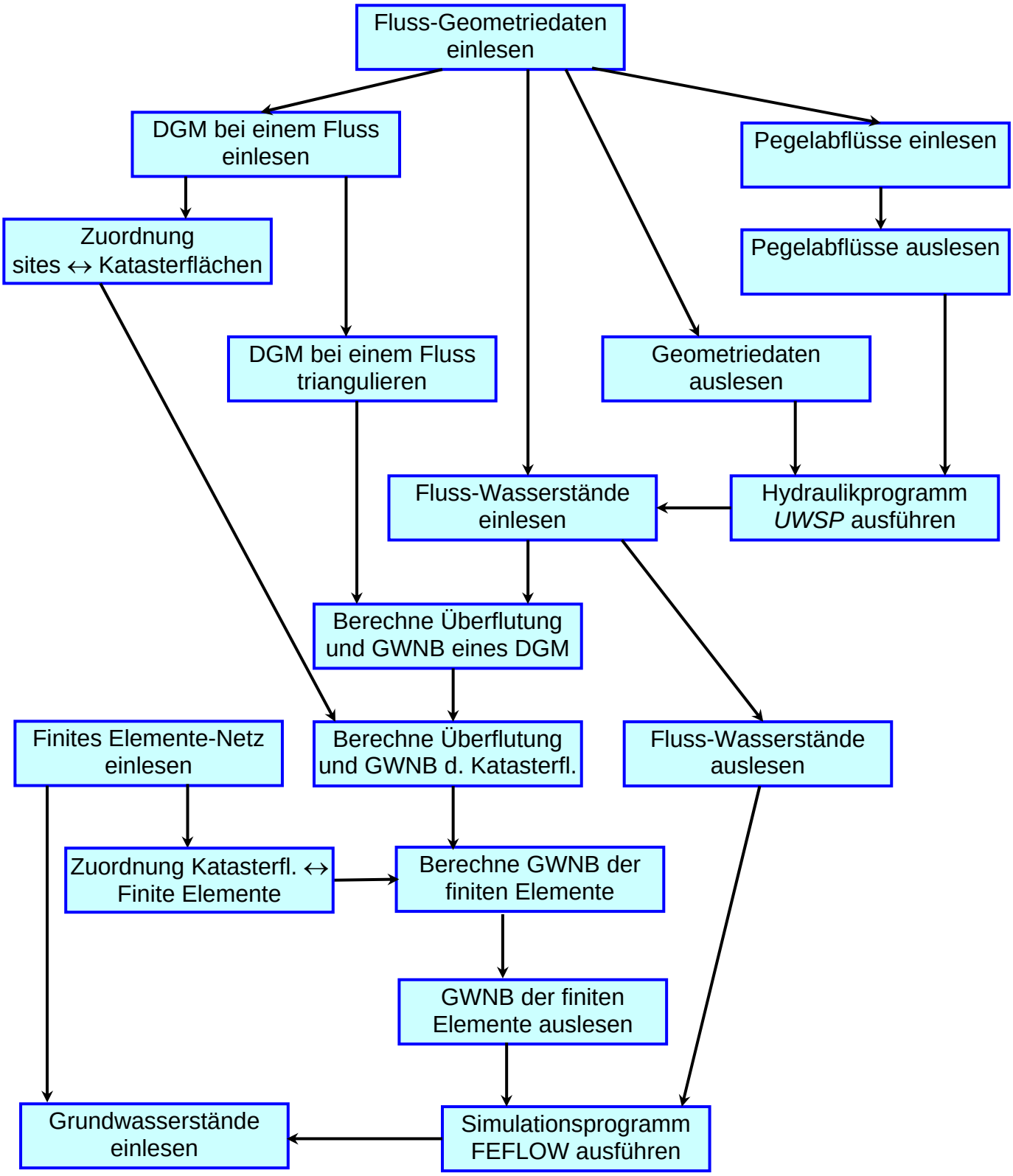


Abb. 6.20: Das Diagramm zeigt die gegenseitigen Beziehungen der Funktionen in OGRUT. Die Pfeile zeigen an, in welcher Reihenfolge die Funktionen ausgeführt werden können. Gehen z.B. Pfeile von drei Funktionen auf eine vierte Funktion zu, so müssen alle drei Funktionen ausgeführt worden sein, bevor die vierte Funktion ausführbar ist.

Die Reihenfolge, in der die verschiedenen Funktionen ausgeführt werden können, entspricht dem Schema in Abb. 6.20. In Tab. 6.5 sind die Vorbedingungen für die einzelnen Funktionen zusammengestellt:

Tab. 6.5: Vorbedingungen der einzelnen Funktionen der OGRUT-Benutzeroberfläche. Sind die Vorbedingungen nicht erfüllt, so wird dies anhand der Kontrollkästchen hinter den Funktionen angezeigt.

	Funktion	**Vorbedingungen**
1	Tin bei einem Fluss einlesen	-
2	Finites Elemente-Netz einlesen	-
3	Geometriedaten für einen Fluss einlesen	-
4	Wasserstandszeitreihen für einen Fluss einlesen	3
5	Grundwasserstands-Zeitreihen einlesen	2
6	Abflusszeitreihen an Pegeln eines Flusses einlesen	3
7	Statische Zuordnung zwischen einem Tin und Katasterflächen festlegen	1
8	Statische Zuordnungen zwischen Katasterflächen und finiten Elementen festlegen	2
9	*UWSP* ausführen	3, 6
10	Ein Tin triangulieren	1
11	Ein Tin von einem Gerinne überfluten	4, 10
12	Katasterflächen überfluten	7, 11
13	Das Finite Element-Netz überfluten	2, 12
14	Geometriedaten eines Flusses in eine externe Datei ausgeben	3
15	Wasserstandszeitreihen für einen Fluss	4
16	Abflusszeitreihen an den Pegeln eines Flusses in eine externe Datei ausgeben	6
17	Grundwasserstands-Zeitreihen in eine externe Datei ausgeben	13

Durch das OGRUT-Edit-Menu sind die Untereditoren für die einzelnen Funktionen erreichbar. Die Beschreibung für diese Editoren findet sich in Anhang C.

Die Vorbedingungen für die Funktionen werden in der Methode `ogrut_project.init()` wie folgt definiert:

```
_method ogrut_project.init()
    # ...
    .necessary << hash_table.new()
    .necessary[:tin_in] << {}
    .necessary[:fe_in] << {}
    .necessary[:geometry_in] << {}
    .necessary[:watertable_in] << {:geometry_in}
```

```
    .necessary[:gwtable_in] << {:fe_in}
    .necessary[:runoff_in] << {:geometry_in}
    .necessary[:tin_area] << {:tin_in}
    .necessary[:area_fe] << {:fe_in}
    .necessary[:run_wsp] << {:geometry_in, :runoff_in}
    .necessary[:triangulate_tin] << {:tin_in}
    .necessary[:flood_tin] << {:triangulate_tin, :watertable_in}
    .necessary[:flood_area] << {:flood_tin, :tin_area}
    .necessary[:flood_fe] << {:flood_area, :fe_in}
    .necessary[:geometry_out] << {:geometry_in}
    .necessary[:watertable_out] << {:watertable_in}
    .necessary[:runoff_out] << {:runoff_in}
    .necessary[:d_gwnb_out] << {:flood_fe}
    # ...
_endmethod
```

6.8 Fazit

OGRUT erweitert *WAQIS* um neue Werkzeuge, die es erlauben, ein Grundwassermodell weitgehend automatisch zu parametrisieren. Dadurch wird die operationelle Nutzung und die Weiterführung von Zeitreihen vereinfacht. Durch die zentrale Verwaltung der relevanten Eingabedaten für das Grundwassermodell wird das Simulationsprogramm *FEFLOW* in das GIS integriert. Eine einheitliche, von *Smallworld* aus zu bedienende Oberfläche verleiht dieser Integrationslösung Nachdruck.

Die Simulationsergebnisse können in die Geodatenbank zurückgelesen werden und stehen dort für weitere Auswertung und Visualisierung bereit.

Die umfangreichen Zeitreihen der Grundwasserneubildung erlauben es *FEFLOW*, Kalibrierung und Modellvorhersage zu verbessern und damit die Leistungsfähigkeit des Modells zu steigern.

Mit *OGRUT* steht dem Trinkwasserversorger innerhalb von *WAQIS* ein weiterer Baustein zu einer umfassenden Geodatenbank zur Verfügung, die im Laufe der Zeit zu einer zentralen Schnittstelle für alle die Wasserqualität betreffenden Informationen und Werkzeuge wird. Das GIS *Smallworld* hat sich in diesem Zusammenhang wie schon bei dem Modul *BOMET* als geeignete Plattform einer integrativen Geodatenbank erwiesen.

Probleme, die die Ausführungsgeschwindigkeit von Berechnungen, insbesondere beim zentralen Überflutungsalgorithmus des digitalen Geländemodells betreffen, sind zwar nicht von der Hand zu weisen. Sie lassen sich aber bei ausreichend hohem "Leidensdruck" lösen, indem der kritische Teil des Programmcodes in ein externes Programm verlagert wird. Dieses kann dann über Systemaufrufe oder über ACP an *Smallworld* angebunden werden. Der Eindruck eines integrativen Systems wird davon nicht gestört, da die Benutzeroberfläche von einer solchen Entscheidung nicht beeinflusst wird.

7 Ergebnisse

7.1 OGRUT und UWSP als neue Bausteine für WAQIS

Aufgabe dieser Arbeit war es, die Leistungsfähigkeit des Grundwassermodells in *WAQIS* zu erhöhen. Die Weiterführung von Zeitreihen und die operationelle Anwendung des Modells sollten erleichtert werden. Die Integration in die Geodatenbank von *WAQIS* sollte verbessert werden. Durch zeitlich und räumlich hochauflösende Randbedingungen als Eingabe für das Modell sollte dessen Genauigkeit verbessert werden.

Um diese Ziele zu erreichen, wurde *WAQIS* um eine weitere Komponente, *OGRUT* ergänzt. Um alle gewünschten Eingabedaten für *FEFLOW* zu erhalten, wurde ausserdem das Gerinnehydraulikprogramm *UWSP* programmiert.

Abb. 7.1 zeigt noch einmal im Überblick die einzelnen Komponenten, um die *WAQIS* erweitert wurde, in ihrem Zusammenspiel. Als Randbedingungen für das Grundwassermodell in *FEFLOW* werden Zeitreihen der Grundwasserneubildung bereitgestellt. Um diese zu erhalten, sind einige Vorarbeiten erforderlich.

Zunächst werden Geometriedaten von Oberflächengwässern in die Geodatenbank aufgenommen. Hinzu kommen Zeitreihen des Abflusses an Pegeln, die ebenfalls in der Datenbank gespeichert werden. Dieser Datenbestand wird in einem geeigneten Ausgabeformat in externe Dateien geschrieben, die als Eingabe für das Gerinnehydraulikprogramm *UWSP* dienen. Das Programm berechnet eindimensionale, stationäre, ungleichförmige Wasserspiegellinien für eine ganze Zeitreihe von Abflussdaten mehrerer Pegel. Dabei wird sowohl der strömende als auch der schiessende Abfluss mit zwei unterschiedlichen Verfahren (BERNOULLI-Gleichung einerseits, Impulssatz andererseits) berechnet. Anhand der spezifischen Kraft wird entschieden, welches Abflussregime kontrolliert. Die Ausgabe der Ergebnisse erfolgt wiederum in eine Datei. Die Ergebnisse werden ihrerseits wieder in die Datenbank transferiert und den Querprofilen der Gerinne zugeordnet. Dieser Teil der Arbeit wird vom Modul *OGRUT OFG* übernommen. Alle hierfür nötigen Funktionen bis hin zum Aufruf von *UWSP* mit den erforderlichen Parametern sind für den Benutzer über die graphische Bedienoberfläche erreichbar. Durch Statusanzeigen für die einzelnen Funktionen und die einzelnen Gewässer wird es dem Benutzer ermöglicht, einen schnellen Überblick über den Stand der Arbeiten zu erhalten und die Funktionen in der richtigen Reihenfolge zu bedienen.

Die nächsten Schritte sind im Teilmodul *OGRUT DGM* enthalten. Dabei wird jedem Gewässer ein digitales Geländemodell zugeordnet. Sind für die einzelnen Gewässer keine separaten Geländemodelle vorhanden, so kann ein grosses Geländemodell durch die Funktion `tin.aufteilen()` in eine Anzahl kleinerer Geländemodelle zerlegt werden, die jeweils einem Gerinne zugeordnet sind. Anhand der Zeitreihen des Wasserstandes an den Gerinnen wird berechnet, in wie weit die einzelnen Geländepunkte des Geländemodells von Überflutung betroffen sind. Der hierfür verwendete Algorithmus wurde detailliert in Kap. 6.4.2 beschrieben. Er ist sehr aufwendig und zeitintensiv, obwohl das Auffinden benachbarter Geländepunkte anhand der Triangulation beschleunigt wird.
Nachdem die durch ein Gerinne überfluteten Geländepunkte eines Zeitpunktes ermittelt wurden, wird die indirekte Grundwasserneubildung an den Geländepunkten berechnet. Dies geschieht unter Verwendung von Durchlässigkeitswerten, die aus Hydropedotopen des *BOMET* ermittelt werden.

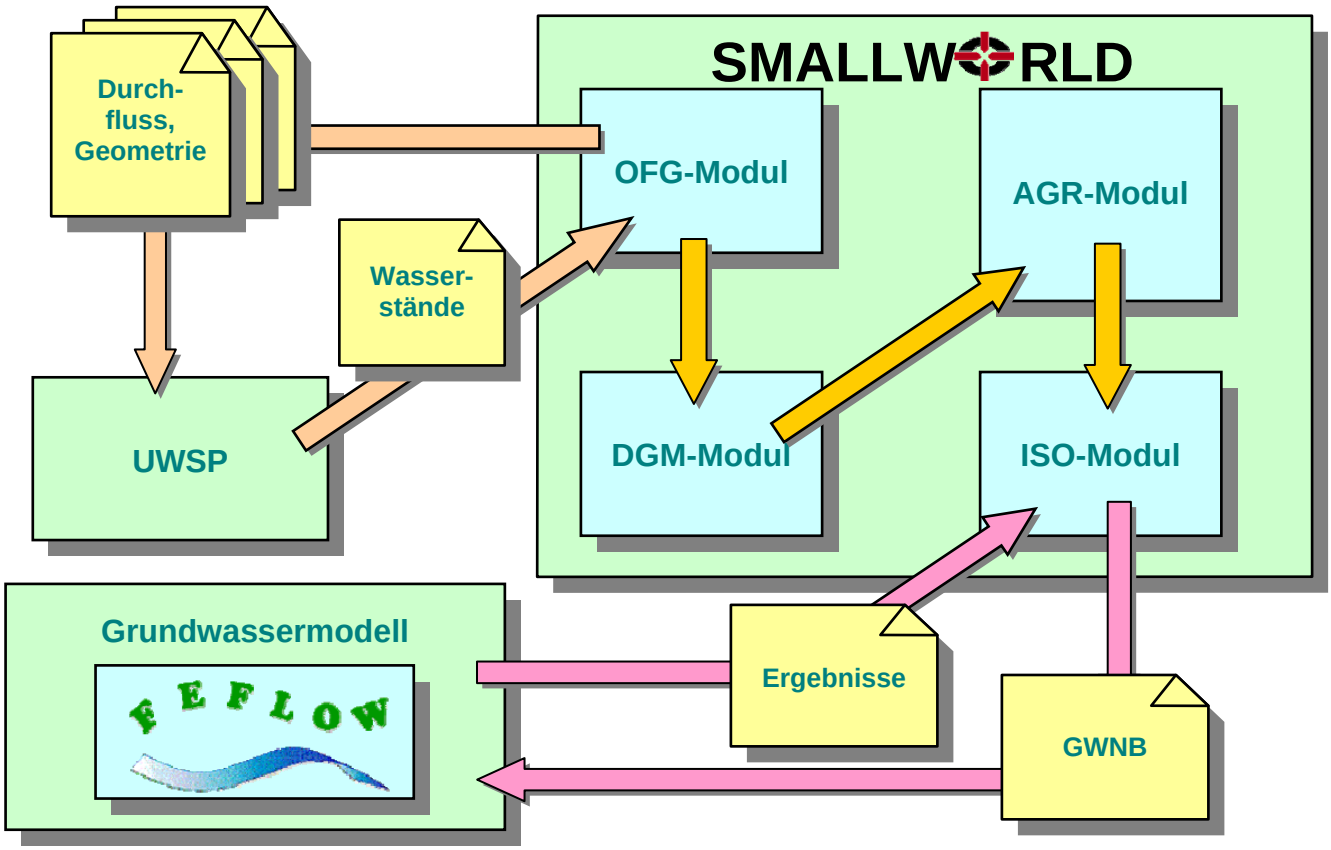


Abb. 7.1: Diagramm des Datenflusses in OGRUT.

Um die Grundwasserneubildung für die einzelnen Katasterflächen berechnen zu können, wird zuerst eine statische Zuordnung zwischen digitalen Geländepunkten und Katasterflächen erzeugt. Diese statische Zuordnung ist bei Weiterführung von Zeitreihen weiterhin gültig. Sie muss also nur dann wieder neu berechnet werden, wenn sich neue Katasterflächen oder neue Geländepunkte ergeben. Anhand der den Katasterflächen zugeordneten Geländepunkte wird die indirekte Grundwasserneubildung für die Katasterflächen berechnet. Diese Zeitreihen werden dann für jeden Zeitschritt mit der direkten Grundwasserneubildung aus Niederschlag ergänzt. Somit erhält man als Summe die gesamte Grundwasserneubildung für die Katasterflächen. Die entsprechenden Funktionen sind Inhalt des Teilmoduls *OGRUT AGR*.

Die letzen Auswertungsschritte vor der Übergabe der Daten an *FEFLOW* sind dem Teilmodul *OGRUT ISO* vorbehalten. Hier wird wiederum eine statische Zuordnung ermittelt, in diesem Fall zwischen Katasterflächen und finiten Elementen des Iso-Modells. Auch diese Einteilung muss erst dann wieder neu errechnet werden, wenn sich entweder neue Katasterflächen oder ein neues Iso-Modell ergeben. Anhand der Zuordnung lassen sich die Zeitreihen der gesamten Grundwasserneubildung von den Katasterflächen auf die finiten Elemente übertragen. Nun liegen sie in der Form vor, in der sie *FEFLOW* als Eingabe erwartet. Die Daten werden in externe Dateien ausgegeben und *FEFLOW* kann aufgerufen werden. Das automatische ausführen von *FEFLOW* von der *OGRUT*-Oberfläche aus ist noch nicht voll umgesetzt. Eine entsprechende Implementation bedarf aber nur noch weniger Schritte.

Zuletzt werden die von *FEFLOW* gelieferten Modellergebnisse nach *OGRUT* zurückgenommen und so ebenfalls Teil der umfassenden Geodatenbank.

Durch die im Iso-Modul enthaltenen Analysewerkzeuge können z.B. Isolinien verschiedener Grundwasserparameter visualisiert werden. Auch einer Weiterverarbeitung der Grundwasserdaten steht nichts im Weg. Die Visualisierbarkeit der Ergebnisdaten aus *FEFLOW* wurde noch nicht im angestrebten Umfang erreicht. Will man z.B. einen bestimmten Zeitpunkt der Zeitreihe darstellen, so ist es zunächst noch nötig alle gewünschten Werte in ein speziell dafür vorgesehenes Feld zu übertragen. Diese Funktion wird gegenwärtig noch nicht angeboten. Dennoch ist auch hier ist der Aufwand für eine Implementation nicht mehr gross. Um nicht den Kern der Funktionalität zu vernachlässigen wurde den Visualisierungsfunktionen eine niedrigere Priorität zugestanden (Prädikat "optional" bis "wünschenswert"), so dass hier noch Ansätze für Verbesserungen gegeben sind.

Den Fluss der Daten versucht die Abb. 7.2 noch einmal zu veranschaulichen. Man erkennt auch hier die einzelnen Stationen des Berechnungsablaufs, um die von *FEFLOW* gewünschten Eingabeparameter zu erhalten.

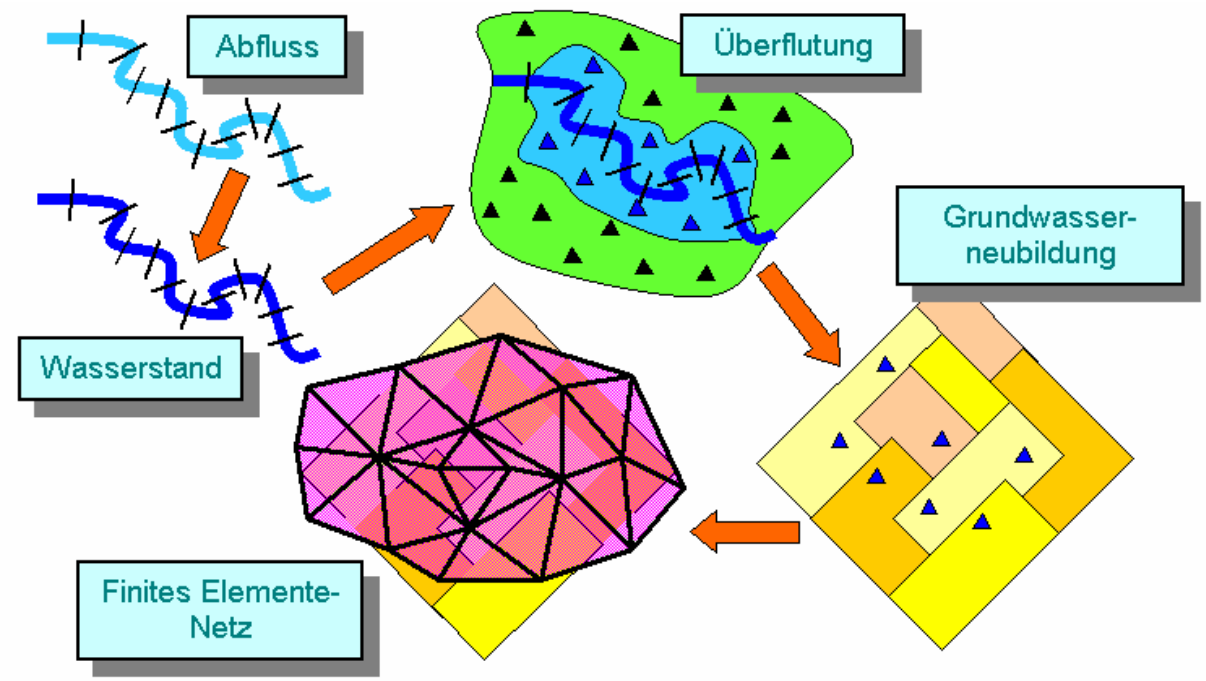


Abb. 7.2: Graphische Übersicht über den Ablauf der Berechnung der Grundwasserneubildungsdaten.

7.2 Erste Berechnungen der Überflutungsflächen

Um erste Ergebnisse bei der Berechnung von Überflutungsflächen zu gewinnen, wurde testweise für ein Hochwasserereignis die Überflutung durch die Gerinne Brugga, Eschbach und Krummbach berechnet. Hierfür wurden vorläufig noch unkalibrierte Wasserstände verwendet, lediglich für den Krummbach wurde eine Kalibrierung vorgenommen.

Nach WASSERWIRTSCHAFTSAMT FREIBURG (1988) bereitet die Festlegung der Ausuferungsflächen im Gelände vor dem Golfplatz (Unterlauf des Krummbachs) und auf dem Golfplatz selbst Schwierigkeiten. Das Bachbett ist hier flach und das Gelände links und rechts des Krummbachs fällt im Norden zur Dreisam bzw. im Süden zur Brugga ab. Bei Hochwasser ufert der Bach auf längeren Strecken aus und ergiesst sich breitflächig nach Norden in die

Dreisam, nach Süden in die Brugga. Eine sinnvolle hydraulische Berechnung der seitlich tatsächlich abfliessenden Wassermassen lässt sich nicht durchführen, d.h. es lässt sich keine Aussage darüber machen, welche Wassermenge aus dem Bett des Krummbachs in die Brugga bzw. in die Dreisam abfliesst und welche Restwassermenge im Bachbett des Krummbachs verbleibt (WASSERWIRTSCHAFTSAMT FREIBURG, 1988).
Die beispielhafte Berechnung der Wasserspiegellagen durch *UWSP* und die anschliessende Berechnung der Überflutungsflächen durch *OGRUT* für ein Hochwasser am 17.12.1989 kommt zu ähnlichen Ergebnissen wie das WASSERWIRTSCHAFTSAMT FREIBURG. Wie aus Abb. 7.3 ersichtlich, ist die durch den Krummbach hauptsächlich von Überflutung betroffene Fläche vor und im Unterlauf des Krummbachs (Bereich Golfplatz) zu finden. Ein flächenhafter Übertritt, insbesondere in die Brugga, konnte bei der Berechnung nicht festgestellt werden. Der Spitzenabfluss für das Hochwasser (auf Tageswerten basierend) lag allerdings mit 14.18 m^3s^{-1} leicht unter dem vom WASSERWIRTSCHAFTSAMT FREIBURG betrachteten Bemessungshochwasser von 15 m^3s^{-1} .

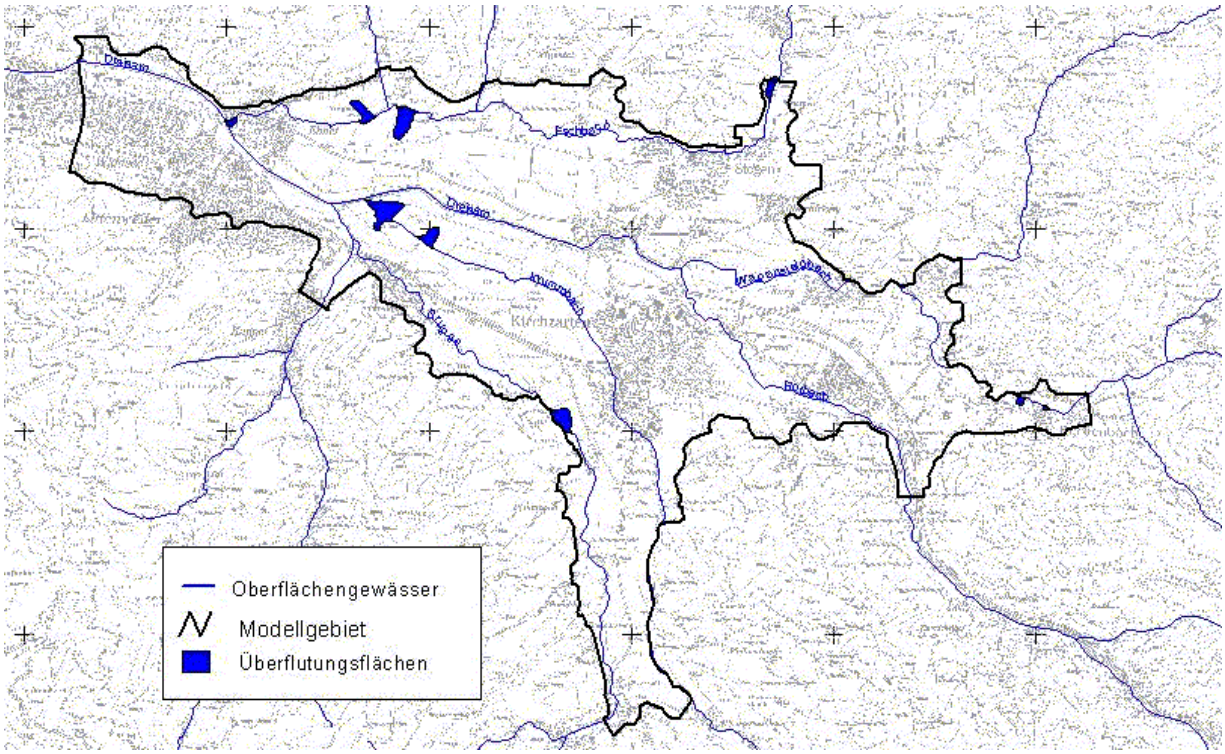


Abb. 7.3: Mit UWSP *und* OGRUT *berechnete Überflutungsflächen des Krummbach, Eschbach und der Brugga für ein Hochwasserereignis am 17.12.1989.*

7.3 Nutzen für das Grundwassermodell

Es stellt sich nun die Frage, inwieweit die durch *OGRUT* zusätzlich bereitgestellten Daten für *FEFLOW* eine Leistungssteigerung des Modells und damit einen Informationsmehrwert bedeuten, die den Aufwand rechtfertig.
Obwohl noch keine längeren Zeitreihen der Grundwasserneubildung berechnet wurden, lassen die bisher testweise durchgeführten Rechnungen doch einige Schlüsse zu. Man betrachte hierzu die in Abb. 7.4 und Abb. 7.4 dargestellten Diagramme. Es handelt sich um Ganglinien der Grundwasserstände an den beiden Grundwasserpegeln *NE*-10 und *NE*-20. Es wird jeweils die gemessene Zeitreihe drei mit *FEFLOW* simulierten (BOLD, 2000) Zeitreihen gegenübergestellt. Dabei handelt es sich erstens um eine Zeitreihe, die ohne zeitvariable Berücksichtigung von indirekten Grundwasserneubildungdaten erstellt wurde. Zweitens ist

eine Zeitreihe abgebildet, bei der die indirekte Grundwasserneubildung im Dezember 1989 pauschal um den Faktor 2.5 erhöht wurde. Die dritte simulierte Ganglinie verwendet die durch *OGRUT* ermittelten indirekten Grundwasserneubildungdaten aus Überflutungsflächen, wie sie im vorhergehenden Kapitel präsentiert wurden.
Während Pegel *NE*-10 vorfluterfern liegt, handelt es sich bei Pegel *NE*-20 um einen vorfluternahen Pegel.

Man erkennt, dass zum Zeitpunkt des Hochwasserereignisses im Dezember 1989 die einfache Zeitreihe am Pegel *NE*-20 keine befriedigende Anpassung an die gemessene Ganglinie ergibt. Dies ist darauf zurückzuführen, dass am Pegel *NE*-20 zu diesem Zeitpunkt hohen Neubildungsraten durch Überflutung auftraten, die in diesem Simulationslauf keinen Eingang fanden.
Beim Vergleich mit der Zeitreihe, bei der die Grundwasserneubildung im Dezember 1989 um den Faktor 2.5 erhöht wurde erkennt man ausserdem, dass zwar eine bessere Anpassung des Modells an die Messwerte des Pegels *NE*-20 erfolgte. Gleichzeitig aber sank die Anpassungsgüte an den vorfluterfernen Pegel *NE*-10. Für diesen Pegel waren auch keine erhöhten Grundwasserneubildungswerte zu erwarten, da er nicht in einem von Überflutung betroffenen Gebiet liegt.
Betrachtet man nun die dritte simulierte Zeitreihe, bei der die Überflutungsdaten flächenvariabel aus *OGRUT* geliefert wurden, so erkennt man einerseits eine Anpassung an Pegel *NE*-10 die ähnlich gut ist wie der normale Simulationslauf. Andererseits findet man für Pegel *NE*-20 eine Anpassung, die gegenüber der normalen Zeitreihe eine merkliche Verbesserung bedeutet. Hinzuzufügen ist, dass die aus *OGRUT* gelieferten Werte zum Zeitpunkt des Kalibrierungslaufes noch keineswegs der endgültigen Qualität entsprachen, da sie in einem relativ frühen Entwicklungsstatium von *OGRUT* in die parallel laufende Arbeit von BOLD einflossen. Es sind also eher noch Verbesserungen der Modellvorhersage zu erwarten, wenn die Kopplung von *FEFLOW* an *OGRUT* dem endgültigen Zustand enspricht.

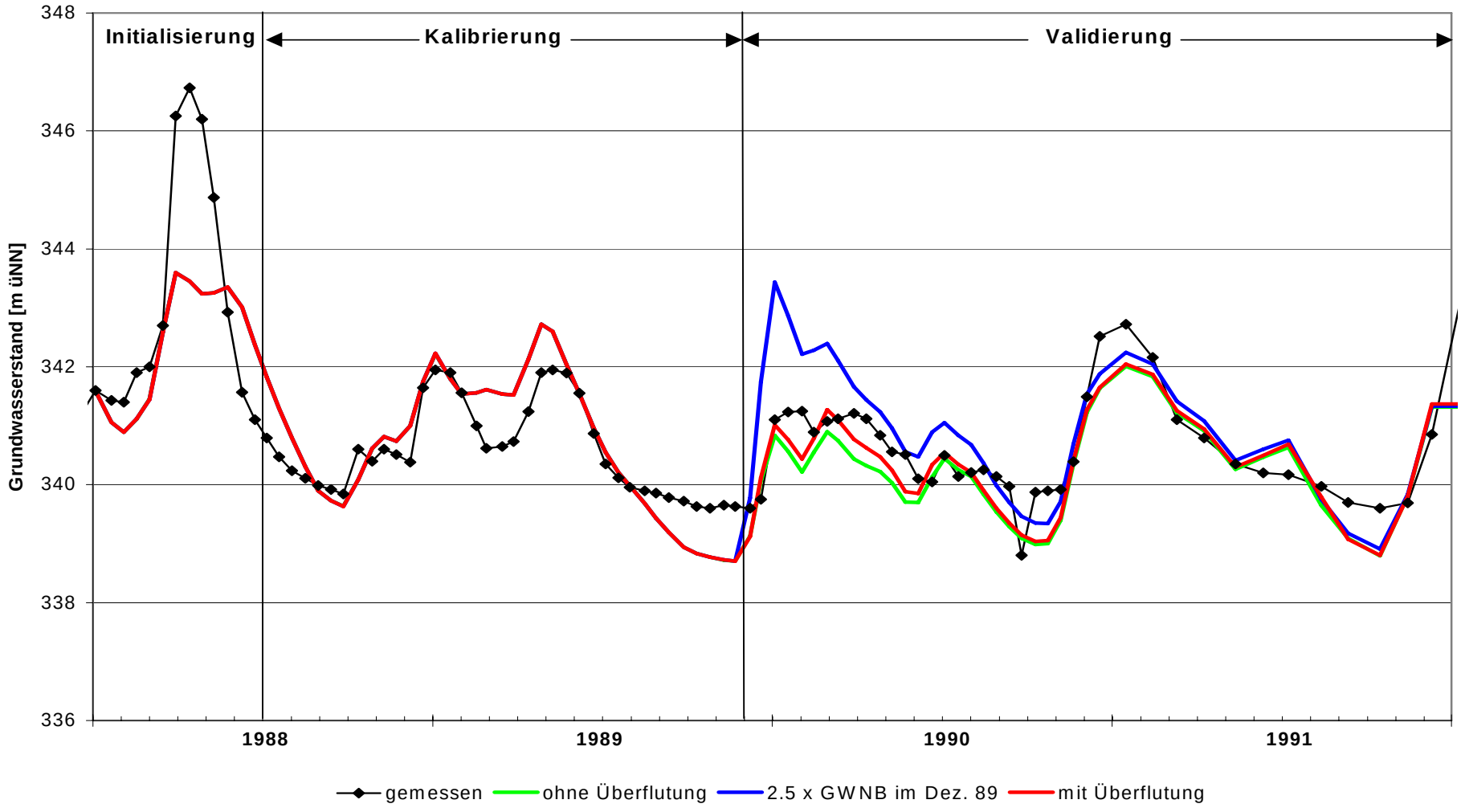


Abb. 7.4: Grundwasserpegel NE-10, vorfluterfern (aus BOLD, 2000).

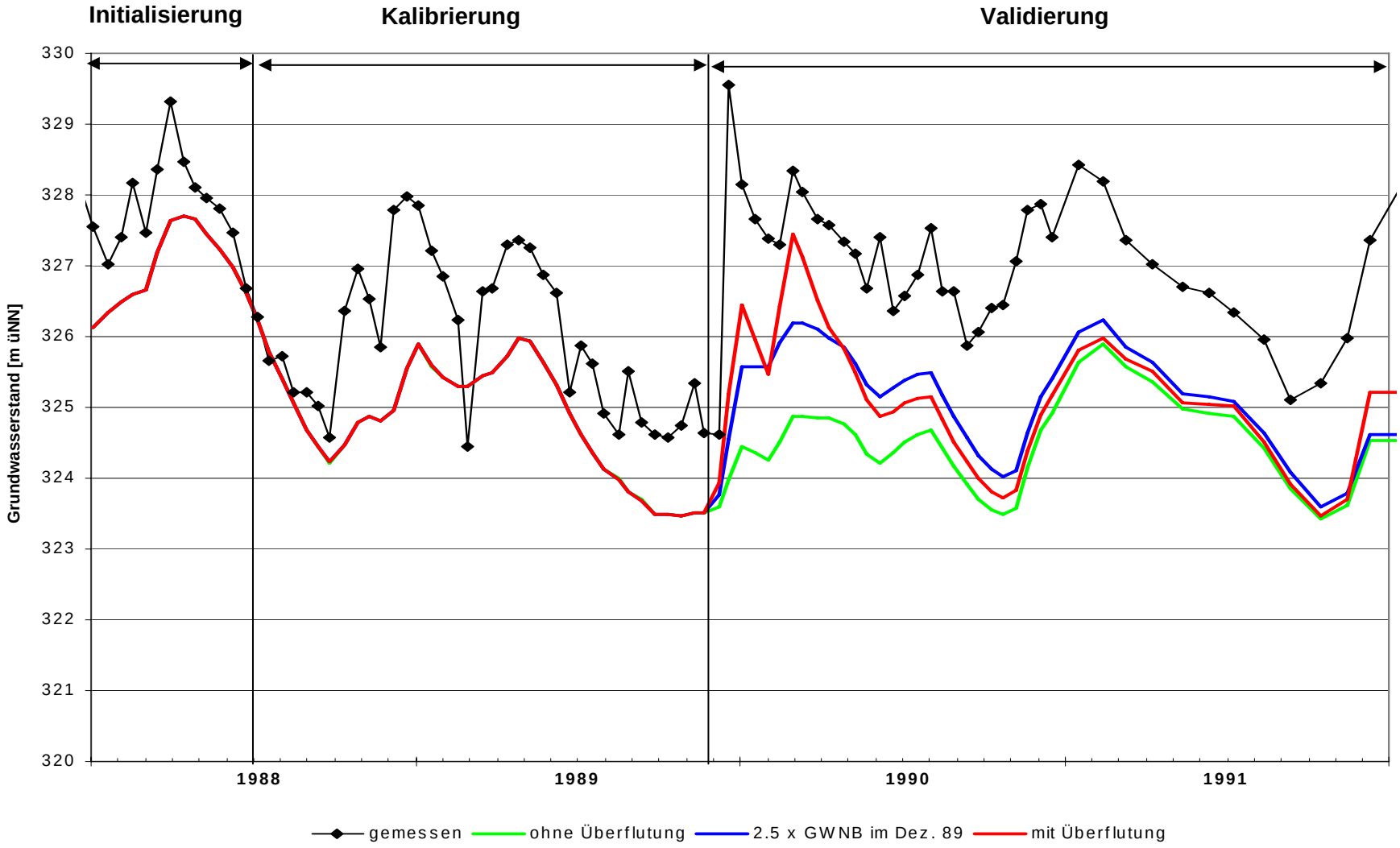


Abb. 7.5: Grundwasserpegel NE-20, vorfluternah (aus BOLD, 2000).

7.4 Fazit

Die ersten Testergebnisse zeigen, dass sich der Einsatz der umfangreichen Werkzeuge des *OGRUT* bezahlt macht. Die Vorhersagegenauigkeit von *FEFLOW* wird durch die exaktere Berücksichtigung der indirekten Grundwasserneubildung in den von Überflutung betroffenen Zeiträumen besser.
Die Fortführung von Modellrechungen und die operationelle Anwendung werden durch die weitgehend automatische Parametrisierung, die zentrale Datenverwaltung und die zentralen Auswertungswerkzeuge erleichtert und sind mit geringerem Aufwand verbunden als bisher.
Für die Ermittlung der Grundwasserneubildung steht dem Benutzer neben der bereits implementierten und in 6.4.1.5 beschriebenen Formel die Möglichkeit offen, weitere Modelle zu integrieren. So wäre es wünschenswert festzustellen, durch welche Modifikationen die verwendete Formel eventuell noch bessere Ergebnisse erzielt.

Das Gerinnehydraulikprogramm *UWSP* berechnet Wasserstandszeitreihen stabil, bedarf aber einer Kalibrierung über den STRICKLER-Beiwert. Es hat sich gezeigt, dass der Kalibrierungsbedarf geringer ist, je grösser die Dichte der zur Verfügung stehenden Geometriedaten ist.
Die Dokumentation von *HEC-RAS* erwähnt an einigen Stellen verschiedene Implementierungsoptionen (z.B. für die Berechnung des Reibungsverlustes S_f). Es wäre interessant festzustellen, in wie weit die verschiedenen Alternativen Einfluss auf die Genauigkeit des Gerinnehydraulikprogramms haben. Hierzu könnte man *UWSP* derart erweitern, dass an den betreffenden Stellen verschiedene Alternativen angeboten werden.

8 Diskussion und Ausblick

Aufgabe dieser Diplomarbeit war die Integration von *FEFLOW* in *Smallworld*. Dies sollte in einer Weise erfolgen, dass die automatische Parametrisierung des Programms unterstützt wird. In dieser Folge ergaben sich als Hautpaufgabe einerseits die Erstellung eines Programms zur Wasserspiegelberechnung (*UWSP*), basierend auf der Methode von *HEC-RAS*, und andererseits die Programmierung des GIS *Smallworld*.
Dabei wurde ein Datenmodell für ein Oberflächengewässer- und ein Grundwassermodul, basierend auf *LIWIS*-Modulen, erstellt. Es wurden die erforderlichen Algorithmen zur Grundwasserneubildung aus Überflutungsflächen implementiert. Bei der gesamten Arbeit wurden Methoden und Werzeuge programmiert, die nötig waren, um die erforderlichen Daten in geeigneter Weise zur Verfügung zu stellen. Dies waren Methoden zum Einlesen und Auselesen von Daten aus externen Dateien und Methoden zum Zugriff auf die *Smallworld*-Datenbank.
Zuletzt wurde eine graphische Benutzeroberfläche erstellt, die es dem Anwender erlaubt, auf einfache und komfortable Art die neuen Möglichkeiten des Moduls *OGRUT* zu erschliessen.

Der Zweck der Arbeit, die Verbesserung des Modells *FEFLOW* durch den Einsatz weiterer bzw. genauerer Randbedingungen wurde nach ersten Ergebnissen erreicht. Auch die operationelle Modellanwendung wird durch die automatische Bereitstellung der Daten ermöglicht.
Somit kann man feststellen, dass sich das *Smallworld*-Modul *OGRUT* und das Wasserspiegel-progamm *UWSP* als weitere Bausteine im integrativen System *WAQIS* einfügen und die Leistungsfähigkeit des Gesamtsystems erhöhen.

Abschliessend werden Vorschläge für die zukünftige Weiterentwicklung von *OGRUT* und *UWSP* gemacht.
Zunächst ist hier zu nennen, dass *UWSP* bisher keine Sonderprofile (Brücken, Wehre, Schütze) berücksichtigt. Diese Einschränkung spielte in dieser Arbeit keine Rolle, da entsprechende Daten für das Anwendungsgebiet nicht vorlagen. Dennoch ist hier Potential für Weiterentwicklungen gegeben. Wünschenswert für *UWSP* wäre auch die Fähigkeit, eine automatische Kalibrierung der k_{St}-Werte anhand bestehender Abfluss-Wasserstands-beziehungen durchzuführen. Da der Quellcode von *UWSP* vorliegt, wäre eine solche Erweiterung wohl mit relativ geringem Aufwand möglich. Überlegenswert wäre ausserdem, für den strömenden Fall als zusätzliche Alternative das Verfahren nach DVWK (1991) zu implementieren, das eine genauere Beschreibung der Reibungsverhältnisse ermöglicht.
Das Modul *OGRUT* bildet noch keine geschlossene Kette aller an der Berechnung beteiligten externen Programme, da *FEFLOW* nach wie vor von Hand aufgerufen werden muss. Die nahtlose Integration von *UWSP* hat jedoch gezeigt, dass das Ziel, auch *FEFLOW* direkt aus *Smallworld* zu steuern, erreichbar ist. Die hierfür nötigen Schritte der Datenbereitstellung und Datenrücknahme sind bereits umgesetzt, so dass für die vollständige Integration von *FEFLOW* kein grosser Aufwand mehr nötig ist.
DIe Umsetzung zusätzlicher Visualisierungsmöglichkeiten, die es erlauben gezielt Zustands-bilder von Grundwasserparametern für einen bestimmten Zeitpunkt graphisch darzustellen, wären sinnvoll. Die im Iso-Modul vorhandenen Isolinien sind hierfür geeignet, müssen aber noch um Funktionen zur Auswahl eines bestimmten Simulationszeitpunktes ergänzt werden.
Weiterhin könnte man den zentralen Überflutungsalgorithmus des digitalen Geländemodells in ein externes *C++*-Programm auszulagern, da hierdurch die Geschwindigkeit der Berechnung deutlich ansteigen dürfte. Die aktuelle Geschwindigkeit ist sehr niedrig, was darauf zurückzuführen ist, dass die Berechnung der Überflutung in einer vierfach

geschachtelten Schleife erfolgt. Für weitere Verbesserungen des Überflutungsalgorithmus wäre eine höhere Geschwindigkeit durch externe Berechnung fast schon eine Voraussetzung. Zuletzt lohnt es sich auch, die Fähigkeiten von *Smallworld* ins Auge zu fassen, externe Programme via alien co-processor direkt über einen Datenkanal mit der GIS-Umgebung zu verbinden. Diese Vorgehensweise würde eine einfachere und flexiblere Kommunikation zwischen *Smallworld* und dem externen Programm ermöglichen.

9 Literaturverzeichnis

ALBERS, H., 1998: *Berechnung der Grundwasserneubildung aus Niederschlag für die Wasserwerke Ebnet und Hausen*, unveröffentlichte Diplomarbeit, Institut für Hydrologie, Albert-Ludwigs-Universität Freiburg im Breisgau.

BARTELME, N., 1989: *GIS-Technologie*, Geoinformationssysteme, Landinformationssysteme und ihre Grundlagen, Springer-Verlag, Berlin, 280 S.

BÖHM, B. et al, 1999: `www.lrz-muenchen.de/~skript_wga/inhalt.html #4.5.2`, *Skript zur Vorlesung Wassergüte und Abfallwirtschaft*, Leibnitz-Rechenzentrum, München.

BOLD, S., 2000: *Instationäre Grundwassermodellierung des Zartener Beckens*, unveröffentlichte Diplomarbeit, Institut für Hydrologie, Albert-Ludwigs-Universität Freiburg im Breisgau.

BRETSCHNEIDER, H., SCHULZ, A., 1982: *Anwendung von Fliessformeln bei naturnahem Gewässerausbau*, Schriftenreihe des DVWK, Heft 72, Verlag Paul Parey, Hamburg und Berlin.

BRUNNER, G. W., 1998: *HEC-RAS River Analysis System - Hydraulic Reference Manual*, Version 2.2, U.S. Army Corps of Engineers, Hydrologic Engineering Center, 175 S.

BUCKINGHAM, E., 1907: *Studies on movement of soil moisture*, Bull. 38, U.S. Dept. of Agr. Bureau of Soils, Washington, D.C.

CARNAHAN, 1987: *Effects of coupled thermal, hydrological and chemical processes on nuclide transport*, Stockholm.

CHOW, V. T., 1959: *Open Channel Flow*, McGraw-Hill Book Company.

CODD, E.F., 1970: *A Relational Model of Data for Large Shared Data Banks*.

D.T. LEE, B.J. SCHACHTER, 1980: *Two Algorithms for Constructing a Delaunay Triangulation*, International Journal of Computer and Information Sciences, Vol. 9, No.3, S. 219-242.

DARCY, H., 1856: *Les fontaines publique de la ville de Dijon*, Dalmont, Paris.

DOMMERMUTH, H., TRAMPF, W., 1990: *Die Verdunstung in der Bundesrepublik Deutschland Zeitraum* 1951-1980. Teil 1, Selbstverlag des Deutschen Wetterdienstes, Offenbach.

DVWK, 1991: *Hydraulische Berechnung von Fliessgewässern*, DK 551.51/54 Fliessgewässer, DK 532.543 Hydraulik, Deutscher Verband für Wasserwirtschaft und Kulturbau e.V. (DVWK), Verlag Paul Parey, Hamburg und Berlin, 62 S.

DYCK, S., PESCHKE, G., 1995: *Grundlagen der Hydrologie*, Verlag für Bauwesen, Berlin, 536 S.

EBERLE, K., 1999: *BOMET – Aufbau eines bodenkundlich-meteorologischen Teil-informationssystems im GIS Smallworld*, unveröffentlichte Diplomarbeit, Institut für Hydrologie, Albert-Ludwigs-Universität Freiburg im Breisgau.

EBNER, M., 1999: *SQL lernen*. - Addison-Wesley, München, 313 S.

EHRMINGER, B., HERDEG, U., 1992: *Abschlussbericht über die isotopenhydrologischen Untersuchungen und die erstellten Grundwassermodelle im Einzugsgebiet des Wasserwerks Ebnet*. Unveröffentlichte Untersuchungen der Freiburger Energie- und Wasserversorgungs-AG und des Geologischen Landesamtes Baden-Württemberg, Freiburg.

ESRI, 2000: `www.esri.com/library/gis`.

FRIEG, B, 1987: *Hydrogeologie und Grundwasserhydraulik des Einzugsgebietes des Wasserwerkes Freiburg-Ebnet*, Inaugural-Dissertation, Naturwissenschaftilch-Mathematische Gesamtfakultät, Ruprecht-Karls-Universität Heidelberg.

GLOMB, G, ZWÖLFER, F, 1990: *Grundwassermodell Zartener Becken. Bodenkundliche Untersuchungen*. Unveröffentlichtes Gutachten des Geologischen Landesamtes Baden-Württemberg Nr.0801.01/89-4765-Zw/Gg. Freiburg im Breisgau.

GOLDBERG, A., ROBSON, D., 1983: *Smalltalk-80: The Language and its Implementation*, Addison-Wesley, Reading, Massachusetts.

GREEN, W.H. & AMPT, G.A., 1911: *Studies on soil physics: I. Flow of air and water through soils*, J. Agric. Sci. 4:1-24.

HAGEN, 1999: `wwwpi6.fernuni-hagen.de/Lehre/Kurse/Geometrie/demo/`, Fernuniversität, Gesamthochschule Hagen.

HEC, 1986: *Accuracy of Computed Water Surface Profiles*, Research, Document 26, Hydrologic Engineering Center, U.S. Army Corps of Engineers, Davis CA.

HERDEG, U, 1993: *Untersuchungen zu den Grundwasserfliesssystemen im Bereich der Wasserwerke von Freiburg i.Br. auf der Grundlage isotopenhydrologischer und geologischer Daten*, Dissertation, Albert-Ludwigs-Universität Freiburg im Breisgau.

HOFIUS, K., NIPPES, K.-R., 1985: *Beziehung zwischen Wasserqualität und Abfluss - modellhafte Untersuchung im Einzugsgebiet Dreisam*. Unveröffentlichter Bericht Forsch.-Vorhaben DFG, Ho 589/1-4, 131 S., Koblenz, Freiburg im Breisgau.

HOLTAN, H.N, 1961: *Concepts for infiltration estimates in watershed engineering*, USDA Res. Ser. Publ.:41-51.

HORTON, R.E., 1940: *An approach towards a physical interpretation of infiltration capacity*, Soil Sci. Soc. Am. Proc. 5:399-417.

ISO, 1987: ISO 9000, International Organization for Standardization.

JUNGHANS, U., 1998: *Grundwassermodell Zartener Becken*, unveröffentlichte Diplomarbeit, Institut für Hydrologie, Albert-Ludwigs-Universität Freiburg im Breisgau.

KIRWALD, E., 1980: *Wasser und Gewässer*. In: Landkreis Breisgau-Hochschwarzwald (Hrsg.): *Breisgau und Hochschwarzwald, Land vom Rhein über den Schwarzwald zur Baar*, 54-60, Freiburg imBreisgau.

KOSTIAKOV, A.N., 1932: *On the dynamics of the coefficient of water-percolation in soils and on the necessity of studying it from a dynamic point of view for purpose of amelioration*, Trans. Sixth Comm. Intl. Soc. Soil Sci., Teil A:17-21.

LEIBUNDGUT, CH. & GÄßLER, G., 1998: *Hydrologische Entscheidungsgrundlagen zur Planung der Überleitung Krummbach-Dreisam*, Bericht des Instituts für Hydrologie der Universiät Freiburg im Breisgau.

MERTENS, W., 1989: *Zur Frage hydraulischer Berechnungen naturnaher Fliessgewässer*, Wasserwirtschaft 79, Heft 4.

METZ, R, 1959: Zur naturräumlichen Gliederung des Schwarzwaldes, Alemann. Jb. 1959, 1-33, Lahr.

NAUDASCHER, E., 1956: *Berechnung der Wasserspiegellage*, Der Bauingenieur, Heft 1.

NAUDASCHER, E., 1987: *Hydraulik der Gerinne und Gerinnebauwerke*, Springer-Verlag, Wien und New York.

PARNAS, D., L., 1972: *On the criteria to be used in decomposing systems into modules*, Communications of the ACM 15, 1053-8.

PASCHE, E., 1984: *Turbulenzmechanismen in natürlichen Fliessgewässern und die Möglichkeit ihrer mathematischen Erfassung*, Dissertation, RWTH Aachen.

PHILIP, J.R., 1957: *The theory of infiltration: 4. Sorptivity and algebraic infiltration equations*, Soil Sci. 84:257-267.

PHILIP, J.R., 1960: *General method of exact solution of the concentration-dependent diffusion equation*, Aust. J. Phys. 13:1-12.

RENGER, M., STREBEL, O., GIESEL, W., 1974: *Beurteilung bodenkundlicher, kulturtechnischer und hydrologischer Fragen mit Hilfe von klimatischer Wasserbilanz und bodenphysikalischen Kennwerten*. 4. Bericht: *Grundwasserneubildung*. Zeitschrift für Kulturtechnik und Flurbereinigung, 15, S.353-366.

RENGER, M., STREBEL, O., MÜNNICH, K.O., SONNTAG, C., 1977: *Methoden zur Bestimmung der Grundwasserneubildungsrate*, Geologisches Jahrbuch, Reihe C, H.19.

RENGER, M., VOIGT, H., STREBEL, O., GIESEL, W., 1974: *Beurteilung bodenkundlicher, kulturtechnischer und hydrologischer Fragen mit Hilfe von klimatischer Wasserbilanz und bodenphysikalischen Kennwerten*. 2. Bericht: *Einfluss des Grundwassers auf die Wasserversorgung der Pflanzen*. Zeitschrift für Kulturtechnik und Flurbereinigung, 15, S.206-221.

ROLKE, A, 1994: *Räumliche Untersuchung von Pestizid- und Nitratbelastung im Raum Freiburg*, Institut für Hydrologie, Albert-Ludwigs-Universität Freiburg.

ROUVÉ (Hrsg.), 1987: *Hydraulische Probleme beim naturnahen Gewässerausbau*, VCH Verlagsges. mbH, Weinheim.

SMALLWORLD SYSTEMS, 1994a: *Smallworld GIS 2 User Guide*, Smallworld Systems.

SMALLWORLD SYSTEMS, 1994b: *Smallworld GIS 2 Magik Manual*, Smallworld Systems.

SMALLWORLD SYSTEMS, 1999: *Smallworld 3 Dokumentation*, Smallworld Systems.

STROUSTRUP, B., 1986: *The C++ Programming Language*, Addison-Wesley, Menlo Park, California.

SUN, 2000: *Java*, `java.sun.com`, Sun Microsystems.

WAQIS, 2000: `http://www.ruf.uni-freiburg.de/hydrology/forsch/waqis`, Institut für Hydrologie, Albert-Ludwigs-Universität Freiburg im Breisgau.

WASSERWIRTSCHAFTSAMT FREIBURG 1988: *Ausweisung von Ausuferungsflächen entlang des Krummbachs*, Beller Consult GmbH, Freiburg.

WATT, D.A., 1996: *Programmiersprachen - Konzepte und Paradigmen*, Carl Hanser Verlag, München, 334 S.

WHATIS, 2000: `www.whatis.com`.

WUNDT, W., 1965: *Oberflächengewässer*. In Statistisches Landesamt Baden-Württemberg: Freiburg im Breisgau, Stadtkreis und Landkreis, Amtliche Kreisbeschreibung, I (3): 98-105, Freiburg.

Anhang A Klassen und Definitionen von UWSP

A.1Klasse Reach

- Datenfelder
  - char **geometrySep**
  - char **flowSep**
  - char **outputSep**
  - TDateTime **dateTime**
  - TimeStep **timeStep**
  - *privat* float **q**
  - *privat* TList* **qList**
  - *privat* TList* **crossSections**
  - *privat* AnsiString **geometryFileName**
  - *privat* bool **calculating**
- Konstruktoren
  - Reach()
- Methoden
  - void **resetComputation**()
  - void **resetFlow**()
  - void **reset**()
  - void **addCrossSection**(CrossSection* crossSection)
  - void **readGeometry**(AnsiString geometryFileName) throw (AnsiString)
  - void **readFlow**(AnsiString flowFileName) throw (AnsiString)
  - void **writeOutput**(AnsiString outputFileName) throw (AnsiString)
  - void **writeOutputAll**(AnsiString outputFileName) throw (AnsiString)
  - void **setFlow**(float q)
  - void **compute**(TForm* form, TStatusBar* statusbar, TProgressBar* progressBar, bool fromList) throw (AnsiString)
  - void **stop**()
  - int **getIndex**(CrossSection* crossSection)
  - int **getSectionsCount**()
  - CrossSection* **getSection**(int i)
  - *privat* void **readDateTime**(Scanner& scanner, TDateTime& dt);
  - *privat* void **readTimeStep**(Scanner& scanner, TimeStep& ts);

A.2Klasse Station

- Datenfelder
  - float **y**, **z**
  - float **kst**
  - int **overbank**
- Konstruktoren
  - **Station**(float y, z, float overbank, float kst)

A.3Klasse CrossSection

- Datenfelder
  - Reach* **reach**
  - FlowData* **flow**[ftCOUNT]
  - FlowData* **selected**

TList* **stations**
Ganglinie* **gangLinie**
Ganglinie* **hLinie**
float **minZ**
float **kst**
float **L**[ovCOUNT]
float **iso**
float **dist**[ovCOUNT]
int **myIndex**
bool **bridge**
bool **temp**

- Konstruktoren
  CrossSection(Reach* aReach)
- Methoden
  void **resetFlow**()
  void **reset**()
  CrossSection* **prevSection**()
  CrossSection* **nextSection**()
  void **addStation**(Station* station)
  void **calcIso**()
  int **getIndex**()
  int **getFirst**(int section)
  int **getLast**(int section)
  float **area**(float h, int first, int last)
  float **perimeter**(float h, int first, int last)

A.4 Klasse FlowData

- Datenfelder
  float **L**
  float **K**[ovCOUNT]
  float **A**[ovCOUNT]
  float **U**[ovCOUNT]
  float **q**[ovCOUNT]
  float **Sf**, **Sf_avg**
  float **C**
  float **v**, **v_head**
  float **alpha**, **beta**
  float **he**
  float **h**
  float **specForce**
  bool **calculated**
  bool **converge**
  CrossSection* **crossSection**
- Konstruktoren
  FlowData(CrossSection* crossSection)
- Methoden
  void **reset**()
  float **step**()
  float **stepMomentum**()
  void **conveyance**(int overbank)
  void **calculate**()

```
void  critical(float aq, int index)
void  subCritical(float aq, int index)
void  superCritical(float aq, int index)
void  normal(float aq, int index)
void  calcSpecForce()
```

A.5 Klasse Ganglinie

- Datenfelder
```
int section
TList* f
```
- Konstruktoren
```
Ganglinie()
Ganglinie(int section)
```
- Methoden
```
void  reset()
void  addF(float f)
float getF(int index)
```

A.6 Klasse Scanner

- Datenfelder
```
Token token
privat char line[2][LINESIZE]
privat char decimalSep
privat int currentLine
privat AnsiString fileName
privat FILE* f
privat char* p
privat char ch
```
- Konstruktoren
```
Scanner()
```
- Methoden
```
void open(AnsiString s, char decimapSep) throw (AnsiString)
void scan() throw (AnsiString)
void scanNumber() throw (AnsiString)
void match(int sym) throw (AnsiString)
bool available()
void nextLine()
void close()
privat void nextCh()
privat void ungetCh()
privat void number()
privat void identifier()
```

A.7 Definition der Overbank-Konstanten (ovXXX)

```
#define ovUNKNOWN       0
#define ovLEFT          1
#define ovRIGHT         2
#define ovCHANNEL       3
#define ovBEFORERIGHT   4
#define ovALL           5
```

A.8 Definition der FlowType-Konstanten (ftXXX)

```
#define ftSUB        0
#define ftSUPER      1
#define ftCRIT       2
#define ftNORMAL     3
```

A.9 Definition der Struktur Token

- Datenfelder

```
Symbol sym
int int_num
float float_num
char id[LINESIZE]
```

Anhang B Klassen und Definitionen von OGRUT

B.1Definition der Datenstruktur von OGRUT OFG

OFG besteht aus den in Abb. 6.3 gezeigten Klassen. Flüsse (Klasse Oberflächengewässer) enthalten Querprofile (Klasse Profile) und Querprofile enthalten wiederum Stützpunkte (Klasse Stützpunkte). Die Legende zu den Symbolen in der ersten Spalte bei den physikalischen Feldern ist in Tab. B.1 erklärt.

Tab. B.1: Legende zu den Symbolen für physikalische Felder im Datenmodell.

*	Schlüsselfeld
S	Sichtbares Feld
I	Vererbtes Feld. Das Feld wird aus einer Klasse entnommen, die die betrachtete Klasse in einer Liste enthält.

B.1.1Klasse Oberflächengewässer

Das Datenmodell der Klasse Oberflächengewässer ist in Tab. B.2 dargestellt. Man erkennt, dass der Name (ofg_name) den Fluss als Schlüsselfeld kennzeichnet. Wichtig für die Berechnungen in *OGRUT* sind die Felder time_step, time_start und zeitreihe_count. Letzteres gibt die Länge der Abflusszeitreihen an, die für diesen Fluss an den einzelnen Pegeln gemessen wurden. time_step, time_start geben das Zeitintervall und den Startzeitpunkt der Messreihe an. Das Feld ogrut_project_name stellt enthält den Namen des *OGRUT*-Projekts, in dem der Fluss enthalten ist. Die Verknüpfungsfelder enthalten unter anderem Verweise auf eine Tabelle mit Querprofilen (lw_profil) und eine Tabelle mit Pegelmessstationen (ofg_pegel_rec). Das Feld ofg_tins enthält eine Tabelle mit dem Fluss zugeordneten digitalen Geländemodellen (*triangulated irregular network*, TIN). Da ein digitales Geländemodell seinerseits keinem Fluss zugeordnet sein muss, ist die Verknüpfung vom Typ 0:*n*. Das Feld ogrut_project verweist auf das *OGRUT*-Projekt.
Das Datenmodell der Klassen Profile und Stützpunkte ist in Tab. B.3 und Tab. B.4

Tab. B.4 dargestellt. In der Klasse Profile (lw_profil) wird das Schlüsselfeld ofg_dar_ofg_name vom Oberflächengewässer (ofg_dar) vererbt, dem das Profil zugeordnet ist. Als weiteres Schlüsselfeld kommt profil_nr hinzu. Diese beiden Attribute zusammen identifiezieren das Querprofil in der Datenbank eindeutig. Beim Einlesen von Geometriedaten aus externen Dateien wird der Abstand (Feld abstand) zwischen zwei aufeinanderfolgenden Profilen aus der Kilometrierung (km) erfasst. Das Feld ofg_dar_rec verweist auf das Gerinnne (ofg_dar), dem das Querprofil zugeordnet ist. Das Feld lw_stuetzpunkte verweist auf die einzelnen Stationen (Querprofilpunkte), die die Geometrie des Querprofils ausmachen. ofg_zeitreihe verweist auf Zeitreihendaten, die dem Profil zugeordnet sind (Abfluss, Wasserstand).

Tab. B.2: Datenmodell der Klasse Oberflächengewässer.

Objektname: ofg_dar (Oberflächengewässer)			
Physikalische Felder		**Feldtyp**	
*S	ofg_name	ds_char_vec (30)	
	ofg_typ_code	ds_uint	
S	datum	ds_time	
S	gew_kennzahl	ds_int	
S	zeitreihe_count	ds_int	
S	time_step	ds_char_vec (30)	
S	time_start	ds_time	
SI	ogrut_project_name	ds_char_vec (30)	
Logische Felder		**Feldtyp**	
	ofg_typ	ds_char_vec (45)	
Geometriefelder		**Feldtyp**	
	geometrie	simple_chain	
	annontation	text	
	flaeche	area	
	linie	chain	
Verknüpfungsfelder		**Beziehungstyp**	**Zu Objekt**
	lw_einleitungen	1:n	lw_einleitungen
	lw_bauwerke	1:n	lw_bauwerke
	lw_segmente	1:n	lw_segmente
	lw_profil	1:n	lw_profil
	lw_richtung_text	0:n	lw_richtung_text
	lw_gewaesserguete	1:n	lw_gewaesserguete
	ofg_pegel_rec	1:n	ofg_pegel
	talsperre_rec	1:n	talsperre
	ofg_tins	0:n	ofg_tin
	ogrut_project	1:n	ogrut_project

ofg_dar.berechne_h_tin(
tin_id, from_date, to_date, delete_old?, display_tin, _**optional** items)

- Diese Methode berechnet die Überflutung eines digitalen Geländemodells (Tin) durch einen Fluss. Es wird für jede site im Tin ein Objekt der Klasse ofg_tin_z_werte erzeugt, an das eine neue Zeitreihe vom Typ ofg_tin_z_reihe angehängt wird. Diese Zeitreihe erhält danach für jeden Zeitreihenschritt ein Objekt der Klasse ofg_tin_z_reihenwert, das den für den gegebenen Zeitschritt berechneten Wasserstand aufnimmt.
- **Parameter** **Typ**
  tin_id sys_id
  from_date string
  to_date string
  delete_old? boolean
  display_tin ofg_tin
  items hash_table
- **Typ des Rückgabewertes** _unset
- Der Parameter tin_id gibt das Tin an, für dessen sites die Überflutung berechnet werden soll. from_date und to_date geben den Anfangs- und Endzeitpunkt in der Zeitreihe an, für die die Berechnung durchgeführt werden soll. Sind sie _unset, so

wird die gesamte Zeitreihe gerechnet. delete_old? legt fest, ob die Ergebnisse früherer Berechnungen gelöscht werden sollen (_true) oder die neuen Ergebnisse einfach hinzugefügt werden (_false). Ist das Feld display_tin nicht _unset, dann werden alle überfluteten sites des letzten Zeitschrittes in der Zeitreihe in das display_tin kopiert. Als *z*-Wert wird der Wasserstand der Überflutung eingetragen. Es empfiehlt sich bei gesetztem display_tin nur einen ausgewählten Zeitschritt zu rechnen (from_date = to_date). Der Parameter items enthält Einträge, die für Multithreading und Statusanzeige verwendet werden. items ist _unset, wenn die Methode nicht von der graphischen Oberfläche von *OGRUT* aufgerufen wird.

ofg_dar.berechne_h_tin_step(
the_tin, ufer_table, date, profil_rope, display_tin, _**optional** items)

- Diese Methode berechnet die Überflutung eines Tins für einen einzelnen Zeitpunkt date. Sie wird aufgerufen von ofg_dar.berechne_h_tin().
- **Parameter** **Typ**

tin_id	sys_id
ufer_table	hash_table[integer](float)
date	date_time
profil_rope	rope(Coordinate)
display_tin	ofg_tin
items	hash_table

- **Typ des Rückgabewertes** _unset
- Der Parameter tin_id gibt das Tin an, für dessen sites die Überflutung berechnet werden soll. In ufer_table sind zum schnellen Vergleich die Uferhöhen für die einzelnen Querprofile enthalten. Dadurch kann sofort festgestellt werden, ob eine Überflutung überhaupt möglich ist. Erst wenn das Wasser über das Ufer tritt, ist mit Überflutung zu rechnen. date ist der Zeitpunkt, für den die Überflutung berechnet wird. profil_rope enthält die Mittelpunktskoordinate für jedes Querprofil, also ein Coordinate-Objekt für jedes Querprofil. display_tin ist _unset oder enthält das Tin, dem die überfluteten sites, z.B. zur Visualisierung zugewiesen werden sollen. Der Parameter items enthält Einträge, die für Multithreading und Statusanzeige verwendet werden. items ist _unset, wenn die Methode nicht von der graphischen Oberfläche von *OGRUT* aufgerufen wird.

ofg_dar.delete_wsp(_**optional** items)

- Diese Methode löscht alte Zeitreihen. Sie wird bei Bedarf von ofg_dar.read_wsp() aufgerufen.
- **Parameter** **Typ**

items	hash_table

- **Typ des Rückgabewertes** _unset
- Der Parameter items enthält Einträge, die für Multithreading und Statusanzeige verwendet werden. items ist _unset, wenn die Methode nicht von der graphischen Oberfläche von *OGRUT* aufgerufen wird.

ofg_dar.get_sorted_profil(a_profil_nr)

- Diese Methode gibt die Stützpunkte des angegebenen Querprofils nach Koordinaten sortiert zurück. Dabei wird die Sortierung umgekehrt, wenn bei normaler, aufsteigender Sortierung das linke Ufer nach dem rechten Ufer ausgegeben würde.
- **Parameter** **Typ**
  a_profil_nr integer
- **Typ des Rückgabewertes** sorted_collection(lw_stuetzpunkt)
- Der Parameter a_profil_nr bezeichnet das Profil, für das die Sortierung vorgenommen wird.

ofg_dar.insert_profil(profil)

- Diese Methode fügt ein neues Querprofil in das Gewässer ein. Danach wird der Abstand des vorhergehenden Profils zum eingefügten Profil berechnet und dem vorhergehenden Querprofil zugewiesen.
- **Parameter** **Typ**
  profil lw_profil
- **Typ des Rückgabewertes** lw_profil
- Der Parameter profil ist das einzufügende Querprofil.

ofg_dar.insert_stuetzpunkt(stuetzpunkt)

- Diese Methode fügt den gegebenen Stützpunkt in ein Querprofil ein. Danach erhalten alle Stützpunkte des Querprofils neue Ordnungsnummern in Abhängigkeit von ihrem Abstand zum Nullpunkt-Stützpunkt.
- **Parameter** **Typ**
  stuetzpunkt lw_stuetzpunkt
- **Typ des Rückgabewertes** lw_stuetzpunkt
- Der Parameter stuetzpunk bezeichnet den einzufügenden Stützpunkt.

ofg_dar.page_headings

- Diese Methode setzt die Namen der einzelnen Seiten des Editorfensters für die Klasse ofg_dar.
- **Typ des Rückgabewertes** simple_vector(string)

ofg_dar.read_abfluss(file_name, delete_old?, **_optional** items)

- Diese Methode liest Abflusszeitreihen aus einer externen Datei ein und ordnet sie einzelnen Pegeln zu. Die Eingabedatei muss eine entsprechende Struktur aufweisen (s. Anhang E).
- **Parameter** **Typ**
  file_name string
  delete_old? boolean
  items hash_table
- **Typ des Rückgabewertes** _unset
- Der Parameter file_name enthält den Namen der Datei, von der gelesen wird. Fehlen Pfadangabe oder Dateiextension, so werden diese aus der Umgebungsvariable ihf_data_store ergänzt. delete_old? legt fest, ob vorhandene Zeitreihen gelöscht (_true) oder ergänzt (_false) werden sollen. Der Parameter items enthält

Einträge, die für Multithreading und Statusanzeige verwendet werden. `items` ist `_unset`, wenn die Methode nicht von der graphischen Oberfläche von *OGRUT* aufgerufen wird.

ofg_dar.read_gewaesser(file_name, delete_old?, **_optional** items)

- Diese Methode liest eine externe Datei mit Geometriedaten ein. Daraus werden Querprofile und Stützpunkte erzeugt und dem Gewässer `_self` hinzugefügt. Es wird automatisch erkannt, ob es sich dabei um ein strukturiertes oder unstrukturiertes Dateiformat handelt.
- **Parameter** **Typ**
  file_name string
  delete_old? boolean
  items hash_table
- **Typ des Rückgabewertes** _unset
- Der Parameter `file_name` enthält den Namen der Datei, von der gelesen wird. Fehlen Pfadangabe oder Dateiextension, so werden diese aus der Umgebungsvariable `ihf_data_store` ergänzt. `delete_old?` legt fest, ob vorhandene Querprofildaten gelöscht (`_true`) oder ergänzt (`_false`) werden sollen. Der Parameter `items` enthält Einträge, die für Multithreading und Statusanzeige verwendet werden. `items` ist `_unset`, wenn die Methode nicht von der graphischen Oberfläche von *OGRUT* aufgerufen wird.

ofg_dar.read_profile(sc, x, profil_nr, region)

- Diese Methode liest Querprofile aus einer externen Datei. Sie wird von `ofg_dar.read_gewaesser()` aufgerufen, falls eine Datei mit strukturiertem Dateiformat zum Einlesen gewählt wurde.
- **Parameter** **Typ**
  sc scanner
  x float
  profil_nr integer
  region symbol(:s_LEFT, :s_CHANNEL, :s_RIGHT)
- **Typ des Rückgabewertes** _unset
- Der Parameter `sc` enthält den Scanner, vom dem die Daten gelesen werden sollen. `x` ist die Kilometrierung des Querprofils. `profil_nr` ist die laufende Nummer des Querprofils. `region` ist der Bereich, in dem sich der Scanner beim Einlesen gerade befindet, möglich sind hier linkes oder rechtes Vorland (`:s_LEFT`, `:s_RIGHT`) oder der Flussschlauch (`:s_CHANNEL`).

ofg_dar.read_unstrukturierte_profile(sc, profil_nr)

- Diese Methode liest ein einzelnes Querprofil aus einer externen Datei. Sie wird von `ofg_dar.read_gewaesser()` aufgerufen, falls eine Datei mit unstrukturiertem Dateiformat zum Einlesen gewählt wurde.
- **Parameter** **Typ**
  sc scanner
  profil_nr integer
- **Typ des Rückgabewertes** _unset
- Der Parameter `sc` enthält den Scanner, vom dem die Daten gelesen werden sollen. `profil_nr` ist die laufende Nummer des Querprofils.

ofg_dar.read_unstrukturierte_stuetzpunkte(sc, profil_nr, stuetzpunkt_nr)

- Diese Methode liest einen einzelnen Stützpunkt aus einer externen Datei. Sie wird von ofg_dar.read_unstrukturierte_profile() aufgerufen, falls eine Datei mit unstrukturiertem Dateiformat zum Einlesen gewählt wurde.
- **Parameter** **Typ**
  sc scanner
  profil_nr integer
  stuetzpunkt_nr integer
- **Typ des Rückgabewertes** _unset
- Der Parameter sc enthält den Scanner, vom dem die Daten gelesen werden sollen. profil_nr ist die laufende Nummer des Querprofils. stuetzpunkt_nr ist die laufende Nummer des Stützpunktes.

ofg_dar.read_wsp(file_name, delete_old?, **_optional** items)

- Diese Methode liest Zeitreihen von Wasserstände an Querprofilen aus einer externen Datei.
- **Parameter** **Typ**
  file_name string
  delete_old? boolean
  items hash_table
- **Typ des Rückgabewertes** _unset
- Der Parameter file_name enthält den Namen der Datei, von der gelesen wird. Fehlen Pfadangabe oder Dateiextension, so werden diese aus der Umgebungsvariable ihf_data_store ergänzt. delete_old? legt fest, ob vorhandene Wasserstandsdaten gelöscht (_true) oder ergänzt (_false) werden sollen. Der Parameter items enthält Einträge, die für Multithreading und Statusanzeige verwendet werden. items ist _unset, wenn die Methode nicht von der graphischen Oberfläche von *OGRUT* aufgerufen wird.

ofg_dar.tin

- Diese Methode gibt das dem Gerinne _self zugeordnete Tin zurück, falls es eindeutig ist. Wenn nicht, wird eine Liste der vorhandenen Tins angezeigt und der Rückgabewert ist _unset.
- **Typ des Rückgabewertes** ofg_tin

ofg_dar.write_abfluss(file_name, **_optional** items)

- Diese Methode schreibt die Abflussdaten an den Pegeln des Gerinnes _self in eine externe Datei.
- **Parameter** **Typ**
  file_name string
  items hash_table
- **Typ des Rückgabewertes** _unset
- Der Parameter file_name enthält den Namen der Datei, in die geschrieben wird. Fehlen Pfadangabe oder Dateiextension, so werden diese aus der Umgebungsvariable ihf_data_store ergänzt. Der Parameter items enthält Einträge, die für Multithreading und Statusanzeige verwendet werden. items ist _unset, wenn die Methode nicht von der graphischen Oberfläche von *OGRUT* aufgerufen wird.

ofg_dar.write_gewaesser_strukturiert(file_name, **_optional** items)

- Diese Methode schreibt die Geometriedaten des Gerinnes _self in eine externe Datei. Dabei wird ein strukturiertes Dateiformat verwendet.
- **Parameter** **Typ**
  file_name string
  items hash_table
- **Typ des Rückgabewertes** _unset
- Der Parameter file_name enthält den Namen der Datei, in die geschrieben wird. Fehlen Pfadangabe oder Dateiextension, so werden diese aus der Umgebungsvariable ihf_data_store ergänzt. Der Parameter items enthält Einträge, die für Multithreading und Statusanzeige verwendet werden. items ist _unset, wenn die Methode nicht von der graphischen Oberfläche von *OGRUT* aufgerufen wird.

ofg_dar.write_gewaesser_unstrukturiert(file_name, **_optional** items)

- Diese Methode schreibt die Geometriedaten des Gerinnes _self in eine externe Datei. Dabei wird ein unstrukturiertes Dateiformat verwendet.
- **Parameter** **Typ**
  file_name string
  items hash_table
- **Typ des Rückgabewertes** _unset
- Der Parameter file_name enthält den Namen der Datei, in die geschrieben wird. Fehlen Pfadangabe oder Dateiextension, so werden diese aus der Umgebungsvariable ihf_data_store ergänzt. Der Parameter items enthält Einträge, die für Multithreading und Statusanzeige verwendet werden. items ist _unset, wenn die Methode nicht von der graphischen Oberfläche von *OGRUT* aufgerufen wird.

ofg_dar.write_wsp_strukturiert(file_name, **_optional** items)

- Diese Methode schreibt die Zeitreihen der Wasserstandsdaten an den einzelnen Querprofilen des Gerinnes _self in eine externe Datei. Dabei wird ein strukturiertes Dateiformat verwendet.
- **Parameter** **Typ**
  file_name string
  items hash_table
- **Typ des Rückgabewertes** _unset
- Der Parameter file_name enthält den Namen der Datei, in die geschrieben wird. Fehlen Pfadangabe oder Dateiextension, so werden diese aus der Umgebungsvariable ihf_data_store ergänzt. Der Parameter items enthält Einträge, die für Multithreading und Statusanzeige verwendet werden. items ist _unset, wenn die Methode nicht von der graphischen Oberfläche von *OGRUT* aufgerufen wird.

ofg_dar.write_wsp_unstrukturiert(file_name, **_optional** items)

- Diese Methode schreibt die Zeitreihen der Wasserstandsdaten an den einzelnen Querprofilen des Gerinnes _self in eine externe Datei. Dabei wird ein unstrukturiertes Dateiformat verwendet.
- **Parameter** **Typ**
  file_name string
  items hash_table
- **Typ des Rückgabewertes** _unset

- Der Parameter `file_name` enthält den Namen der Datei, in die geschrieben wird. Fehlen Pfadangabe oder Dateiextension, so werden diese aus der Umgebungsvariable `ihf_data_store` ergänzt. Der Parameter `items` enthält Einträge, die für Multithreading und Statusanzeige verwendet werden. `items` ist `_unset`, wenn die Methode nicht von der graphischen Oberfläche von *OGRUT* aufgerufen wird.

`ofg_dar``.state_concerning(topic, project)`

- Diese Methode prüft, inwieweit die Durchführung von `topic` im *OGRUT*-Projekt `project` ausgeführt wurde, bzw. ausführbar ist. `topic` bezeichnet dabei eines der folgenden möglichen Symbole:
  `:run_wsp, :tin_area, :runoff_in, :d_gwnb_out, :tin_in, :flood_tin, :watertable_in, :flood_area, :geometry_in, :triangulate_tin,`
  Diese Themen bezeichnen jeweils einen Arbeitsschritt bei der Berechnung von Daten in *OGRUT*. Als Rückgabewert liefert die Methode eine der drei Konstanten `:calculable`, `:not_calculable` oder `:calculated`, je nachdem ob die entsprechende Funktion im Moment durchführbar ist, nicht durchführbar ist (weil noch Vorarbeiten fehlen) oder schon durchgeführt wurde.
- **Parameter** — **Typ**
  `topic` — `symbol(`
  `:geometry_in,`
  `:runoff_in,`
  `:watertable_in,`
  `:tin_in,`
  `:triangulate_tin,`
  `:tin_area,`
  `:flood_tin,`
  `:flood_area,`
  `:d_gwnb_out,`
  `:run_wsp)`
  `project` — `ogrut_project`
- **Typ des Rückgabewertes** — `symbol(`
  `:calculable,`
  `:not_calculable,`
  `:calculated)`
- Der Parameter `topic` bezeichnet die Funktion, deren Status abgefragt wird. `project` verweist auf das *OGRUT*-Projekt, in dem das Gerinne enthalten ist.

B.1.2 Klasse Profile

In der Klasse Profile (`lw_profil`) wird das Schlüsselfeld `ofg_dar_ofg_name` vom Oberflächengewässer (`ofg_dar`) vererbt, dem das Profil zugeordnet ist. Als weiteres Schlüsselfeld kommt `profil_nr` hinzu. Diese beiden Attribute zusammen identifizieren das Querprofil in der Datenbank eindeutig. Beim Einlesen von Geometriedaten aus externen Dateien wird der Abstand (Feld `abstand`) zwischen zwei aufeinanderfolgenden Profilen aus der Kilometrierung (`km`) erfasst. Das Feld `ofg_dar_rec` verweist auf das Gerinnne (`ofg_dar`), dem das Querprofil zugeordnet ist. Das Feld `lw_stuetzpunkte` verweist auf die einzelnen Stationen (Querprofilpunkte), die die Geometrie des Querprofils ausmachen. `ofg_zeitreihe` verweist auf Zeitreihendaten, die dem Profil zugeordnet sind (Abfluss, Wasserstand).

Tab. B.3: Datenmodell der Klasse `Profile`.

Objektname: lw_profil (Profile)			
Physikalische Felder		**Feldtyp**	
*SI	ofg_dar_ofg_name	ds_char_vec (30)	
*S	profil_nr	ds_int	
S	datum	ds_time	
S	km	ds_float	
S	abstand	ds_float	
S	festpunkt	ds_float	
Geometriefelder		**Feldtyp**	
	annotation	text	
	lage	simple_chain	
Verknüpfungsfelder		**Beziehungstyp**	**Zu Objekt**
	ofg_dar_rec	1:n	ofg_dar
	lw_stuetzpunkte	1:n	lw_stuetzpunkte
	ofg_zeitreihe	1:n	ofg_zeitreihe

B.1.3Klasse Stützpunkte

Die Klasse Stütztpunkte (`lw_stuetzpunkte`) enthält drei Schlüsselfelder, zwei davon sind vererbt: `lw_profil_ofg_dar_ofg_name` vom Oberflächengewässer, `lw_profil_profil_nr` vom Querprofil, das den Stützpunkt enthält. Hinzu kommt das neue Schlüselfeld `id`.. Die Koordinaten eines Stützpunktes werden durch Rechts- und Hochwerte (Felder `hw` und `rw`) charakterisiert. Dabei wird beim Einlesen der Geometriedaten die signifikanteste (höchstwertige) Ziffer abgeschnitten, da das Koordinatensystem von *LIWIS* nur mit den nachgeordneten Stellen arbeitet, also z.B. (x, y) = (3418943.623 *m*, 5316279.496 *m*) wird in der Datenbank als (418943623 *mm*, 316279496 *mm*) abgespeichert. Beim Einlesen wird die Ordnung der Stützpunkte nach den Koordinaten ermittelt. Der Stützpunkt mit der niedrigsten Koordinate erhält die Nummer 1 (Feld `stuetzpunkt_nr`), die anderen Stützpunkte werden entsprechend hochgezählt. Dann wird der Abstand (Feld `abstand`) der einzelnen Stützpunkte vom ersten Stützpunkt berechnet. Das Feld `hoehe` nimmt die Geländehöhe (*z*-Koordinate) des Stützpunktes auf. Der STRICKLER-Beiwert wird im Feld `kst` gespeichert. Das Feld `ufer` enthält einen der Werte `"linkes Ufer"`, `"rechtes Ufer"`, `"nicht gesetzt"` . Diese zeigen an, ob der Stützpunkt sich am Übergang zwischen linkem Vorland und Flusschlauch bzw. rechtem Vorland und Flussschlauch befindet und werden von *UWSP* bei der Aufteilung des Querprofils verwendet. `lw_profil_rec` zeigt auf das Querprofil, dem der Stützpunkt zugeordnet ist.

Tab. B.4 Datenmodell der Klasse Stützpunkte.

Objektname: lw_stuetzpunkt (Stützpunkte)		
Physikalische Felder	**Feldtyp**	
* I lw_profil_ofg_dar_ofg_name		
*SI lw_profil_profil_nr		
*S id	sys_id	
S hw	ds_double	
S rw	ds_double	
S stuetzpunkt_nr	ds_int	
S abstand	ds_float	
S hoehe	ds_float	
S ufer	ufer	
S kst	ds_float	
Verknüpfungsfelder	**Beziehungstyp**	**Zu Objekt**
lw_profil_rec	1:n	lw_profil

B.1.4Klasse OFG Zeitreihe

Die Klasse OFG Zeitreihe (ofg_zeitreihe) enthält die den Querprofilen zugeordneten Zeitreihenwerte von Abfluss (Feld q) und Wasserstand (Feld h). *OGRUT* verwendet nur das Feld h.. Neben den vererbten Schlüsselfeldern des Oberflächengewässers (Feld ofg_name) und des Querprofils (profil_nr) dient das Feld zeit als weiteres Schlüsselfeld. Es kennzeichnet den einzelnen Zeitreihenwert eindeutig. Das Verknüpfungsfeld lw_profil verweist zurück auf das Querprofil, dem der Zeitreihenwert zugeordnet ist.

Tab. B.5: Datenmodell der Klasse OFG Zeitreihe.

Objektname: ofg_zeitreihe (OFG Zeitreihe)		
Physikalische Felder	**Feldtyp**	
* I ofg_name	ds_char_vec (30)	
* I profil_nr	ds_int	
*S zeit	ds_time	
S q	ds_float	
S h	ds_float	
Verknüpfungsfelder	**Beziehungstyp**	**Zu Objekt**
lw_profil	1:n	lw_profil

B.1.5Klasse Oberflächengewässerpegel

Die Klasse Oberflächengewässerpegel (ofg_pegel) hat als Schlüsselfelder das von ofg_dar (Oberflächengewässer) vererbte Feld ofg_dar_ofg_name und zusätzlich das Feld int_bez (interne Bezeichnung). int_bez enthält den Namen des Oberflächengewässerpegels. Das Verknüpfungsfeld messdaten_rec zeigt die dem Pegel zugeordneten Wasserstandszeitreihe (Klasse ofg_pegel_daten). Das Verknüpfungsfeld mengen_rec zeigt auf die dem Pegel zugeordnete Durchflussmenge (Klasse ofg_pegel_mengen). ofg_dar_rec verweist auf das Oberflächengewässer zurück, dem der Pegel zugeordnet ist.
Das Feld profil enthält die Nummer des Querprofils, dem der Oberflächengewässerpegel zugeordnet ist. Diese Zuordnung wird verwendet, um bei der Berechnung der Wasserstände in *UWSP* zu ermitteln, welcher Abfluss an welchen Querprofil anliegt. Der oberste Pegel gibt

den Abfluss (bzw. die Abflusszeitreihe) vom obersten Querprofil bis zum nächsttieferen Pegel vor. Der nächste Pegel wird wiederum für das angegebene Querprofil und die stromabwärts liegenden Querprofile bis zum nächsten Pegel verwendet, usw. Daraus folgt, dass mindestens ein Pegel, und zwar für das oberste Querprofil, angegeben werden muss.

Tab. B.6: Datenmodell der Klasse Oberflächengewässerpegel.

Objektname: ofg_pegel (Oberflächengewässerpegel)			
Physikalische Felder		**Feldtyp**	
*S	int_bez	ds_char_vec (25)	
*SI	ofg_dar_ofg_name	ds_char_vec (30)	
S	ext_bez_1	ds_char_vec (25)	
	gemarkung	ds_char_vec (35)	
	lgb_nr	ds_char_vec (35)	
	wsg_zone	ds_char_vec (15)	
	nn_hoehe	ds_float	
	g_ordnung	ds_char_vec (35)	
	messturnus	ds_charci_vec (25)	
	termin	ds_date	
	vermerk	ds_char_vec (75)	
S	profil	ds_uint	
Logische Felder		**Feldtyp**	
	x_coord	ds_char_vec (20)	
	y_coord	ds_char_vec (20)	
Geometriefelder		**Feldtyp**	
	punkt	point	
	diagramm_punkt	simple_point	
	annotation	text	
Text-Verknüpfungsfelder			
	text		
Verknüpfungsfelder		**Beziehungstyp**	**Zu Objekt**
	messdaten_rec	1:n	ofg_pegel_daten
	mengen_rec	1:n	ofg_pegel_mengen
	qualitaet_rec	1:n	ofg_pegel_qualitaet
	ofg_dar_rec	1:n	ofg_dar

ofg_pegel.get_or_create(int_bez, ofg_dar_name)

- Diese Methode sucht nach dem Pegel mit dem Namen int_bez im Gerinne ofg_dar_name. Wird kein entsprechender Pegel gefunden, so wird ein neuer erzeugt.
- **Parameter** **Typ**
  int_bez string
  ofg_dat_name string
- **Typ des Rückgabewertes** ofg_pegel
- Der Parameter int_bez ist der Namen des gesuchten bzw. zu erzeugenden Pegels, ofg_dar_name ist der Name des Gerinnes.

ofg_pegel.read_abfluss(file_name, position)

- Diese Methode liest eine Abflusszeitreihe aus einer externen Datei ein. Sie wird an den Pegel _self angehängt.

- **Parameter** **Typ**
  file_name string
  position integer
- **Typ des Rückgabewertes** _unset
- Der Parameter file_name enthält den Namen der Datei, in die geschrieben wird. Fehlen Pfadangabe oder Dateiextension, so werden diese aus der Umgebungsvariable ihf_data_store ergänzt. Das Feld position gibt die Spalte in der Datei an, aus der die Abflusszeitreihe gelesen werden soll.

B.1.6 Klasse Wasserstände

Die Klasse Wasserstände (ofg_pegel_daten) bildet die einzelnen Zeitreihenwerte für die Wasserstände an den Oberflächengewässerpegeln ab. Die Schlüsselfelder int_bez (geerbt von ofg_pegel), ofg_pegel_ofg_dar_ofg_name (geerbt von ofg_dar) und datum kennzeichnen den Zeitreihenwert eindeutig. Der Wasserstand wird im Feld wert gespeichert.

Tab. B.7: Datenmodell der Klasse Wasserstände.

Objektname: ofg_pegel_daten (Wasserstände)		
Physikalische Felder	**Feldtyp**	
*S datum	ds_time	
*SI int_bez	ds_char_vec (25)	
* I ofg_pegel_ofg_dar_ofg_name		ds_char_vec (30)
S wert	ds_float	
Verknüpfungsfelder	**Beziehungstyp**	**Zu Objekt**
ofg_pegel_daten	1:n	ofg_pegel

B.1.7 Klasse Durchflussmenge

Die Klasse Durchflussmenge (ofg_pegel_mengen) bildet die einzelnen Zeitreihenwerte für die Durchflüsse an den Oberflächengewässerpegeln ab. Die Schlüsselfelder int_bez (geerbt von ofg_pegel), ofg_pegel_ofg_dar_ofg_name (geerbt von ofg_dar) und datum kennzeichnen den Zeitreihenwert eindeutig. Der Durchfluss wird im Feld wert gespeichert.

Tab. B.8: Datenmodell der Klasse Durchflussmenge.

Objektname: ofg_pegel_mengen (Durchflussmenge)		
Physikalische Felder	**Feldtyp**	
*S datum	ds_time	
*SI int_bez	ds_char_vec (25)	
* I ofg_pegel_ofg_dar_ofg_name	ds_char_vec (30)	
S wert	ds_float	
vermerk	ds_char_vec (35)	
Verknüpfungsfelder	**Beziehungstyp**	**Zu Objekt**
ofg_pegel_rec	1:n	ofg_pegel

ofg_pegel_mengen.get_or_create(wert, datum, int_bez, ofg_dar_name)

- Diese Methode sucht nach dem ofg_pegel_mengen -Objekt mit den in den Parametern angegebenen Eigenschaften (datum, int_bez und ofg_dar_name). Wird kein entsprechendes Objekt gefunden, so wird ein neues erzeugt.
- **Parameter** **Typ**
  wert float
  datum string
  int_bez string
  ofg_dar_name string
- **Typ des Rückgabewertes** ofg_pegel_mengen
- Der Parameter wert ist der zu setzende Wert im Zeitreihenwert zum Zeitpunkt date der Abflusszeitreihe. int_bez ist der Name des Pegels, ofg_dar_name der Name des Gerinnes.

B.1.8 Prozeduren von OGRUT OFG

Die folgenden vier Prozeduren werden genutzt, um die einzelnen Stützpunkte eines Querprofils nach aufsteigenden Koordinaten zu ordnen. Falls dann das rechte Ufer vor dem linken Ufer kommt, so wird die Reihenfolge umgekehrt, so dass dann eine absteigende Ordnung entsteht.

max_rw_order_proc (e1, e2)

- Diese Prozedur vergleicht zwei Stützpunkte bezüglich ihres Rechtswertes. Zurückgegeben wird das Ergebnis des Vergleichsoperators _cf (KLEENE-Operator), der die drei Zustände _true, _false und _maybe hat. Diese entsprechen den Vergleichsergebnissen grösser, kleiner und gleich.
- **Parameter** **Typ**
  e1, e2 lw_stuetzpunkt
- **Typ des Rückgabewertes** kleene
- Die Parameter e1 und e2 sind die beiden Stützpunkte, die verglichen werden.

min_rw_order_proc (e1, e2)

- Diese Prozedur vergleicht zwei Stützpunkte bezüglich ihres Rechtswertes. Zurückgegeben wird das Ergebnis des Vergleichsoperators _cf (KLEENE-Operator), der die drei Zustände _true, _false und _maybe hat. Diese entsprechen den Vergleichsergebnissen grösser, kleiner und gleich.
- **Parameter** **Typ**
  e1, e2 lw_stuetzpunkt
- **Typ des Rückgabewertes** kleene
- Die Parameter e1 und e2 sind die beiden Stützpunkte, die verglichen werden.

max_hw_order_proc (e1, e2)

- Diese Prozedur vergleicht zwei Stützpunkte bezüglich ihres Hochwertes. Zurückgegeben wird das Ergebnis des Vergleichsoperators _cf (KLEENE-Operator), der die drei Zustände _true, _false und _maybe hat. Diese entsprechen den Vergleichsergebnissen grösser, kleiner und gleich.
- **Parameter** **Typ**
  e1, e2 lw_stuetzpunkt

- **Typ des Rückgabewertes** `kleene`
- Die Parameter `e1` und `e2` sind die beiden Stützpunkte, die verglichen werden.

`min_hw_order_proc (e1, e2)`

- Diese Prozedur vergleicht zwei Stützpunkte bezüglich ihres Hochwertes. Zurückgegeben wird das Ergebnis des Vergleichsoperators `_cf` (KLEENE-Operator), der die drei Zustände `_true`, `_false` und `_maybe` hat. Diese entsprechen den Vergleichsergebnissen grösser, kleiner und gleich.
- **Parameter** **Typ**
  `e1, e2` `lw_stuetzpunkt`
- **Typ des Rückgabewertes** `kleene`
- Die Parameter `e1` und `e2` sind die beiden Stützpunkte, die verglichen werden.

`dist (x1, y1, x2, y2)`

- Diese Prozedur berechnet die Distanz zweier Punkte ($x1$, $y1$) und ($x2$, $y2$)
- **Parameter** **Typ**
  `x1, y1, x2, y2` `float`
- **Typ des Rückgabewertes** `float`
- Die Parameter sind Koordinaten der Punkte, deren Distanz berechet wird.

`min_pegel_order_proc(e1, e2)`

- Diese Prozedur vergleicht zwei Pegel aufgrund des Querprofils, mit dem sie assoziiert sind. Zurückgegeben wird das Ergebnis des Vergleichsoperators `_cf` (KLEENE-Operator), der die drei Zustände `_true`, `_false` und `_maybe` hat. Diese entsprechen den Vergleichsergebnissen grösser, kleiner und gleich. Entsprechend liegt der erste Pegel stromabwärts oder stromaufwärts relativ zum anderen Pegel oder ist mit ihm identisch. Der Vergleich wird verwendet, um die Abfluss-zeitreihen an Pegeln in der Reihenfolge ihres Auftretens am Gerinne in eine Datei auszugeben.
- **Parameter** **Typ**
  `e1, e2` `ofg_pegel`
- **Typ des Rückgabewertes** `kleene`
- Die Parameter `e1` und `e2` sind die beiden Pegel, die verglichen werden.

B.2Definition der Datenstrukturen von OGRUT DGM

B.2.1Klasse OFG Tin

Die Klasse OFG Tin (ofg_tin) enthält als Geometriefeld ein Tin (*triangulated irregular network*). Dieses besteht aus einzelnen digitalen Geländepunkten (Klasse sw_tin!primal_site). Jeder dieser Geländepunkte oder sites hat drei Koordinaten. *x* und *y* geben den Rechts- und Hochwert an, *z* ist die Geländehöhe. Ein digitales Geländemodell kann einem Oberflächengewässer zugeordnet sein (Verknüpfungsfeld ofg_dar). Dies ermöglicht die Berechnung der Überflutung des Tin durch das Gewässer.

Tab. B.9: Datenmodell der Klasse OFG Tin.

Objektname: ofg_tin (OFG Tin)		
Physikalische Felder	**Feldtyp**	
*S id	sys_id	
Geometriefelder	**Feldtyp**	
tin	tin	
Verknüpfungsfelder	**Beziehungstyp**	**Zu Objekt**
ofg_dar	0:n	ofg_dar

ofg_tin.add_site_to_kat_flaeche(katasterflaeche, site)

- Diese Methode erzeugt für jede Katasterfläche ein Objekt der Klassen kat_flaeche_tins und kat_flaeche_sites, falls diese Objekte nicht schon existieren. Dann wird ein Objekt der Klasse kat_flaeche_site erzeugt und die Verbindung zwischen der site und der Katasterfläche eingetragen. Die Katasterfläche und die site werden verbunden, da die site räumlich in der Katasterfläche liegt.
- **Parameter** **Typ**
  katasterflaeche ihf_katasterflaeche_auswertung
  site sw_tin!primal_site
- **Typ des Rückgabewertes** _unset
- Der Parameter katasterflaeche ist die Katasterfläche, die mit dem digitalen Geländepunkt, der durch site angegeben ist, verbunden werden soll.

ofg_tin.aufteilen(river_name_vec)

- Diese Methode teilt das Tin _self in eine Reihe kleinerer Tins auf. Dabei wird jede site in dem Tin daraufhin untersucht, welches Gerinne in der in river_name_vec angegebenen Menge von Gerinnen die geringste Entfernung zur site hat. Diesem Gerinne wird die site dann zugeordnet. Es entstehen also aus einem Tin *n* neue Tins, wenn river_name_vec *n* Gerinne enthält.
- **Parameter** **Typ**
  river_name_vec simple_vector(string)
- **Typ des Rückgabewertes** _unset

- Der Parameter river_name_vec enthält eine Menge von Strings (Zeichenketten), die dem Feld ofg_name in der Klassse ofg_dar (Oberflächengewässer) entspricht. Die einzelnen Gerinne werden also durch ihre Namen übergeben.

ofg_tin.calculate_gwnb(h, kf_wert)

- Diese Methode berechnet die indirekte die Grundwasserneubildung für einen Zeitschritt.
- **Parameter** **Typ**
  h float
  kf_wert float
- **Typ des Rückgabewertes** _unset
- Der Parameter h ist die Überstauhöhe des Wassers, kf_wert ist der gesättigte hydraulische Durchlässigkeitsbeiwert.

ofg_tin.create_time_series(
delete_old?, start_date, stop_date, _**optional** items)

- Diese Methode berechnet für jede Katasterfläche und für jede kat_flaeche_sites-Menge, die dieser Katasterfläche zugeordnet ist, die Zeitreihe der indirekten Grundwasserneubildung. Bei dieser Berechung wird die Grundwasserneubildung direkt vom digitalen Geländemodell auf die Isoflächen übertragen. Daher wird hier nur die indirekte Grundwasserneubildung aus Überflutungsflächen berücksichtigt.
- **Parameter** **Typ**
  delete_old? boolean
  start_date string
  stop_date string
  items hash_table
- **Typ des Rückgabewertes** _unset
- Der Parameter delete_old? legt fest, ob bestehende Zeitreihen gelöscht oder ergänzt werden sollen. start_date und stop_date sind der Anfangzeitpunkt und der Endzeitpunkt der zu berechnenden Zeitreihe. Wenn sie _unset sind, wird die gesamte Zeitreihe berechnet. Der Parameter items enthält Einträge, die für Multithreading und Statusanzeige verwendet werden. items ist _unset, wenn die Methode nicht von der graphischen Oberfläche von *OGRUT* aufgerufen wird.

ofg_tin.delete_all_sites_from_kat_flaeche()

- Diese Methode löscht alle kat_flaeche_site -Objekte, die eine Verbindung zwischen sites des Tins _self und Katasterflächen herstellen.
- **Typ des Rückgabewertes** _unset

ofg_tin.delete_h(_**optional** items)

- Diese Methode löscht die ofg_tin_z_reihenwert -Objekte des Tins _self. Es werden also die Wasserstandszeitreihen des Tins gelöscht.
- **Parameter** **Typ**
  items hash_table
- **Typ des Rückgabewertes** _unset
- Der Parameter items enthält Einträge, die für Multithreading und Statusanzeige verwendet werden. items ist _unset, wenn die Methode nicht von der graphischen Oberfläche von *OGRUT* aufgerufen wird.

ofg_tin.new()

- Diese Methode erzeugt eine neue Instanz der Klasse ofg_tin.
- **Typ des Rückgabewertes** ofg_tin

ofg_tin.read_data_menu(file_name, delete_old_sites?, items)

- Diese Methode liest die Daten eines digitalen Geländemodells aus einer externen Datei ein.
- **Parameter** **Typ**
  file_name string
  delete_old_sites? boolean
  items hash_table
- **Typ des Rückgabewertes** _unset
- Der Parameter file_name bezeichnet die Datei, aus der gelesen wird. delete_old_sites? legt fest, ob vorhandene Geländepunkte gelöscht (_true) oder ergänzt (_false) werden. Der Parameter items enthält Einträge, die für Multithreading und Statusanzeige verwendet werden. items ist _unset, wenn die Methode nicht von der graphischen Oberfläche von *OGRUT* aufgerufen wird.

ofg_tin.reinit_tin()

- Diese Methode löscht die Geländepunkte eines Tins.
- **Typ des Rückgabewertes** _unset

ofg_tin.set_kat_flaeche(delete_old?, **_optional** items)

- Diese Methode ermittelt für alle sites im Tin _self jeweils die Katasterfläche, in der die site liegt. Die sites werden der gefundenen Katasterfläche zugeordnet und das Gewicht jeder site für die Katasterfläche berechnet.
- **Parameter** **Typ**
  delete_old? boolean
  items hash_table
- **Typ des Rückgabewertes** _unset
- Der Parameter delete_old? legt fest, ob vorhandene Zuordnungen gelöscht (_true) oder ergänzt (_false) werden sollen. Der Parameter items enthält Einträge, die für Multithreading und Statusanzeige verwendet werden. items ist _unset, wenn die Methode nicht von der graphischen Oberfläche von *OGRUT* aufgerufen wird.

ofg_tin.update_h(h, date, site, display_tin)

- Diese Methode erzeugt für die angegebene site ein Objekt ofg_tin_z_werte, an das eine neue Zeitreihe vom Typ ofg_tin_z_reihe angehängt wird. Diese Zeitreihe erhält für den angegebenen Zeitreihenschritt ein Objekt ofg_tin_z_reihenwert. Existiert das Objekt schon, dann wird der bestehende Wasserstand betrachtet und modifiziert, wenn der neue Wasserstand h höher ist. Wurde das ofg_tin_z_reihenwert -Objekt modifiziert (bzw. neu erzeugt), so wird _true zurückgegeben, sonst _false.
- **Parameter** **Typ**
  h float
  date date_time

site	sw_tin!primal_site
display_tin	ofg_tin

- **Typ des Rückgabewertes** boolean
- Der Parameter h ist der Wasserstand, der für den digitalen Geländepunkt site gesetzt werden soll. date ist der Zeitpunkt der Zeitreihe, an dem dieser Wasserstand auftritt. Wenn display_tin nicht _unset ist, so wird die site in dieses Tin kopiert, falls update_h() den Wasserstand der site setzt bzw. ändert.

ofg_tin.write_tins(path_string, **_optional** max)

- Diese Methode gibt die einzelnen sites des Tin _self in eine externe Datei aus.
- **Parameter** **Typ**

path_string	string
max	x

- **Typ des Rückgabewertes** _unset
- Der Parameter path_string gibt den Pfad und Namen der Datei an, in die geschrieben wird. Mit max kann festgelegt werden, dass nur sites mit einem *z*-Wert < max bei der Ausgabe berücksichtigt werden.

B.2.2 Klasse OFG Tin Z-Werte

In der Klasse OFG Tin Z-Werte (ofg_tin_z_werte) werden die beiden Felder triangulation_id und id als Schlüsselfelder verwendet. Dies sind auch genau die Schlüsselfelder einer site. Die Wahl dieser Schlüsselfelder führt also zu einer 1:1-Zuordnung zwischen sw_tin!primal_site und OFG Tin Z-Werte. Es handelt sich hierbei aber um keine Relation, wie man aus den Verknüpfungsfeldern in Tab. B.10 erkennt. Hier ist nur eine Relation z_wert aufgeführt, die auf eine Liste von Zeitreihen (Klasse ofg_tin_z_reihe) verweist. Das Feld kat_flaeche_sys_id zeigt auf die Katasterfläche, in der die site liegt (falls hierfür eine Katasterfläche vorhanden ist). Der im Feld kf_wert gespeicherte gesättigte hydraulische Durchlässigkeitsbeiwert wird aus der Hydropedotopfläche (Klasse ihf_hydropedotop) gewonnen, in der die site liegt.

Tab. B.10: Datenmodell der Klasse OFG Tin Z-Werte.

Objektname: ofg_tin_z_werte (OFG Tin Z-Werte)		
Physikalische Felder	**Feldtyp**	
*S triangulation_id	ds_uint	
*S id	ds_uint	
S kat_flaeche_sys_id	sys_id	
S kf_wert	ds_float	
Verknüpfungsfelder	**Beziehungstyp**	**Zu Objekt**
z_wert	1:n	ofg_tin_z_reihe

ofg_tin_z_werte.get_or_create(site)

- Diese Methode sucht das ofg_tin_z_werte -Objekt für die angegebene site und gibt es zurück. Wenn es nicht gefunden wurde, so wird ein neues Objekt erzeugt.
- **Parameter** **Typ**

site	sw_tin!primal_site

- **Typ des Rückgabewertes** ofg_tin_z_werte

- Der Parameter site ist die site, für die die Zeitreihe ermittelt werden soll.

B.2.3Klasse OFG Tin Z-Reihe

Die Klasse OFG Tin Z-Reihe (ofg_tin_z_reihe) stellt eine Zeitreihe von Werten an einer site dar (s. Tab. B.11). Die Schlüsselfelder id und triangulation_id sind von der site vererbt, der diese Zeitreihe zugeordnet ist. Hinzu kommt das Schlüsselfeld typ, das die Zeitreihe kennzeichnet (z.B. Wasserstand, Grundwasserneubildung). Das Verknüpfungsfeld z_reihe zeigt auf die einzelnen Zeitreihenwerte. Das Verknüpfungsfeld z_werte zeigt zurück auf die site.

Tab. B.11: Datenmodell der Klasse OFG Tin Z-Reihe.

Objektname: ofg_tin_z_reihe (OFG Tin Z-Reihe)		
Physikalische Felder	**Feldtyp**	
*S typ	ds_char_vec (30)	
* I triangulation_id	ds_uint	
* I id	ds_uint	
Verknüpfungsfelder	**Beziehungstyp**	**Zu Objekt**
z_reihe	1:n	ofg_tin_z_reihenwert
z_werte	1:n	ofg_tin_z_werte

ofg_tin_z_reihe.get_or_create(typ, site)

- Diese Methode sucht das ofg_tin_z_reihe -Objekt mit den in den Parametern angegebenen Eigenschaften und gibt es zurück. Wenn es nicht gefunden wurde, so wird ein neues Objekt erzeugt.
- **Parameter** **Typ**
  typ string
  site sw_tin!primal_site
- **Typ des Rückgabewertes** ofg_tin_z_reihe
- Der Parameter typ ist der Typ des zu ermittelnden bzw. zu erzeugenden Zeitreihenwertes (z.B. "h" für Wasserstandswerte). site ist der digitale Geländepunkt, für den die Zeitreihe ermittelt werden soll.

B.2.4Klasse OFG Tin Z-Reihenwert

Die Klasse OFG Tin Z-Reihenwert (ofg_tin_z_reihenwert) enthält einzelne Werte einer Zeitreihe von Daten an einer site (s. Tab. B.12). Die Schlüsselfelder id und triangulation_id sind von ofg_z_werte vererbt und kennzeichnen die site, der dieser Zeitreihenwert zugeordnet ist. Das vererbte Schlüsselfeld typ kennzeichnet die Zeitreihe, in der dieser Wert enthalten ist. das Feld zeit ist von der Klasse ds_time und identifiziert den einzelnen Zeitreihenwert eindeutig. Der eigentliche Wert ist im Feld wert gespeichert.
Das Verknüpfungsfeld z_wert zeigt zurück zur Zeitreihe, der dieser Zeitreihenwert zugeordnet ist.

Tab. B.12: Datenmodell der Klasse OFG Tin Z-Reihenwert.

Objektname: ofg_tin_z_reihenwert (OFG Tin Z-Reihenwert)		
Physikalische Felder	**Feldtyp**	
*S zeit	ds_time	
* I typ	ds_char_vec (30)	
* I triangulation_id	ds_uint	
* I id	ds_uint	
S wert	ds_float	
Verknüpfungsfelder	**Beziehungstyp**	**Zu Objekt**
z_wert	1:n	ofg_tin_z_reihe

ofg_tin_z_reihenwert.get_or_create(date, typ, site)

- Diese Methode sucht das ofg_tin_z_reihenwert -Objekt mit den in den Parametern angegebenen Eigenschaften und gibt es zurück. Wenn es nicht gefunden wurde, so wird ein neues Objekt erzeugt.
- **Parameter** **Typ**
  date date_time
  typ string
  site sw_tin!primal_site
- **Typ des Rückgabewertes** ofg_tin_z_reihenwert
- Der Parameter date gibt den Zeitpunkt an, für den ein bestehender Zeitreihenwert gesucht bzw. erzeugt werden soll. typ ist der Typ des Zeitreihenwertes (z.B. "h" für Wasserstandswerte). site ist der Geländepunkt, für die die Zeitreihe ermittelt werden soll.

B.2.5 Klasse IHF-Hydropedotop

Der k_f-Wert, der in der Klasse ofg_tin_z_werte gespeichert wird, wird aus dem Hydropedotop ermittelt, in dem die site liegt. Er ist im Feld kf_flaeche der Klasse ihf_hydropedotop zu finden (s. Tab. B.13).

Tab. B.13: Datenmodell der Klasse IHF Hydropedotop.

Objektname: ihf_hydropedotop (IHF Hydropedotop)			
Physikalische Felder		**Feldtyp**	
*S	bodtypflaechennr	ds_uint	
*S	hptnr	ds_uint	
S	pwp_flaeche	ds_float	
S	fk_flaeche	ds_float	
S	nfk_flaeche	ds_float	
S	lk_flaeche	ds_float	
S	gpv_flaeche	ds_float	
S	kf_klasse_flaeche	ihf_code_ft	
S	kf_flaeche	ds_float	
S	kakpot_flaeche	ds_float	
S	kakeff_flaeche	ds_float	
S	we_flaeche	ds_float	
S	pwpwe_flaeche	ds_float	
S	fkwe_flaeche	ds_float	
S	nfkwe_flaeche	ds_float	
S	lkwe_flaeche	ds_float	
S	gpvwe_flache	ds_float	
	text_id	ds_uint	
S	flache	ds_charci_vec(10)	
S	umfang	ds_charci_vec(10)	
Geometriefelder		**Feldtyp**	
S	polygon	area	
S	annotation	text	
Text-Verknüpfungsfelder		**Beziehungstyp**	
S	bemerkungen	textfeld(text_id)	
Verknüpfungsfelder		**Beziehungstyp**	**Zu Objekt**
S	ihf_bodtyp_flaeche	0:n	ihf_bodtyp_flaeche
S	ihf_hpt_statistiks	0:n	ihf_htp_statistik
S	ihf_kf_klass_katalog	0:n	ihf_kf_klass_katalog

B.3Definition der Datenstrukturen von OGRUT AGR

B.3.1Klasse Kat.Flaeche Tins

Die Klasse Kat.Fläche Tins (kat_flaeche_tins) ist einer Katasterfläche zugeordnet, und zwar einer Instanz der Klasse ihf_katasterflaeche_auswertung (s. Tab. B.14). Das Schlüsselfeld kat_flaeche_id entspricht daher dem Schlüsselfeld des assoziierten ihf_katasterflaeche_auswertung -Objektes. Es wird jedoch keine Relation verwendet. Das Verknüpfungsfeld tins verweist auf eine Menge von digitalen Geländemodellen. Diese wiederum bestehen aus sites, die in der betrachteten Katasterfläche liegen. Es wird also durch diese Datenstrukturen für eine beliebige Anzahl von Tins die statische Zuordnung der darin enthaltenen sites zur jeweiligen Katasterfläche abgebildet. Dies erlaubt später das schnelle Auffinden aller sites eines digitalen Geländemodells, die in einer gegebenen Katasterfläche liegen. Diese Information wird benötigt, um Zeitreihendaten von digitalen Geländemodellen auf Katasterflächen übertragen zu können.

Tab. B.14: Datenmodell der Klasse Kat.Fläche Tins.

Objektname: kat_flaeche_tins (Kat.Fläche Tins)			
Physikalische Felder		**Feldtyp**	
*S	kat_flaeche_id	sys_id	
Verknüpfungsfelder		**Beziehungstyp**	**Zu Objekt**
	tins	1:n	kat_flaeche_sites

kat_flaeche_tins.get_or_create(katasterflaeche_id)

- Diese Methode sucht das kat_flaeche_sites -Objekt, für die Katasterfläche mit der Nummer katasterflaeche_id. Wenn keines existiert, dann wird ein neues erzeugt. Durch das Objekt wird eine Beziehung zwischen einer Katasterfläche und einer Anzahl digitaler Geländemodelle hergestellt.
- **Parameter** **Typ**
  katasterflaeche_id sys_id
- **Typ des Rückgabewertes** kat_flaeche_tins
- Der Parameter katasterflaeche_id bezeichnet die Katasterfläche, für die ein Eintrag gefunden oder hergestellt werden soll.

B.3.2Klasse Kat.Fläche Sites

Die Klasse Kat.Fläche Sites (kat_flaeche_sites) stellt die Verbindung zwischen einer Katasterfläche und einem digitalen Geländemodell her (s. Tab. B.15). kat_flaeche_id ist das vererbte Schlüsselfeld der Katasterfläche, tin_id das Schlüsselfeld, das die Beziehung zum verknüpften Tin bildet. Beide Verbindungen sind keine Relationen. Das Verknüpfungsfeld tins_parent verweist zurück zum Objekt der Klasse kat_flaeche_tins, das die Verbindung zur Katasterfläche herstellt. Das Verknüpfungsfeld sites verweist auf eine Liste von sites aus dem gegebenen Tin, die räumlich in der betreffenden Katasterfläche liegen.

Tab. B.15: Datenmodell der Klasse Kat.Fläche Sites.

Objektname: kat_flaeche_sites (Kat.Fläche Sites)			
Physikalische Felder		**Feldtyp**	
*S	tin_id	sys_id	
* I	kat_flaeche_id	sys_id	
Verknüpfungsfelder		**Beziehungstyp**	**Zu Objekt**
	sites	1:n	kat_flaeche_site
	tins_parent	1:n	kat_flaeche_tins

kat_flaeche_sites.get_or_create(tin_id, katasterflaeche_id)

- Diese Methode sucht das kat_flaeche_sites -Objekt, für das die Felder tin_id und katasterflaeche_id gelten. Wenn keines existiert, dann wird ein neues erzeugt. Durch das Objekt wird eine Beziehung zwischen einer Katasterfläche und einem digitalen Geländemodell hergestellt. Daraufhin können einzelne Punkte aus dem DGM aufgenommen werden, die in der Katasterfläche liegen.

- **Parameter** **Typ**
  tin_id sys_id
  katasterflaeche_id sys_id
- **Typ des Rückgabewertes** kat_flaeche_sites
- Der Parameter tin_id bezeichnet das digitale Geländemodell, für das eine Verbindung zur Katasterfläche mit der Nummer katasterflaeche_id hergestellt werden soll.

B.3.3 Klasse Kat.Fläche Site

Die Klasse Kat.Fläche Site (kat_flaeche_site) verweist auf eine site, die in einer Katasterfläche liegt (s. Tab. B.16). Zu den vererbten Schlüsselfeldern tin_id, das das Tin kennzeichnet und kat_flaeche_id, das auf die Katasterfläche verweist, kommen die Schlüsselfelder der site hinzu (triangulation_id und id). Das Feld gewicht enthält die Gewichtung der site. Daraus errechnet sich der Beitrag dieser site zur Grundwasserneubildung der Katasterfläche.
Das Verknüpfungsfeld sites_parent verweist zurück auf das kat_flaeche_sites -Objekt, das auf alle sites eines Tins verweist, die in der Katasterfläche mit dem gegebenen kat_flaeche_id liegen.

Tab. B.16: Datenmodell der Klasse Kat.Fläche Site.

Objektname: kat_flaeche_site (Kat.Fläche Site)		
Physikalische Felder	**Feldtyp**	
*S triangulation_id	sys_id	
*S id	sys_id	
* I tin_id	sys_id	
*SI kat_flaeche_id	sys_id	
S gewicht	ds_float	
Verknüpfungsfelder	**Beziehungstyp**	**Zu Objekt**
sites_parent	1:n	kat_flaeche_sites

kat_flaeche_site.get_or_create(site, tin_id, katasterflaeche_id)

- Diese Methode sucht das kat_flaeche_site -Objekt, dessen Felder mit site.id, site.triangulation_id, tin_id und katasterflaeche_id übereinstimmen. Wenn noch kein solches Objekt existiert, dann wird ein neues erzeugt. Das Objekt stellt eine Beziehung zwischen einer Katasterfläche und einer site her, die in dieser Katasterfläche liegt.
- **Parameter** **Typ**
  site sw_tin!primal_site
  tin_id sys_id
  katasterflaeche_id sys_id
- **Typ des Rückgabewertes** kat_flaeche_site
- Der Parameter site bezeichnet den digitalen Geländepunkt aus dem digitalen Geländemodell mit der Nummer tin_id, für den eine Verbindung zur Katasterfläche mit der Nummer katasterflaeche_id hergestellt werden soll.

B.3.4Klasse AGR Grundfläche

Die Klasse AGR Grundfläche (agr_grundflaeche) enthält eine Reihe von landwirtschaftlichen Daten und Felder zur Einordnung der Agrarfläche ins Kataster (gemarkung, flurstuecknr_zaehler, flurstuecknr_nenner). Das Geometriefeld gebiet enthält den Polygonzug, der die Katasterfläche begrenzt (s. Tab. B.17). Diese Fläche wird verwendet um festzustellen, welche Punkte des digitalen Geländemodells in welcher Katasterfläche liegen.
Die Zeitreihendaten für den Überflutungszustand einer Katasterfläche werden nicht direkt als eine Relation auf die Klasse agr_grundflache gespeichert, sondern es wird eine weitere Indirektion verwendet, die bereits von *BOMET* definiert wurde.
BOMET verwendet zu jeder Agrar-Grundfläche eine Klasse IHF Katasterfläche Auswertung (ihf_katasterfläche_auswertung), die auf Zeitreihen von Daten verweist, die der Katasterfläche zugeordnet sind (s. Tab. B.18). Der Grund hierfür ist vor allem die übersichtliche Trennung von bestehenden Modulen und solchen Datenstrukturen, die in *BOMET* neu hinzukommen (EBERLE, 1999).

Tab. B.17: Datenmodell der Klasse AGR Grundfläche.

Objektname: agr_grundflaeche (AGR Grundfläche)

Physikalische Felder		Feldtyp
*	sys_id	sys_id
S	flurstuecknr_zaehler	ds_uint
S	flurstuecknr_nenner	ds_uint
S	gemarkung	ds_char_vec(30)
S	flur	ds_char_vec(30)
S	stand	ds_date
S	bezeichnung	ds_char_vec(30)
S	kat_groesse	ds_uint
S	sap_id	ds_char_vec(30)
S	gemeinde	ds_char_vec(30)
	text_id	ds_uint
S	flaechennutzung	ds_char_vec(30)
S	bodenart	ds_char_vec(30)
S	bodentyp	ds_char_vec(30)
S	geologie	ds_char_vec(30)
S	gruendigkeit	ds_char_vec(30)
S	standort	ds_char_vec(30)
S	hangneigung	ds_char_vec(30)
S	bodenschutz	ds_char_vec(30)
S	acker_gruen_zahl	ds_uint
S	steinanteil	ds_char_vec(30)
	text_id_2	ds_uint
I	agr_person_katalog_sys_id	sys_id
S	teilflaechen_nr	ds_byte
S	ihf_alk_flurstueck_area_id	ds_int_vec(3)
S	groesse	ds_uint
S	beschriftung	text

Geometriefelder		Feldtyp
S	gebiet	area
S	punkt	point

Text-Verknüpfungsfelder		Beziehungstyp
S	bemerkung	Textfeld(text_id)
S	location	Textfeld(text_id_2)

Verknüpfungsfelder		Beziehungstyp	Zu Objekt
S	agr_nitratrueckhaltung_rec	1:0	agr_nitratrueckhaltung
S	agr_bodentyp_rec	n:m	agr_bodentyp
S	agr_entwaesserungssystem_rec	0:n	agr_entwaesserungssystem
S	agr_geologie_rec	n:m	agr_geologie
S	agr_bewirtflaeche_rec	n:m	agr_bewirtflaeche
S	agr_bodenart_rec	n:m	agr_bodenart
S	agr_person_katalog_rec	1:n	agr_person_katalog
S	ihf_katasterflaeche_auswertung	1:0	ihf_kataserflaeche_auswertung

B.3.5 Klasse IHF Katasterfläche (Auswertung)

Die Klasse ihf_katasterflaeche_auswertung und die angegliederten Klassen für Zeitreihen (ihf_kataster_meteo_zeitreihe, ihf_zeitreihe_meteo_abgeleitet) werden in EBERLE (1999) beschrieben. Sie werden verwendet, um Zeitreihen an einzelnen Katasterflächen zu verwalten.

Tab. B.18: Datenmodell der Klasse IHF Katasterfläche (Auswertung).

Objektname: ihf_katasterflaeche_auswertung (IHF Katasterfläche (Auswertung))		
Physikalische Felder		**Feldtyp**
*S	sys_id	sys_id
S	flurstueck_nr_zaehler	ds_uint
S	flurstueck_nr_nenner	ds_uint
S	gemarkung	ds_charci_vec(30)
S	versiegelungsstufe	ds_int
S	nfk_kat_flaechen_mittel	ds_float
S	fk_kat_flaechen_mittel	ds_float
S	zentroid_geaendert?	ds_kleene
S	teilflaechen_nr	ds_byte
S	gemeinde	ds_char_vec(50)
S	versiegel_extrakt	ds_float
I	agr_grundflaeche_sys_id	sys_id
Geometriefelder		**Feldtyp**
S	zentroid	point
Verknüpfungsfelder		**Beziehungstyp**	**Zu Objekt**
S	ihf_versiegelungs_tabelle	0:n	ihf_versiegelungs_tabelle
S	ihf_kataster_meteo_zeitreihen	1:n	ihf_kataster_meteo_zeitreihe
S	ihf_zeitreihe_meteo_abgeleitets	1:n	ihf_zeitreihe_meteo_abgeleitet
S	ihf_n_modell_input_files_s	1:n	ihf_n_modell_input_files
S	agr_grundflaeche	0:1	agr_grundflaeche

ihf_katasterflaeche_auswertung.calculate_time_series(
the_tin, date, step, count, **_optional** items)

- Diese Methode berechnet die indirekte Grundwasserneubildung-Zeitreihe für die betrachtete Katasterfläche _self. Es wird ein Objekt der Klasse ihf_zeitreihe_meteo_abgeleitet erzeugt. Für jeden Zeitschritt wird ein Objekt der Klasse ihf_abgeleit_zeitreihen_wert erzeugt.
- **Parameter** **Typ**
  the_tin ofg_tin
  date date_time
  step time_interval
  count integer
  items hash_table
- **Typ des Rückgabewertes** _unset
- Der Parameter the_tin gibt das digitale Geländemodell an, dessen Grundwasserneubildungsdaten übertragen werden sollen. Die Parameter date und step geben den Startzeitpunkt und die zeitliche Auflösung der Zeitreihe an. Der Parameter items enthält Einträge, die für Multithreading und Statusanzeige verwendet werden. items ist _unset, wenn die Methode nicht von der graphischen Oberfläche von *OGRUT* aufgerufen wird.

ihf_katasterflaeche_auswertung.create_time_series(
type, eval_method, time_step, start, **_optional** items)

- Diese Methode erzeugt ein Objekt der Klasse `ihf_zeitreihe_meteo_abgeleitet` für die betrachtete Katasterfläche.
- **Parameter** **Typ**
  `type` `string`
  `eval_method` `string`
  `time_step` `time_interval`
  `start` `date_time`
  `items` `hash_table`
- **Typ des Rückgabewertes** `_unset`
- Der Parameter `type` gibt den Typ der Zeitreihe an, *OGRUT* verwendet hier `"flood"` für die Zeitreihe der Grundwasserneubildung. `eval_method` gibt die Auswertungsmethode an, `start` und `time_step` den Startzeitpunkt und die zeitliche Auflösung der Zeitreihe. Der Parameter `items` enthält Einträge, die für Multithreading und Statusanzeige verwendet werden. `items` ist `_unset`, wenn die Methode nicht von der graphischen Oberfläche von *OGRUT* aufgerufen wird.

B.3.6 Klasse IHF Zeitreihe Meteo (abgeleitet)

`ihf_zeitreihe_meteo_abgeleitet``.add(value, date, time_step)`

- Diese Methode fügt ein Objekt der Klasse `ihf_abgeleit_zeitreihen_wert` zur Katasterfläche hinzu, falls noch kein entsprechendes Objekt existiert. Ansonsten wird lediglich das Feld `wert` des Objektes zu `value` geändert.
- **Parameter** **Typ**
  `value` `float`
  `date` `date_time`
  `time_step` `time_interval`
- **Typ des Rückgabewertes** `_unset`
- Der Parameter `value` ist der zu setzende Zeitreihenwert für den Zeitpunkt `date`. `time_step` ist die zeitliche Auflösung der Zeitreihe.

B.4 Definition der Datenstrukturen von OGRUT ISO

B.4.1 Mixin lw_mixin

`lw_mixin``.delete_ganglinie(param_name, param_value)`

- Diese Methode löscht eine Ganglinie an einem Iso-Element oder einem Iso-Knoten (beide sind von `lw_mixin` abgeleitete Klassen).
- **Parameter** **Typ**
  `param_name` `string`
  `param_value` `string`
- **Typ des Rückgabewertes** `_unset`
- Der Parameter `param_name` gibt den Parameter der Ganglinie an, dessen Wert ermittelt werden soll. Stimmt der Wert mit `param_value` überein, so ist dies die gesuchte Gangline und sie wird gelöscht.

B.4.2 Klasse Iso-Projekt

Die Klasse Iso-Projekt (iso_project) verwaltet eine Menge finiter Elemente und Knoten (s. Tab. B.19). Der Name des Projektes (Feld name) dient als Schlüsselfeld. Das Verknüpfungsfeld iso_elements verweist auf die Elemente des Projektes, das Feld iso_nodes auf die Knoten. Ein Iso-Projekt kann selbst wiederum in einem oder mehreren OGRUT-Projekten enthalten sein. Diese Verbindung ist durch das Verknüpfungsfeld ogrut_project gegeben. Das Verknüpfungsfeld param_groups verweist auf eine Menge von Parametern, die die Zeitreihen an den einzelnen Elementen bzw. Knoten des Projektes beschreiben.

Tab. B.19: Datenmodell der Klasse Iso-Projekt.

Objektname: iso_project (Iso-Projekt)		
Physikalische Felder	**Feldtyp**	
*S name	ds_char_vec (30)	
info_id	ds_uint	
current?	ds_bool	
netz_n	ds_uint	
Logische Felder	**Feldtyp**	
current_text	ds_char_vec (10)	
project_area	ds_char_vec (10)	
Geometriefelder	**Feldtyp**	
boundary	simple_area	
identification	text	
Text-Verknüpfungsfelder		
info		
Verknüpfungsfelder	**Beziehungstyp**	**Zu Objekt**
iso_elements	1:n	iso_element
iso_nodes	field_mapping	iso_node
iso_stoerung	1:n	iso_stoerung
iso_schnitt	1:n	iso_schnitt
hl_rec	field_mapping	iso_hoehenlinie
hl_params_rec	field_mapping	iso_hl_params
ogrut_projects	0:n	ogrut_project
param_groups	1:n	iso_param_group

iso_project.berechne_direkte_gwnb(delete_old?, _**optional** items)

- Diese Methode berechnet die direkte Grundwasserneubildung für alle Isoflächen des Iso-Projektes _self.
- **Parameter** **Typ**
  delete_old? boolean
  items hash_table
- **Typ des Rückgabewertes** _unset
- Der Parameter delete_old? legt fest, ob die Ergebnisse früherer Berechnungen gelöscht werden (_true) oder die neuen Ergebnisse hinzugefügt werden (_false). Der Parameter items enthält Einträge, die für Multithreading und Statusanzeige verwendet werden. items ist _unset, wenn die Methode nicht von der graphischen Oberfläche von *OGRUT* aufgerufen wird.

iso_project.berechne_indirekte_gwnb(the_tin_set, weight?)

- Diese Methode berechnet ohne Umweg über Katasterflächen die indirekte Grundwasserneubildung direkt aus digitalen Geländepunkten.
- **Parameter** **Typ**
  the_tin_set set(ofg_tin)
  weight? boolean
- **Typ des Rückgabewertes** _unset
- Der Parameter the_tin_set enthält die Tins, deren sites bei der Berechnung berücksichtigt werden sollen. weight? gibt an, ob zuerst eine Gewichtung aller sites für alle in the_tin_set enthaltenen Tins für die einzelnen Iso-Elemente durchgeführt wird (_true).

iso_project.delete_indirekte_gwnb_gewichtung()

- Diese Methode löscht die Gewichtung zwischen Sites und Iso-Elementen für alle Iso-Elemente im Iso-Projekt _self.
- **Typ des Rückgabewertes** _unset

iso_project.delete_nodes_and_elements()

- Diese Methode löscht alle Knoten und Elemente im Iso-Projekt _self.
- **Typ des Rückgabewertes** _unset

iso_project.get_or_create_param_group_with(info)

- Diese Methode sucht nach einer Parametergruppe im Iso-Projekt _self. Wird die entsprechende Parametergruppe nicht gefunden, so wird sie erzeugt.
- **Parameter** **Typ**
  info string
- **Typ des Rückgabewertes** iso_param_group
- Der Parameter info ist der Wert, den das Feld info der gesuchten Parametergruppe gesetzt hat.

iso_project.get_param_group_with(info)

- Diese Methode sucht nach einer Parametergruppe im Iso-Projekt _self.
- **Parameter** **Typ**
  info string
- **Typ des Rückgabewertes** iso_param_group
- Der Parameter info ist der Wert, den das Feld info der gesuchten Parametergruppe gesetzt hat.

iso_project.gewichte_direkte_gwnb(delete_old?, **_optional** items)

- Diese Methode erstellt eine gewichtete Beziehung zwischen Katasterflächen der Datenbank und Iso-Flächen des Iso-Projektes _self. Es werden jeweils die Katasterflächen und Iso-Flächen assoziiert, die gemeinsame Teilflächen haben.
- **Parameter** **Typ**
  delete_old? boolean
  items hash_table
- **Typ des Rückgabewertes** _unset

- Der Parameter delete_old? legt fest, ob die Ergebnisse früherer Berechnungen gelöscht werden (_true) oder die neuen Ergebnisse hinzugefügt werden (_false). Der Parameter items enthält Einträge, die für Multithreading und Statusanzeige verwendet werden. items ist _unset, wenn die Methode nicht von der graphischen Oberfläche von *OGRUT* aufgerufen wird.

iso_project.gewichte_indirekte_gwnb(the_tin)

- Diese Methode erstellt eine gewichtete Beziehung zwischen sites des Tins the_tin und Iso-Flächen des Iso-Projektes _self. Es werden jeweils die sites mit den Iso-Flächen assoziiert, in denen sie räumlich liegen.
- **Parameter** **Typ**
  the_tin ofg_tin
- **Typ des Rückgabewertes** _unset
- Der Parameter the_tin ist das digitale Geländemodell, dessen sites mit den Isoflächen in Beziehung gesetzt werden sollen.

iso_project.insert_element(anz, k1, k2, k3, det_node, det_fe)

- Diese Methode fügt ein neues Iso-Element in die Menge der in _self enthaltenen Iso-Elemente ein.
- **Parameter** **Typ**
  anz integer
  k1, k2, k3 integer
  det_node iso_node
  det_fe iso_element
- **Typ des Rückgabewertes** _unset
- Der Parameter anz gibt die Nummer des neuen Iso-Elementes an. k1, k2 und k3 geben die Nummern der Knoten an, die die Ecken des Iso-Elementes bilden sollen.

iso_project.read_data_menu(node_file_name, elements_file_name, delete_old_nodes?, delete_old_elements?, items)

- Diese Methode liest Knoten und Iso-Elemente aus einer externen Datei ein und fügt sie in das Iso-Projekt _self ein.
- **Parameter** **Typ**
  node_file_name string
  elements_file_name string
  delete_old_nodes? boolean
  delete_old_elements? boolean
  items hash_table
- **Typ des Rückgabewertes** _unset
- Der Parameter elements_file_name bezeichnet die Quelldatei für die Iso-Elemente, node_file_name ist der Name der Quelldatei für die Iso-Knoten. delete_old_nodes? legt fest, ob die vorhandenen Knoten gelöscht werden (_true) oder die neu eingelesenen Knoten hinzugefügt werden (_false). delete_old_elements? legt fest, ob die vorhandenen Elemente gelöscht werden (_true) oder die neu eingelesenen Elemente hinzugefügt werden (_false). Der Parameter items enthält Einträge, die für Multithreading und Statusanzeige verwendet werden. items ist _unset, wenn die Methode nicht von der graphischen Oberfläche von *OGRUT* aufgerufen wird.

iso_project.read_nodes(path_string, **_optional** items)

- Diese Methode liest Iso-Knoten aus einer externen Datei in das Iso-Projekt _self ein.
- **Parameter** **Typ**
  path_string string
  items hash_table
- **Typ des Rückgabewertes** _unset
- Der Parameter path_string bezeichnet die Quelldatei für die Iso-Knoten, items enthält Einträge, die für Multithreading und Statusanzeige verwendet werden. items ist _unset, wenn die Methode nicht von der graphischen Oberfläche von *OGRUT* aufgerufen wird.

iso_project.read_elements(path_string, **_optional** items)

- Diese Methode liest Iso-Elemente aus einer externen Datei in das Iso-Projekt _self ein.
- **Parameter** **Typ**
  path_string string
  items hash_table
- **Typ des Rückgabewertes** _unset
- Der Parameter path_string bezeichnet die Quelldatei für die Iso-Elemente, items enthält Einträge, die für Multithreading und Statusanzeige verwendet werden. items ist _unset, wenn die Methode nicht von der graphischen Oberfläche von *OGRUT* aufgerufen wird.

iso_project.read_grundwasserstand(
path_string, delete_old?, **_optional** items)

- Diese Methode liest Grundwasserstandsdaten aus einer externen Datei in das Iso-Projekt _self ein. Die Daten werden von *FEFLOW* geliefert und als Zeitreihen an die Iso-Knoten angehängt.
- **Parameter** **Typ**
  path_string string
  delete_old? boolean
  items hash_table
- **Typ des Rückgabewertes** _unset
- Der Parameter path_string bezeichnet die Quelldatei, aus der Grundwasserstandsdaten einelesen werden. delete_old? legt fest, ob die vorhandenen Daten gelöscht werden (_true) oder die neu eingelesenen Daten hinzugefügt werden (_false). Der Parameter items enthält Einträge, die für Multithreading und Statusanzeige verwendet werden. items ist _unset, wenn die Methode nicht von der graphischen Oberfläche von *OGRUT* aufgerufen wird.

iso_project.write_direkte_gwnb(file_name, **_optional** items)

- Diese Methode schreibt Daten der Grundwasserneubildung in eine externe Datei, wo sie von *FEFLOW* zur Simulation verwendet werden können.
- **Parameter** **Typ**
  file_name string
  items hash_table
- **Typ des Rückgabewertes** _unset

- Der Parameter file_name bezeichnet die Datei, in die die Daten geschrieben werden. Der Parameter items enthält Einträge, die für Multithreading und Statusanzeige verwendet werden. items ist _unset, wenn die Methode nicht von der graphischen Oberfläche von *OGRUT* aufgerufen wird.

B.4.3 Klasse Iso-Knoten

Die Klasse Iso-Knoten (iso_node) repräsentiert ein punktförmiges Knotenobjekt (s. Tab. B.20). Jeder Knoten enthält neben dem von ogrut_project vererbten Schlüsselfeld pro_name eine eindeutige Nummer num. Im Geometriefeld node vom Typ point sind die Koordinaten des Knotens gespeichert.
Das Verknüpfungsfeld iso_projects verweist zurück auf das Iso-Projekt, dem der Knoten zugeordnet ist. iso_element1, iso_element2 und iso_element3 zeigen auf drei finite Element-Dreiecke, für die dieser Knoten ein Eckpunkt ist.
Das Verknüpfungsfeld iso_node_ganglinies zeigt auf die Zeitreihen, die mit diesem Knoten verbunden sind.

Tab. B.20: Datenmodell der Klasse Iso-Knoten.

Objektname: iso_node (Iso-Knoten)		
Physikalische Felder	**Feldtyp**	
*S pro_name	ds_char_vec (30)	
*S num	ds_uint	
S geologie_ok	ds_float	
Logische Felder	**Feldtyp**	
x_coord	ds_char_vec (15)	
y_coord	ds_char_vec (15)	
Geometriefelder	**Feldtyp**	
node	point	
annotation	text	
Verknüpfungsfelder	**Beziehungstyp**	**Zu Objekt**
iso_projects	field_mapping	iso_project
iso_element_1	field_mapping	iso_element
iso_element_2	field_mapping	iso_element
iso_element_3	field_mapping	iso_element
iso_node_ganglinies	1:n	iso_node_ganglinie

iso_node.create_ganglinie_with(param_name, param_value)

- Diese Methode erzeugt ein iso_node_ganglinie -Objekt und einen iso_node_parameter. Der iso_node_parameter erhält den Namen param_name und den Wert param_value.
- **Parameter** **Typ**
  param_name string
  param_value string
- **Typ des Rückgabewertes** _unset
- param_name gibt den Namen des Ganglinienparameters an, param_value den Wert.

iso_node.create_parameter_with(ganglinie_id, param_name, param_value)

- Diese Methode erzeugt einen iso_node_parameter in der angegeben Ganglinie und mit angegebenem Namen und Wert.
- **Parameter** **Typ**
  ganglinie_id sys_id
  param_name string
  param_value string
- **Typ des Rückgabewertes** _unset
- Der Parameter ganglinie_id gibt die Ganglinie an, für die eine Parameter erzeugt werden soll. param_name ist der Name des Parameters, param_value der Wert des Parameters.

iso_node.get_ganglinie(param_name, param_value)

- Diese Methode gibt die iso_node_ganglinie mit dem angegebenen Name-Wert-Paar (param_name, param_value) zurück.
- **Parameter** **Typ**
  param_name string
  param_value string
- **Typ des Rückgabewertes** iso_node_ganglinie
- param_name gibt den Namen, param_value den Wert des Parameters an, für den die Ganglinie gesucht wird.

B.4.4 Klasse Iso-Element

Die Klasse Iso-Element (iso_element) repräsentiert ein finites Element, also ein ebenes Dreieck (s.Tab. B.21). Diese Dreiecke werden durch Triangulation einer Punktemenge (die Knoten) erzeugt. Als Schlüsselfelder dient neben dem vom Iso-Projekt vererbten Feld pro_name eine eindeutige Nummer für das Element (Feld num). Die Felder node_1, node_2 und node_3 enthalten die Nummern der drei Knoten, die die Ecken des Elements bilden. Der Mittelpunkt des Elements ist im Feld element_centre als Vektor mit zwei Gleitkommazahlen für die Koordinaten gespeichert. Das Geometriefeld element vom Typ area beschreibt die Lage des Dreiecks.
Das Verknüpfungsfeld iso_project zeigt auf das Projekt, in dem das Element enthalten ist. Die Verknüpfungsfelder iso_node_1, iso_node_2 und iso_node_3 zeigen auf die drei Eckpunkte des Elements.
iso_elem_ganglinies verweist auf eine Liste von Zeitreihen, die dem Element angehängt sind. Das Feld iso_direkte_gwnbs verweist auf die statischen Zuordnungen zwischen Katasterflächen und Iso-Elementen, iso_indirekte_gwnb verweist auf die statischen Zuordnungen zwischen sites des digitalen Geländemodells und Iso-Elementen.

Tab. B.21: Datenmodell der Klasse `Iso-Element`.

Objektname: iso_element (Iso-Element)			
Physikalische Felder		**Feldtyp**	
*S	num	ds_uint	
* I	pro_name	ds_char_vec (30)	
S	node_1	ds_uint	
S	node_2	ds_uint	
S	node_3	ds_uint	
	mb_katalog_id	sys_id	
	mat_bereichen_set_name	ds_char_vec (30)	
	element_centre	ds_float_vec (2)	
Geometriefelder		**Feldtyp**	
	element	area	
	annotation	text	
Verknüpfungsfelder		**Beziehungstyp**	**Zu Objekt**
	iso_project	1:n	iso_project
	iso_node_1	field_mapping	iso_node
	iso_node_2	field_mapping	iso_node
	iso_node_3	field_mapping	iso_node
	iso_elem_ganglinies	1:n	iso_elem_ganglinie
	iso_direkte_gwnbs	1:n	iso_direkte_gwnb
	iso_indirekte_gwnbs	1:n	iso_indirekte_gwnb

iso_element.berechne_direkte_gwnb(delete_old?, **_optional** items)

- Diese Methode berechnet die Zeitreihe der direkten Grundwasserneubildung aus den Zeitreihen der Katasterflächen, die mit dem Iso-Element `_self` assoziiert sind.
- **Parameter** **Typ**
  delete_old? boolean
  items hash_table
- **Typ des Rückgabewertes** _unset
- Der Parameter `delete_old?` legt fest, ob die Ergebnisse früherer Berechnungen gelöscht werden (`_true`) oder die neuen Ergebnisse hinzugefügt werden (`_false`). Der Parameter `items` enthält Einträge, die für Multithreading und Statusanzeige verwendet werden. `items` ist `_unset`, wenn die Methode nicht von der graphischen Oberfläche von *OGRUT* aufgerufen wird.

iso_element.berechne_indirekte_gwnb(delete_old?)

- Diese Methode berechnet die indirekte Grundwasserneubildung für das Iso-Element `_self` direkt aus der Überflutung der digitalen Geländepunkte, die mit dem Iso-Element assoziiert sind. Es wird nur die indirekte Grundwasserneubildung aus Überflutungsflächen berechnet, da die Zeitreihen der direkten Grundwasserneubildung aus Niederschlag nur für Katasterflächen verfügbar sind.
- **Parameter** **Typ**
  delete_old? boolean
- **Typ des Rückgabewertes** _unset
- Der Parameter `delete_old?` legt fest, ob die Ergebnisse früherer Berechnungen gelöscht werden (`_true`) oder die neuen Ergebnisse hinzugefügt werden (`_false`).

iso_element.create_ganglinie_with(param_name, param_value)

- Diese Methode erzeugt ein neues Ganglinienobjekt und ein Parameterobjekt für die Ganglinie. Das Parameterobjekt erhält die das Wertepaar (param_name, param_value).
- **Parameter** **Typ**
  param_name string
  param_value string
- **Typ des Rückgabewertes** _unset
- Der Parameter param_name ist der Name des Parameters für die Ganglinie. param_value ist der Wert des Parameters.

iso_element.create_parameter_with(ganglinie_id, param_name, param_value)

- Diese Methode erzeugt in der angegebenen Ganglinie einen Parmeter.
- **Parameter** **Typ**
  ganglinie_id sys_id
  param_name string
  param_value string
- **Typ des Rückgabewertes** _unset
- Der Parameter ganglinie_id gibt die Ganglinie an, für die ein Parameter erzeugt werden soll. param_name ist der Name des Parameters, param_value der Wert.

iso_element.get_ganglinie(param_name, param_value)

- Diese Methode ermittelt die Ganglinie am Iso-Element _self, die einen Parameter mit dem Name-Wert-Paar (param_name, param_value) enthält.
- **Parameter** **Typ**
  param_name string
  param_value string
- **Typ des Rückgabewertes** iso_element_ganglinie
- Der Parameter param_name gibt den Namen des gesuchten Parameters an, param_value den Wert. Die Ganglinie mit dem passenden Name-Wert-Paar wird zurückgegeben, sonst _unset.

iso_element.write_direkte_gwnb(output_stream, **_optional** items)

- Diese Methode schreibt die Zeitreihe für die direkte Grundwasserneubildung am Iso-Element _self in den Stream output_stream (z.B. eine Datei).
- **Parameter** **Typ**
  output_stream text_output_stream
  items hash_table
- **Typ des Rückgabewertes** _unset
- Der Parameter output_stream ist der Ausgabestream, auf den geschrieben wird. Der Parameter items enthält Einträge, die für Multithreading und Statusanzeige verwendet werden. items ist _unset, wenn die Methode nicht von der graphischen Oberfläche von *OGRUT* aufgerufen wird.

B.4.5 Klasse Iso Direkte GWNB

Die Klasse Iso Direkte GWNB (iso_direkte_gwnb) enthält eine Beziehung zu einer Katasterfläche, die einen Teil ihrer Fläche mit dem Iso-Element gemeinsam hat (s. Tab. B.22). Es wird also in der Liste der iso_direkte_gwnb -Objekte festgehalten, welche Katasterflächen Flächenanteile mit der betrachteten Iso-Fläche haben. Das Verknüpfungsfeld iso_element zeigt zurück auf dieses Element.
Die Schlüsselfelder iso_element_pro_name und iso_element_num sind die vom Iso-Projekt bzw. vom Iso-Element vererbten Schlüsselfelder. Das Feld gewicht zeigt an, welchen relativen Anteil die im Feld kat_flaeche_sys_id angegebene Katasterfläche an der Gesamtfläche des finiten Elements ausmacht. Die Summe der Gewichte aller iso_direkte_gwnb -Objekte am Iso-Element ist eins.

Tab. B.22: Datenmodell der Klasse Iso Direkte GWNB.

Objektname: iso_direkte_gwnb (Iso Direkte GWNB)			
Physikalische Felder		**Feldtyp**	
* I	iso_element_num	ds_uint	
* I	iso_element_pro_name	ds_char_vec (30)	
S	gewicht	ds_float	
S	kat_flaeche_sys_id	sys_id	
Verknüpfungsfelder		**Beziehungstyp**	**Zu Objekt**
	iso_element	1:n	iso_element

B.4.6 Mixin iso_ganglinie_mixin

iso_ganglinie_mixin.get_param(param_name)

- Diese Methode liefert den Parameter einer Ganglinie mit dem angegebenen Namen zurück Sowohl die Klasse iso_node_ganglinie als auch die Klasse iso_element_ganglinie sind von iso_ganglinie_mixin abgeleitet und können daher diese Methode verwenden.
- **Parameter** **Typ**
  param_name string
- **Typ des Rückgabewertes** iso_parameter | iso_node_parameter
- param_name ist der Name des gesuchten Ganglinienparameters.

B.4.7 Klasse Iso Element Ganglinie

Die Klasse Iso Element Ganglinie (iso_elem_ganglinie) enthält die einem finiten Element zugeordneten Zeitreihen von Daten, bei *OGRUT* sind dies Grundwasserneubildungsdaten (s. Tab. B.23). Jede Ganglinie enthält neben den vom Iso-Projekt und Iso-Element vererbten Schlüsselfeldern iso_element_pro_name bzw. iso_element_num eine eindeutige Nummer id. Das Verknüpfungsfeld iso_element verweist zurück auf das Iso-Element, dem die Ganglinie zugeordnet ist. Die eigentlichen Zeitreihenwerte werden in der Liste iso_werte gespeichert. Das Feld iso_parameters enthält Parameter, die die Ganglinie kennzeichnen.

Tab. B.23: Datenmodell der Klasse Iso Element Ganglinie.

Objektname: iso_elem_ganglinie (Iso Element Ganglinie)			
Physikalische Felder		**Feldtyp**	
*S	id	sys_id	
* I	iso_element_num	ds_uint	
* I	iso_element_pro_name	ds_char_vec (30)	
Verknüpfungsfelder		**Beziehungstyp**	**Zu Objekt**
	iso_element	1:n	iso_element
	iso_werte	1:n	iso_wert
	iso_parameters	1:n	iso_parameter

B.4.8 Klasse Iso Wert

Die Klasse Iso Wert (iso_wert) enthält Zeitreihenwerte von Ganglinien, die Iso-Elementen zugeordnet sind (s. Tab. B.24).
Neben den vom Iso-Projekt, Iso-Element und der Ganglinie vererbten Schlüsselfeldern iso_element_pro_name, iso_element_num und iso_element_ganglinie_id enthält jeder Ganglinienwert einen eindeutigen Zeitwert im Schlüsselfeld datum_zeit. Der eigentliche Zeitreihenwert wird im Feld wert gespeichert.
Das Feld iso_ganglinie verweist zurück auf die Zeitreihe, in der der Zeitreihenwert enthalten ist.

Tab. B.24: Datenmodell der Klasse Iso Wert.

Objektname: iso_wert (Iso Wert)			
Physikalische Felder		**Feldtyp**	
*SI	iso_elem_num	ds_uint	
*SI	iso_elem_pro_name	ds_char_vec (30)	
*SI	iso_elem_ganglinie_id	sys_id	
*S	datum_zeit	ds_time	
S	wert	ds_float	
Verknüpfungsfelder		**Beziehungstyp**	**Zu Objekt**
	iso_ganglinie	1:n	iso_elem_ganglinie

B.4.9 Klasse Iso Parameter

Die Klasse Iso Parameter (iso_parameter) enthält Parameter, die die Ganglinie kennzeichnen und von anderen Ganglinien unterscheidbar machen (s. Tab. B.25). Für jede Ganglinie kann es nur einen Parameter mit einem gegebenen Namen geben, daher wird das Feld name als Schlüsselfeld eingeführt. Hinzu kommen die vererbten Schlüsselfelder iso_elem_pro_name (vom Iso-Projekt), iso_element_num (vom Iso-Element) und iso_elem_ganglinie_id (von der Ganglinie). Das Feld wert enthält den Parameterwert. Das Paar (name, wert) bildet also den Parameter. Hier wird von *OGRUT* z.B. ("Art", "indirekte gwnb") oder ("Art", "direkte gwnb") gesetzt.

Tab. B.25: Datenmodell der Klasse Iso Parameter.

Objektname: iso_parameter (Iso Parameter)			
Physikalische Felder		**Feldtyp**	
*S	name	ds_char_vec (30)	
*SI	iso_element_num	ds_uint	
*SI	iso_elem_pro_name	ds_char_vec (30)	
*SI	iso_elem_ganglinie_id	sys_id	
S	wert	ds_char_vec (30)	
Verknüpfungsfelder		**Beziehungstyp**	**Zu Objekt**
	iso_ganglinie	1:n	iso_elem_ganglinie

B.4.10 Klasse Iso Indirekte GWNB

Die Klasse Iso Indirekte GWNB (iso_indirekte_gwnb) enthält Verbindungen zwischen einzelnen sites eines digitalen Geländemodells und einem finiten Element (s. Tab. B.26). Es wird in einer statischen Zuordnung abgebildet, welche sites in welcher Elementfläche liegen. Damit lässt sich - ohne Umweg über Katasterflächen - die indirekte Grundwasserneubildung aus den sites eines digitalen Geländemodells berechnen.

Tab. B.26: Datenmodell der Klasse Iso Indirekte GWNB.

Objektname: iso_indirekte_gwnb (Iso Indirekte GWNB)			
Physikalische Felder		**Feldtyp**	
* I	iso_element_num	ds_uint	
* I	iso_element_pro_name	ds_char_vec (30)	
*S	id	sys_id	
*S	triangulation_id	sys_id	
S	gewicht	ds_float	
Verknüpfungsfelder		**Beziehungstyp**	**Zu Objekt**
	iso_element	1:n	iso_element

iso_indirekte_gwnb.get_or_create(site, iso_element)

- Diese Methode ermittelt, ob es zwischen der angegeben site und dem angegebenen Iso-Element bereits eine Verbindung gibt. Ist dies nicht der Fall, so wird eine Verbindung erzeugt.
- **Parameter** — **Typ**
  site — sw_tin!primal_site
  iso_element — iso_element
- **Typ des Rückgabewertes** — iso_indirekte_gwnb
- Der Parameter site ist der digitale Geländepunkt und iso_element die Isofläche, zwischen denen eine Verbindung erstellt werden soll.

B.4.11 Klasse Iso Node Ganglinie

Die Klasse Iso Node Ganglinie (iso_node_ganglinie) enthält die einem Netzknoten zugeordneten Zeitreihen von Daten, bei *OGRUT* sind dies Grundwasserneubildungsdaten (s. Tab. B.27). Jede Ganglinie enthält neben den vom Iso-Projekt und Iso-Knoten vererbten Schlüsselfeldern iso_node_pro_name bzw. iso_node_num eine eindeutige Nummer id. Das

Verknüpfungsfeld iso_node verweist zurück auf den Iso-Knoten, dem die Ganglinie zugeordnet ist. Die eigentlichen Zeitreihenwerte werden in der Liste iso_werte gespeichert. Das Feld iso_parameters enthält Parameter, die die Ganglinie kennzeichnen.

Tab. B.27: Datenmodell der Klasse Iso Node Ganglinie.

Objektname: iso_node_ganglinie (Iso Node Ganglinie)			
Physikalische Felder		**Feldtyp**	
*S	id	sys_id	
* I	iso_node_num	ds_uint	
* I	iso_node_pro_name	ds_char_vec (30)	
Verknüpfungsfelder		**Beziehungstyp**	**Zu Objekt**
	iso_werte	1:n	iso_node_wert
	iso_parameters	1:n	iso_node_parameter
	iso_node	1:n	iso_node

B.4.12 Klasse Iso Node Wert

Die Klasse Iso Node Wert (iso_node_wert) enthält Zeitreihenwerte von Ganglinien, die Iso-Knoten zugeordnet sind (s. Tab. B.28).
Neben den vom Iso-Projekt, Iso-Knoten und der Ganglinie vererbten Schlüsselfeldern iso_node_pro_name, iso_node_num und iso_node_ganglinie_id enthält jeder Ganglinienwert einen eindeutigen Zeitwert im Schlüsselfeld datum_zeit. Der eigentliche Zeitreihenwert wird im Feld wert gespeichert.
Das Feld iso_ganglinie verweist zurück auf die Zeitreihe, in der der Zeitreihenwert enthalten ist.

Tab. B.28: Datenmodell der Klasse Iso Node Wert.

Objektname: iso_node_wert (Iso Node Wert)			
Physikalische Felder		**Feldtyp**	
*SI	iso_node_num	ds_uint	
*SI	iso_node_pro_name	ds_char_vec (30)	
*SI	iso_node_ganglinie_id	sys_id	
*S	datum_zeit	ds_time	
S	wert	ds_float	
Verknüpfungsfelder		**Beziehungstyp**	**Zu Objekt**
	iso_ganglinie	1:n	iso_node_ganglinie

B.4.13 Klasse Iso Node Parameter

Die Klasse Iso Node Parameter (iso_node_parameter) enthält Parameter, die die Ganglinie kennzeichnen und von anderen Ganglinien unterschiedbar machen (s. Tab. B.29). Für jede Ganglinie kann es nur einen Parameter mit einem gegebenen Namen geben, daher wird das Feld name als Schlüsselfeld eingeführt. Hinzu kommen die vererbten Schlüsselfelder iso_node_pro_name (vom Iso-Projekt), iso_node_num (vom Iso-Knoten) und iso_node_ganglinie_id (von der Ganglinie). Das Feld wert enthält den Parameterwert. Das Paar (name, wert) bildet also den Parameter. Hier wird von *OGRUT* z.B. ("Art", "gw_stand"), ("start", "1.1.1996 12:0:0") oder ("step", "0,0,1,0,0,0") gesetzt. Der letze Parameter step gibt die Schrittweite der Zeitreihe an, in diesem Beispiel 1 Tag.

Tab. B.29: Datenmodell der Klasse Iso Node Parameter.

Objektname: iso_node_parameter (Iso Node Parameter)			
Physikalische Felder		**Feldtyp**	
*S	name	ds_char_vec (30)	
*SI	iso_node_num	ds_uint	
*SI	iso_node_pro_name	ds_char_vec (30)	
*SI	iso_node_ganglinie_id	sys_id	
S	wert	ds_char_vec (30)	
Verknüpfungsfelder		**Beziehungstyp**	**Zu Objekt**
	iso_ganglinie	1:n	iso_node_ganglinie

B.4.14 Klasse Parameter-Gruppe

Die Klasse Parameter-Gruppe (iso_param_group) enthält Parameter, die die Ganglinie an den Isoflächen des Projektes betreffen (s. Tab. B.30).

Da jeder Netzknoten eine Ganglinie besitzt (sofern überhaupt welche eingelesen wurden), können die Parameter "start" und "step" direkt als Ganglinienparameter an den Ganglinien der Knoten gespeichert werden. Zwar gibt es für diese Ganglinien eine eigene Parameterklasse, in der z.B. der Parameter "Art" gespeichert wird. Im Unterschied zu Knoten muss aber nicht jedes finite Element eine Ganglinie enthalten, falls z.B. nur die indirekte Grundwasserneubildung berechnet wurde, die nur für wenige Isoflächen wirksam ist. Es wäre also zu aufwendig, um die zugeordneten Parameter zu erfahren zunächst ein Iso-Element zu suchen, das eine Ganglinie enthält. Daher werden die Angaben "start" und "step" für die Isoflächen-Ganglinien direkt im Iso-Projekt gespeichert.
Hierzu dienen die Parameter, die in der Parameter-Gruppe enthalten sind. Das Verknüpfungsfeld parameters verweist auf die einzelnen Parameter. Das Verknüpfungsfeld iso_project verweist zurück auf das Iso-Projekt. Das Schlüsselfeld id ist eine eindeutige Nummer, project_name ist das vom Iso-Projekt vererbte Schlüsselfeld. Das Feld info wird bei der von *OGRUT* erzeugten Parametergruppe mit dem Wert "gwnb" belegt.

Tab. B.30: Datenmodell der Klasse Parameter-Gruppe.

Objektname: iso_param_group (Parameter-Gruppe)			
Physikalische Felder		**Feldtyp**	
*S	id	sys_id	
* I	project_name	ds_char_vec (30)	
S	info	ds_char_vec (50)	
Verknüpfungsfelder		**Beziehungstyp**	**Zu Objekt**
	iso_project	1:n	iso_project
	parameters	1:n	iso_param

iso_param_group.create_param_with(name, value)

- Diese Methode erzeugt einen Parameter mit dem angegebenen Name-Wert-Paar in der Parametergruppe _self.
- **Parameter** — **Typ**
  name — string
  value — string

- **Typ des Rückgabewertes** iso_param
- name gibt den Namen, value den Wert des Parameters an, der erzeugt wird.

iso_param_group.get_param(name)

- Diese Methode sucht einen Parameter in der Parametergruppe _self und gibt ihn zurück.
- **Parameter** **Typ**
  name string
- **Typ des Rückgabewertes** iso_param
- name ist der Name des gesuchten Parameters.

B.4.15 Klasse Parameter

Die Klasse Parameter (iso_param) enthält Parameter, die die Ganglinien an den Isoflächen betreffen (s. Tab. B.31).
Für jede Ganglinie kann es nur einen Parameter mit einem gegebenen Namen geben, daher wird das Feld name als Schlüsselfeld eingeführt. Hinzu kommen die vererbten Schlüsselfelder project_name (vom Iso-Projekt) und param_group_id (von der Parametergruppe). Das Feld wert enthält den Parameterwert. Das Paar (name, wert) bildet also den Parameter. Hier wird von *OGRUT* z.B. ("start", "1.1.1996 12:0:0") oder ("step", "0,0,1,0,0,0") gesetzt. Der letze Parameter step gibt die Schrittweite der Zeitreihe an, in diesem Beispiel 1 Tag. Der Parameter "Art" wird dagegen wie bei den Knoten direkt an der Ganglinie der Isofläche gespeichert.

Tab. B.31: Datenmodell der Klasse Parameter.

Objektname: iso_param (Parameter)			
Physikalische Felder		**Feldtyp**	
*S	name	ds_char_vec (50)	
* I	project_name	ds_char_vec (30)	
* I	param_group_id	sys_id	
S	wert	ds_char_vec (50)	
Verknüpfungsfelder		**Beziehungstyp**	**Zu Objekt**
	param_group	1:n	iso_param_group

Nicht näher beschrieben werden hier die Klassen Iso-Schnitt, Iso-Störungen, Parameter für Iso-Höhenlinien, Iso-Höhe, Iso-Höhenlinien und Iso-Höhenlinien/Punkte. Sie werden von *LIWIS* bereitgestellt und in *OGRUT* nicht weiter verändert. Die Klassen im Zusammenhang mit Iso-Höhenlinien werden zur graphischen Darstellung von Grundwasserständen in Form von Isolinien verwendet.

B.5 Weitere Klassen und Methoden

Im Zusammenhang mit der Entwicklung von *OGRUT* ergab sich an verschiedenen Stellen die Notwendigkeit, unterstützende Klassen und Methoden zu definieren. Diese erledigen keine speziell hydrologischen Aufgaben sondern können als Grundlage für beliebige Module von *Smallworld* genutzt werden. Neben einigen globalen Prozeduren gehört hierzu die Klasse data_store, die eine spezielle Hash-Tabelle darstellt und die Klasse scanner, die einen Zeichenstrom in einen Symbolstrom umwandelt.

B.5.1 Klasse data_store

Diese Klasse bietet eine Art Hash-Tabelle an, in der Variablen und deren Werte gespeichert werden können.

data_store.new()

- Diese Methode erzeugt eine neue Instanz der Klasse data_store.
- **Typ des Rückgabewertes** data_store

data_store.init()

- Diese Methode initialisiert eine neue Instanz der Klasse data_store.
- **Typ des Rückgabewertes** _unset

data_store.get_value_for(name)

- Diese Methode gibt den Wert der Variablen mit dem Namen name zurück.
- **Parameter** **Typ**
  name *
- **Typ des Rückgabewertes** *

data_store.print_values()

- Diese Methode gibt eine Liste der enthaltenen Variablen und Werte am Bildschirm aus.
- **Typ des Rückgabewertes** _unset

data_store.set_value_for(name, value)

- Diese Methode setzt für die Variable name den Wert value.
- **Parameter** **Typ**
  name *
  value *
- **Typ des Rückgabewertes** _unset
- Der Parameter name ist der Name der Variablen, value der neue Wert.

data_store.read_values(path_string)

- Diese Methode liest eine Reihe von Variablen-Werte-Paaren aus einer externen Datei ein.
- **Parameter** **Typ**
  path_string string
- **Typ des Rückgabewertes** _unset
- Der Parameter path_string ist der Name der externen Datei.

B.5.2 Klasse scanner

scanner.add_symbol(sym, text)

- Diese Methode fügt der Symboltabelle des Scanners ein weiteres Symbol hinzu.
- **Parameter** **Typ**
  sym symbol
  text string

- **Typ des Rückgabewertes** _unset
- Der Parameter sym ist der Name des Symbols, das vom Scanner zurückgegeben werden soll, wenn in der Eingabe die Zeichenfolge text auftaucht.

scanner.add_symbols(sym)

- Diese Methode fügt dem Scanner eine Anzahl Symbole hinzu. Dies geschieht durch Aufruf von _self.add_symbol() für alle Elemente von sym.
- **Parameter** **Typ**
  sym simple_vector(symbol)
- **Typ des Rückgabewertes** _unset
- Der Parameter sym enthält eine Menge von Symbolen, die in die Symboltabelle aufgenommen werden.

scanner.available?()

- Diese Methode liefert als Antwort, ob der Scanner am Ende des Eingabestroms angelangt ist (_false), oder noch Symbole verfügbar sind (_true).
- **Typ des Rückgabewertes** boolean

scanner.close()

- Diese Methode beendet den Scanner und schliesst die Quelldatei.
- **Typ des Rückgabewertes** _unset

scanner.debug(path_string, interrupt_count)

- Diese Methode wird für Debugging-Zwecke verwendet, um die vom Scanner gelesenen Symbole am Bildschirm auszugeben.
- **Parameter** **Typ**
  path_string string
  interrupt_count integer
- **Typ des Rückgabewertes** _unset
- Der Parameter path_string bezeichnet die Eingabedatei, aus der gelesen wird. Wenn interrupt_count nicht _unset ist, dann wird das Einlesen nach der dort angegebenen Zahl von Symbolen abgebrochen.

scanner.error(s)

- Diese Methode dient intern zur Ausgabe von Fehlermeldungen beim Scannen.
- **Parameter** **Typ**
  s string
- **Typ des Rückgabewertes** _unset
- Der Parameter s ist die auszugebende Fehlermeldung.

scanner.identifier()

- Diese Methode liest einen Bezeichner aus dem Eingabestrom.
- **Typ des Rückgabewertes** _unset

scanner.init()

- Diese Methode initialisiert den Scanner.

- **Typ des Rückgabewertes** scanner

scanner.key_word?()

- Diese Methode prüft, ob die gelesene Zeichenfolge ein Schlüsselwort darstellt.
- **Typ des Rückgabewertes** boolean

scanner.match(sym)

- Diese Methode vergleicht das aktuelle Symbol auf Übereinstimmung mit dem geforderten Symbol. Ist die Prüfung erfolgreich, so wird das nächste Symbol gelesen, sonst ein Fehler ausgegeben.
- **Parameter** **Typ**
  sym symbol
- **Typ des Rückgabewertes** _unset
- Der Parameter sym ist das erwartete Symbol, das mit dem tatsächlich gelesenen Symbol verglichen wird.

scanner.match_all(sym_list)

- Diese Methode vergleicht die anstehende Symbolfolge mit der erwarteten Symbolfolge auf Übereinstimmung.
- **Parameter** **Typ**
  sym_list simple_vector(symbol)
- **Typ des Rückgabewertes** _unset
- Der Parameter sym_list ist die erwartete Symbolfolge, die mit den tatsächlich gelesenen Symbolen verglichen wird.

scanner.match_alternative(sym_list)

- Diese Methode vergleicht, ob das gelesene Symbol mit einer Auswahl von Symbolen übereinstimmt. Wird eine Übereinstimmung gefunden, so wird das gelesene Symbol zurückgegeben, andernfalls wird eine Fehlermeldung ausgegeben.
- **Parameter** **Typ**
  sym_list simple_vector(symbol)
- **Typ des Rückgabewertes** symbol
- Der Parameter sym_list enthält eine Liste von Symbolen, von denen eines mit dem gerade gelesenen Symbol übereinstimmen soll.

scanner.next_ch()

- Diese Methode liest das nächste Zeichen aus dem Eingabestrom.
- **Typ des Rückgabewertes** _unset

scanner.next_line()

- Diese Methode beendet das scannen der aktuellen Textzeile und geht umgehend zur nächsten Zeile.
- **Typ des Rückgabewertes** _unset

scanner.number()

- Diese Methode liest eine Ganzzahl oder Gleitkommazahl ein.

- **Typ des Rückgabewertes** _unset

scanner.open(file_name)

- Diese Methode öffnet eine Datei zum Lesen.
- **Parameter** **Typ**
  file_name string
- **Typ des Rückgabewertes** _unset
- Der Parameter file_name bezeichnet die Datei, die zum Lesen geöffnet wird.

scanner.scan()

- Diese Methode liest das nächste Symbol.
- **Typ des Rückgabewertes** _unset

scanner.scan_number()

- Diese Methode liest das nächste Symbol und erwartet, dass es sich dabei um eine Zahl handelt.
- **Typ des Rückgabewertes** _unset

scanner.scan_timestring()

- Diese Methode liest das nächste Symbol und erwartet, dass es sich dabei um die Folge "Y,M,D,h,m,s" handelt, wobei Y bis s Platzhalter für Zahlen von Jahr bis Sekunde sind. Das Ergebnis wird als String zurückgegeben.
- **Typ des Rückgabewertes** string

scanner.unget_ch(ch)

- Diese Methode legt ein gelesenes Symbol zurück, um es beim nächsten Aufruf von scanner.scan() wieder auszugeben.
- **Typ des Rückgabewertes** _unset

B.5.3 Methode coordinate.geom_category

coordinate.geom_category

- Diese Methode liefert den Wert :point zurück. Dies erlaubt es, Punkt-Geometrie-methoden auch auf Objekte der Klasse coordinate anzuwenden.
- **Typ des Rückgabewertes** :point

B.5.4 Globale Prozeduren

as_time_interval(date_time)

- Diese Prozedur wandelt einen String in ein time_interval-Objekt um.
- **Parameter** **Typ**
  date_time string
- **Typ des Rückgabewertes** date_time
- Der Parameter date_time muss folgenden Aufbau haben: "Y,M,D,h,m,s", also z.B. "0,0,1,0,0,0" für ein Intervall von einem Tag.

as_string(x)

- Diese Prozedur wandelt ein Objekt der Klasse `integer` in eine Zeichenkette um. Kaum zu glauben, aber so was gibts in *Magik* noch nicht!
- **Parameter** **Typ**
  `x` `integer`
- **Typ des Rückgabewertes** `string`
- Der Parameter `x` ist die Integer-Zahl (Ganzzahl), die in einen String (Zeichenkette) umgewandelt wird.

`date_from_format(string, format)`

- Diese Methode wandelt einen String in ein `date_time`-Objekt um.
- **Parameter** **Typ**
  `string` `string`
  `format` `string`
- **Typ des Rückgabewertes** `date_time`
- Der Parameter `string` ist die Zeichenkette, die in ein `date_time`-Objekt umgewandelt werden soll. `format` ist eine weitere Zeichenkette die das Format angibt, in dem der String vorliegt. Mögliche Aufrufe sind also z.B.

```
date_from_format(x, "D.M.Y h:m:s")
date_from_format("1.1.1999", "D.M.Y")
date_from_format("17/12/1986 - 13:00:00", "D/M/Y - h:m:s")
```

`date_to_format(dt, format)`

- Diese Methode wandelt das Objekt vom Typ `date_time` in eine Zeichenkette um.
- **Parameter** **Typ**
  `dt` `date_time`
  `format` `string`
- **Typ des Rückgabewertes** `string`
- Der Parameter `dt` ist das date_time -Objekt, das in eine Zeichenkette umgewandelt werden soll. `format` ist das Format, in dem das Objekt zurückgegeben werden soll. Mögliche Aufrufe sind hier z.B.

```
date_to_format(x, "D.M.Y")
date_to_format(x, "D.M.Y h:m:s")
date_to_format(x, "D/M/Y - h:m:s")
```

`get_full_path(file_name, path_string, path_variable, extension_variable)`

- Diese Methode ergänzt eine Zeichenkette zu einer kompletten Pfadangabe.
- **Parameter** **Typ**
  `file_name` `string`
  `path_string` `string`
  `path_variable` `string`
  `extension_variable` `string`
- **Typ des Rückgabewertes** `string`
- Der Parameter `path_string` ist die Zeichenkette, die unter Umständen ergänzt werden soll. Mögliche Ergänzungen sind der Dateiname (aus `file_name`), der Pfad (aus `path_variable`) und die Dateiextension (aus `extension_variable`).

B.6Definitionen der graphischen Benutzeroberfläche von OGRUT

B.6.1Klasse ogrut_project

ogrut_project.init()

- Diese Methode initialisiert eine neue Instanz der Klasse ogrut_project
- **Typ des Rückgabewertes** ogrut_project

ogrut_project.worst_case(a, b)

- Diese Methode vergleicht Vorbedingungen zur Durchführung einer Funktion daraufhin, welche am wenigsten weit fortgeschritten ist. Eine Vorbedingung für eine Funktion kann entweder noch nicht berechenbar (:not_calculable), berechenbar (:calculable) oder schon berechnet sein (:calculated). Von zwei Vorbedingungen zur Durchführung einer Funktion entscheidet die mit den grösseren Einschränkungen (worst case). Daher wird der schlechtere Fall als Ergebnis zurückgegeben.
- **Parameter** **Typ**
  a, b symbol
- **Typ des Rückgabewertes** symbol
- Die Parameter a und b sind die beiden Zustände vom Typ symbol(not_calculable, :calculable, :calculated), die verglichen werden.

ogrut_project.preconditions(topic)

- Diese Methode prüft alle Vorbedingungen einer Funktion daraufhin, ob sie bereits erfüllt sind. Ist dies der Fall, so ist die Funktion berechenbar, ansonsten fehlen noch Voraussetzungen.
- **Parameter** **Typ**
  topic symbol
- **Typ des Rückgabewertes** boolean
- Der Parameter topic ist die gewünschte Funktion, deren Status geprüft wird.

ogrut_project.river_preconditions(topic, river)

- Diese Methode prüft alle Vorbedingungen einer Funktion daraufhin, ob sie bereits erfüllt sind. Ist dies der Fall, so ist die Funktion berechenbar, ansonsten fehlen noch Voraussetzungen. Die Funktion bezieht sich auf ein Gerinne, also z.B. das Einlesen von Querprofildaten oder die Berechnung der Wasserstandszeitreihe.
- **Parameter** **Typ**
  topic symbol
  river ofg_dar
- **Typ des Rückgabewertes** boolean
- Der Parameter topic ist die gewünschte Funktion, deren Status geprüft wird. river ist das Gerinne, für das die Funktion geprüft wird.

ogrut_project.get_color_for(topic, river)

- Diese Methode liefert eine Farbe zurück, in der der Fluss in einer Auswahlliste dargestellt wird. Die Farbe zeigt an, ob die eine bestimmte Funktion auf den Fluss bereits angewendet wurde bzw. anwendbar ist.

- **Parameter** **Typ**
  `topic` `symbol`
  `river` `ofg_dar`
- **Typ des Rückgabewertes** `:red | :black | :green`
- Der Parameter `topic` gibt die Funktion an, für die eine Farbe gesucht ist. `river` ist der Fluss, auf den sich die Funktion beziehen soll.

`ogrut_project``.changed()`

- Diese Methode gibt dem OGRUT-Projekt _self zur Kenntnis, dass sich etwas geändert hat, z.B. dass eine Funktion berechnet wurde. Daraufhin werden verschiedene Anzeigen aktualisiert.
- **Typ des Rückgabewertes** `_unset`

B.6.2 Klasse ogrut_project_editor

`ogrut_project_editor``.define_application_buttons()`

- Diese Methode definiert einen zusätzlichen Button, damit vom OGRUT-Projekt-Editor aus das OGRUT-Menu-Dialogfenster erreichbar ist.
- **Typ des Rückgabewertes** `_unset`

`ogrut_project_editor``.aufruf()`

- Diese Methode öffnet das OGRUT-Menu-Dialogfenster
- **Typ des Rückgabewertes** `_unset`

B.6.3 Klasse ogrut_project_menu

`ogrut_project_menu``.new(rec)`

- Diese Methode erzeugt eine neue Instanz der Klasse `ogrut_project_menu`. Dadurch wird ein Editorfenster geöffnet.
- **Parameter** **Typ**
  `rec` `ogrut_project`
- **Typ des Rückgabewertes** `ogrut_project_menu`
- Der Parameter `rec` ist das aktuelle `ogrut_project` aus dem `ogrut_project_editor`.

`_private ogrut_project_menu``.init(rec)`

- Diese Methode initialisiert die statischen Elemente des OGRUT-Projektmenu-Editors.
- **Parameter** **Typ**
  `rec` `ogrut_project`
- **Typ des Rückgabewertes** `ogrut_project_menu`
- Der Parameter `rec` ist das aktuelle `ogrut_project` aus dem `ogrut_project_editor`.

`ogrut_project_menu``.edit()`

- Diese Methode öffnet ein Fenster der Klasse `ogrut_edit_menu` vom OGRUT-Projektmenu-Editor aus.

- **Typ des Rückgabewertes** _unset

ogrut_project_menu.change_fe_project()

- Diese Methode gibt dem OGRUT-Projektmenu-Editor zur Kenntnis, dass sich das aktuelle Finite-Elemente-Projekt geändert hat. Dies führt dazu, dass die Anzeige aktualisiert wird.
- **Typ des Rückgabewertes** _unset

ogrut_project_menu.remove_river()

- Diese Methode entfernt diejenigen Flüsse aus der Liste der aktuell im Projekt enthaltenen Flüsse, die gerade markiert sind.
- **Typ des Rückgabewertes** _unset

ogrut_project_menu.add_river()

- Diese Methode fügt der Liste der aktuell im Projekt enthaltenen Flüsse diejenigen Flüsse hinzu, die in der Liste der überhaupt vorhandenen Flüsse gerade markiert sind.
- **Typ des Rückgabewertes** _unset

ogrut_project_menu.update_display()

- Diese Methode aktualisiert die Anzeige des OGRUT-Projektmenu-Dialogfensters.
- **Typ des Rückgabewertes** _unset

ogrut_project_menu.show_window(rec)

- Diese Methode zeigt das OGRUT-Projektmenu-Dialogfenster an. Wenn noch keine Instanz des Fensters existitert, so wird eine neue Instanz erzeugt.
- **Parameter** **Typ**
  rec ogrut_project
- **Typ des Rückgabewertes** _unset
- Der Parameter rec ist das aktuelle OGRUT-Projekt.

ogrut_project_menu.activate_in(a_frame)

- Diese Methode aktiviert das OGRUT-Projektmenu-Dialogfenster. Die Methode wird nur einmal beim Erstellen des Fensters aufgerufen und erzeugt die statischen Dialogelemente.
- **Parameter** **Typ**
  a_frame frame
- **Typ des Rückgabewertes** _unset
- Der Parameter a_frame ist der frame, in dem die Fensterelemente angezeigt werden.

B.6.4 Klasse ogrut_list_item

ogrut_list_item.draw_on(window, style, x, y, str, next, my_model)

- Diese Methode zeichnet die einzelnen Einträge der Listen in OGRUT-Dialogfenstern. Gerinne werden in einigen Fenstern farbig aufgelistet, je nach Fortschritt der Berechnung der verschiedenen Funktionen für die jeweiligen Gerinne.

- **Parameter** **Typ**
  window window
  style text_style
  x,y integer
  str string
- **Typ des Rückgabewertes** _unset
- Der Parameter window ist der Fensterbereich, in dem das Element der Liste gezeichnet wird, x und y sind die Koordinaten. str ist die zu zeichnende Zeichenkette. style gibt den Stil an, in dem die Zeichenkette gezeichnet wird (Zeichensatz, Farbe, usw.).

ogrut_list_item.new(text, index, obj, color)

- Diese Methode erzeugt eine neue Instanz der Klasse ogrut_list_item.
- **Parameter** **Typ**
  text string
  index index
  obj *
  color symbol
- **Typ des Rückgabewertes** ogrut_list_item
- Der Parameter text gibt die Zeichenkette an, die in der Liste angezeigt werden soll. index ist die Nummer, die das Objekt kennzeichnet. obj ist ein Verweis auf ein Objekt, das durch den Listeneintrag repräsentiert wird (z.B. ein Oberflächengewässer-Objekt). color ist die Farbe, in der der Eintrag in der Liste gezeichnet werden soll.

ogrut_list_item.init(text, index, obj, color)

- Diese Methode initialisiert ein ogrut_list_item-Objekt.
- **Parameter** **Typ**
  text string
  index index
  obj *
  color symbol
- **Typ des Rückgabewertes** ogrut_list_item
- Der Parameter text gibt die Zeichenkette an, die in der Liste angezeigt werden soll. index ist die Nummer, die das Objekt kennzeichnet. obj ist ein Verweis auf ein Objekt, das durch den Listeneintrag repräsentiert wird (z.B. ein Oberflächengewässer-Objekt). color ist die Farbe, in der der Eintrag in der Liste gezeichnet werden soll.

B.6.5 Klasse ogrut_edit_menu

ogrut_edit_menu.new(rec)

- Diese Methode erzeugt eine neue Instanz der Klasse ogrut_edit_menu. Dadurch wird ein neuer Editor erzeugt, in dem ein OGRUT-Projekt bearbeitet werden kann.
- **Parameter** **Typ**
  rec ogrut_project
- **Typ des Rückgabewertes** ogrut_edit_menu

- Der Parameter `rec` ist das aktuelle OGRUT-Projekt, das in diesem Editor bearbeitet wird.

_private ogrut_edit_menu.`init(rec)`

- Diese Methode initialisiert einen OGRUT-Editmenu-Editor.
- **Parameter** **Typ**
  `rec` `ogrut_project`
- **Typ des Rückgabewertes** `ogrut_edit_menu`
- Der Parameter `rec` ist das aktuelle OGRUT-Projekt, das in diesem Editor bearbeitet wird.

ogrut_edit_menu.`update_display()`

- Diese Methode aktualisiert die Anzeige des OGRUT-Editmenu-Editors.
- **Typ des Rückgabewertes** `_unset`

ogrut_edit_menu.`show_window(rec)`

- Diese Methode zeigt das OGRUT-Editmenu-Fenster an.
- **Parameter** **Typ**
  `rec` `ogrut_project`
- **Typ des Rückgabewertes** `_unset`
- Der Parameter `rec` ist das aktuelle OGRUT-Projekt, das in diesem Editor bearbeitet wird.

ogrut_edit_menu.`read_fe()`

- Diese Methode ruft ein Untereditorfenster auf. In diesem Dialogfenster kann ein finites Elemente-Netz eingelesen werden. Das Editorfenster wird beim Betätigen des entsprechenden Buttons vom OGRUT-Editmenu-Dialogfenster aus geöffnet.
- **Typ des Rückgabewertes** `_unset`

ogrut_edit_menu.`read_tin()`

- Diese Methode ruft ein Untereditorfenster auf. In diesem Dialogfenster kann ein digitales Geländemodell für ein Gerinne eingelesen werden. Das Editorfenster wird beim Betätigen des entsprechenden Buttons vom OGRUT-Editmenu-Dialogfenster aus geöffnet.
- **Typ des Rückgabewertes** `_unset`

ogrut_edit_menu.`read_geometry()`

- Diese Methode ruft ein Untereditorfenster auf. In diesem Dialogfenster kann eine Geometriedatei mit Querprofildaten für ein Gerinne eingelesen werden. Das Editorfenster wird beim Betätigen des entsprechenden Buttons vom OGRUT-Editmenu-Dialogfenster aus geöffnet.
- **Typ des Rückgabewertes** `_unset`

ogrut_edit_menu.`read_runoff()`

- Diese Methode ruft ein Untereditorfenster auf. In diesem Dialogfenster kann eine Abflussdatei mit Abflusszeitreihen für die Pegel eines Gerinnes eingelesen werden.

Das Editorfenster wird beim Betätigen des entsprechenden Buttons vom OGRUT-Editmenu-Dialogfenster aus geöffnet.

- **Typ des Rückgabewertes** `_unset`

ogrut_edit_menu.read_watertable()

- Diese Methode ruft ein Untereditorfenster auf. In diesem Dialogfenster kann eine Datei eingelesen werden, die Wasserstandszeitreihen für die Querprofile eines Gerinnes enthält. Das Editorfenster wird beim Betätigen des entsprechenden Buttons vom OGRUT-Editmenu-Dialogfenster aus geöffnet.
- **Typ des Rückgabewertes** `_unset`

ogrut_edit_menu.read_gwtable()

- Diese Methode ruft ein Untereditorfenster auf. In diesem Dialogfenster kann eine Datei mit Grundwasserständen eingelesen werden, die aus der Simulation von *FEFLOW* zurückgegeben werden. Die Grundwasserstandszeitreihen werden an die Knoten des Finite-Elemente-Projektes im aktuellen OGRUT-Projekt angehängt. Das Editorfenster wird beim Betätigen des entsprechenden Buttons vom OGRUT-Editmenu-Dialogfenster aus geöffnet.
- **Typ des Rückgabewertes** `_unset`

ogrut_edit_menu.attach_sites_areas()

- Diese Methode ruft ein Untereditorfenster auf. In diesem Dialogfenster können statische Verbindungen zwischen den einzelnen sites eines digitalen Geländemodells und den Katasterflächen in der Datenbank erstellt werden. Das Editorfenster wird beim Betätigen des entsprechenden Buttons vom OGRUT-Editmenu-Dialogfenster aus geöffnet.
- **Typ des Rückgabewertes** `_unset`

ogrut_edit_menu.attach_areas_elements()

- Diese Methode ruft ein Untereditorfenster auf. In diesem Dialogfenster können statische Verbindungen zwischen den einzelnen Katasterflächen in der Datenbank und den Elementen des Finite-Elemente-Projektes im aktuellen OGRUT-Projekt erstellt werden. Das Editorfenster wird beim Betätigen des entsprechenden Buttons vom OGRUT-Editmenu-Dialogfenster aus geöffnet.
- **Typ des Rückgabewertes** `_unset`

ogrut_edit_menu.calculate_run_wsp()

- Diese Methode ruft ein Untereditorfenster auf. In diesem Dialogfenster kann das externe Programm *UWSP* gestartet werden, um für ein Gerinne die Wasserstandszeitreihen zu berechnen. Das Editorfenster wird beim Betätigen des entsprechenden Buttons vom OGRUT-Editmenu-Dialogfenster aus geöffnet.
- **Typ des Rückgabewertes** `_unset`

ogrut_edit_menu.calculate_triangulate_tin()

- Diese Methode ruft ein Untereditorfenster auf. In diesem Dialogfenster kann ein Tin trianguliert werden. Die Triangulation lässt sich im Gegensatz zu allen anderen Funktionen nicht abbrechen, da die interne Ausführung der Triangulation durch *Smallworld* in einem Schritt (atomar) erfolgt. Das Editorfenster wird beim

Betätigen des entsprechenden Buttons vom OGRUT-Editmenu-Dialogfenster aus geöffnet.

- **Typ des Rückgabewertes** _unset

ogrut_edit_menu.calculate_flooding_tin()

- Diese Methode ruft ein Untereditorfenster auf. In diesem Dialogfenster kann die Überflutung und indirekte Grundwasserneubildung des digitalen Geländemodells durch ein Gerinne berechnet werden. Das Editorfenster wird beim Betätigen des entsprechenden Buttons vom OGRUT-Editmenu-Dialogfenster aus geöffnet.
- **Typ des Rückgabewertes** _unset

ogrut_edit_menu.calculate_flooding_areas()

- Diese Methode ruft ein Untereditorfenster auf. In diesem Dialogfenster kann die Überflutung und indirekte Grundwasserneubildung der Katasterflächen in der Datenbank durch das digitale Geländemodell eines Gerinnes berechnet werden. Das Editorfenster wird beim Betätigen des entsprechenden Buttons vom OGRUT-Editmenu-Dialogfenster aus geöffnet.
- **Typ des Rückgabewertes** _unset

ogrut_edit_menu.calculate_flooding_elements()

- Diese Methode ruft ein Untereditorfenster auf. In diesem Dialogfenster kann die Überflutung und indirekte Grundwasserneubildung an den Elementen eines Finiten-Elemente-Netzes aus Katasterflächen berechnet werden. Das Editorfenster wird beim Betätigen des entsprechenden Buttons vom OGRUT-Editmenu-Dialogfenster aus geöffnet.
- **Typ des Rückgabewertes** _unset

ogrut_edit_menu.write_geometry()

- Diese Methode ruft ein Untereditorfenster auf. In diesem Dialogfenster können die Geometriedaten der Querprofile eines Gerinnes in eine externe Datei ausgegeben werden. Das Editorfenster wird beim Betätigen des entsprechenden Buttons vom OGRUT-Editmenu-Dialogfenster aus geöffnet.
- **Typ des Rückgabewertes** _unset

ogrut_edit_menu.write_watertable()

- Diese Methode ruft ein Untereditorfenster auf. In diesem Dialogfenster können die Wasserstandszeitreihen an den Querprofilen eines Gerinnes in eine externe Datei ausgegeben werden. Das Editorfenster wird beim Betätigen des entsprechenden Buttons vom OGRUT-Editmenu-Dialogfenster aus geöffnet.
- **Typ des Rückgabewertes** _unset

ogrut_edit_menu.write_runoff()

- Diese Methode ruft ein Untereditorfenster auf. In diesem Dialogfenster können die Abflusszeitreihen an den Pegeln eines Gerinnes in eine externe Datei ausgegeben werden. Das Editorfenster wird beim Betätigen des entsprechenden Buttons vom OGRUT-Editmenu-Dialogfenster aus geöffnet.
- **Typ des Rückgabewertes** _unset

ogrut_edit_menu.write_d_gwnb()

- Diese Methode ruft ein Untereditorfenster auf. In diesem Dialogfenster können die Grundwasserneubildungs-Zeitreihen der Elemente eines finiten Elemente-Netzes in eine externe Datei ausgegeben werden. Das Editorfenster wird beim Betätigen des entsprechenden Buttons vom OGRUT-Editmenu-Dialogfenster aus geöffnet.
- **Typ des Rückgabewertes** _unset

ogrut_edit_menu.activate_in(a_frame)

- Diese Methode aktiviert das OGRUT-Editmenu-Dialogfenster und erzeugt die statischen Dialogelemente des Fensters.
- **Parameter** **Typ**
  a_frame frame
- **Typ des Rückgabewertes** _unset
- Der Parameter a_frame ist der Rahmen, in dem die Dialogelemente des Fensters angezeigt werden.

B.6.6 Klassen für die einzelnen Untereditoren

Die nachfolgenden Methoden sind für jeden Untereditor des Hautpeditorfensters (ogrut_edit_menu) vorhanden und erfüllen dort vergleichbare Funktionen. Daher werden sie nicht für jeden Untereditor einzeln beschrieben, sondern nur einmal repräsentativ. Der Platzhalter XXX_menu kann daher im folgenden ersetzt werden durch:

- flooding_areas_menu
- flooding_elements_menu
- flooding_tin_menu
- read_fe_menu
- read_geometry_menu
- read_gwtable_menu
- read_runoff_menu
- read_tin_menu
- read_watertable_menu
- attach_areas_elements_menu
- attach_sites_areas_menu
- calculate_runoff_menu
- triangulate_tin_menu
- write_d_gwnb_menu
- write_geometry_menu
- write_runoff_menu
- write_watertable_menu

Jeder Editor entspricht dabei einer Funktion, die auf einzelne Elemente des aktuellen OGRUT-Projektes anwendbar ist

XXX_menu.new(rec)

- Diese Methode erzeugt eine neue Instanz der Klasse XXX_menu, die ein Untereditor zur Bearbeiten eines OGRUT-Projektes bietet.
- **Parameter** **Typ**
  rec ogrut_project
- **Typ des Rückgabewertes** XXX_menu
- Der Parameter rec ist das aktuelle OGRUT-Projekt, das in diesem Untereditor bearbeitet werden soll.

_private **XXX_menu**.init(rec)

- Diese Methode initialisiert eine neue Instanz der Klasse XXX_menu.
- **Parameter** **Typ**

rec	ogrut_project

- **Typ des Rückgabewertes** XXX_menu
- Der Parameter rec ist das aktuelle OGRUT-Projekt, das in diesem Untereditor bearbeitet werden soll.

XXX _menu.update_display()

- Diese Methode aktualisiert die Anzeige des Untereditors _self. Sie wird aufgerufen, wenn sich durch Ausführen einer Funktion der Zustand des aktuellen OGRUT-Projektes geändert hat.
- **Typ des Rückgabewertes** _unset

XXX _menu.show_window(rec, parent_editor)

- Diese Methode zeigt ein XXX_menu-Dialogfenster. Falls noch keines existiert, wird ein neues Fenster erstellt.
- **Parameter** **Typ**

rec	ogrut_project
parent_editor	ogrut_edit_menu

- **Typ des Rückgabewertes** _unset
- Der Parameter rec ist das aktuelle OGRUT-Projekt, das in diesem Untereditor bearbeitet werden soll. ogrut_edit_menu verweist zurück auf das übergeordnete OGRUT-Editorfenster. Dies ermöglicht z.B. die Weitergabe der Information, dass die Anzeige der Fenster nicht mehr aktuell ist und erneuert werden muss.

XXX _menu.start_stop()

- Diese Methode startet die Ausführung einer Funktion, die in diesem Untereditor angeboten wird. Ist die Funktion bereits gestartet, so wird die Ausführung stattdessen beendet. Dies macht es erforderlich, dass ein separater Hintergrund-Thread gestartet wird. Er überprüft regelmässig, ob er beendet werden soll. Dies wird über den Zustand eines booleschen Feldes angezeigt. Wird der Thread durch einen Fehler beendet, so wird die Fehlerbehandlung durch einen _protect-Block abgefangen und die Anzeige im Dialogfenster aktualisiert bevor der Fehler weitergeleitet wird.
- **Typ des Rückgabewertes** _unset

XXX _menu.activate_in(a_frame)

- Diese Methode aktiviert ein Fenster der Klasse XXX_menu und initialisert die statischen Dialogelemente im Fenster.
- **Parameter** **Typ**

a_frame	frame

- **Typ des Rückgabewertes** _unset
- Der Parameter a_frame ist der frame, in dem die Dialogelemente angezeigt werden.

Anhang C Beschreibung der Funktions-Editoren

Jede Funktion der graphischen Benutzeroberfläche ist durch einen Button des OGRUT-Edit-Menus erreichbar. Wenn die Funktion für einzelne Elemente des OGRUT-Projektes noch nicht durchführbar ist bzw. noch nicht durchgeführt wurde, so wird dies im OGRUT-Edit-Menu durch ein Symbol hinter der Funktion angezeigt. Dies heisst jedoch nicht, dass die Funktion für *kein* Element anwendbar ist. So kann z.B. für einen Fluss bereits ein digitales Geländemodell eingelesen worden sein, für einen anderen noch nicht. Daraus folgt, das z.B. eine Triangulation für den ersten Fluss möglich (bzw. vielleicht bereits durchgeführt) ist, für den zweiten noch nicht. Diese Unterschiede werden in den entsprechenden Untereditoren ebenfalls angezeigt. Es werden Listen von Flüssen dargestellt, die farbig gekennzeichnet sind. Dabei haben die einzelnen Farben folgende Bedeutung:

- rot Die Funktion ist für diesen Fluss noch nicht ausführbar.
- schwarz Die Funktion ist ausführbar.
- grün Die Funktion ist ausführbar und wurde bereits ausgeführt.

Nun zu den einzelnen Funktions-Editoren.

C.1.1Der Editor *"Finites Elemente-Netz einlesen"*

Der Editor *"Finites Elemente-Netz einlesen"* wird über den Button geöffnet. Es wird das Fenster in Abb. C.1 geöffnet.

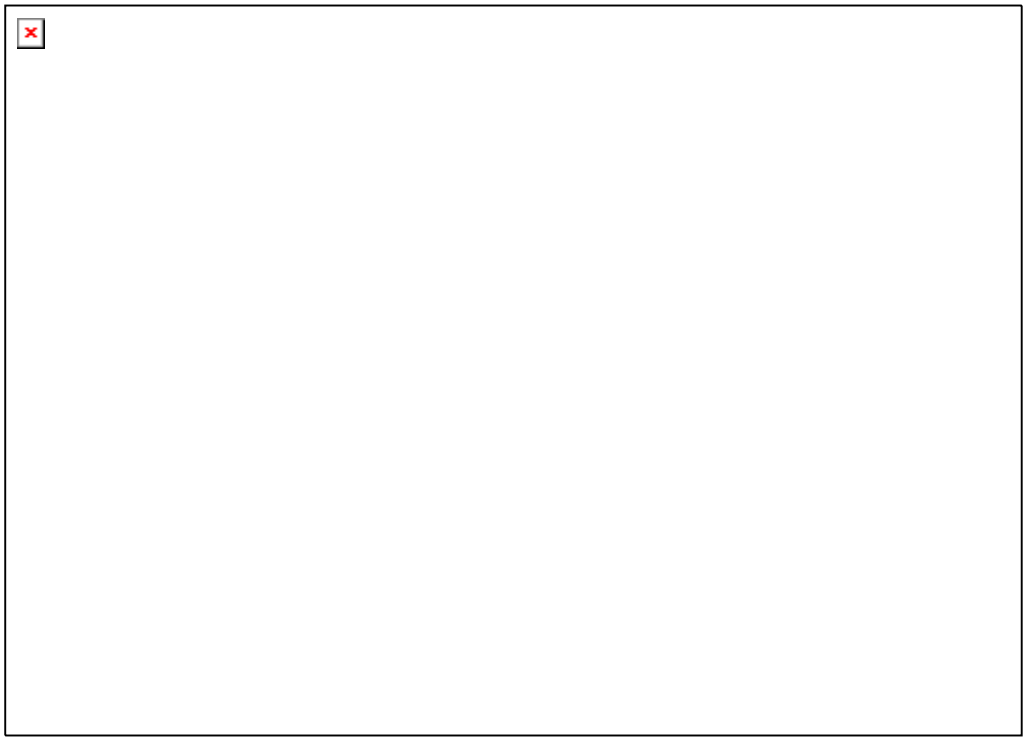


Abb. C.1: Der Editor ogrut_read_fe_menu dient zum Einlesen eines Finiten Elemente-Netzes.

In der Statuszeile wird das aktuelle OGRUT-Projekt angezeigt (*"Zartener Becken"*). In der zweiten Zeile wird das Finite Elemente-Projekt angezeigt, das dem OGRUT-Projekt zugeordnet ist. Die Zahl der Knoten und Elemente im Finite-Elemente-Projekt wird in

Klammern angegeben. Mit diesem Editor lässt sich ein finites Elemente-Netz aus einer externen Datei einlesen. Dabei kann angegeben werden, ob Knoten (Kontrollkästchen *"Knoten Einlesen"*), Elemente (Kontrollkästchen *"Elemente einlesen"*) oder beides eingelesen werden soll. Weiterhin kann festgelegt werden, ob alte Knoten bzw. Elemente vor Beginn des Einlesens gelöscht werden sollen, oder die neuen Knoten oder Elemente zu den bestehenden hinzugefügt werden sollen. Über den Buttons wird jeweils ein Fenster geöffnet, das einen Dateiauswahl anbietet (s. Abb. C.2).

Abb. C.2: Dateiauswahlbox, die die Auswahl einer Datei zum lesen oder speichern von Daten erlaubt.

Durch Betätigen des Buttons wird die Funktion gestartet. Sie kann jederzeit wieder abgebrochen werden, indem der Start-Button erneut betätigt wird. Während die Funktion läuft, wird der Text des Buttons von *"Start"* zu *"Stop"* geändert.

C.1.2Der Editor *"DGM bei einem Fluss einlesen"*

Der Editor *"DGM bei einem Fluss einlesen"* wird über den Button geöffnet. Es wird das Fenster in Abb. C.3 geöffnet.

Abb. C.3: Der Editor ogrut_read_tin_menu dient zum Einlesen eines digitalen Geländemodells für einen Fluss.

In der Statuszeile wird das aktuelle OGRUT-Projekt angezeigt (*"Zartener Becken"*). In der Liste *"Flüsse"* werden die im OGRUT-Projekt enthaltenen Flüsse dargestellt. Dies geschieht, wie oben bereits erwähnt, mit unterschiedlichen Farben um anzuzeigen, inwieweit die Funktion für die einzelnen Gerinne angewendet wurde bzw. anwendbar ist. Aus der Liste kann nun ein Fluss markiert werden und durch Betätigen des Buttons ausgewählt werden. Er wird dann als *"Gewählter Fluss"* angezeigt. Über den Button wird ein Fenster geöffnet, das eine Dateiauswahl anbietet (s. Abb. C.2).
Durch Anwahl der Option *"alte sites löschen"* werden vor dem Einlesen vorhandene digitale Geländepunkte gelöscht. Andernfalls werden die neuen Geländepunkte zu den bestehenden hinzugefügt. Durch Betätigen des Buttons wird die Funktion gestartet.

C.1.3Der Editor *"Fluss-Geometriedaten einlesen"*

Der Editor *"Fluss-Geometriedaten einlesen"* wird über den Button geöffnet. Es wird das Fenster in Abb. C.4 geöffnet.

Abb. C.4: Der Editor ogrut_read_geometry_menu dient zum Einlesen von Querprofildaten für ein Oberflächengewässer.

In der Statuszeile wird das aktuelle OGRUT-Projekt angezeigt (*"Zartener Becken"*). In der Liste *"Flüsse"* werden die im OGRUT-Projekt enthaltenen Flüsse angezeigt. Durch Markieren eines Flusses in der Liste und Betätigen des Buttons wird der Fluss ausgewählt. Über den Button wird ein Fenster geöffnet, das eine Dateiauswahl anbietet (s. Abb. C.2). Die Option *"Alte Geometriedaten löschen"* legt fest, ob vorhandene Querprofildaten gelöscht werden oder die neuen Querprofile hinzugefügt werden. Durch Betätigen des Buttons wird das Einlesen der Querprofildaten gestartet.

C.1.4Der Editor *"Pegelabflüsse einlesen"*

Der Editor *"Pegelabflüsse einlesen"* wird über den Button geöffnet. Es wird das Fenster in Abb. C.5 geöffnet.

Abb. C.5: Der Editor ogrut_read_runoff_menu dient zum Einlesen von Abflusszeitreihen an Pegeln eines Gerinnes.

In der Statuszeile wird das aktuelle OGRUT-Projekt angezeigt (*"Zartener Becken"*). Mit diesem Editor lassen sich die Abflussdaten an den Pegeln eines Gerinnes einlesen. Durch Markieren eines Flusses in der Liste und Betätigen des Buttons wird der Fluss ausgewählt. Über den Button wird ein Fenster geöffnet, das eine Dateiauswahl anbietet (s. Abb. C.2). Die Kontrollbox *"alte Abflusszeitreihen löschen"* legt fest, ob vorhandene Pegeldaten gelöscht werden sollen oder die neuen Daten hinzugefügt werden sollen. Durch Betätigen des Buttons wird das Einlesen der Daten gestartet.

C.1.5 Der Editor *"Fluss-Wasserstände einlesen"*

Der Editor *"Fluss-Wasserstände einlesen"* wird über den Button geöffnet. Es wird das Fenster in Abb. C.6 geöffnet.

Abb. C.6: Der Editor ogrut_read_watertable_menu dient zum Einlesen der Wasserstandsdaten an den Querprofilen eines Gerinnes.

In der Statuszeile wird das aktuelle OGRUT-Projekt angezeigt (*"Zartener Becken"*). Mit diesem Editor lassen sich Wasserstandszeitreihen für die Querprofile eines Gerinnes aus einer externen Datei einlesen. Durch Markieren eines Flusses in der Liste und Betätigen des Buttons wird der Fluss ausgewählt. Über den Button wird ein Fenster geöffnet, das eine Dateiauswahl anbietet (s. Abb. C.2). Die Checkbox *"alte Wasserstandszeitreihen löschen"* legt fest, ob vorhandene Daten gelöscht werden oder die neuen Daten zu den vorhandenen Zeitreihen hinzugefügt werden. Durch Betätigen des Buttons wird die Funktion gestartet.

C.1.6 Der Editor *"Grundwasserstände einlesen"*

Der Editor *"Grundwasserstände einlesen"* wird über den Button geöffnet. Es wird das Fenster in Abb. C.7 geöffnet.

Abb. C.7: Der Editor ogrut_read_gwtable_menu dient zum Einlesen von Grundwasserstandsdaten aus einer externen Datei.

In der Statuszeile wird das aktuelle OGRUT-Projekt angezeigt (*"Zartener Becken"*). In der zweiten Zeile wird das dem OGRUT-Projekt zugeordnete Iso-Projekt angezeigt. Mit diesem Editor lassen sich Grundwasserstandsdaten, die durch *FEFLOW* berechnet wurden in das Finite Elemente-Projekt einlesen. Über den Button wird ein Fenster geöffnet, das einen Dateiauswahl anbietet (s. Abb. C.2). Die Kontrollbox *"alte Grundwasserdaten löschen"* legt fest, ob vorhandene Grundwasserstände gelöscht werden oder die neuen Daten den bestehenden Zeitreihen hinzugefügt werden. Durch Betätigen des Buttons wird die Funktion ausgeführt.

C.1.7Der Editor *"Zuordnung sites ↔Katasterflächen"*

Der Editor *"Zuordnung sites ↔Katasterflächen"* wird über den Button geöffnet. Es wird das Fenster in Abb. C.8 geöffnet.

Abb. C.8: Der Editor ogrut_attach_sites_areas_menu dient dazu, statische Zuordnungen zwischen dem digitalen Geländemodell eines Gewässers und Katasterflächen durchzuführen..

In der Statuszeile wird das aktuelle OGRUT-Projekt angezeigt (*"Zartener Becken"*). Mit diesem Editor lassen sich Zuordnungen zwischen den einzelnen sites des digitalen Geländemodells eines Gerinnes und Katasterflächen erstellen. Dabei werden die Geländepunkte jeweils der Katasterfläche zugeordnet, in der sie räumlich liegen. Durch Markieren eines Gerinnes in der Liste und Betätigen des Buttons wird ein Gerinne ausgewählt. Die Checkbox *"Alte Zuordnungen löschen"* legt fest, ob bestehende Zuordnungen entfernt werden oder die neuen Zuordnungen den bestehenden hinzugefügt werden. Durch Betätigen des Buttons wird die Funktion gestartet.

C.1.8Der Editor *"Zuordnungen Katasterflächen ↔ finite Elemente"*

Der Editor *"Zuordnungen Katasterflächen ↔ finite Elemente"* wird über den Button geöffnet. Es wird das Fenster in Abb. C.9 geöffnet.

Abb. C.9: Der Editor ogrut_attach_areas_elements_menu dient dazu, statische Zuordnungen zwischen Katasterflächen und finiten Elementen erstellen.

In der Statuszeile wird das aktuelle OGRUT-Projekt angezeigt (*"Zartener Becken"*). in der zweiten Zeile wird das finite Elemente-Projekt angezeigt. Mit diesem Editor lassen sich statische Zuordnungen zwischen Katasterflächen und den Elementen des finiten Elemente-Projekts im aktuellen OGRUT-Projekt erstellen.. Dabei wird eine Flächengewichtung berechnet, die den Flächenanteil von Katasterfläche an den einzelnen Elementen ergibt. Die Kontrollbox *"Alte Zuordnungen löschen"* legt fest, ob bestehende Zuordnungen entfernt werden oder die neuen Zuordnungen den bestehenden hinzugefügt werden. Durch Betätigen des Buttons wird die Funktion gestartet.

C.1.9Der Editor *"Hydraulik-Programm starten"*

Der Editor *"Hydraulik-Programm starten"* wird über den Button geöffnet. Es wird das Fenster in Abb. C.10 geöffnet.

Abb. C.10: Der Editor ogrut_calculate_run_wsp_menu dient zum Aufruf des externen Programms UWSP.

In der Statuszeile wird das aktuelle OGRUT-Projekt angezeigt (*"Zartener Becken"*). Mit diesem Editor kann das externe Hydraulikprogramm *UWSP* ausgeführt werden. Durch Markieren eines Gerinnes in der Liste und Betätigen des Buttons wird das Gerinne ausgewählt. Durch Betätigen des Buttons wird die Funktion gestartet.

C.1.10 Der Editor *"DGM bei einem Fluss triangulieren"*

Der Editor *"DGM bei einem Fluss triangulieren"* wird über den Button geöffnet. Es wird das Fenster in Abb. C.11 geöffnet.

Abb. C.11: Der Editor ogrut_triangulate_tin_menu dient zum Triangulieren des digitalen Geländemodells bei einem Gerinne.

In der Statuszeile wird das aktuelle OGRUT-Projekt angezeigt (*"Zartener Becken"*). Mit diesem Editor lässt sich die Triangulation für das Tin bei einem Gerinne durchführen. Durch Markieren eines Gerinnes in der Liste und Betätigen des Buttons wird das Gerinne ausgewählt. Durch Betätigen des Buttons wird die Funktion gestartet.

C.1.11 Der Editor *"Berechne Überflutung eines DGM"*

Der Editor *"Berechne Überflutung eines DGM"* wird über den Button geöffnet. Es wird das Fenster in Abb. C.12 geöffnet.

Abb. C.12: Der Editor ogrut_flooding_tin_menu dient zur Berechnung der Überflutung eines digitalen Geländemodells durch das zugeordnente Gerinne.

In der Statuszeile wird das aktuelle OGRUT-Projekt angezeigt (*"Zartener Becken"*). Mit diesem Editor lässt sich die Überflutung eines digitalen Geländemodells durch das zugeordnete Gerinne berechnen. Durch Markieren eines Gerinnes in der Liste und Betätigen des Buttons wird das Gerinne ausgewählt. Die Checkbox *"Alte Überflutungsdaten löschen"* legt fest, ob bestehende Daten entfernt werden oder die neuen Daten den bestehenden Zeitreihen hinzugefügt werden. Die Auswahlschalter *"Rechnungsbeginn"* und *"Rechnungsende"* legen fest, welcher Ausschnitt der verfügbaren Zeitreihe berechnet wird. In der ersten Zeile sind der minimale Anfangszeitpunkt bzw. der maximale Endzeitpunkt angegeben. Alternativ kann in der zweiten Zeile jeweils ein beliebiger Zeitpunkt festgelegt werden. Durch Betätigen des Buttons wird die Funktion gestartet.

C.1.12 Der Editor *"Berechne Überflutung und GWNB der Katasterflächen"*

Der Editor *"Berechne Überflutung und GWNB der Katasterflächen"* wird über den Button geöffnet. Es wird das Fenster in Abb. C.13 geöffnet.

Abb. C.13: Der Editor ogrut_flooding_areas_menu dient zur Berechnung der Überflutung von Katasterflächen durch ein digitales Geländemodell.

In der Statuszeile wird das aktuelle OGRUT-Projekt angezeigt (*"Zartener Becken"*). Mit diesem Editor lässt sich die Überflutung und Grundwasserneubildung von Katasterflächen durch das digitale Geländemodell bei einem Gerinne berechnen. Durch Markieren eines Gerinnes in der Liste und Betätigen des Buttons wird das Gerinne ausgewählt. Die Checkbox *"Alte Überflutungsdaten löschen"* legt fest, ob bestehende Daten entfernt werden oder die neuen Daten den bestehenden Zeitreihen hinzugefügt werden. Die Auswahlschalter *"Rechnungsbeginn"* und *"Rechnungsende"* legen fest, welcher Ausschnitt der verfügbaren Zeitreihe berechnet wird. In der ersten Zeile sind der minimale Anfangszeitpunkt bzw. der maximale Endzeitpunkt angegeben. Alternativ kann in der zweiten Zeile jeweils ein beliebiger Zeitpunkt festgelegt werden. Durch Betätigen des Buttons wird die Funktion gestartet.

C.1.13 Der Editor *"Berechne GWNB der finiten Elemente"*

Der Editor *"Berechne GWNB der finiten Elemente"* wird über den Button geöffnet. Es wird das Fenster in Abb. C.14 geöffnet.

Abb. C.14: Der Editor ogrut_flooding_elements_menu dient zur Berechnung der Grundwasserneubildung von finiten ELementen aus Zeitreihen an Katasterflächen.

In der Statuszeile wird das aktuelle OGRUT-Projekt angezeigt (*"Zartener Becken"*). in der zweiten Zeile wird das Finite Elemente-Projekt im OGRUT-Projekt angezeigt. Mit diesem Editor lässt sich die Grundwasserneubildung an finiten Elementen durch die Grundwasserneubildung auf Katasterflächen berechnen. Die Kontrollbox *"Alte Grundwasser-neubildungsdaten löschen"* legt fest, ob bestehende Daten entfernt werden oder die neuen Daten den bestehenden Zeitreihen hinzugefügt werden. Durch Betätigen des Buttons wird die Funktion gestartet.

C.1.14 Der Editor *"Geometriedaten auslesen"*

Der Editor *"Geometriedaten auslesen"* wird über den Button geöffnet. Es wird das Fenster in Abb. C.15 geöffnet.

Abb. C.15: Der Editor ogrut_write_geometry_menu dient zum Auslesen von Querprofildaten in eine externe Datei.

In der Statuszeile wird das aktuelle OGRUT-Projekt angezeigt (*"Zartener Becken"*). Mit diesem Editor lassen sich Querprofildaten eines Gerinnes in eine externe Datei auslesen. Durch Markieren eines Gerinnes in der Liste und Betätigen des Buttons wird das Gerinne ausgewählt. Durch Betätigen des Buttons wird ein Dateiauswahl-Dialogfenster geöffnet. Darin kann eine Ausgabedatei angegeben werden. Die Auswahlbox *"strukturierte Dateien"* legt fest, dass ein strukturiertes Dateiformat verwendet wird (s. Anhang E), bei *"unstrukturierte Dateien"* wird ein unstrukturiertes Ausgabeformat gewählt. Durch Betätigen des Buttons wird die Ausgabe der Daten gestartet.

C.1.15 Der Editor *"Pegelabflüsse auslesen"*

Der Editor *"Pegelabflüsse auslesen"* wird über den Button geöffnet. Es wird das Fenster in Abb. C.16 geöffnet.

Abb. C.16: Der Editor ogrut_write_runoff_menu dient zum Auslesen von Abflussdaten in eine externe Datei.

In der Statuszeile wird das aktuelle OGRUT-Projekt angezeigt (*"Zartener Becken"*). Mit diesem Editor lasen sich Abflusszeitreihen an den Pegeln eines Gerinnes in eine externe Datei auslesen. Durch Markieren eines Gerinnes in der Liste und Betätigen des Buttons wird das Gerinne ausgewählt. Durch Betätigen des Button wird ein Dateiauswahl-Dialogfenster geöffnet. Darin kann eine Ausgabedatei angegeben werden. Durch Betätigen des Buttons wird die Ausgabe der Daten gestartet.

C.1.16 Der Editor *"Fluss-Wasserstände auslesen"*

Der Editor *"Fluss-Wasserstände auslesen"* wird über den Button geöffnet. Es wird das Fenster in Abb. C.17 geöffnet.

Abb. C.17: Der Editor ogrut_write_watertable_menu dient zum Auslesen von Wasserstandsdaten in eine externe Datei.

In der Statuszeile wird das aktuelle OGRUT-Projekt angezeigt (*"Zartener Becken"*). Mit diesem Editor lassen sich Wasserstandszeitreihen an einem Gerinne in eine externe Datei auslesen. Durch Markieren eines Gerinnes in der Liste und Betätigen des Buttons wird das Gerinne ausgewählt. Durch Betätigen des Buttons wird ein Dateiauswahl-Dialogfenster geöffnet. Darin kann eine Ausgabedatei angegeben werden. Die Auswahlbox *"strukturierte Dateien"* legt fest, dass ein strukturiertes Dateiformat verwendet wird (s. Anhang E), bei *"unstrukturierte Dateien"* wird ein unstrukturiertes Ausgabeformat gewählt. Durch Betätigen des Buttons wird die Ausgabe der Daten gestartet.

C.1.17 Der Editor *"GWNB der finiten Elemente auslesen"*

Der Editor *"GWNB der finiten Elemente auslesen"* wird über den Button geöffnet. Es wird das Fenster in Abb. C.18 geöffnet.

Abb. C.18: Der Editor ogrut_write_d_gwnb_menu dient zum Auslesen von Grundwasserneubildungsdaten finiter Elemente in eine externe Datei.

In der Statuszeile wird das aktuelle OGRUT-Projekt angezeigt (*"Zartener Becken"*). In der zweiten Zeile wird das finite Elemente-Projekt angezeigt, das im OGRUT-Projekt enthalten ist. Mit diesem Editor lassen sich Grundwasserneubildungszeitreihen des finiten Elemente-Netzes in eine externe Datei auslesen. Durch Betätigen des Buttons wird ein Dateiauswahl-Dialogfenster geöffnet. Darin kann eine Ausgabedatei angegeben werden. Durch Betätigen des Buttons wird die Ausgabe der Daten gestartet.

Anhang D Verzeichnisstruktur der externen Dateien

Zur einfachen und sicheren Kommunikation zwischen *Smallworld* und dem externen Programm *UWSP* sowie zwischen *Smallworld* und *FEFLOW* wird für externe Dateien eine feste Verzeichnisstruktur verwendet, in der die Ein- und Ausgabedaten in festgelegten Ordnern und unter festen Namen abgelegt sind. *UWSP* erfährt die gewünschten Verzeichnisse über Kommandozeilenparameter, so dass diese Struktur nicht zwingend ist. *OGRUT* verwendet sie jedoch, wenn es *UWSP* automatisch aufruft. Die Verzeichnisstruktur ist in Abb. D.1 dargestellt.

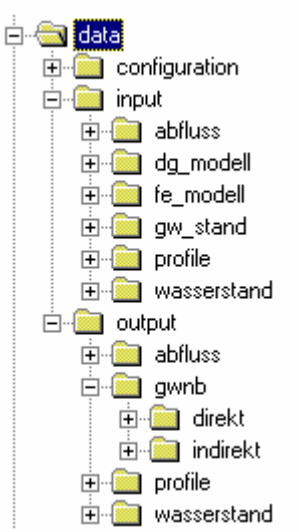


Abb. D.1: Verzeichnisstruktur, in der externe Dateien von OGRUT und UWSP verwaltet werden.

Anhang E Format der externen Dateien

E.1.1Geometriedateien

Diese Dateien liegen in den Verzeichnissen `data/input/profile` und `data/output/profile`. Sie enthalten Querprofildaten von Gerinnen. Die Beispieldateien sind wegen ihrer Länge an der Stelle "..." gekürzt. Die Namen der Dateien sind per Konvention die der Flüsse, auf die sie sich beziehen.

Datei "Krummbach.txt" - unstrukturiertes Format

```
// Krummbach
// Profil-Nr  Abstand  Punktabstand  Höhe  Uferkante   Strickler-Beiwert   Rechtswert  Hochwert
1    0        -82      330.29            33      3419309.671    5316228.517
1    0        -50.41   330.44            33      3419323.859    5316256.634
1    0        -19.56   330.46  1         34      3419337.715    5316284.091
1    0        -5.62    328.96            35      3419343.976    5316296.498
1    0        -3.39    327.15            35      3419344.977    5316298.483
1    0        0        327.27  2         34      3419346.5      5316301.5
1    0        1.3      328.9             33      3419347.084    5316302.657
1    0        29.4     330.67            33      3419359.705    5316327.667
2    0.0406   -50.04   334.19            33      3419327.545    5316234.694
2    0.0406   -30.67   330.77            33      3419342.914    5316246.425
2    0.0406   -4.1     331.27  1         34      3419363.997    5316262.517
2    0.0406   -3.64    330.54            35      3419364.362    5316262.795
2    0.0406   -0.35    330.63            35      3419366.972    5316264.788
2    0.0406   0        331.09  2         34      3419367.25     5316265
2    0.0406   26.56    331.33            33      3419388.325    5316281.086
3    0.10008          -53.57  335.04          33      3419378.753    5316172.359
3    0.10008          -23.99  335.21          33      3419388.967    5316200.046
3    0.10008          -5.3    332.8   1       34      3419395.42     5316217.539
3    0.10008          -3.42   331.67          35      3419396.069    5316219.299
3    0.10008          -0.91   331.81          35      3419396.936    5316221.648
3    0.10008          0       332.96  2       34      3419397.25     5316222.5
3    0.10008          47.03   333.59          33      3419413.489    5316266.519
3    0.10008          92.83   333.58          33      3419429.303    5316309.388
```

Datei "Krummbach.txt" - strukturiertes Format

```
// Krummbach
//    Höhe    Strickler-Beiwert        Rechtswert      Hochwert

river = {
      section = {
                x = 0.00000
                left = {
                330.290                 33      3419309.671    5316228.517
                330.440                 33      3419323.859    5316256.634
                }
                channel = {
                330.460                 34      3419337.715    5316284.091
                327.150                 33      3419344.977    5316298.483
                }
                right = {
                327.270                 34      3419346.500    5316301.500
                328.900                 33      3419347.084    5316302.657
                }
      }
      section = {
                x = 0.04060
                left = {
                334.190                 33      3419327.545    5316234.694
                330.770                 33      3419342.914    5316246.425
                }
                channel = {
                331.270                 34      3419363.997    5316262.517
                330.630                 33      3419366.972    5316264.788
                }
                right = {
                331.090                 34      3419367.250    5316265.000
                331.330                 33      3419388.325    5316281.086
                }
      }
      section = {
                x = 0.10008
                left = {
                335.040                 33      3419378.753    5316172.359
                335.210                 33      3419388.967    5316200.046
                }
                channel = {
                332.800                 34      3419395.420    5316217.539
                331.810                 35      3419396.936    5316221.648
                }
                right = {
                332.960                 34      3419397.250    5316222.500
                333.590                 33      3419413.489    5316266.519
                333.580                 33      3419429.303    5316309.388
                }
      }
      ...
      ...
}
```

E.1.2 Dateien für digitale Geländemodelle

Diese Dateien liegen im Verzeichnis `data/input/dg_modell`. Sie enthalten Koordinatendaten digitaler Geländepunkte. Die Beispieldatei ist wegen ihrer Länge an der Stelle "..." gekürzt. Die Namen der Dateien sind per Konvention die der Flüsse, auf die sie sich beziehen.

Datei "Brugga.txt"

```
1  0.34217435000000000D+07  0.53136300000000000D+07  0.3970000E+03
1  0.34217410000000000D+07  0.53135910000000000D+07  0.3980000E+03
1  0.34217315000000000D+07  0.53136265000000000D+07  0.3970000E+03
1  0.34190798912999998D+07  0.53161074989000000D+07  0.3261300E+03
1  0.34190792289000000D+07  0.53161070502000004D+07  0.3273500E+03
1  0.34190273985000001D+07  0.53161922573999995D+07  0.3274400E+03
1  0.34216412500000000D+07  0.53118700000000000D+07  0.4350000E+03
1  0.34216527500000000D+07  0.53118795000000000D+07  0.4301000E+03
1  0.34216490000000000D+07  0.53118700000000000D+07  0.4301000E+03
1  0.34215727500000000D+07  0.53126685000000000D+07  0.4120000E+03
1  0.34215740000000000D+07  0.53126575000000000D+07  0.4120000E+03
1  0.34215607500000000D+07  0.53126605000000000D+07  0.4120000E+03
1  0.34216677500000000D+07  0.53136530000000000D+07  0.3950000E+03
1  0.34216255000000000D+07  0.53136910000000000D+07  0.3940000E+03
1  0.34216340000000000D+07  0.53137005000000000D+07  0.3940000E+03
...
...
```

E.1.3 Dateien für Finite Elemente-Netze

Diese Dateien liegen im Verzeichnis `data/input/fe_modell`. Sie enthalten Daten für das finite Elemente-Netz an Pegeln. Die Beispieldateien sind wegen ihrer Länge an der Stelle "..." gekürzt. Die Datei `nodes.txt` enthält Knoten-Koordinaten, `elements.txt` enthält Finite-Elemente.

Datei "nodes.txt"

```
fesim nodes_header
  10767
    1  +3.417330E+08  +5.317524E+08
    2  +3.417314E+08  +5.317448E+08
    3  +3.417292E+08  +5.317368E+08
    4  +3.417266E+08  +5.317292E+08
    5  +3.417236E+08  +5.317201E+08
    6  +3.417205E+08  +5.317113E+08
    7  +3.417197E+08  +5.317041E+08
    8  +3.417170E+08  +5.316959E+08
    9  +3.417148E+08  +5.316884E+08
   10  +3.417127E+08  +5.316813E+08
   11  +3.417105E+08  +5.316742E+08
   12  +3.417083E+08  +5.316670E+08
   13  +3.417060E+08  +5.316596E+08
   14  +3.417041E+08  +5.316523E+08
   15  +3.417021E+08  +5.316454E+08
  ...
...
```

Datei "elements.txt"

```
fesim fe_header
  20720
    1    15    16    82
    2    16    17    82
    3    17    18    82
    4    81     1    83
    5     1     2    83
    6    81    83    80
    7    44    45    84
    8    45    46    84
    9    42    43    85
   10    43    44    85
   11    44    84    85
   12    53    54    86
   13    54    55    86
   14    52    53    87
   15    53    86    87
...
...
```

E.1.4 Grundwasserstands-Dateien

Diese Dateien liegen in dem Verzeichnis `data/input/gw_stand`. Sie enthalten Grundwasserstandsdaten aus der Simulation von *FEFLOW*. Die Beispieldatei ist wegen ihrer Länge an der Stelle "..." gekürzt.

Datei "gw_stand.txt"

```
time = {
      start = { 1987,1,1, 0,0,0 }        // years. months. days. hours. minutes. seconds
      step =  {    0,0,1, 0,0,0 }        // years. months. days. hours. minutes. seconds
}

values = {
   1     +3.017330E+08 +3.027330E+08 +3.037330E+08 +3.047330E+08,
   2     +3.117314E+08 +3.127314E+08 +3.137314E+08 +3.147314E+08,
   3     +3.217292E+08 +3.227292E+08 +3.237292E+08 +3.247292E+08,
   4     +3.317266E+08 +3.327266E+08 +3.337266E+08 +3.347266E+08,
   5     +3.417236E+08 +3.427236E+08 +3.437236E+08 +3.447236E+08,
   6     +3.517205E+08 +3.527205E+08 +3.537205E+08 +3.547205E+08,
   7     +3.617197E+08 +3.627197E+08 +3.637197E+08 +3.647197E+08,
   8     +3.717170E+08 +3.727170E+08 +3.737170E+08 +3.747170E+08,
   9     +3.817148E+08 +3.827148E+08 +3.837148E+08 +3.847148E+08,
  10     +3.917127E+08 +3.927127E+08 +3.937127E+08 +3.947127E+08
      ...
      ...
}
```

E.1.5Wasserstandsdateien

Diese Dateien liegen in den Verzeichnissen `data/input/wasserstand` und `data/output/wasserstand`. Sie enthalten Wasserstandszeitreihen an Pegeln. Die Beispieldateien sind wegen ihrer Länge an der Stelle "..." gekürzt. Die Namen der Dateien sind per Konvention die der Flüsse, auf die sie sich beziehen.

Datei "Eschbach.txt" - unstrukturiertes Format

```
// Eschbach
Geometrie-Datei = \\eine\smallworld\liwis\johe\data\output\profile\Eschbach.txt
Zahl der Querprofile: 60
Start: 01.01.1987 00:00:00
Step: 0,0,1,0,0,0          // year,month,day,hour,minute,second
Zahl der Zeitreihenwerte: 3592
    327.720    330.849    332.785    335.403    336.700    338.015    376.981
    327.819    330.916    334.767    335.486    336.802    338.068    376.995
    327.683    330.786    334.780    335.358    336.787    337.942    377.059
    327.622    330.751    334.133    335.320    336.734    337.889    376.969
    327.586    330.731    333.786    335.295    336.702    337.858    376.926
    327.558    330.717    333.582    335.276    336.682    337.833    376.900
    327.493    330.688    333.273    335.229    336.641    337.774    376.859
...
...
```

Datei "Brugga.txt" - strukturiertes Format

```
river = {
      name = "Brugga",
      start = { 1.1.1987 24:0:0 },       // years. months. days. hours. minutes. seconds
      step =  {  0,0,1,0,0,0 },   // year,month,day,hour,minute,second
      sections_count = 64,
      h = {
              323.894                    325.350                    326.504
              327.470                    328.834                    329.169
              330.129                    330.552                    331.386
              333.690                    334.465                    335.480
              340.051                    340.847                    341.411
              341.855                    342.930                    344.153
              345.607                    347.235                    348.197
              ...
              ...
      }
}
```

E.1.6 Abflussdateien

Diese Dateien liegen in den Verzeichnissen `data/input/abfluss` und `data/output/abfluss`. Sie enthalten Abflusszeitreihen an Pegeln. Die Beispieldatei ist wegen ihrer Länge an der Stelle "..." gekürzt. Die Namen der Dateien sind per Konvention die der Flüsse, auf die sie sich beziehen.

Datei "Eschbach.txt"

```
// Eschbach

time = {
      start = { 1987,1,1, 0,0,0 }        // years. months. days. hours. minutes. seconds
      step =  {    0,0,1, 0,0,0 }        // years. months. days. hours. minutes. seconds
}


section = {              // der letzte Wert muss >= der Nummer des obersten Querprofils sein.
      Zufluss2 = 57,
      Zufluss1 = 62,
      Oberlauf = 999            // dies ist der Anfangsabfluss
}


Q = {                                    // Abflüsse für die angegebenen Querprofile
6.1672      5.4556          4.744
8.20885     7.261675        6.3145
5.28905     4.678775        4.0685
3.82525     3.383875        2.9425
3.1824      2.8152          2.448
2.86715     2.536325        2.2055
2.353       2.0815          1.81
1.99615     1.765825        1.5355
1.82715     1.616325        1.4055
1.6588      1.4674          1.276
1.4157      1.25235         1.089
1.3117      1.16035         1.009
1.30975     1.158625        1.0075
1.21745     1.076975        0.9365
...
...
}
```

Anhang F Normalisierung einer Datenbank

Um eine Datenbank fehlerfrei zu machen und sie zu optimieren, steht die Methode der Normalisierung zur Verfügung. Es gibt mehrere *Normalformen*. Eine Datenbank befindet sich beispielsweise in der dritten Normalform, wenn sie den Bedingungen der ersten, zweiten und dritten Normalform genügt.

Bedingungen der *ersten Normalform* sind:

- Die Datenbank enthält keine doppelten Datensätze.
- Die Reihenfolge der Datensätze und Attribute muss beliebig sein dürfen. Die Funktionalität der Datenbank darf also nicht davon abhängen, dass auf eine Zeile oder Spalte eine bestimmte nächste Zeile oder Spalte folgt. Die Zeilen werden ausschliesslich über den Primärschlüssel, die Spalten ausschliesslich über den Spaltennamen angesprochen.
- Alle Attributwerte unterliegen einer Domäne und sind somit unteilbar (atomar).

Zusätzliche Bedingung der *zweiten Normalform* ist:

- Jedes Schlüsselattribut ist funktional abhängig vom Gesamtschlüssel, nicht aber von dessen Teilen.

Zusätzliche Bedingung der *dritten Normalform* ist:

- Es gibt keine funktionalen Abhängigkeiten zwischen Attributen, die nicht als Schlüssel definiert sind.

Man betrachte zum Beispiel Tab. F.1.

Tabelle **Mitarbeiter**

Nr.	Name	Telefon	Abt.-Nr.	Abt.-Name	Abt.-Chef	Produkt-Nr.	Produkt-Name
123	Müller	56	7	Verkauf	Maier	8, 9	Brot, Spiele

Tab. F.1: Relation genügt nicht der ersten Normalform

Diese Tabelle könnte die Mitarbeiterdatei einer Firma sein. Sie entspricht noch nicht einmal der ersten Normalform, da hier zwei Spalten Werte haben, die nicht atomar sind (`Produkt-Nr.` und `Produkt-Name`). Ein solcher Aufbau führt zu Problemen, wenn über diese Attribute eine Referenz gebildet werden soll. Die Tabelle in Tab. F.2 genügt der ersten Normalform

Tabelle **Mitarbeiter**

Nr.	Name	Telefon	Abt.-Nr.	Abt.-Name	Abt.-Chef	Produkt-Nr.	Produkt-Name
123	Müller	56	7	Verkauf	Maier	8	Brot
123	Müller	56	7	Verkauf	Maier	9	Spiele

Tab. F.2: Relation genügt der ersten Normalform

Gegen die *zweite Normalform* kann nur verstossen werden, wenn der Primärschlüssel aus mehreren Attributen zusammengesetzt ist, was hier nicht der Fall ist, da wir die Spalte `Nr.` als Primärschlüssel annehmen.

Nach den Regeln der *dritten Normalform* sind keine funktionalen Abhängigkeiten zwischen Spalten erlaubt, die nicht als Primärschlüssel definiert sind. In Tab. F.2 finden sich solche Abhängigkeiten. `Abt.-Nummer`, `Abt.-Name`, und `Abt.-Chef` sind voneinander abhängig, denn Abteilung `7` ist immer `Verkauf` und hat immer den Chef `Maier`. Ebenso sind `Produkt-Nummer` und `Produkt-Name` voneinander abhängig. Diese Abhängigkeiten bedeuten Redundanzen. Um solche Abhängigkeiten aufzulösen, wird die Tabelle in mehrere kleine Tabellen aufgeteilt, wie dies Abb. F.1 zeigt.

Tabelle **Mitarbeiter**

Nr.	Name	Telefon	Abt.-Nr.
123	Müller	56	7

Tabelle **Abteilungen**

Nr.	Name	Chef
7	Verkauf	Maier

Tabelle **Aufgaben**

Nr.	Mitarbeiter-Nr.	Produkt-Nr.
1	123	8
2	123	9

Tabelle **Produkte**

Nr.	Name
8	Brot
9	Spiele

Abb. F.1: Aufteilung in mehrere Tabellen

Ehrenwörtliche Erklärung

Hiermit erkläre ich, dass ich diese Arbeit selbständig und nur unter Verwendung der angegebenen Hilfsmittel angefertigt habe.

Freiburg, den 10.08.2000