**Talked with Nick 12/20/13**

**About turner cut:**

1. **Compare dsm2 results with real good data to see if the phase problem still exist**
2. **If problem exist, compare model results with the other model**
3. **If difference still there, find solution.**

**About velocity profiles:**

1. **Get field data**
2. **Verify the model**
3. **Using model results to establish velocity profile, correlation between point velocities and flow and other factors**
4. **Implement the velocity profiles to ptm**
5. **Verify the ptm with the real results vs. other model results.**

**Useful utilities:**

**Convert HECTime to calendar time** PTMHydroInput.convertHecTime

Convert internal node number to real node number PTMHydroInput.getNodeIntFromExt

Convert internal node number to real node number PTMHydroInput.getIntFromExt

Convert real node number to internal node number PTMHydroInput.getIntFromExt

**Loop over time period first and then all particles**

**Add a Particle type interface and in the helper add method**

**Velocity is constant for the sub-time step, only updated for the dsm2 time step.**

**Velocity is inteperated linearly from up to down node.**

**MainPTM line 64 (line 62 set Globals.environment) set environmental variables, insert read node look up table**

**Current Barrier Settings:**

1. Initialize Barriers info in PTMEnv (line 58):

In PTMHydroInput (line 79)

1. Read in file (barrier IDs and operate info)
2. create a list \_barriers
3. create method PTMHydroInput.getBarriers()
4. install barriers to water body in PTMEnv (line113 setWaterbodyInfo()):

The code assumes that all channels are at the beginning of the water body array

Make sure this is true. This is not good coding practice need to be rewritten

for every waterbody that loop through, check to see if barriers != null, if yes:

1. convert the internal waterbody Id to external (Channel EnvIndexId) using PTMHydroInput.getExtfromInt()
2. Iterate through the barriers list to find a match.
3. If find a match in the list, install barrier in the channel

install barriers to node in PTMEnv (line 478 setNodeInfo()):

same as above

1. Update Barrier Operation info:
2. Start from: In MainPTM (line 180): PTMEnv.getHydroInfo(currentTime)
3. obtain hydroinfo:

In PTMEnv (line 521): PTMHydroInput.getNextChunk(currentTime)

In PTMHydroInput(line 171): setCurrentBarrierOp(currentTime) to set current operation inside of NonPhysicalBarrier objects

1. Update hydroinfo in channels:

In PTMEnv(line 522) hydroInput.updateWaterBodiesHydroInfo(…)

**Current Particle Behavior settings:**

1. Create HelperHandlers in MainPTM (line 129) and install to particles
2. Water body type could be channel, reservoir, boundary, conveyor PTMFixedInput(line180)

**how ptm read in data from file:**

1. PTMEnv line 55 instantiate PTMFixedInput
2. PTMFixedInput line48 instantiate FTMfixedData(inputFileName)
3. PTMFixedData (String) line 55:using native functions to read in file
4. PTMFixedData line 89 getParticleFixedData(): instantiate and set variables in ParicleFixedData
5. PTMFixedInput line 51 get ParticlefixedData: get particle info back to PTMFixedInput
6. ParticleFixedInfo line 358 PTMFixedInput.getPTMFixedInfo(…) instantiate and set ParticleFixedInfo PaticlefixedInfo can be accessed by package
7. PTMEnv line 104 set and get info from ParticleFixedInfo and set particleType
8. PTMEnv line 446 getParticleType()

**How PTM index system works**

It assumes that all node, waterbody, and xsection numbering systems are continuous from 1-maxmum number. I guess this is what outputs in dss have. Therefore all translations from real node# and channel# to coded ones are done in native methods.

**PTM Trace File**

1. Loop through time peiod -> loop through number of particles ->updatePostion for every particle (MainPTM line 235)
2. In every particle only when insert a particle and make node decision the ParticleObserver.observeChange() (Particle line 844, 727, 737))called
3. Then ParticleObserver.observeChange (ParticleObserver line 66) call specific observe change function but observeNodeChange() doesn’t do anything. observeDeath has never been used.
4. All observe change functions (except observe node change) write particle#, node#, wb# and entry time to trace file via PTMTraceOutput
5. Write to Trace file: ParticleObserver->PTMTraceOutput (line 84)
6. When both loops (time period and particles) are done, FluxMonitor reads in the trace file: FluxMonitor.calcuateFlux()-> FluxMonitor.createTraceArray() ->PTMTraceInput (line 118)
7. Traces are read in traceArray with particle number as index and ParticleTrace as element. Every element hold entire trace of the particle, which include vectors of wb#, node# and entry times.
8. FluxMonitor.calcuateFlux()->NodeFlux.calculateFlux (line 94):

For each particle:

For each time step:

Loop over the particle trace

If trace time matches current time step

If trace node matches input node

If in/out waterbodies matches input in/out waterbodies

Add to particle flux

If in/out waterbodies reverses (in==out, out==in)

Minus to particle flux

Else nothing

add particle flux into Flux[time] when finishes the particle loop

calculate the flux: in NodeFlux.caculateFlux line 75, actual calculation: line 94. The direction of flux is defined in the input file. If a particle leaves then comes back, the particle from be removed for the flux counting.

The trace file:

Time particle# node# wb#

1:00 3 1 2

1:00 2 1 2

…

1:05 1 1 2

…

1:10 2 1 1

FluxMonitor initialize all the particle trace objects in createTraceArray(..) with and stored in traceArray.

**Particle class: insertion time is set by setInsertionInfo in line 313**

**There is varable insertionTime**

Age =+ Time to advance (Particle line 462).