Modellierung von Sedimenttransportprozessen an der Donau

Anforderungen aus der Praxis und technische Umsetzung mittels Shared-Memory-Datenaustausch

Prof. Dr. Michael Tritthart
Institut für Wasserbau, Hydraulik und Fließgewässerforschung
BOKU University

BOKU UNIVERSITY

Inhalt

- Zielsetzung
- Software iSed
- Modellkoppelung iSed - HydroAS

BOKU UNIVERSITY

Inhalt

- **Zielsetzung**
- Software iSed
- Modellkoppelung iSed - HydroAS

Ziel

- Berechnung des Sedimenttransports und der Morphodynamik
- Einfluss unterschiedlicher Wehrsteuerungsvarianten am Fallbeispiel der Donau:
  - Speziell bei kleineren Hochwasserereignissen
  - **Wissenschaftliche Betrachtung!**

Koppelung zwischen hydrodynamischen und morphodynamischen Modell erforderlich!

HW-Ereignis Juni 2024

Quelle: Maximilian Zauner

BOKU UNIVERSITY

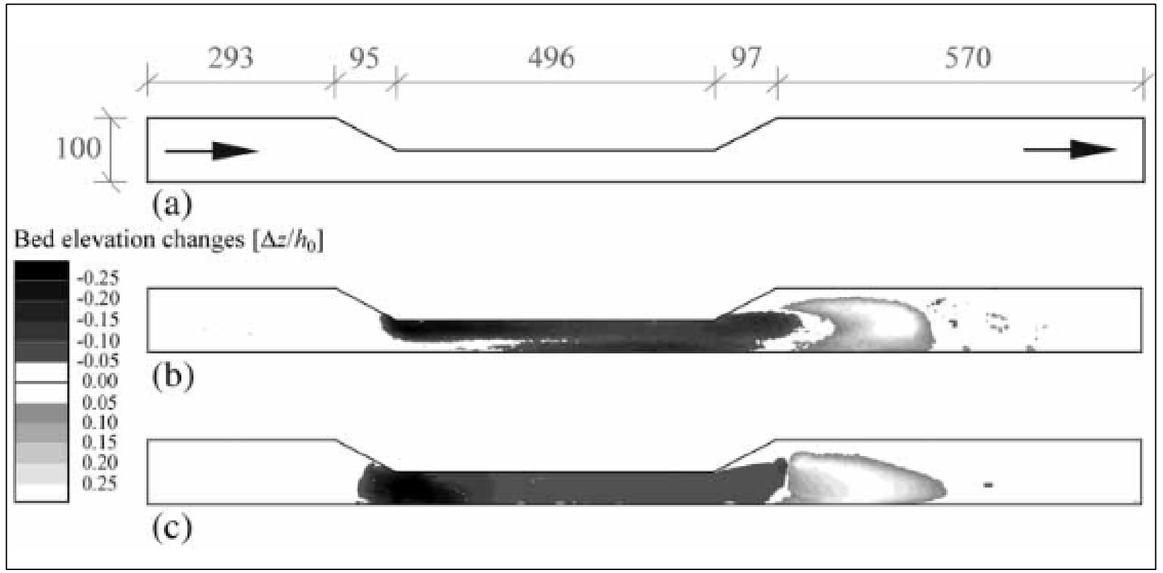
Inhalt

- Zielsetzung
- **Software iSed**
- Modellkoppelung iSed - HydroAS

Software iSed (Tritthart et al., 2011)

„integriertes Sedimenttransportmodell“ – Universität für Bodenkultur Wien

- Eine Software für die Berechnung von Schwebstoff- & Geschiebetransport sowie Morphodynamik
- iSed berechnet keine Hydrodynamik
  - Benötigt Koppelung (z.B. RSim-2D, RSim-3D, HydroAS, …)
  - Berücksichtigung einer unbegrenzten Anzahl von Sohlschichten und Korngrößenfraktionen
- **Schwebstofftransport:** Advektions – Diffusions – Gleichung
- **Geschiebetransport:** Lösung einer empirischen Formel (Meyer-Peter und Müller, van Rijn, Egiazaroff, Hunziker, Wu et al.)



Vergleich (in cm) von Grundriss (a) und Draufsicht (b) gemessene und (c) berechnete normalisierte Höhenänderungen der Sohle

Quelle: Tritthart, M., Schober, B. & Habersack, H. (2011). Non-uniformity and layering in sediment transport modelling 1: flume simulations. Journal of Hydraulic Research, 49(3), 325–334.

Software iSed (Tritthart et al., 2011)

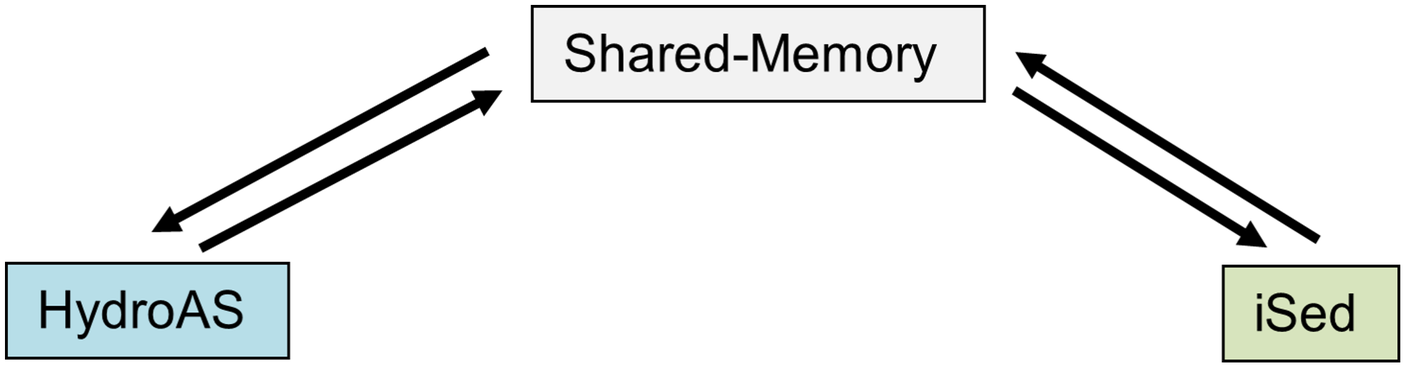
„integriertes Sedimenttransportmodell“ – Universität für Bodenkultur Wien

BOKU UNIVERSITY

Inhalt

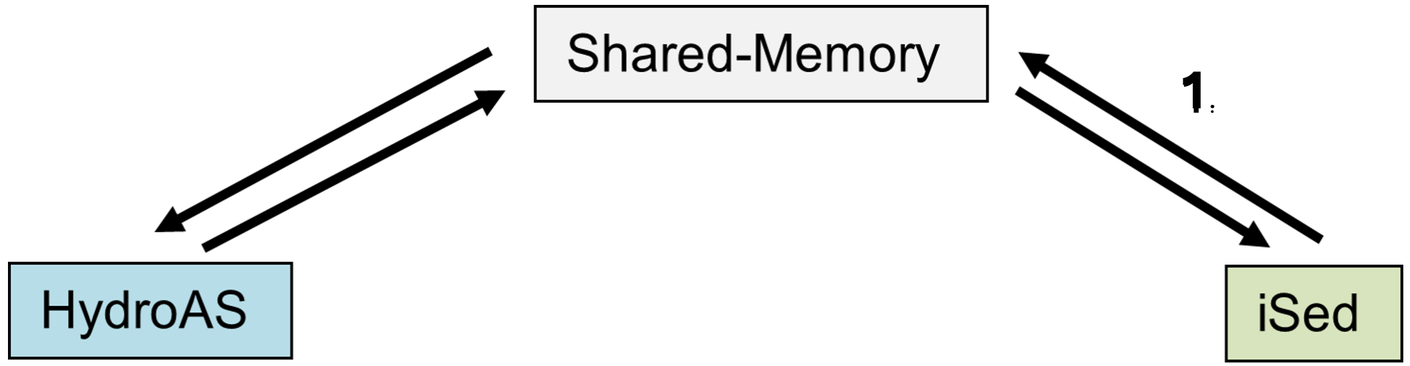
- Zielsetzung
- Software iSed
- **Modellkoppelung iSed - HydroAS**

Modellkoppelung HydroAS – iSed



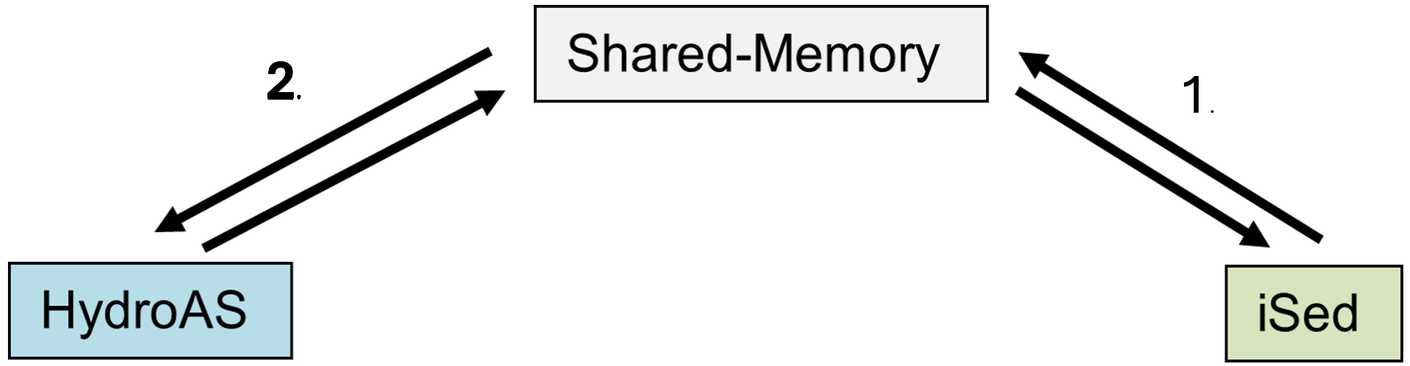
Modellkoppelung HydroAS – iSed

- Schema der Koppelung
  - **1. Schritt:** iSed – Erzeugen/Lesen der Netz-Geometrien und Setzen der Sedimentrandbedingungen



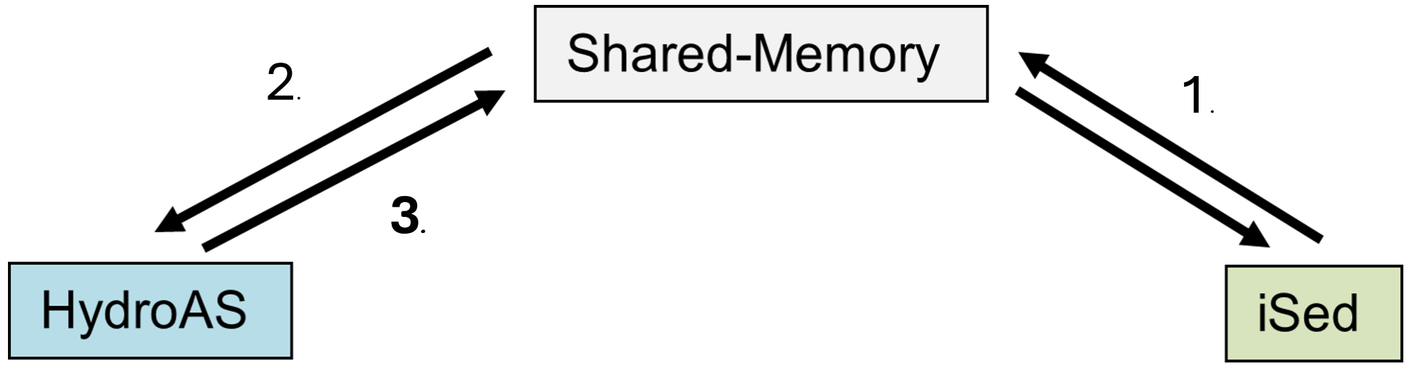
Modellkoppelung HydroAS – iSed

- Schema der Koppelung
  - **2. Schritt:** HydroAS – übernimmt Daten und startet daraufhin die hydrodynamische Berechnung



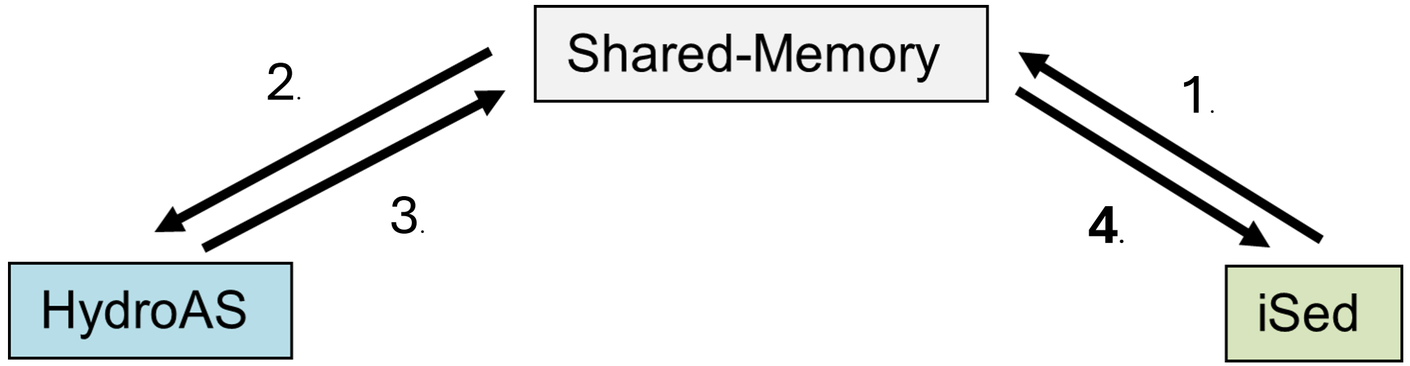
Modellkoppelung HydroAS – iSed

- Schema der Koppelung
  - **3. Schritt:** HydroAS – Die notwendigen berechneten Parameter (v, h, τ) für den Geschiebe- & Schwebstofftransport werden an den Shared-Memory übergeben



Modellkoppelung HydroAS – iSed

- Schema der Koppelung
  - **4. Schritt:** iSed – Auf Basis der ermittelten hydrodynamischen Kenngrößen wird die neue Sohlhöhe z(t) berechnet, welche anschließend wieder von HydroAS für den nächsten Zeitschritt verwendet wird



Modellkoppelung HydroAS - iSed

- Koppelung erfolgt mittels LUA-Scripting
  - Eigenes LUA-Script neben den bestehenden (z.B. für die Anwendung des Wehrsteuerungsmoduls)
  - Adaptierung der „control.lua“
  - Neues LUA-Script „ised.lua“ für Koppelung befindet sich im „Data-in“ Ordner

control.lua *

```
local wbo = require("wbo.WBOBuilder")
local ised = require("ised")

return
{
  step = function()
    wbo:step()
    ised:step()
  end,
  writeResult = function()
    --wbo:writeResult()
    ised:writeResult()
  end,
  close = function()
    wbo:close()
    ised:close()
  end
}
```

Modellkoppelung HydroAS - iSed

ised.lua

```
require("hydroas.LockUndeclared").lock(_G)
local hydroas = require("hydroas")

--- Semaphore & Shared-Memory ---
local sem1 = hydroas.Semaphore.newSemaphore("/HydroAS")
local sem2 = hydroas.Semaphore.newSemaphore("/iSed")

local shm_hydroAS_variables = hydroas.SharedMemory.openSysV(1233, hydroas.SharedMemory.TypeInt32, 2)
local shm_dt = hydroas.SharedMemory.openSysV(1234, hydroas.SharedMemory.TypeFloat32, 2)
local shm_z = hydroas.SharedMemory.openSysV(1235, hydroas.SharedMemory.TypeFloat32, hydroas.Node.count())
local shm_depth = hydroas.SharedMemory.openSysV(1236, hydroas.SharedMemory.TypeFloat32, hydroas.Node.count())
local shm_v = hydroas.SharedMemory.openSysV(1237, hydroas.SharedMemory.TypeFloat32, hydroas.Node.count())
local shm_sst = hydroas.SharedMemory.openSysV(1238, hydroas.SharedMemory.TypeFloat32, hydroas.Node.count())

local status, T, Tdef

local function copyvar(T)
  shm_v:copyVelocFromHA()
  shm_depth:copyDepthFromHA()
  shm_sst:copyShearstressFromHA()
  shm_dt:set(0, T)
  shm_hydroAS_variables:set(1, hydroas.Global.timestepIndex())
  return
end

if sem1:isOpen() and sem2:isOpen() then
   print("Semaphores are open.")

   if shm_hydroAS_variables:isOpen() and shm_dt:isOpen() and shm_z:isOpen() and shm_depth:isOpen() and shm_v:isOpen() and shm_sst:isOpen() then
      print("Shared memory segments are open.")
   shm_hydroAS_variables:set(0, 0) --HydroAS is running
   else
      error("Could not open all shared memory segments.")
   end
else
   error("Could not open semaphores.")
end
```

```
return {
   step = function()
      Tdef = shm_dt:get(1)
      T = hydroas.Global.timestepStart()
      --print("HydroAS calculated...", T)

      if (T >= Tdef) then
 copyvar(T)
        --Continue with iSed
        sem1:post()
        --Wait until iSed has finished..
        sem2:wait()
        status = shm_hydroAS_variables:get(0)
        if (status == 2) then
          hydroas.Global.stopSimulation()
        end
        shm_z:copyZToHA()
      end
   end,

   writeResult = function()
   end,

   close = function()
      shm_hydroAS_variables:set(0, 1) --HydroAS is finished
      T = hydroas.Global.timestepStart()
      --print("HydroAS finished...", T)

 copyvar(T)
      sem1:post() --Continue iSed after the time loop
      sem1:close()
      sem2:close()
      shm_sst:close()
      shm_v:close()
      shm_depth:close()
      shm_z:close()
      shm_dt:close()
      shm_hydroAS_variables:close()
   end
}
```

Modellkoppelung HydroAS - iSed

ised.lua

```
require("hydroas.LockUndeclared").lock(_G)
local hydroas = require("hydroas")

--- Semaphore & Shared-Memory ---
local sem1 = hydroas.Semaphore.newSemaphore("/HydroAS")
local sem2 = hydroas.Semaphore.newSemaphore("/iSed")

local shm_hydroAS_variables = hydroas.SharedMemory.openSysV(1233, hydroas.SharedMemory.TypeInt32, 2)
local shm_dt = hydroas.SharedMemory.openSysV(1234, hydroas.SharedMemory.TypeFloat32, 2)
local shm_z = hydroas.SharedMemory.openSysV(1235, hydroas.SharedMemory.TypeFloat32, hydroas.Node.count())
local shm_depth = hydroas.SharedMemory.openSysV(1236, hydroas.SharedMemory.TypeFloat32, hydroas.Node.count())
local shm_v = hydroas.SharedMemory.openSysV(1237, hydroas.SharedMemory.TypeFloat32, hydroas.Node.count())
local shm_sst = hydroas.SharedMemory.openSysV(1238, hydroas.SharedMemory.TypeFloat32, hydroas.Node.count())

local status, T, Tdef

local function copyvar(T)
  shm_v:copyVelocFromHA()
  shm_depth:copyDepthFromHA()
  shm_sst:copyShearstressFromHA()
  shm_dt:set(0, T)
  shm_hydroAS_variables:set(1, hydroas.Global.timestepIndex())
  return
end

if sem1:isOpen() and sem2:isOpen() then
   print("Semaphores are open.")

   if shm_hydroAS_variables:isOpen() and shm_dt:isOpen() and shm_z:isOpen() and shm_depth:isOpen() and shm_v:isOpen() and shm_sst:isOpen() then
      print("Shared memory segments are open.")
   shm_hydroAS_variables:set(0, 0) --HydroAS is running
   else
      error("Could not open all shared memory segments.")
   end
else
   error("Could not open semaphores.")
end
```

Shared – Memory den Variablen zuordnen

```
return {
        if (T >= Tdef) then
 copyvar(T)
        --Continue with iSed
        sem1:post()
        --Wait until iSed has finished..
        sem2:wait()
        status = shm_hydroAS_variables:get(0)
        if (status == 2) then
          hydroas.Global.stopSimulation()
        end
        shm_z:copyZToHA()
      end
   end,

   writeResult = function()
   end,

   close = function()
      shm_hydroAS_variables:set(0, 1) --HydroAS is finished
      T = hydroas.Global.timestepStart()
      --print("HydroAS finished...", T)

 copyvar(T)
      sem1:post() --Continue iSed after the time loop
      sem1:close()
      sem2:close()
      shm_sst:close()
      shm_v:close()
      shm_depth:close()
      shm_z:close()
      shm_dt:close()
      shm_hydroAS_variables:close()
   end
}
```

Modellkoppelung HydroAS - iSed

ised.lua

```
require("hydroas.LockUndeclared").lock(_G)
local hydroas = require("hydroas")

--- Semaphore & Shared-Memory ---
local sem1 = hydroas.Semaphore.newSemaphore("/HydroAS")
local sem2 = hydroas.Semaphore.newSemaphore("/iSed")

local shm_hydroAS_variables = hydroas.SharedMemory.openSysV(1233, hydroas.SharedMemory.TypeInt32, 2)
local shm_dt = hydroas.SharedMemory.openSysV(1234, hydroas.SharedMemory.TypeFloat32, 2)
local shm_z = hydroas.SharedMemory.openSysV(1235, hydroas.SharedMemory.TypeFloat32, hydroas.Node.count())
local shm_depth = hydroas.SharedMemory.openSysV(1236, hydroas.SharedMemory.TypeFloat32, hydroas.Node.count())
local shm_v = hydroas.SharedMemory.openSysV(1237, hydroas.SharedMemory.TypeFloat32, hydroas.Node.count())
local shm_sst = hydroas.SharedMemory.openSysV(1238, hydroas.SharedMemory.TypeFloat32, hydroas.Node.count())

local status, T, Tdef

local function copyvar(T)
  shm_v:copyVelocFromHA()
  shm_depth:copyDepthFromHA()
  shm_sst:copyShearstressFromHA()
  shm_dt:set(0, T)
  shm_hydroAS_variables:set(1, hydroas.Global.timestepIndex())
  return
end

if sem1:isOpen() and sem2:isOpen() then
   print("Semaphores are open.")

   if shm_hydroAS_variables:isOpen() and shm_dt:isOpen() and shm_z:isOpen() and shm_depth:isOpen() and shm_v:isOpen() and shm_sst:isOpen() then
      print("Shared memory segments are open.")
   shm_hydroAS_variables:set(0, 0) --HydroAS is running
   else
      error("Could not open all shared memory segments.")
   end
else
   error("Could not open semaphores.")
end
```

Daten von HydroAS auf Shared – Memory kopieren

```
return {
   step = function()
      Tdef = shm_dt:get(1)
      T = hydroas.Global.timestepStart()
      --print("HydroAS calculated...", T)

      if (T >= Tdef) then
 copyvar(T)
        --Continue with iSed
        sem1:post()
        --Wait until iSed has finished..
        sem2:wait()
        status = shm_hydroAS_variables:get(0)
        if (status == 2) then
          hydroas.Global.stopSimulation()
   writeResult = function()
   end,

   close = function()
      shm_hydroAS_variables:set(0, 1) --HydroAS is finished
      T = hydroas.Global.timestepStart()
      --print("HydroAS finished...", T)

 copyvar(T)
      sem1:post() --Continue iSed after the time loop
      sem1:close()
      sem2:close()
      shm_sst:close()
      shm_v:close()
      shm_depth:close()
      shm_z:close()
      shm_dt:close()
      shm_hydroAS_variables:close()
   end
}
```

Modellkoppelung HydroAS - iSed

ised.lua

```
require("hydroas.LockUndeclared").lock(_G)
local hydroas = require("hydroas")

--- Semaphore & Shared-Memory ---
local sem1 = hydroas.Semaphore.newSemaphore("/HydroAS")
local sem2 = hydroas.Semaphore.newSemaphore("/iSed")

local shm_hydroAS_variables = hydroas.SharedMemory.openSysV(12
local shm_dt = hydroas.SharedMemory.openSysV(1234, hydroas.Sha
local shm_z = hydroas.SharedMemory.openSysV(1235, hydroas.Shar
local shm_depth = hydroas.SharedMemory.openSysV(1236, hydroas.
local shm_v = hydroas.SharedMemory.openSysV(1237, hydroas.Shar
local shm_sst = hydroas.SharedMemory.openSysV(1238, hydroas.Sha

local status, T, Tdef

local function copyvar(T)
  shm_v:copyVelocFromHA()
  shm_depth:copyDepthFromHA()
  shm_sst:copyShearstressFromHA()
  shm_dt:set(0, T)
  shm_hydroAS_variables:set(1, hydroas.Global.timestepIndex())
  return
end

if sem1:isOpen() and sem2:isOpen() then
   print("Semaphores are open.")

   if shm_hydroAS_variables:isOpen() and shm_dt:isOpen() and shm_z:isOpen() and shm_depth:isOpen() and shm_v:isOpen() and shm_sst:isOpen() then
      print("Shared memory segments are open.")
   shm_hydroAS_variables:set(0, 0) --HydroAS is running
   else
      error("Could not open all shared memory segments.")
   end
else
   error("Could not open semaphores.")
end
```

- Abgleich der Zeitschritte von HydroAS und iSed
- iSed startet erst, wenn die Summe der benutzerdefinierten Zeitschrittweite von HydroAS die benutzerdefinierte Zeitschrittweitweite von iSed erreicht

```
return {
   step = function()
      Tdef = shm_dt:get(1)
      T = hydroas.Global.timestepStart()
      --print("HydroAS calculated...", T)

      if (T >= Tdef) then
 copyvar(T)
        --Continue with iSed
        sem1:post()
        --Wait until iSed has finished..
        sem2:wait()
        status = shm_hydroAS_variables:get(0)
        if (status == 2) then
          hydroas.Global.stopSimulation()
        end
        shm_z:copyZToHA()
      end
   end,

   writeResult = function()
   end,

   close = function()
      shm_hydroAS_variables:set(0, 1) --HydroAS is finished
      T = hydroas.Global.timestepStart()
      --print("HydroAS finished...", T)

 copyvar(T)
      sem1:post() --Continue iSed after the time loop
      sem1:close()
      sem2:close()
      shm_sst:close()
      shm_v:close()
      shm_depth:close()
      shm_z:close()
      shm_dt:close()
      shm_hydroAS_variables:close()
   end
}
```

Modellkoppelung HydroAS - iSed

ised.lua

```
require("hydroas.LockUndeclared").lock(_G)
local hydroas = require("hydroas")

--- Semaphore & Shared-Memory ---
local sem1 = hydroas.Semaphore.newSemaphore("/HydroAS")
local sem2 = hydroas.Semaphore.newSemaphore("/iSed")

local shm_hydroAS_variables = hydroas.SharedMemory.openSysV(1233, hydroas.SharedMemory.TypeInt32, 2)
local shm_dt = hydroas.SharedMemory.openSysV(1234, hydroas.SharedMemory.TypeFloat32, 2)
local shm_z = hydroas.SharedMemory.openSysV(1235, hydroas.SharedMemory.TypeFloat32, hydroas.Node.count())
local shm_depth = hydroas.SharedMemory.openSysV(1236, hydroas.SharedMemory.TypeFloat32, hydroas.Node.count())
local shm_v = hydroas.SharedMemory.openSysV(1237, hydroas.SharedMemory.TypeFloat32, hydroas.Node.count())
local shm_sst = hydroas.SharedMemory.openSysV(1238, hydroas.SharedMemory.TypeFloat32, hydroas.Node.count())

local status, T, Tdef

local function copyvar(T)
  shm_v:copyVelocFromHA()
  shm_depth:copyDepthFromHA()
  shm_sst:copyShearstressFromHA()
  shm_dt:set(0, T)
  shm_hydroAS_variables:set(1, hydroas.Global.timestepIndex())
  return
end

if sem1:isOpen() and sem2:isOpen() then
   print("Semaphores are open.")

   if shm_hydroAS_variables:isOpen() and shm_dt:isOpen() and shm_z:isOpen() and shm_depth:isOpen() and shm_v:isOpen() and shm_sst:isOpen() then
      print("Shared memory segments are open.")
   shm_hydroAS_variables:set(0, 0) --HydroAS is running
   else
      error("Could not open all shared memory segments.")
   end
else
   error("Could not open semaphores.")
end
```

- An HydroAS werden die neuen Sohlhöhen übergeben

```
return {
   step = function()
      Tdef = shm_dt:get(1)
      T = hydroas.Global.timestepStart()
      --print("HydroAS calculated...", T)

      if (T >= Tdef) then
 copyvar(T)
        --Continue with iSed
        sem1:post()
        --Wait until iSed has finished..
        sem2:wait()
        status = shm_hydroAS_variables:get(0)
        if (status == 2) then
          hydroas.Global.stopSimulation()
        end
        shm_z:copyZToHA()
      end
   end,

   writeResult = function()
   end,

   close = function()
      shm_hydroAS_variables:set(0, 1) --HydroAS is finished
      T = hydroas.Global.timestepStart()
      --print("HydroAS finished...", T)

 copyvar(T)
      sem1:post() --Continue iSed after the time loop
      sem1:close()
      sem2:close()
      shm_sst:close()
      shm_v:close()
      shm_depth:close()
      shm_z:close()
      shm_dt:close()
      shm_hydroAS_variables:close()
   end
}
```

BOKU
UNIVERSITY

Vielen Dank für Ihre Aufmerksamkeit!

Assoc. Prof. Dipl.-Ing. Dr. techn. Michael Tritthart
Institut für Wasserbau, Hydraulik und Fließgewässerforschung (IWA)
Department für Wasser-Atmosphäre-Umwelt (WAU)

T +43 1 47654-81910
michael.tritthart@boku.ac.at

BOKU University
Am Brigittenauer Sporn 3, 1200 Wien
boku.ac.at