White Paper

CRiSP 1.7 Model Working Document:

Changes to and Information on the Juvenile Salmon Downstream Passage Model Since Version 1.6

David H. Salinger

Columbia Basin Research

School of Aquatic and Fishery Sciences

University of Washington

Box 358218

Seattle, WA 98195

salinger@cbr.washington.edu

September 15, 2004

Introduction

This document keeps track of notable problems or improvements in the Crisp1.7 model as well as information on project editing. Eventually, this information will make it to the manual.

Bypass/transport round-off bug (1/2002)

This feature arises when a large percentage of fish that enter the bypass are subsequently routed to transport.   With only a small portion of the fish left to exit the bypass, accumulated round-off, from having this happen over many timesteps, can throw off the expected bypass survival rate.

As an example: 1653190 fish enter Lower Monumental Dam and 774934.44 of these are routed to bypass. Of these, 774893.5 are subsequently routed to transport leaving only 40.94 fish to pass through the bypass system.   Crisp output reports that 39.06/40.94 = 0.954079 fish survived the bypass. But the bypass survival was set at 0.95 – meaning that only 38.89 fish should have survived the bypass. This means that 774934.61 fish exited the bypass, even though only 774934.44 entered, or an extra 0.17 fish.   (Note: the reported transport, spillway and turbine survivors are also off by a similar number of fish, but since there are .5 – 1.2 million fish passing via these routes in this example, the error is inconsequential.) The issue is only that “something” appeared to be wrong when 39.06 survived the bypass rather than the 38.89 expected.

Insufficient flow warning bug (12/2004)

The issue is a warning that looks like:

```
   Warning: insufficient flow (2.40) on day 15 at N F Clearwater;
    … adjusting (old loss:  -1.15,  new loss:  -1.15,  flow min:  3.56)
```

It appears that the new loss should be -1.16 (2.40 - 3.56) and that on printing a new .dat file, the issue should self-correct. The problem results from: flow_min is specified as

3.555 in the columbia.desc. The new loss is actually calculated properly as -1.155, but is then truncated as -1.55 on both the warning message and on printing a new .dat file – making it appear that the new loss is the same as the old loss and that there should not be a warning.

To prevent or remove this problem, specify the flow_min in columbia.desc using only two decimal digits. This seemed a preferable fix to changing the code. It is likely that similar issues can arise from specifying 3 digits for environmental variables in either columbia.desc or base.dat files.

New Parameters

Growth Related Parameters (8/2002)

length_weight_b0	-used to convert between length and weight - non-zero if growth is desired. (Set once for the model.)
length_weight_b1	-used to convert between length and weight - non-zero if growth is desired. (Set once for the model.)
prey_energy_density	-non-zero if growth is desired. (Set once for the model.) This parameter is the flag: 0 indicates no growth modeling.
pprime_a, pprime_b	-regression parameters for computing P'. - Not both zero if growth is desired. (Set by species)

Output Parameter

output_settings	-sets output for summary.dat. =1024 for mean fish length at that river feature. OR =1025 for fish length AND passage info (see Susannah's cheat sheet for more info). (Set for each river feature.) =2048 for turbidity output [reach, dam, headwater] - prints daily turbidity for each river feature. (added 5/2003)

X-T model parameters

time_coef, distance_coef - (Set by Species.)

If time_coef = 1 and distance_coef = 0, then the X-T model is NOT being used. If either of the parameters is set otherwise, than the X-T model is being used.

These parameters are set at the species level only (and can not be set at the stock level).

Water temperature specification parameters (5/2003)

input_temp [set by reach or dam].

Usage: input_temp Off (is the default if no token).

input_temp On …followed by an array of 366 daily temperature values; replaces the modeled temperature values at that river feature, and propagates those temperatures downstream (via the old temperature model) unless the following feature has input_temp On. There is no GUI for this parameter, though water temps can be viewed at all features. The water temperature at the headwater is still set via token water_temp with its array of (366) daily temperature values.

Turbidity specification parameters (5/2003)

input_turbidity [set by reach, dam or headwater].

Usage: input_ turbidity Off (is the default if no token).

input_ turbidity On …followed by an array of 366 daily temperature values; replaces the modeled temperature values at that river feature, and propagates those temperatures downstream (via a copy of the temperature model) unless the following feature has input_turbidity On. It is not necessary to have headwater turbidities (default will be 0), as long as turbidity is specified upstream (and upfork) of any modeling having to do with turbidity.

At this point, turbidity is not used anywhere in the model (eg. for predation)

Reach Classes

Each reach is designated to belong to a particular reach class. Many parameters and equations now need to be designated by reach class. All species and stock -level parameters pertaining to reaches (not dams) can be designated by reach class. Many species and stock -level equations can be designated by reach class (not, e.g. the Flush equations or the dam delay or gas mort equations which pertain to dams, not reaches).

Reach classes are defined (and numbered successively) at the top of the .dat file as follows:

```
num_reach_classes 2
      reach_class_name 0 Class_0
      reach_class_name 1 lgr_res
```

Parameters in reach class number 0 (named “Class_0” in this case) are taken as the default parameters for all other reach classes. In this way, parameters in the other reach classes need only be defined if they are different from those in the default reach class.

Command Line Flags (8/2002)

In addition to all the old command line flags, there are two new flags in Crisp1.7:
“-t” indicates travel time calibration mode. In this mode, what was formerly called “Zabel’s Pirate Crisp” is invoked… this recomputes travel rather than using the (faster but discontinuous) travel tables.
“-a” followed by a file name indicates that this file name is the Adult Model .config file. This is never to be used when running the Crisp Juvenile model, but is needed for the adult model which calls the old Crisp Code to manage environmental conditions and run-time flags including batch mode etc.

Also:
“-g” gives full giu mode. (This is not new, just re-discovered 4/03)

Notes on Project Editing

On PC

The source code on my PC is in directory:

C:\Documents and Settings\salinger\Workshop\crisp1\Src

with “original” copies at

C:\Documents and Settings\salinger\Workshop\crisp1\src_original

Editing is done in ..\Src sirecotry using Microsoft Visual C++. Tab “Build” – “set active configuration” lets you choose between debug mode and release mode. When editing and testing is done, use WinDiff to compare Src and src_original directories to confirm that expected changes (in Src) were the only ones made.

ON UNIX

The source code resides in /proj/crisp/src under SCCS control. The files in the directory are read-only. To get a writable copy, one must check it out from SCCS using the command:

sccs edit *fname*

Other useful SCCS commands:

sccs delget *fname* (checks in edited version to SCCS and replaces read-only copy in src directory).

sccs unedit *fname* (throws away edited copy, retaining previous version in SCCS).

sccs diffs *fname* (compares edited version SCCS)

Build (compile) the code on UNIX in /proj/crisp/z_obj (see step 5 below). The executable files, `crisp1.execuatable` and `crisp1bat`, are created in /proj/crisp/z_obj.

Editing:

1. Development and testing on PC… build final executable.
2. WinDiff Src and src_original directories to confirm changes.

3. On UNIX, check out (from SCCS control) files needing change in /proj/crisp/src using: sccs edit *fname*
4. Transfer changed files from PC or make changes on UNIX using SSH.
5. Build (compile) on UNIX in /proj/crisp/z_obj using “make debug” or “make optimize” command. (Since I do most debugging on PC, I usually build here using “make optimize”).
6. Testing. I usually compare output from PC and UNIX runs to be sure changes worked consistently.
7. Check in new src files in /proj/crisp/src using: sccs delget *fname*. The SCCS program lets you make comments about code changes at this point.
8. Load new master files from in /proj/crisp/src to \src_original on PC (and WinDiff Src and src_original directories to confirm).