

Modellierung von Hydrosystemen
"Numerische und daten-basierte Methoden"
BHYWI-22-17 @ 2018
Gerinnehydraulik - Übungen
Selke-Modell

Olaf Kolditz

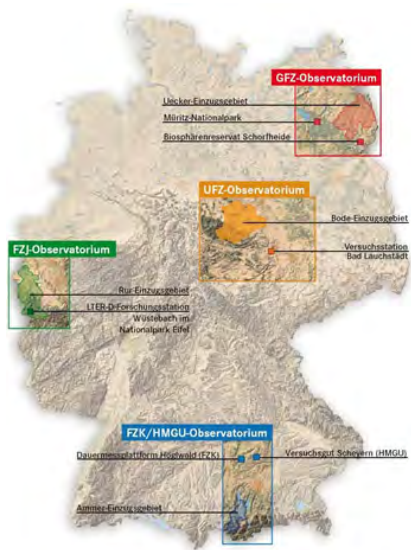
*Helmholtz Centre for Environmental Research – UFZ

¹Technische Universität Dresden – TUDD

²Centre for Advanced Water Research – CAWR

29.06.2018 - Dresden

- ▶ Kurzer Rückblick (Stabilitätskriterium)
- ▶ Vorstellen der Case Study: Selke Catchment (Bode)
- ▶ OGSDataExplorer
- ▶ Aktive und Inaktive Knoten
- ▶ FDM - erstmal QAD Programmierung (USA3)



- ▶ TERENO Projekt
- ▶ Steffen Zacharias
- ▶ Ute Wollschläger
- ▶ VISLab (Karsten Rink)

Case Study: Bode Einzugsgebiet



Figure: Digitales Geländemodell (DEM) des Bode-Einzugsgebietes

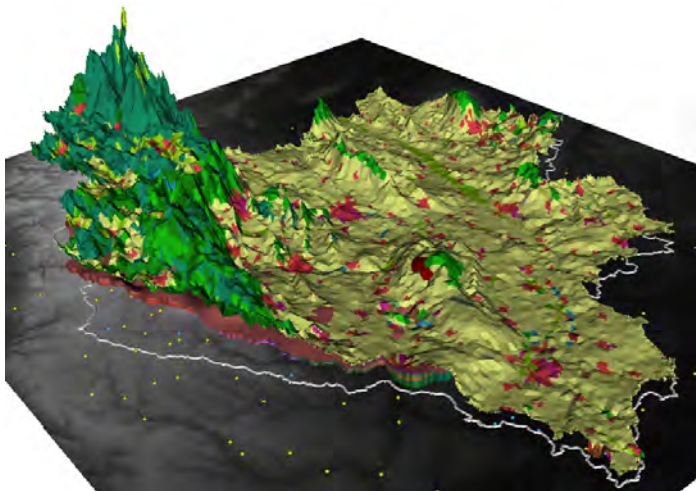


Figure: DEM mit Landnutzung

Case Study: Bode Einzugsgebiet

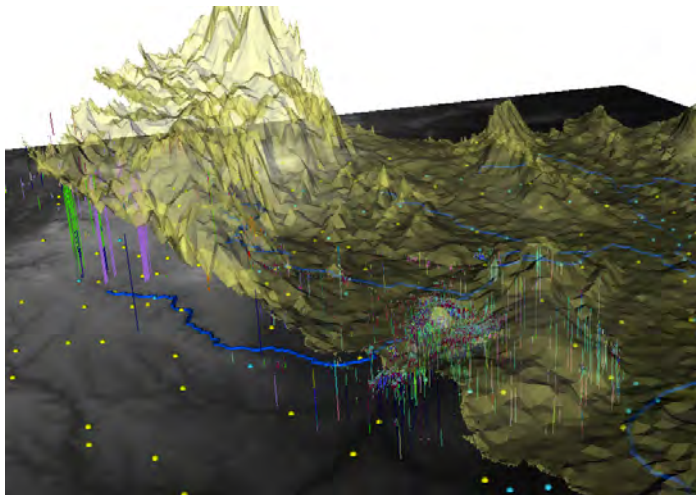


Figure: DEM mit geologischen Daten

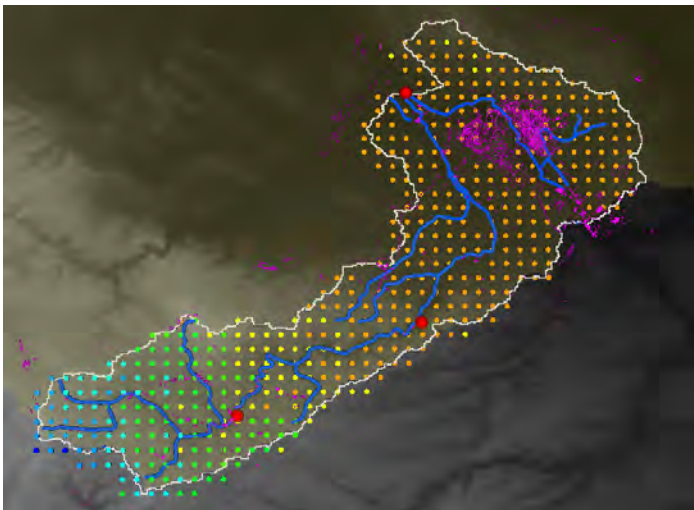
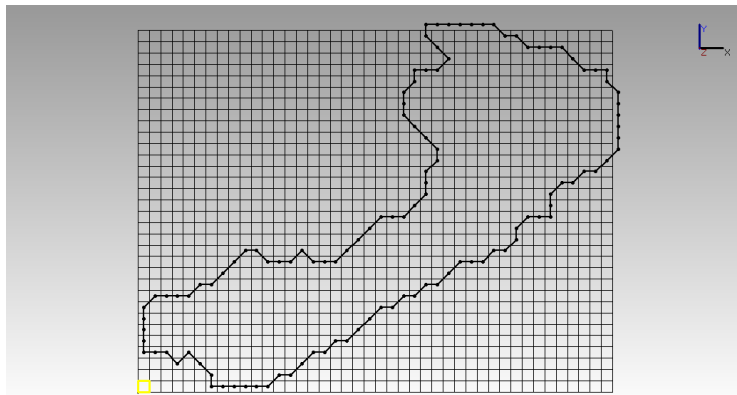


Figure: Untersuchungsgebiet - Selke

Selke Einzugsgebiet

Die Abb. zeigt uns eine mögliche Approximation des Selke-Einzugsgebietes mit einer relativ geringen Anzahl von FD Zellen ($32 \times 42 = 1344$).

- ▶ Aus wie vielen FD Knoten besteht das FD Mesh?



Wie bekommen wir aus unserem regelmäßigen rechteckigen Raster ein eher unregelmäßiges Catchment herausgeschnitten? Der Trick besteht darin, einzelne Zellen zu deaktivieren. Das klingt schon wieder nach Arbeit, ist aber machbar, dafür gibt's die nächste Übung (GW2). Die geometrische Analyse mit OGS liefert uns zunächst eine Liste von Gitterpunkten die ausserhalb des Catchments liegen (siehe Übung GW2):

- ▶ ExtractedSelkeMeshIDs.txt
- ▶ selke.gli

Diese Files können wir mal mit dem OGS-DatenExplorer (ogs-gui.exe) laden.

- ▶ Download von der Lehre-Seite
- ▶ Manual
- ▶ Übung

OGS-DE: Selke Einzugsgebiet

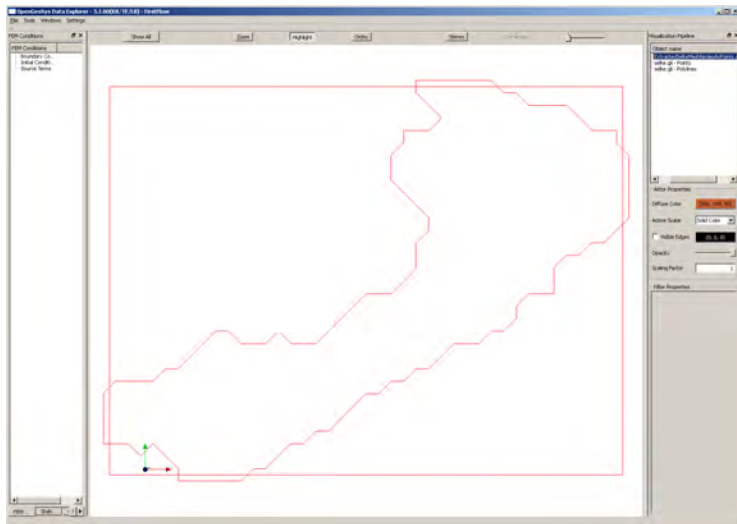


Figure: Das sind zwar die Daten, sieht aber noch nach nix aus ...

OGS-DE: Selke Einzugsgebiet

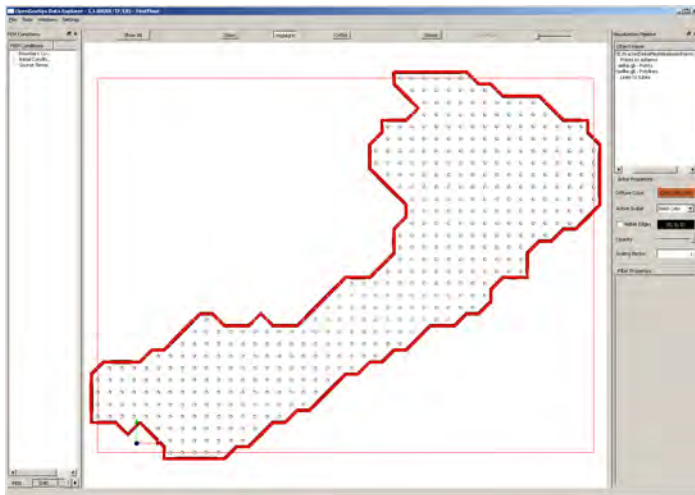
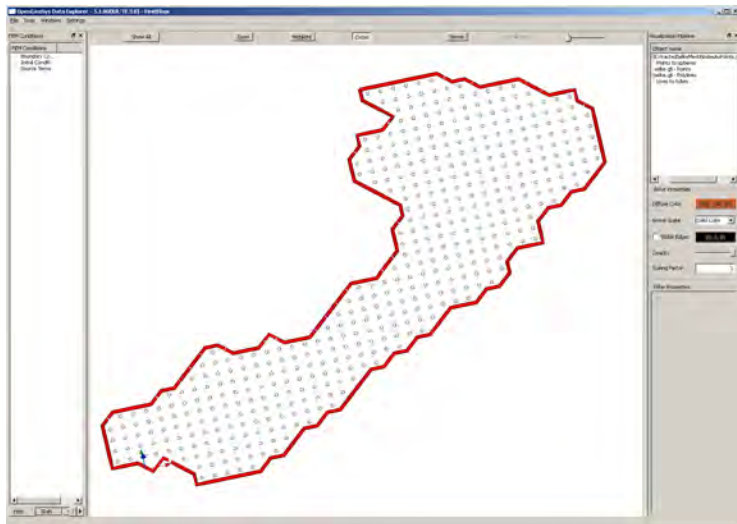


Figure: OGS hat ein paar nette VTK-Filter, um geometrische Objekte herauszuheben

OGS-DE: Selke Einzugsgebiet



Das ist zwar alles schön und gut, was wir aber brauchen sind die Knoten ausserhalb des Catchments, damit wir diese für das FD Verfahren deaktivieren können (Tafelbild). Also müssen wir doch selber ran. Unser Plan ist wie folgt:

- 1 Aktive Knoten lesen und speichern.
- 2 Aktive Knoten sortieren (Gruß an Hydroinformatik I - Hantieren mit Listen)
- 3 (das Zwischenergebnis zur Sicherheit mal rausschreiben)
- 4 Alle Knoten rausfischen, die NICHT aktiv sind.
- 5 Dabei kommt eine neue Hilfs-Funktion `NodeInList` ins Spiel (siehe unten).
- 6 Wir überzeugen uns vom Ergebnis (File schreiben) ...
- 7 ... und natürlich graphisch, wozu haben wir denn Visual C++ gelernt!

▶ ActiveNodes.txt

173

216

259

302

345

174

217

260

303

346

175

218

261

304

347

...

- ▶ Aktive Knoten lesen und speichern.
- ▶ Aktive Knoten sortieren (Gruß an Hydroinformatik I - Hantieren mit Listen)

```
std::list<int>nodes_active;
std::ifstream active_nodes_file;
active_nodes_file.open("ActiveNodes.txt");
int na;
while(!active_nodes_file.eof())
{
    active_nodes_file >> na;
    nodes_active.push_back(na);
}
nodes_active.sort();
```


- ▶ Das Zwischenergebnis zur Sicherheit mal rausschreiben

```
std::ofstream active_nodes_file_test;
active_nodes_file_test.open("ActiveNodesSorted.txt");
list<int>::const_iterator p = nodes_active.begin();
while(p!=nodes_active.end())
{
    active_nodes_file_test << *p << endl;
    ++p;
}
active_nodes_file_test.close();
```

- ▶ ActiveNodesSorted.txt

50

51

52

53

54

93

94

95

96

97

98

...

- ▶ E: Überzeugen sie sich, ob unser Unterfangen erfolgreich war, indem sie die Elementanzahl der beiden Listen bestimmen und rausschreiben.

- ▶ Alle Knoten rausfischen, die NICHT aktiv sind.
- ▶ Dabei kommt eine neue Hilfs-Funktion NodeInList ins Spiel (siehe unten).

```
for(j=0;j<jy;j++)
{
    nn = j*ix;
    for( i=0;i<ix;i++)
    {
        n = nn+i;
        if(!NodeInList(n,nodes_active))
            nodes_inactive.push_back(n);
    }
}
```

Die nützliche Hilfs-Funktion, die alle Knoten raussucht, die NICHT in nodes_activestehen.

```
bool NodeInList(int n, std::list<int> nodes_active)
{
    list<int>::const_iterator p = nodes_active.begin();
    while(p != nodes_active.end())
    {
        if(n == *p)
            return true;
        ++p;
    }
    return false;
}
```

- ▶ Wir überzeugen uns vom Ergebnis (File schreiben) ...

```
std::ofstream inactive_nodes_file;  
inactive_nodes_file.open("InactiveNodes.txt");  
for(i=0;i<nodes_inactive.size();i++)  
{  
    inactive_nodes_file << nodes_inactive[i] << endl;  
}  
inactive_nodes_file.close();
```

- ▶ ... und natürlich graphisch, wozu haben wir denn Visual C++ gelernt!

