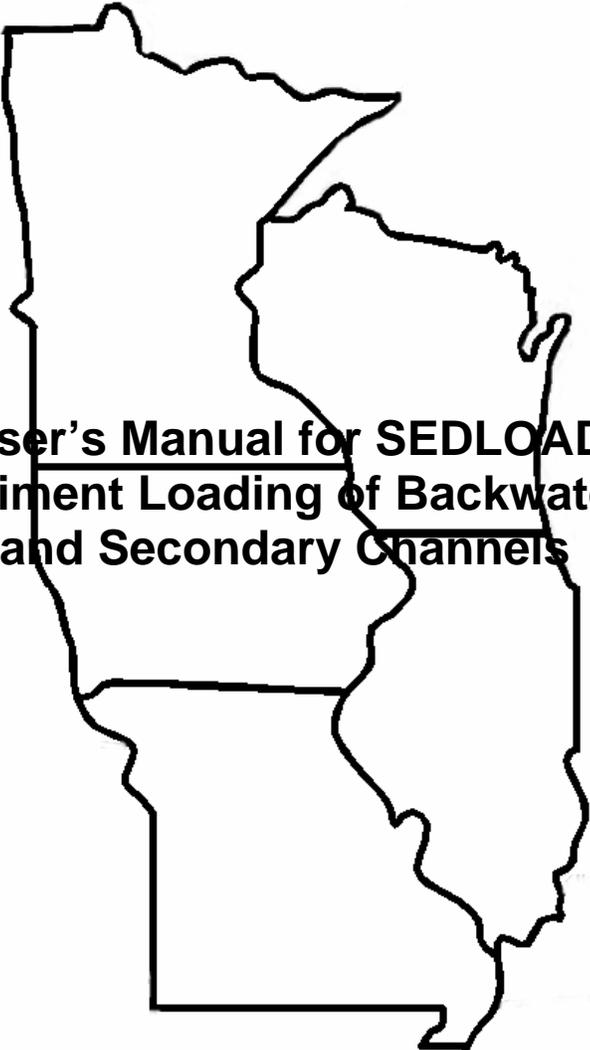
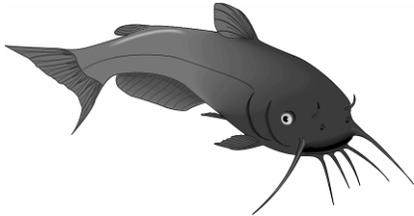
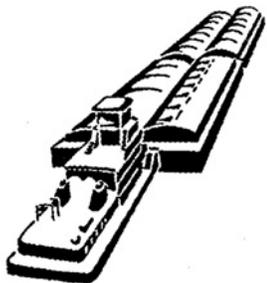


Interim Report For The Upper Mississippi River – Illinois Waterway System Navigation Study



**User's Manual for SEDLOAD
Sediment Loading of Backwaters
and Secondary Channels**



**US Army Corps
of Engineers**

August 2004

Rock Island District
St. Louis District
St. Paul District

User's Manual for SEDLOAD Sediment Loading of Backwaters and Secondary Channels

Clay LaHatte , Stephen T. Maynard

*Coastal and Hydraulics Laboratory
U.S. Army Engineer Research and Development Center
3909 Halls Ferry Road
Vicksburg, MS 39180-6199*

Interim report

Approved for public release; distribution is unlimited.

Prepared for U.S. Army Engineer District, Rock Island
Rock Island, IL 61204-2004
U.S. Army Engineer District, St. Louis
St. Louis, MO 63103-2833
U.S. Army Engineer District, St. Paul
St. Paul, MN 55101-1638

ABSTRACT:

The SEDLOAD package is a group of programs designed to determine the sediment load delivered to the inlets of backwaters and secondary channels of a river due to river traffic, specifically tow traffic. This user's manual describes how to set up batch files, input and output file formats, example outputs, and source code listings in order to run the program.

DISCLAIMER: The contents of this report are not to be used for advertising, publication, or promotional purposes. Citation of trade names does not constitute an official endorsement or approval of the use of such commercial products. All product names and trademarks cited are the property of their respective owners. The findings of this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Contents

Preface	v
1—Overview	1
General	1
SEDLOAD Components	1
Data Flow Chart	2
System Requirements	4
Disk Space Requirements	4
2—Installation	5
Step By Step	5
Set Up Directories	7
Project Directory	7
Data Directories	7
Backwater and Secondary Channel Specific Subdirectories	8
Programs	8
Input Data Files	9
Overview	9
Separate Backwater and Secondary Channel Data	9
3—Executing the Programs	11
Types of Runs	11
Full Run	11
Traffic Scenario Run	11
Making a Full Run	11
Making a Traffic Scenario Run	12
4—Output	13
Final Results	13
Appendix A—File Formats	1
General	1
Pxx_TOWNNUMBER.DAT	1
Pxx_TRAFMNTH.DAT	1
Pxx_KLCONC.DAT	2
Pxx_BWSECAREA.DAT	3
Pxx_BWLENGTH.DAT	3
PxxBWSEC.TXT	4

Appendix B—VOLUME Program.....	1
Files.....	1
Input	1
Output.....	1
Source Code.....	2
Appendix C—SEDMASS Program	1
Files.....	1
Input	1
Output.....	1
Source Code.....	2
Appendix D—BWMASSPROB Program	1
Files.....	1
Input	1
Output.....	1
Source Code.....	2
Appendix E – SED2BW Program	1
Files.....	1
Input	1
Output.....	1
Source Code.....	2

SF 298

Preface

The work reported herein was conducted as part of the Upper Mississippi River—Illinois Waterway (UMR-IWW) System Navigation Study. The information generated for this interim effort will be considered as part of the plan formulation process for the System Navigation Study.

The UMR-IWW System Navigation Study is being conducted by the U.S. Army Engineer Districts of Rock Island, St. Louis, and St. Paul under the authority of Section 216 of the Flood Control Act of 1970. Commercial navigation traffic is increasing, and in consideration of existing system lock constraints, will result in traffic delays that will continue to grow in the future. The system navigation study scope is to examine the feasibility of navigation improvements to the Upper Mississippi River and Illinois Waterway to reduce delays to commercial navigation traffic. The study will determine the location and appropriate sequencing of potential navigation improvements on the system, prioritizing the improvements for the 50-year planning horizon from 2000 through 2050. The final product of the System Navigation Study is a Feasibility Report, which is the decision document for processing to Congress.

This study was conducted in the Coastal and Hydraulics Laboratory (CHL), U.S. Army Engineer Research and Development Center (ERDC), Vicksburg, MS. The work was conducted under the direction of Mr. Thomas A. Richardson, Director, CHL. This report was written by Mr. W. Clay Lahatte and Dr. Stephen T. Maynard, CHL, ERDC.

At the time of publication of this report, Director of ERDC was Dr. James R. Houston. COL James R. Rowan was Commander, ERDC.

This manual was last modified on Tuesday, March 30, 2004.

1 Overview

General

The SEDLOAD package is a group of programs designed to determine the sediment load delivered to the inlets of backwaters and secondary channels of a river due to river traffic, specifically tow traffic. The program treats backwaters and secondary channels differently. Technical background for this user's manual is provided in Pokrefke et al. (2003).

The programs in the SEDLOAD package are run in series, meaning the results from one program are used by the next program.

The final result for a given backwater or secondary channel area will be expressed as

- a.* Volume delivered to the backwater or secondary channel through all of the inlets of that backwater or secondary channel (acre-ft/year).
- b.* Rate of accumulation averaged over the entire area of a backwater or secondary channel (cm/year). This assumes all sediment delivered to the backwater or secondary channel there.
- c.* Intensity of sediment delivery at an individual inlet (acre-ft/year/meter of inlet opening width).

For a given section of a river, called a "pool," an entire sediment loading scenario can be run by issuing one command. This automated method is accomplished using MS-DOS batch files to execute the various programs in the SEDLOAD package.

SEDLOAD Components

SEDLOAD is composed of four Fortran codes that are run sequentially. The first three codes are independent of the amount of traffic. Description of the codes follows.

- a.* **VOLUME.EXE** - Determines the vessel-induced exchange of water between the main channel and backwater or secondary channel as a result of the

drawdown created by the vessel. Program formulation is described in Appendix A of Pokrefke et al. (2003). The program outputs volume exchanged for all 108 tow types at each cell that represents a backwater or secondary channel for each of the nine combinations of stage and sailing line. Input and output files and source listing are provided in Appendix B.

b. **SEDMASS.EXE** - Uses as input the volume exchange caused by the tow drawdown from VOLUME.exe, the river flow into the backwater or secondary channel, and the sediment concentration at the edge of the main channel (from NAVSED). Determines the mass of sediment delivered to the backwater or secondary channel inlet per tow from the tow drawdown and from the river flow while the sediment concentration is elevated due to tow passage. Output is for all 108 tow types, each of the nine combinations of stage and sailing line, for each month of the navigation season. Input and output files and source listing are provided in Appendix C.

c. **BWMASSPROB.EXE** - Determines the sediment delivered to the inlet per tow in a probability of non-exceedance by a single tow ranging from 0.0 to 1.0 in increments of 0.1 based on a random sample of 5,000 tows. A probability of 0.0 represents the minimum value from the random sample of 5,000 tows. A probability of 1.0 represents the maximum value from the 5,000 random events. This step combines all 108 tow types and all nine combinations of stage and sailing line but keeps separate each month of the navigation season. Input and output files and source listing are provided in Appendix D.

d. **SED2BW.EXE** - The final code is the only step where the amount of traffic is an input. Based on a selected probability (such as 0.5 in UMR-IWW backwater/secondary channel studies), the sediment mass/tow for each month of the navigation season is multiplied by the number of tows for that month. The monthly sediment mass is summed to define the annual sediment mass delivered to the inlet. Input and output files and source listing are provided in Appendix E.

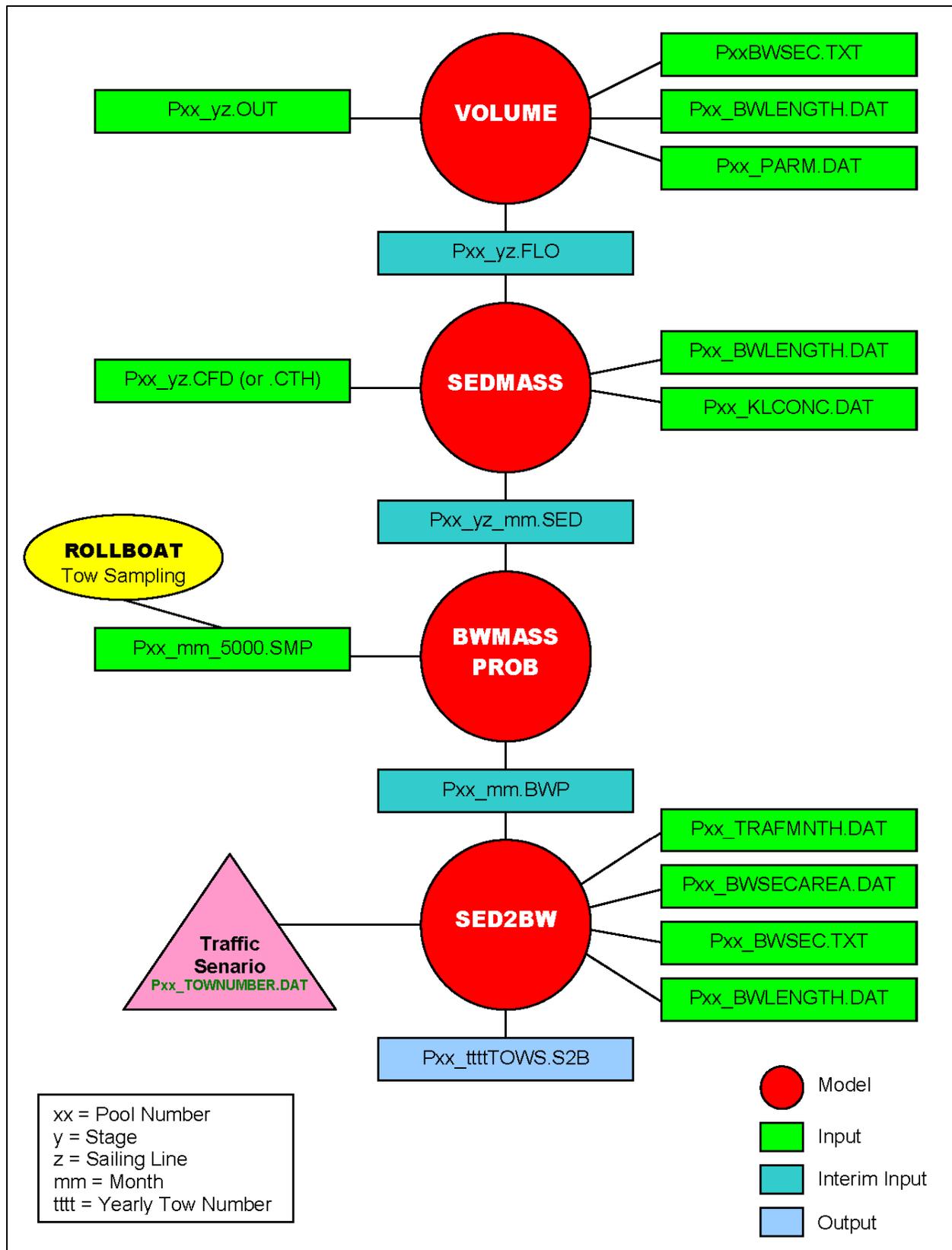
There are primarily two ways to run the SEDLOAD programs:

a. **Full Run** - This may be a new project or significant changes to an existing project. Prior to SEDLOAD, NAVEFF must be run to determine tow drawdown and NAVSED must be run to determine sediment concentration at the edge of the main channel. VOLUME.exe, SEDMASS.exe, BWMASSPROB.exe, and SED2BW.exe are then run sequentially.

b. **Traffic Scenario Run** - In cases where the fleet characteristics, sediment type, channel, and backwater geometry are established and don't change but different levels of traffic must be evaluated, only SED2BW.exe must be run.

Data Flow Chart

The flow of data through the SEDLOAD process is shown in the flow chart below.



System Requirements

► **FYI:** These programs have been tested only on a Windows-based personal computer.

No exhaustive testing has been done with these programs concerning system requirements. It is assumed you will need at least a Pentium or AMD processor-based computer with 128 MB RAM.

Disk space requirements

Because of the sizes of the files involved with the backwater loading modeling processes, a large amount of hard disk space will generally be required. Typically you may want to have at least three GB of free space available for an entire project, and more if it is a large project with many pools and backwater inlets.

2 Installation

Step By Step

Listed below are the steps required to prepare for a new project and a complete run of the SEDLOAD package. Details can be found in the sections to follow.

Set Up Directories:

For more information, see “Set Up Directories” on page 7.

- (1) Create a Project directory for the study.
- (2) In the Project directory, create a subdirectory for each Pool to be studied (**poolxx**).
- (3) In each Pool directory, create two subdirectories: **bw** and **sc**.

Install Programs:

For more information, see “Programs” on page 8.

- (4) Copy all SEDLOAD executable (.exe) and batch files (.bat) into the Project directory.

Install Pool Data Files:

- (5) Copy the nine NAVEFF output files (.out) for the **Pool** of interest to the matching **Pool** subdirectory under the Project directory. (These nine files are the combinations of three stages—low, medium, and high flow, and three sailing lines—left, middle, and right.)
- (6) Copy the nine NAVSED output files (.cfd or .cth) for the **Pool** of interest to the matching **Pool** subdirectory under the Project directory.
- (7) Copy the 12 (or less) monthly traffic rollup files (.smp) for the **Pool** of interest to the matching **Pool** subdirectory under the Project directory.

(8) Create a text file called **pxx_townumber.dat** in the Pool subdirectory. Edit this file in a text editor. The first two lines of this file are reserved for comments. The data must begin on the third line. On the third line, put an integer value specifying the total number of tows for the year. This number can be changed to model traffic conditions for past or future years. Follow the format indicated in Appendix A.

(9) Create a text file called **pxx_trafmnth.dat** in the Pool subdirectory. This file contains the percentage of tows to pass in this Pool for each month of the year. The actual number of monthly tows is calculated based on these values and the total tows for a year. Edit this file in a text editor. The first two lines of this file are reserved for comments. The data must begin on the third line. Follow the format indicated in Appendix A.

(10) Create a text file called **pxx_klconc.dat** in the Pool subdirectory. This file contains the ambient concentration data for this Pool. Edit this file in a text editor. The first two lines of this file are reserved for comments. The data must begin on the third line. Follow the format indicated in Appendix A.

Install backwater data files

(11) Create a text file called **pxx_bwsecarea.dat** in the **bw** subdirectory of the Pool subdirectory. This file contains the surface areas of the backwaters of interest in this Pool. Edit this file in a text editor. The first two lines of this file are reserved for comments. The data must begin on the third line. Follow the format indicated in Appendix A.

(12) Create a text file called **pxx_bwlength.dat** in the **bw** subdirectory of the Pool subdirectory. This file contains the lengths of the backwaters, as well as the sediment types. Edit this file in a text editor. The data must begin on the first line. Follow the format indicated in Appendix A. If this file already exists, be sure it contains only data related to backwaters (no secondary channel data).

(13) Copy the file **pxxbwsec.txt** (no underscore) to the **bw** subdirectory of the Pool subdirectory. This file is from the GIS. Edit this file and remove any secondary channel (SECxx) data, leaving only backwater data. This file contains backwater inlet data. Follow the format indicated in Appendix A. *Be sure it contains only backwater data.*

Install secondary channel files

(14) Create a text file called **pxx_bwsecarea.dat** in the **sc** subdirectory of the Pool subdirectory. This file contains the surface areas of the backwaters of interest in this Pool. Edit this file in a text editor. The first two lines of this file are reserved for comments. The data must begin on the third line. Follow the format indicated in Appendix A.

(15) Create a text file called **pxx_bwlength.dat** in the **sc** subdirectory of the Pool subdirectory. This file contains the lengths of the backwaters, as well

as the sediment types. Edit this file in a text editor. The data must begin on the first line. Follow the format indicated in Appendix A. If this file already exists, be sure it contains only data related to secondary channels (no backwater channel data).

(16) Copy the file `pxxbwsec.txt` (no underscore) to the `sc` subdirectory of the Pool subdirectory. This file is from the GIS. Edit this file and remove any backwater channel (BKW`xx`) data, leaving only secondary channel data. This file contains secondary channel inlet data. Follow the format indicated in Appendix A. *Be sure it contains only secondary channel data.*

After these steps are completed, you should be ready to make an automated run. See “Making a Full Run” on page 11.

Set Up Directories

To successfully run the SEDLOAD package, a specific directory structure must be used. A general outline showing the directory structure is shown below.

- Project Directory	Root of the project, executables are here.
- Pool 01	One of the pool data directories.
- bw	Subdirectory for backwater data for the pool.
- sc	Subdirectory for secondary channel data for the pool.
- Pool 02	Another pool data directory.
.	
.	... and so on, for all pools in this project.

► **Tip:** You can create the data directories as you install the input data files for the pools that you need.

Project directory

The project directory is the root of the project. All data associated with this project will be located in subdirectories of this project directory.

All SEDLOAD executable programs and batch files are to be stored in the project directory.

Example:

Project Directory:
C:\Windows\Desktop\Uppermiss

Data directories

The Data Directories will hold the pool data. Each pool will have its own data directory.

Example:

One or more Data Directories:

C:\Windows\Desktop\Uppermiss\pool~~xx~~

Where ~~xx~~ = pool number

Example:

C:\Windows\Desktop\Uppermiss\pool113

Backwater and secondary channel specific subdirectories

Each data directory also should contain a subdirectory for input data specific to backwaters (bw), and a subdirectory for input data specific to secondary channels (sc), in that pool.

Example:

C:\Windows\Desktop\Uppermiss\pool113\bw

C:\Windows\Desktop\Uppermiss\pool113\sc

These are separated because there can be cases in which the same cell number will apply to both a backwater and a secondary channel.

Programs

All the executable programs and batch files associated with the SEDLOAD package are to be installed in the project directory. The SEDLOAD package consists of these files:

Batch Files

GO_SEDLOAD.BAT
GO_SED2BW.BAT
GO_VOLUME.BAT
GO_SEDMASS.BAT
GO_BWMASSPROB.BAT

Executable Files

VOLUME.EXE
SEDMASS.EXE
BWMASSPROB.EXE
SED2BW.EXE

To install the program files:

- (1) Create a directory to contain the current project.
- (2) Copy all SEDLOAD executable (.exe) and batch files (.bat) into the project directory.

Input Data Files

Overview

You will need all the files described below in order to make a complete automated run.

Data Directories (Pools). These files should be in the pool data directories.

- NAVEFF output (.out).
- NAVSED output (.ctb or .cfd) (.cfd files result from assigning a fixed depth when running NAVSED).
- .SMP files (traffic rollup sampling files).
- Pxx_KLCONC.DAT (monthly ambient concentration data).
- Pxx_TRAFMNTH.DAT (percentage of traffic for each month).

Backwater (bw) and Secondary Channel (sc) Subdirectories. These files must be in the **bw** and **sc** subdirectories for the pool.

- PxxBWSEC.TXT (from GIS, file name contains no underscore).
- Pxx_BWLENGTH.DAT (contains the sediment type information).
- Pxx_PARM.DAT (tow characteristics).
- Pxx_BWSECAREA.DAT (surface area in acres of each backwater).

Separate backwater and secondary channel data

The backwater and secondary channel input data must be kept separate for the results to be accurate. You should make sure the backwater data is in its own input files in the **bw** subdirectory, and secondary channel data is in its own files in the **sc** subdirectory. The two files that might originally have both types combined are the following:

- Pxx_BWLENGTH.DAT. Backwater data contains **BW** in the last column. Secondary Channel data contains **SEC** in the last column.
- PxxBWSEC.TXT. Backwater data begins with **BWK**. Secondary Channel data begins with **SEC**

If not already separate, split these two files into four files, with the two backwater related files going into the **bw** subdirectory, and the two secondary channel related files going into the **sc** subdirectory.

► **Important:** Be sure this data is kept separate. To ensure reliable results, all the cell numbers in any one file should be unique.

3 Executing the Programs

Types of Runs

There are primarily two ways to run the SEDLOAD programs:

Full Run

This may be a new project or significant changes to an existing project. Prior to SEDLOAD, NAVEFF must be run to determine tow drawdown and NAVSED must be run to determine sediment concentration at the edge of the main channel (these programs are outside the scope of this document). VOLUME.exe, SEDMASS.exe, BWMASSPROB.exe, and SED2BW.exe are then run sequentially.

Traffic Scenario Run

In cases where the fleet characteristics, sediment type, and channel and backwater geometry are established and don't change, but different levels of traffic must be evaluated, only SED2BW.exe needs to be run.

Making a Full Run

The process for running the entire SEDLOAD package has been made fairly simple. Once all the input files are in the proper locations, starting a run involves only the following steps:

- (1) If in Windows, open an MS-DOS command session.
- (2) Navigate to the directory of the pool in which you are interested.
Example: `cd C:\Windows\Desktop\Uppermiss\pool13`
- (3) Start the processes by typing `..\go_sedload xx` where `xx` = the pool number. Example: `..\go_sedload 13`. This will execute the batch file `go_sedload.bat` located in the project directory, i.e., `c:\Windows\Desktop\Uppermiss`, and begin the SEDLOAD run.

Making a Traffic Scenario Run

If you need to make a change only to the yearly traffic number for the pool, you need to rerun only `SED2BW.EXE` to obtain the new results. The input files required to run only `SED2BW` are shown in “Data Flow Chart” on page 2. The new yearly tow number can either be input via the keyboard or read from the `Pxx_TOWNUMBER.DAT` file, depending on the choice made while answering the `SED2BW` prompts.

If all the previous steps of the `SEDLOAD` process already have been performed, do the following to make a new traffic scenario run:

- (1) If in Windows, open an MS-DOS command session.
- (2) Navigate to the backwater or secondary channel data directory of the pool in which you are interested. Example:

```
cd C:\Windows\Desktop\Uppermiss\pool113\bw
```
- (3) Run `SED2BW.EXE` by typing `..\..\sed2bw xx` where `xx` = the pool number. Example: `..\..\sed2bw 13`

The results are stored in the backwater (`bw`) or secondary channel (`sc`) subdirectory in a file called `Pxx_ttttTOWS.DAT`, where `xx` is the pool number and `tttt` is the yearly number of tows used for the run.

4 Output

Final Results

The final output of SED2BW consists of a file called **p_{xx}_t_{ttt}tows.s2b** written in both the **bw** and **sc** subdirectories of the pool subdirectory, where *xx* is the pool and *ttt* is the yearly number of tows used. This file reports the backwater or secondary channel, and the corresponding worst case sediment loading based on the input traffic scenario.

The result for a backwater or secondary channel is given as a level of significance for the three loading types as follows:

Loading by...	BLUE Negligible Impact Potential	YELLOW Medium Impact Potential	RED High Impact Potential
Volume	< 1.0 acre-ft/year	≥ 0.1 acre-ft/year	No criteria
Rate	< 0.1 cm/year	≥ 0.1 cm/year < 1.0 cm/year	≥ 1.0 cm/year
Inlet Intensity	< 0.01 acre-ft/year/m	> 0.01 acre-ft/year/m	No criteria

The following are included in the output:

- The area in acres of the backwater or secondary channel
- The cell ID associated with the backwater or secondary channel
- Sediment type in the backwater/secondary channel
- Traffic Impact by Volume given as acre-ft/year, and as a severity color
- Traffic Impact by Rate given as cm/year, and as a severity color
- Traffic Impact by Unit Volume given as acre-ft/year/meter, and as a severity color
- The worst of the three traffic impact types is provided in summary for the backwater/secondary channel as a severity color.

Example Output

SEDIMENTS TO BACKWATER OR SECONDARY CHANNEL (SEE FEATURE), POOL13
 BASED ON 50 percent ROLLUP
 Total tons for year: 2361

FEATURE	AREA Acres	CELLID	SEDIMENT_TYPE	IMPACT_BY_VOLUME acre-ft/year	IMPACT_BY_RATE cm/year	IMPACT_BY_UNIT_VOLUME acre-ft/year/meter	WORST_CASE
BKW01_01	746	315L5560	Cohesive-Medium	.0001 BLUE	.0003 BLUE	.0000 BLUE	BLUE
BKW01_02	746	105L5555	Cohesive-Medium	.0071 BLUE	.0003 BLUE	.0001 BLUE	BLUE
BKW02_01	33	65L5520	Noncohesive	.00384 BLUE	.00355 BLUE	.0003 BLUE	BLUE
BKW04_01	876	115R5465	Noncohesive	.0000 BLUE	.0000 BLUE	.0000 BLUE	BLUE
BKW05_03	782	355L5400	Cohesive-Medium	.0000 BLUE	.01968 BLUE	.0000 BLUE	BLUE
BKW05_05	782	135L5390	Cohesive-Medium	.18336 BLUE	.01968 BLUE	.00075 BLUE	BLUE
BKW05_06	782	125L5390	Cohesive-Medium	.32091 BLUE	.01968 BLUE	.00137 BLUE	BLUE
BKW05_07	782	85L5385	Cohesive-Medium	.0068 BLUE	.01968 BLUE	.0000 BLUE	BLUE
BKW06_01	934	195R5410	Cohesive-Soft	.47150 BLUE	.01539 BLUE	.00112 BLUE	BLUE
BKW07_01	399	265L5370	Noncohesive	.0000 BLUE	.0000 BLUE	.0000 BLUE	BLUE
BKW07_03	399	395L5345	Noncohesive	.0000 BLUE	.0000 BLUE	.0000 BLUE	BLUE
BKW07_04	399	385L5345	Noncohesive	.0000 BLUE	.0000 BLUE	.0000 BLUE	BLUE
BKW07_05	399	545L5340	Noncohesive	.0000 BLUE	.0000 BLUE	.0000 BLUE	BLUE
BKW07_06	399	565L5335	Noncohesive	.0000 BLUE	.0000 BLUE	.0000 BLUE	BLUE
BKW09_01	127	235R5345	Cohesive-Medium	.00571 BLUE	.00137 BLUE	.00025 BLUE	BLUE
BKW10_01	1181	315R5335	Noncohesive	.0000 BLUE	.0000 BLUE	.0000 BLUE	BLUE
BKW10_04	1181	165R5310	Noncohesive	.0000 BLUE	.0000 BLUE	.0000 BLUE	BLUE
BKW10_05	1181	325R5305	Noncohesive	.0001 BLUE	.0000 BLUE	.0000 BLUE	BLUE
BKW10_07	1181	685R5300	Noncohesive	.0000 BLUE	.0000 BLUE	.0000 BLUE	BLUE
BKW11_01	1122	455L5330	Cohesive-Soft	.30954 BLUE	.44665 YELLOW	.00525 BLUE	YELLOW
BKW11_02	1122	445L5330	Cohesive-Soft	.01047 BLUE	.44665 YELLOW	.00024 BLUE	YELLOW
BKW11_03	1122	435L5330	Cohesive-Soft	.01314 BLUE	.44665 YELLOW	.00037 BLUE	YELLOW
BKW11_07	1122	95L5325	Cohesive-Soft	4.02252 YELLOW	.44665 YELLOW	.02174 YELLOW	YELLOW
BKW11_08	1122	475L5310	Cohesive-Soft	.33900 BLUE	.44665 YELLOW	.00030 BLUE	YELLOW
BKW11_09	1122	75L5300	Cohesive-Soft	.20700 BLUE	.44665 YELLOW	.00397 BLUE	YELLOW
BKW11_12	1122	15L5290	Cohesive-Soft	11.54011 YELLOW	.44665 YELLOW	.02410 YELLOW	YELLOW

Appendix A

File Formats

General

This section contains some of the file formats for files used as input to the various components of the SEDLOAD package. These file format descriptions are provided for reference in the event custom input files may need to be created.

Pxx_TOWNNUMBER.DAT

The number of tows passing through a given pool for the period of a year.

Used by: SED2BW

ASCII Text File, Free field. Data starts on the third line.

Line	Column	Type	Description
1, 2	Any	N/A	User Comments.
3	1	Integer	Number of tows for the year.

Example:

Total tows for year 2000 – Pool 13

Other comments

2306

Pxx_TRAFMNTH.DAT

Traffic percentage by month.

Used by: SED2BW

ASCII Text File, Free field. Data starts on the third line.

Line	Column	Type	Description
1, 2	Any	N/A	User Comments.
3-14	1	Real	Values for months 1 through 12, one per line, listed as the percentage of the total number of tows to pass for the particular month.

Example:

Traffic percentage by month - Pool13

Other comments

0.0

0.0

0.0761

0.1196

0.1313

0.1184

0.1353

0.1223

0.0979

0.1039

0.0849

0.0105

All these values should sum to 1.

► **Note:** 0 . 0 indicates no navigation for that month, usually due to winter shutdown.

Pxx_KLCONC.DAT

File contains ambient concentration values for 12 months.

Used by: SEDMASS

ASCII Text File, Free field. Data starts on the third line.

Line	Column	Type	Description
1, 2	Any	N/A	User Comments.
3-14	1	Character*2	Two-character number specifying the month.
3-14	2	Real	The ambient concentration for the month specified.

Example:

Ambient Concentration Values – Pool 13

Other comments

01 -1.0

02 -1.0

03 51.6

04 83.0

05 72.0

06 58.0
 07 48.5
 08 33.1
 09 34.3
 10 34.7
 11 30.1
 12 -1.0

► **Note:** -1 . 0 indicates no navigation for that month, usually due to winter shutdown.

Pxx_BWSECAREA.DAT

File contains surface areas in acres of the specified backwaters/secondary channels.

Used by: SED2BW

ASCII Text File, Free field. Data starts on the third line.

Line	Column	Type	Description
1, 2	Any	N/A	User Comments.
3-??	1	Character	The backwater/secondary channel, in the form of: BKWxx or SECxx, where xx is a two-digit number.
3-??	2	Integer	The surface area in acres of the backwater/secondary channel specified.

Example:

Backwater Areas – Pool 13
 Other comments
 BKW01 746
 BKW02 33
 BKW03 83
 BKW04 876
 BKW05 782
 BKW06 934
 BKW07 399
 BKW09 127
 BKW10 1181
 BKW11 1122

Pxx_BWLENGTH.DAT

File contains the lengths and sediment types of the backwaters/secondary channels.

Used by: VOLUME, SEDMASS, SED2BW

ASCII Text File, Free field. Data starts on the first line.

Line	Column	Type	Description
Any	1	Character	Cell ID at backwater/secondary channel inlet.
	2	Integer	Length in meters of the backwater/secondary channel.
	3	Integer	Sediment Type. Cohesive sediments = 2 Non-cohesive sediments = 3
	4	Integer	Sediment Class—applicable only to cohesive sediments. Soft = 1 Medium = 2 Hard = 3
	5	Character	Indicates the backwater/secondary channel number.

Example:

315L5560	1200	2	2	BW1
105L5555	400	2	2	BW1
65L5520	800	3	9999	BW2
455L5475	850	9999	9999	BW3
115R5465	1700	3	9999	BW4
345R5455	3000	9999	9999	BW4
355L5400	1600	2	2	BW5
135L5390	300	2	2	BW5

► **Note:** 9999 indicates no data or not applicable.

PxxBWSEC.TXT

File contains various types of data from the GIS.

Used by: VOLUME, SED2BW

Line 1 is for user comments. In the case of the example below, the meanings of the values are presented.

The data starts on line 2. The file contains these values, separated by commas:

- (1) Feature, in the form of BKWxx_yy or SECxx_yy, where xx = backwater/secondary channel number, and yy = inlet/outlet number.
- (2) Cell ID at backwater/secondary channel inlet
- (3) Actual river mile
- (4) Inlet width, meters
- (5) Low flow cross-sectional area of inlet, square meters
- (6) Medium flow cross-sectional area of inlet, square meters

- (7) High flow cross-sectional area of inlet, square meters
- (8) Low flow average inlet depth, meters
- (9) Medium flow average inlet depth, meters
- (10) High flow average inlet depth, meters
- (11) Low flow rate of inlet, cubic meters per second
- (12) Medium flow rate of inlet, cubic meters per second
- (13) High flow rate of inlet, cubic meters per second
- (14) Specifies whether the channel is an inlet or outlet. SEDLOAD deals only with inlets.

Example:

```

FEATURE, CELL_ID, ACTUAL_RM, WIDTH, L_XAREA, M_XAREA, H_XAREA, L_AVGDEP, M_AVGDEP, H_AVGDEP, L_FLOWRATE, M_FLOWRATE, H_FLOWRATE, IN_OUT
BKW01_01, 315L5560, 555.9, 91.0, 124.0, 193.3, 382.0, 1.3, 2.1, 4.2, 1.90, 6.07, -999.00, INLET
BKW01_02, 105L5555, 555.7, 121.5, 99.9, 192.5, 436.9, 0.8, 1.6, 3.6, 2.28, 6.71, -999.00, INLET
BKW01_03, 125L5555, 555.5, 116.7, 185.9, 274.9, 509.6, 1.6, 2.3, 4.4, 2.29, 5.62, -999.00, OUTLET
BKW01_04, 145L5500, 550.0, 378.0, 715.5, 957.5, 1602.7, 1.9, 2.5, 4.2, 4.72, 14.58, -999.00, OUTLET
BKW02_01, 65L5520, 552.1, 133.0, 150.6, 239.7, 483.0, 1.1, 1.8, 3.6, 2.83, 8.85, -999.00, INLET
BKW02_02, 155L5505, 550.5, 217.3, 183.6, 322.8, 700.4, 0.8, 1.5, 3.2, 3.89, 11.16, -999.00, OUTLET
BKW03_01, 445L5475, 547.4, 267.9, 30.3, 185.5, 601.9, 0.1, 0.7, 2.3, -999.00, -999.00, -999.00, INLET
BKW04_01, 115R5465, 546.2, 27.0, 24.3, 38.3, 77.8, 0.9, 1.4, 2.9, 0.58, 1.42, -999.00, INLET
BKW04_02, 345R5455, 545.4, 37.6, 64.2, 83.7, 137.5, 1.7, 2.2, 3.6, -999.00, -999.00, -999.00, INLET
BKW04_03, 335R5415, 541.3, 123.5, 162.0, 210.9, 361.5, 1.3, 1.7, 2.9, 2.62, 9.06, -999.00, OUTLET
BKW05_03, 355L5400, 540.1, 49.5, 14.5, 34.2, 88.5, 0.3, 0.7, 1.8, 0.16, 0.92, -999.00, INLET
BKW05_05, 135L5390, 539.1, 243.5, 728.5, 810.1, 1069.9, 3.0, 3.3, 4.4, 31.71, 80.76, -999.00, INLET

```

A value of -999.00 indicates the inlet is submerged at that flow condition.

► **Note:** The first line has been wrapped in the example. It should be one long line of text.

Appendix B

VOLUME Program

Files

Input

PxxBWSEC.TXT (from GIS)

Pxx_PARM.DAT (tow characteristics)

Pxx_yz.OUT (NAVEFF output file)

Pxx_BWLENGTH.DAT (length of the channels)

where

xx = Pool Number

y = Stage, z = Sailing Line

Output

Pxx_yz.flo

where

xx = Pool Number

y = Stage, z = Sailing Line

Brief Output Description.

Field	Description
1	Cell ID
2	Traffic scenario
3	Stage/sailing line
4	Wave period, in seconds
5	Inlet area, in square meters
6	Channel length, in meters
7	Water depth, in meters
8	Input drawdown, in meters

9	Modified drawdown at the inlet, in meters
10	Channel type (backwater or secondary channel)
11,13,15	Discharge for wave
12,14,16	Wave duration, in seconds
17	Low flow rate, cubic meters per second
18	Medium flow rate, cubic meters per second
19	High flow rate, cubic meters per second
20	Total volume, in cubic meters

Source Code

```

      program compute_flux
c
c The program reads the lengths of channels from chlength.dat,
c the cell_ids and dimensions of channels from the
c Backwater/Secondary-channel data file,
c and the drawdowns from NAVEFF output.
c
c The program then calls subroutines BACKWATER and SIDE_CHANNEL to
c compute the flow rate of water. It is assumed that the rate is constant
c for each cycle as a step function.
c
c The results units are m**3/sec

      character*2  pool, flowchan
      character*200 PATH, naveffpath

      real         rm
      character*1  dir, speed, size, draft, turbine, stage, sl_pos

      character*8  feature, id1, id2, id3, ch_type*2, cellid(1000)
      character*6  in_out
      character*256 lin

      real*8       q(3), time(3)
      real         n
      real*8       width
      real*8       ldepth, mdepth, hdepth
      real*8       lxarea, mxarea, hxarea
      real*8       lfrate, mfrate, hfrate
      real*8       drawdown, drawdownx

      character*5  traffic, stg_sl*2

      character*64 BWDAT, BWDATTMP, NAVOUTDAT, NOUTDATTMP, CHANLENDAT, OUTDAT

cccc ccccc ccccccc ccccccccccccccccccccccccccccccccccc

      write (*,*) 'This program requires files...'
      write (*,*) 'PxxBWSEC.TXT'
      write (*,*) 'Pxx_yz.OUT'
      write (*,*) 'CHLENGTH.DAT'
      write (*,*) '  where xx=Pool Number'
      write (*,*) '          y=Stage, z=Sailing line'
      write (*,*) '----'
      write (*,*) ' '

      n = 0.020  ! Default

      write(*,*) 'Enter the Pool Number (2 characters - Ex: 13):'
      read (*,*) pool
      write(*,*) 'Enter the Flow/Channel (2 characters - Ex: LR):'
      read (*,*) flowchan
*   write(*,*) 'Enter directory for reading data (. for current):'
*   write(*,*) '  **** DO NOT ADD TRAILING BACKSLASH ****'
*   read (*,*) PATH

```

```

*  PATH = 'c:\windows\desktop\uppermiss\pool'//trim(pool)//'\
PATH='

write(*,*) 'Enter the path for the NAVEFF output files.'
write(*,*) 'If the same as the default, enter 0 (zero)'
read(*,*) naveffpath

BWDAT=trim(PATH)//'p'//pool//'bwsec.txt'
BWDATTMP=trim(PATH)//'p'//pool//'bwsec.tmp'
if(naveffpath .eq. '0')then
  NAVOUTDAT=trim(PATH)//'p'//pool//'_ '//flowchan//'.out'
else
  NAVOUTDAT=trim(naveffpath)//'p'//pool//'_ '//flowchan//'.out'
end if
NOUDDATTMP=trim(PATH)//'VOLUME.TMP'
CHANLENDAT=trim(PATH)//'P'//pool//'_bwlength'//'.dat'
OUTDAT=trim(PATH)//'P'//pool//'_ '//flowchan//'.flo'

c  goto 12

      open(20,file=BWDAT,status='old')
      open(21,file=BWDATTMP,status='unknown')
      open(30,file=NAVOUTDAT,status='old')
      open(35,file=NOUDDATTMP,status='unknown')

write(*,*) 'Extracting useful information from ',NAVOUTDAT
read(20,*) lin ..... ! Skip first line of text.
nid = 0
do
  read(20,*,end=100) feature, id1, actual_rm, width,
&                    lxarea,mxarea,hxarea,
&                    ldepth,mdepth,hdepth,
&                    lfrate,mfrate,hfrate,
&                    in_out
  i_pos=scan(in_out,'I')
  if (i_pos .ne. 0) then
*   if (in_out .eq. 'INLET ') then
      if (lfrate.eq.-999.0 .and. mfrate.eq.-999.0
&.and. hfrate.eq.-999.0) then
        ! Ignore cells with no flow data.
      else
        nid = nid + 1
        cellid(nid) = id1
c       do i=1, nid
c         if (cellid(i) .eq. id1) then
c           num_occur(i) = num_occur(i) + 1
c         end if
c       end do
        write(21,21) feature, id1, actual_rm, width,
&                    ldepth,mdepth,hdepth,
&                    lxarea,mxarea,hxarea,
&                    lfrate,mfrate,hfrate,
&                    in_out
21      format(a8,',',a8,',',f6.2,',',f9.3,',',lx
&            ,f9.3,',',f9.3,',',f9.3,',',lx
&            ,f9.3,',',f9.3,',',f9.3,',',lx
&            ,f9.3,',',f9.3,',',f9.3,',',
&            ,a6)
        end if
      end if
    end do
100  write(*,*) nid, ' cells with INLETs used from ',BWDAT

! Write to temp file.
do
  read(30,*,end=299) rm,dir,speed,size,draft,
&                    turbine,stage,sl_pos,
&                    traffic, stg_sl,
&                    id1,depth,ret_vel,drawdown

  do j = 1, nid
    if (trim(id1).eq.trim(cellid(j))) then
      write(*,*) id1,' ', traffic,' ', stg_sl
      write(35,200) id1,traffic,stg_sl,drawdown
    end if
  end do
end do

```

```

        end if
      end do
    end do
c 202 format(a10,2x,a5,2x,a2,f10.3)
299 close(35)
    close(30)
    close(21)
    close(20)

12  open(10,file=CHANLENDAT,status='old')
    open(20,file=BWDATTMP,status='old')
    open(30,file=NOUDDATTMP,status='old')
    open(50,file=OUTDAT,status='unknown')

    i = 0
    DO
c 90  read(10,*,end=999) id1, ch_length ! , ch_type
    format(2x,a8,f10.1,3x,a2)

*    ch_length = ch_length*41000.0*2.54/100.0 ! Use Only if input file
is in inches.

    rewind (20)
    DO
      read(20,*,end=500) feature,id2, actual_rm, width,
&          ldepth,mdepth,hdepth,
&          lxarea,mxarea,hxarea,
&          lfrate,mfrate,hfrate,
&          in_out
95    format(a8,2x,a8,7f8.1,3f8.2,a8)

      if (trim(id1).eq.trim(id2)) then
        rewind (30)
        DO
200      read(30,200,end=501) id3, traffic, stg_sl,drawdown
        format(a8,4x,a5,2x,a2,f12.6)
        if (trim(id3).eq.trim(id1)) then
          speed = traffic(2:2)
          size = traffic(3:3)
          stage = stg_sl(1:1)

          call conv(j,speed,size,stage,
&          ldepth,mdepth,hdepth,
&          lxarea,mxarea,hxarea,
&          T,Area,H)

c          ~~~~~ Computing the volume of water ~~~~~

          Perimeter = width + 2.0*H
          R = Area/Perimeter

          drawdownx = drawdown..... ! Retain drawdown for writing      out

          if (lfrate.eq.0.0 .and. mfrate.eq.0.0
&          .and. hfrate.eq.0.0) then
            ch_type = 'BW'
            n = 0.020
            call backwater(drawdownx,H,Area,R,
&          ch_length,n,T,q,time,vsum)
          else
            ch_type = 'SC'
            n = 0.020
            call side_channel(drawdownx,H,Area,R,
&          ch_length,n,T,q,time,vsum)
          &
          end if

          write(50,300) id1,traffic,stg_sl,T,Area,ch_length,
&          H,drawdown,drawdownx,ch_type,
&          q(1),time(1),q(2),time(2),q(3),time(3),
&          lfrate,mfrate,hfrate,vsum
300      format(a8,2x,a5,2x,a2,3f10.2,
&          f10.3,2f10.4,3x,a2,2x,

```

```

&                3(f11.4,f12.2),3f11.2,2x,f11.3)
                end if
            END DO
501        end if
            END DO
500    i = i + 1
        write(*,*) 'Processed cell ', i
    END DO

999 close(50)
    close(30, status='delete')
    close(20, status='delete')

    stop
    end

```

cc

```

subroutine conv(j,speed,size,stage,
&                ldepth,mdepth,hdepth,
&                lxarea,mxarea,hxarea,
&                T,Area,H)

```

```

real            Lb

real*8          ldepth,mdepth,hdepth
real*8          lxarea,mxarea,hxarea
real*8          lfrate,mfrate,hfrate
character*1     speed,size,draft,stage

```

```

if(speed.eq.'F') Vb = 3.58
if(speed.eq.'M') Vb = 2.91
if(speed.eq.'S') Vb = 2.24

if(size.eq.'B') Lb = 297.18
if(size.eq.'M') Lb = 237.74
if(size.eq.'S') Lb = 178.31
if(size.eq.'L') Lb = 45.72

```

```

if(stage.eq.'H') then
    H = hdepth
    Area = hxarea
else if(stage.eq.'M') then
    H = mdepth
    Area = mxarea
else if(stage.eq.'L') then
    H = ldepth
    Area = lxarea
end if

```

```

T = 2.0 * Lb/Vb

```

```

return
end

```

cc

```

SUBROUTINE SIDE_CHANNEL(A0,H,AREA,R,L,n,T,q,time,vsum)

```

```

c
c The routine computes the volumetric flux (q) for a side channel
c due to the drawdown. Initial three cycles are used. In calculating
c the water volume only V1 & V4 are used.
c
c Inputs:
c   L   channel length
c   n   0.050
c   H   water depth

```



```

V4 = Area*A4*sqrt(g/H)*T/pi          ! Add volume using A4
CALL FRICTION(A4,A5,H,R,n,X)         ! friction loss
CALL ENT_LOSS(A5,A6,H)

A0 = A6

stuff=4.0*L/sqrt(g*H)
if(stuff.gt.400.0)then !After 1200 seconds, most activity is done.
  stuff=400.0
end if
time(i) = float(i)*stuff
q(i) = (V1 + V4) /stuff *(4.0*L/sqrt(g*H))

vsum=vsum+V1+V4

END DO

RETURN
END

```

Appendix C

SEDMASS Program

Files

Input

Pxx_yz.CTH (or CFD (fixed depth), currently used - 11/14/01) (NAVSED output file)

Pxx_BWLENGTH.DAT (lengths of channels)

Pxx_yz.flo (Output from VOLUME program)

Pxx_KLCONC.DAT (Monthly ambient concentration data)

where

xx = Pool Number

y = Stage, z = Sailing Line

Output

Pxx_yz_mm.SED

where

xx = Pool Number

y = Stage, z = Sailing Line

mm = month

Brief Output Description.

Field	Description
1	Cell ID
2, 3	Traffic scenario
4	Mass added due to tow drawdown, in kilograms
5	Mass added due to disturbed sediment added to the river from tow passing, in kilograms
6	Total mass added to the backwater/secondary channel, in kilograms
7	Total volume, in cubic meters

Source Code

```
program sedmass

c The program computes the mass of sediment due to boat passage given
c flows and concentration values for each cellid and tow configuration
c given.
c
c From TMPDATA:
c The program reads times and calculates time increments.
c The program reads concentration time histories in mg/l
c (concentration at each time) for appropriate cell_id's.
c
c From FLOWDATA:
c The program reads flows (q1, q2, and q3)
c at times (t1, t2, and t3), and river flows at low, medium,
c and high stages for each cell_id.
c
      common/com/timehist

character*10 PROGNAME

character*256 lin

character*2 pool, flowchan
character*255 PATH, navsedpath, navsedfileext, klconcpath

      real      reflq(3), refltime(3)
real timehist(1000)
integer deltaT(1000)
real conc(1000)

real*8 massriv, massboat !, mass_sc
      integer   noflow, nohflow
character  cnoflow*2, cnohflow*10

character*2 month, m

      real lfrate,mfrate,hfrate
      real q_river

c   values from FLOW file.
character .....cellid_f*9, last_cellid_f*9, traffic_f*5, stgsl_f*2
c   values from CONCENTRATION file.
character .....cellid_c*9, traffic_c*5, stgsl_c*2

character.....cell_c*18, cell_f*18 !cellid+traffic+stgsl

      character*2 ch_type

integer npts ! read from CONCENTRATION file.
Integer nlms, nrm

real klac

integer index0.....! array index of time zero in CONCENTRATION file.

Integer errfile1

real      flowlins, conclins, pcentcmp

character*1 ccc

character*64 TMPDATA, FLOWDATA, OUTFILE1, ERRFILE
character*64 CHANDATA, CTHDATA
integer UNITCONC, UNITFLOW, UNITOUT1, UNITERR

integer WRITE_AT

!number of reflections in secondary channel or backwater
integer REFLECTS
```

```

cccc ccccccc cccccccc cccccccccccccccccccccccccccccccccccccc
c      Define 'constant' values...

PROGRAM= 'SEDMASS'

      write (*,*) 'This program requires files...'
      write (*,*) 'Pxx_yz.CTH'
      write (*,*) 'CHLENGTH.DAT'
      write (*,*) ' where xx=Pool Number'
      write (*,*) '          y=Stage, z=Sailing line'
      write (*,*) ' '
      write (*,*) 'Output file is Pxx_yz.SED'
      write (*,*) '----'
      write (*,*) ' '

      write(*,*) 'Enter the Pool Number (2 characters - Ex: 13):'
      read (*,*) pool
      pool=trim(adjustr(pool))
      if(pool(1:1) .eq. '') then
        pool(1:1) = '0'
      endif
      write(*,*) 'Enter the Flow/Channel (2 characters - Ex: LR):'
      read (*,*) flowchan
      * write(*,*) 'Enter directory for reading data (. for current):'
      * write(*,*) ' **** DO NOT ADD TRAILING BACKSLASH ****'
      * read (*,*) PATH
      write(*,*) 'Enter the number corresponding to the month (2 chars)'
      read (*,*) month
      month=trim(adjustr(month))
      if(month(1:1) .eq. '') then
        month(1:1) = '0'
      endif

      * write(*,*) 'Enter the Kevin Landwehr ambient concentration:'
      * read (*,*) klac

      * PATH = 'c:\windows\desktop\uppermiss\pool'//trim(pool)//'\
      * PATH='

      write(*,*) 'Enter the path for the concentration file KLCONC.DAT'
      write(*,*) 'If the same as the default, enter a period (.)'
      read(*,*) klconcpath
      ! Add backslash if necessary
      i_ccc=len(trim(klconcpath))
      ccc=klconcpath(i_ccc:i_ccc+1)
      if (ccc .ne. '\') then
        .....klconcpath = trim(klconcpath)//'\
      end if

      write(*,*) 'Enter the path for the NAVSED output files'
      write(*,*) 'If the same as the default, enter a period (.)'
      read(*,*) navsedpath

      write(*,*) 'Enter the NAVSED file extension (cth or cfd)'
      read(*,*) navsedfileext
      ! Remove any leading period
      if (navsedfileext(1:1) .eq. '.') then
        .....navsedfileext = navsedfileext(2:199)
      end if

      klac = -1.0
      open(unit=23,file=trim(PATH)//trim(klconcpath)//'P'//pool//
      & '_klconc.dat', status='old',action='read')
      ! Read two comment lines
      read(23,*) lin
      read(23,*) lin
      ! Read the data
      do
        read(23,*, end=23) m, klac
        if(m .eq. month) then
          goto 24
        end do

```

```

        end if
    end do
23 klac = -1.0
24 close(23)
    if(klac .eq. -1.0) then
        write(*,*) ' '
        write(*,*) '*** No KL Concentration for month ', month
        stop
    end if

    CHANDATA= trim(PATH)//'P'//pool//'_bwlength'//'.dat'
    if(navsedpath .eq. '0' .or. navsedpath .eq. '.')then
        CTHDATA = trim(PATH)//'P'//pool//'_ '//flowchan//'. '//
& trim(navsedfileext)
    else
        CTHDATA = trim(navsedpath)//'P'//pool//'_ '//flowchan//'. '//
& trim(navsedfileext)
    endif
    TMPDATA = trim(PATH)//trim(PROGNAME)//'.tmp'
    FLOWDATA= trim(PATH)//'P'//pool//'_ '//flowchan//'.flo'
    OUTFILE1= trim(PATH)//'p'//pool//'_ '//flowchan//'_ '//
&
& month//'.sed'
    ERRFILE= trim(PATH)//'nocels'//flowchan//'.txt'
    UNITCONC= 40
    UNITFLOW= 50
    UNITOUT1= 60
    UNITERR= 90

    WRITE_AT= 10000

    !number of reflections in secondary channel or backwater
    REFLECTS= 3

cccc cccccc cccccc ccccccccccccccccccccccccccccccccccc

    write (*,*) 'Program running...'

    write (*,*) 'Output file is ',OUTFILE1
    write(*,*) ' '

c    Extract necessary cellid's from concentration data file.
c    Write a temporary and smaller concentration data file
c    for processing by this program.

    call concmake(CHANDATA,CTHDATA,TMPDATA)

c    Count the number of lines in flow data file (used for percent
complete).
    call cntlines(FLOWDATA, flowlins)
    ! flowlins is returned.

    open(unit=UNITCONC,file=TMPDATA,status='old')
    open(unit=UNITFLOW,file=FLOWDATA,status='old')
    open(unit=UNITOUT1,file=OUTFILE1,status='unknown')

c    Read the times.
    call gettimes(UNITCONC,starttm,endtm,npts,nlms,nrm)
    ! All parameters except UNITCONC are returned.

c    Find time zero.
    call gettime0(timehist, npts, index0)
    ! index0 is returned.

    if (index0 .lt. 0) then
        write(*,*) 'No time zero in file ', TMPDATA
        stop
    end if

c    Get the time intervals.
    call getdeltaT(timehist, npts, deltaT)
    ! deltaT() is returned.

c    Print a heading at the top of the output file.
    call outhead(UNITOUT1)

```

```

        conclins = 0.0

do
301 ..... read(UNITFLOW,*, end=901) cellid_f,traffic_f,stgsl_f,T,Area,
        *      ch_length,H,dddummy,drawdown,ch_type,
        *      reflq(1),refltime(1),reflq(2),refltime(2),
        *      reflq(3),refltime(3),
        *      lfrate,mfrate,hfrate,vsum
300  format(a9,2x,a5,2x,a2,3f10.2,3f6.3,3x,a2,
&      3(f8.4,f8.0),3f8.2,2x,f9.3)

*  if(cellid_f .eq. '345R5315 ')then
*  d=1
*  endif

        conclins = conclins + 1
c      Calculate percent complete.
        pcentcmp = 100 * (1 - (flowlins - conclins)/flowlins)

        cell_f = cellid_f//' '//traffic_f//' '//stgsl_f
! Check for immediately following duplicates
        if (cell_f .eq. cell_c) then
            goto 301
        end if

c      Write something on the screen to show activity.
        if(cellid_f .ne. last_cellid_f)then
            write(*,15) cell_f, pcentcmp
15      format(' Processing cell ', a18, 2x, f5.1, '%')
            last_cellid_f = cellid_f
            i_cell_found=0
        end if

c      Skip past the times in the concentration time history file.
        call resetconcfiler(UNITCONC, nlms, nrm).....

do

        read(UNITCONC,*, end=902) cellid_c, traffic_c, stgsl_c

        cell_c = cellid_c//' '//traffic_c//' '//stgsl_c

        if(trim(cell_f) .eq. trim(cell_c)) then
            ! The Meat!!!

            i_cell_found=-1

            call getconc(UNITCONC, nlms, nrm, klac, ch_type, conc)
! conc() is returned.

            call pickq_river(stgsl_f,lfrate,mfrate,hfrate,q_river)
! q_river is returned.

            if(cell_f .eq. '325R5305  UMBO HM')then
d=1
endif

            call calcmass(npts,timehist,deltaT,conc,reflq,refltime,
        *      index0,q_river,REFLECTS,massriv,massboat,
        *      noflow,nohflow)
            ! massriv, massboat, and noflow and nohflow are returned.

            cnoflow = ' '
            cnohflow = ' '
            if (noflow .eq. 1) then
                cnoflow = 'BW'
            end if
            if (nohflow .eq. 1) then
                cnohflow = 'No Hi Flow'
                goto 810 ! Skip this info
            end if

            write(UNITOUT1,20) cell_f, massriv, massboat,

```

```

*          massriv+massboat,vsum,cnoflow,cnohflow
20      format(1x,a18,1x,f13.2,2x,f13.2,2x,f13.2, 2x,f11.3,
&          3x,a2,3x,a10)

      ! Go on to the next cell/tow_config in FLOWDATA.
      goto 800
      else
c          Skip past the current set of values in the concentration time
history file.
      call skipconc(UNITCONC, nlns, nrm).....

      end if

810 end do

      if(i_cell_found .eq. 0) then
      write(*,*) '** Cell ',cell_f,' not found in ',TMPDATA
      end if

902 continue
*      if (errfile1 .eq. 0) then
*          errfile1=1
*          ..... open(unit=UNITERR,file=ERRFILE,status='unknown')
*          write(UNITERR,*) trim(PROGNAME), ' Program'
*          write(UNITERR,201) TMPDATA
* 201      format('Cell ids and tow configurations that were not'
*          *          ' found in data file ', a64, /)
*          end if
*          write(UNITERR,*) cell_f
800 continue

      end do

901 close(unit=UNITFLOW)
      close(unit=UNITOUT1)
      close(unit=UNITERR)
      close(unit=UNITCONC, status='delete')

      stop
      end

c
c -----
c
c
c      subroutine gettimes(UNITCONC,starttm,endtm,npts,
*          nlns,nrm)
c          All parameters except UNITCONC are returned.
          common/com/timehist

      real..... timehist(1000)
      integer..... npts
      integer..... nlns, nrm
      integer..... i,j,k
      integer..... UNITCONC

      read(UNITCONC,*)
      read(UNITCONC,*) starttm, endtm, npts
      read(UNITCONC,*)
      nlns = npts/ 10
      nrm = mod(npts,10)
      k=1
      do i = 1, nlns
          read(UNITCONC,15) (timehist(j), j=k, k+9)
          k=k+10
      end do
      if (nrm .gt. 0) then
          nrmm1=nrm-1
          read(UNITCONC,15) (timehist(j), j=k,k+nrmm1)
      end if

15 format(10f10.2)

```

```

        return
    end

c -----
c -
c
c      subroutine resetconcfilE(UNITCONC, nlms, nrm)
c          Skips past the times in the concentration time history file.

        integer nlms, nrm
        integer UNITCONC

        rewind(unit=UNITCONC)

        read(UNITCONC,*)
        read(UNITCONC,*)
        read(UNITCONC,*)
        do i = 1, nlms
            read(UNITCONC,*)
        end do
        if (nrm .gt. 0) then
            read(UNITCONC,*)
        end if

        return
    end

c -----
c -
c
c      subroutine skipconcfilE(UNITCONC, nlms, nrm)
c          Skips past the current set of values in the concentration time
c          history file.

        integer nlms, nrm
        integer UNITCONC
        integer flag
        real    dummy

        read(UNITCONC, *) flag, dummy

        if (flag .eq. 1) then

            do i = 1, nlms
                read(UNITCONC,*)
            end do
            if (nrm .gt. 0) then
                read(UNITCONC,*)
            end if

        end if

        return
    end

c -----
c -
c
c      subroutine gettime0(timehist, npts, index0)
c          index0 is returned.

        real.....timehist(1000)
        integer.....npts, index0

        do index0 = 1, npts
            if(timehist(index0) .eq. 0) then
                return
            end if
        end do

        index0 = -1

        return
    end

```

```

end

c
c -----
c
c
c subroutine getdeltaT(timehist, npts, deltaT)
c   deltaT() is returned.

real.....timehist(1000)
integer.....npts
integer.....deltaT(1000)

integer i

do i = 1, npts - 1
  deltaT(i) = timehist(i+1)-timehist(i)
end do

return
end

c
c -----
c
c
c subroutine getconc(UNITCONC, nlns, nrm, klac, ch_type, conc)
c   conc() is returned.

integer.....nlns, nrm
real.....klac, conc(1000)
character*2 ch_type
integer UNITCONC
integer flag
real ambientc
integer i, j, k

  read(UNITCONC, *) flag, ambientc

if (ch_type .eq. 'BW') then

  if (flag .eq. 1) then
    k=1
    do i = 1, nlns
      read(UNITCONC, 15) (conc(j),j=k,k+9)
      do j=k,k+9
        conc(j)=conc(j)-ambientc+klac
      end do
      k=k+10
    end do
    if (nrm .gt. 0) then
      nrmm1=nrm-1
      read(UNITCONC, 15) (conc(j),j=k,k+nrmm1)
      do j=k,k+nrmm1
        conc(j)=conc(j)-ambientc+klac
      end do
    end if
  else
    conc(1) = klac
    do i = 2, (nlns*10) + nrm
      conc(i) = klac
    end do
  end if

else

  if (flag .eq. 1) then
    k=1
    do i = 1, nlns
      read(UNITCONC, 15) (conc(j),j=k,k+9)
      k=k+10
    end do
    if (nrm .gt. 0) then
      nrmm1=nrm-1
      read(UNITCONC, 15) (conc(j),j=k,k+nrmm1)
    end if
  end if

```

```

        else
            conc(1) = ambientc
            do i = 2, (nlms*10) + nrm
                conc(i) = ambientc
            end do
        end if

    end if

15 format(10f10.2)

    return
end

c
c -----
c
c
c      subroutine pickq_river(stgsl,lfrate,mfrate,hfrate,q_river)
c      q_river is returned.

character.....stgsl*2, stg*1
real .....lfrate,mfrate,hfrate
real.....q_river

    if (lfrate.le.0.0 .and. mfrate.le.0.0 .and. hfrate.le.0.0) then
        q_river = -1.0 !Backwater
        goto 800
    end if
    if (hfrate.le.0.0) then
        q_river = -2.0 !No computation to make
        goto 800
    end if

    stg = stgsl

    if (stg .eq. 'L') then
        q_river = lfrate
    else if (stg .eq. 'M') then
        q_river = mfrate
    else if (stg .eq. 'H') then
        ! Check for no value for hfrate.
        if (hfrate .le. 0.0) then
            ! Return error code.
            q_river = -3.0 ! no hi flow value
            goto 800
        end if
        q_river = hfrate
    else
        q_river = -9.0 !No stage value found.
    end if

800 return
end

c
c -----
c
c
c      subroutine calcmass(npts,timehist,deltaT,conc,reflq,refltime,
c      *                    index0,q_river,REFLECTS,massriv,massboat,
c      *                    noflow,nohflow)
c      massriv, massboat, and noflow and nohflow are returned.

integer.....npts
real.....timehist(1000), conc(1000)
integer.....deltaT(1000)
real.....reflq(3), refltime(3), q_reflect
integer      REFLECTS

real*8.....mass_sc, massriv, massboat, thisdirt
integer.....noflow, nohflow

real      con

integer i, j, k, addons

```

```

massriv = 0.0
massboat= 0.0
mass_sc = 0.0

c   Add extra time increments if necessary to go through flow reflection
times.
    addons = 0
do j = 1, REFLECTS
    if (refltime(j) .gt. timehist(npts+addons)) then
        addons = addons + 1
        timehist(npts + addons) = refltime(j)
        deltaT(npts + addons-1) = refltime(j)-timehist(npts+addons-1)
    end if
end do
c

q_reflect = reflq(1)
k = 1
noflow = 0
nohflow = 0

do i = index0+1, npts + addons
    if(k.le.REFLECTS)then
        if(timehist(i) .gt. refltime(k)) then
            k = k + 1
            if (k .lt. REFLECTS+1) then
                q_reflect = reflq(k)
            else
                q_reflect = 0.0.....! no more reflection flow
            end if
        end if
    else
        q_reflect = 0.0 .....! no more reflection flow
    end if

    !if q_river is indicating the 'no value condition'...
    if (q_river .lt. 0.0) then
        if (q_river .eq. -1.0) then
            noflow = 1      !set flag indicating cell has no initial flow
        end if
        if (q_river .eq. -3.0) then
            nohflow = 1
            return
        end if
        q_river = 0.0.....      !set q_river to 0 for calculation below...
    end if

    if (i .gt. npts) then
        con = conc(1)
    else
        con = ((conc(i-1)+conc(i))/2)
    end if

    thisdirt = (con-conc(1)) * q_river * deltaT(i-1) /1000
    if (thisdirt .lt. 0.0) then
        thisdirt = 0.0
    end if
    massriv = massriv + thisdirt

    if(noflow .ne. 0) then
        thisdirt = con * q_reflect * deltaT(i-1) /1000
    else
        thisdirt = (con-conc(1)) * q_reflect * deltaT(i-1) /1000
    end if
    if (thisdirt .lt. 0.0) then
        thisdirt = 0.0
    end if
    massboat = massboat + thisdirt

end do

return
end
c
c -----

```

```

-
c
subroutine outhead(UNITOUT1)

integer UNITOUT1

write(UNITOUT1,*) 'MASS per cell of sediment moved due to'
write(UNITOUT1,*) 'the river and to the boat passage (units: kg)'
write(UNITOUT1,*) ' '
write(UNITOUT1,20)
write(UNITOUT1,30)
20 format(' Cell      Tow Config      kg(River)      kg(Boat)'
*      '          Total      Volume      ---- Other ----')
30 format('-----')
*      '-----')
return
end

c
-----
-
c
subroutine cntlines(filename, numlines)
! counts number of lines in the given file.
! Returns numlines

character*64 filename
real          numlines

open(unit=10,file=filename,status='old')
numlines = 0.0
do
read(10,*,end=10)
numlines = numlines + 1
end do
10 close(10)

return
end

c
-----
-
c
subroutine concmake(CHANDATA,CTHDATA,TMPDATA)

common/com/timehist

character*64 CHANDATA,CTHDATA,TMPDATA

integer npts, nlns, nrm

character*9 cellid(1000), cell
character  traffic*5, stgsl*2
real.....conc(1000)

real starttm, endtm
real timehist(1000)

integer flag
real  ambientc

integer i, j, k, l, cellcnt

write (*,*) ' Extracting useful data from ', CTHDATA
write (*,*) ''

open(unit=10,file=CHANDATA,status='old')
open(unit=11,file=CTHDATA,status='old', action='read')
open(unit=12,file=TMPDATA,status='unknown')

do cellcnt = 1, 1000
read (10,*, end=910) cellid(cellcnt)
end do
910 .....close (10)

```

```

        cellcnt=cellcnt-1
c      Read the times.
      call gettimes(11,starttm,endtm,npts,nlms,nrm)
        ! All parameters except the first are returned.

      write(12,*)
      write(12,*) starttm, endtm, npts
      write(12,*)
      k=1
      do i = 1, nlms
        write(12,15) (timehist(j), j=k, k+9)
        k=k+10
      end do
      if (nrm .gt. 0) then
        nrmm1=nrm-1
        write(12,15) (timehist(j), j=k,k+nrmm1)
      end if

      do

        read (11,*, end=911) cell, traffic, stgs1

      do l = 1, cellcnt
        if(trim(cell) .eq. trim(cellid(l))) then
          read(11, *) flag, ambientc
          if (flag .eq. 1) then
            k=1
            do i = 1, nlms
              read(11, 15) (conc(j),j=k,k+9)
              k=k+10
            end do
            if (nrm .gt. 0) then
              nrmm1=nrm-1
              read(11, 15) (conc(j),j=k,k+nrmm1)
            end if
          end if

          write (12,120) cell, traffic, stgs1
120       format (a9, 2x, a5, 1x, a2)

          write (12,121) flag, ambientc
121       format (1x, i1, 1x, f8.2)

          if (flag .eq. 1) then
            k=1
            do i = 1, nlms
              write (12, 15) (conc(j),j=k,k+9)
              k=k+10
            end do
            if (nrm .gt. 0) then
              nrmm1=nrm-1
              write (12, 15) (conc(j),j=k,k+nrmm1)
            end if
          end if

          goto 400

        end if

      end do !end of l

      read(11, *, end=911) flag, ambientc
      if (flag .eq. 1) then
        k=1
        do i = 1, nlms
          read(11, 15) (conc(j),j=k,k+9)
          k=k+10
        end do
        if (nrm .gt. 0) then
          nrmm1=nrm-1
          read(11, 15) (conc(j),j=k,k+nrmm1)
        end if
      end if

```

```
        end if
    end if
400 end do
911 .....close (11)
    close (12)
15 .....format(10f10.2)
    write(*,*) ' '
    write(*,*) ' '
return
end
```


Appendix D

BWMASSPROB Program

Files

Input

Pxx_mm_5000.SMP (Tow data)

Pxx_yz_mm.SED (SEDMASS Output file)

where

xx = Pool Number

y = Stage, z = Sailing Line

mm = month

Output

Pxx_mm.BWP (sediment loading probabilities)

where

xx = Pool Number

mm = month

Brief Output Description. The BWMASSPROB program output contains three sets of 11 values. Each set is a range of probabilities from 0.0 to 1.0 in increments of tenths.

The data sets are as follows:

Set 1:	Total mass added
Set 2:	Mass added due to tow drawdown
Set 3:	Mass added due to disturbed sediment added to the river from tow passing

Source Code

```
program bwmassprob

USE MSFLIB

character*7 flowtype
character*5 towconfig(50000) !, last_tr, tc_last
character dumb, d1, d2, d3 !, d4*2
integer linesread, numevents(9) !, done_tc
character*2 sl(9)

integer confignum(50000)
dimension i_noread(9), i_cellcnt(9) !, i_numread(9)
dimension i_done_file(9) !, i_done_sorted(9), i_skip_rmfile(9)

character*254 PATH, samplefile, smppath
character*2 monthnum

character*3 month
integer i_month

character*22 dum22

character*6 rm, last_rm, current_rm !, first_rm, rmx(9)
character*10 id1, last_id1

character cellid(800000)*10, traf_sed(10000)*5
character*254 line(800000)

character*2 pool

LOGICAL(4) result

data sl/'hl', 'hm', 'hr', 'ml', 'mm', 'mr', 'll', 'lm', 'lr'/

i_debug=0
ih_debug=0

write(*,*) 'What is the pool (2 characters - Ex: 08)?'
read(*,*) pool

write(*,*)

write(*,*) 'What is the month (2 characters - Ex: 04)?'
read(*,*) monthnum
monthnum=trim(adjustr(monthnum))
if(monthnum(1:1) .eq. '') then
    monthnum(1:1) = '0'
endif

c      This one is for when running in the debugger
*      PATH = 'c:\windows\desktop\uppermiss\mass\pool'//trim(pool)//'\

PATH = ''

write(*,*) 'What is the path to the .SMP file (0 for current)?'
read(*,*) smppath
if (smppath .eq. '0') then
    smppath = PATH
end if

samplefile=trim(smppath)//'p'//trim(pool)//'_'//trim(monthnum)//
& '_5000.smp'

open(300, file=samplefile, status='old', err=12000
& , action='read')

i_probout=60

do
read(300, *, end=305) flowtype
linesread=linesread+1
```

```

c          Pull out the month...
          if(linesread.eq.6) then
            read(300,304) dum22, month, i_month
304        format(a22,a3, 3x,i2)
            goto 305
          end if
          end do
305        rewind(300)

          i_firsttime= -1

          write(*,*) ' '

          rewind(300)
          linesread=0
          do
            read(300,*) flowtype
            linesread=linesread+1

            if (flowtype .eq. 'File.HL') then
              rewind(300)
              goto 20
            end if
          end do
20        do 100 i=1, linesread-1
          read(300,*) dumb
100       continue

          i_cont=0

          i_infile_tmp=20
          i_celfile=30
          i_tmpsortfile=40

          open(i_tmpsortfile,file=trim(PATH)//'tmppsort.tmp',
            & status='unknown')

          open(i_celfile,file=trim(PATH)//'cel.tmp',
            & status='unknown')

          do i_infile=1,9
c          Initialize value
            i_line=0
            i_done_file(i_infile)=0

c          Get the tow data
            read(300,*) flowtype,d1,d2,d3,numevents(i_infile)
            do 110 i=1, numevents(i_infile)
              read(300,*) d1, confignum(i), towconfig(i)
110          continue
              write(*,*) flowtype, ' ', sl(i_infile)

              if (numevents(i_infile).eq.0) then
                if(i_infile.eq.9) then
                  last_rm = current_rm
                  current_rm = rm
                end if
                goto 221
              end if
              call heapsort(numevents(i_infile), confignum, towconfig)

              open(i_infile+10, file=trim(PATH)//'p'//trim(pool)//'_'//
                & sl(i_infile)//'_'//trim(monthnum)//'.sed',status='old'
                , action='read')
              write(*,*) 'Opened .sed file ', sl(i_infile)

              call resetinfile(i_infile+10)

              i_cellcnt(i_infile)=0
              i_noread(i_infile)=0

              do
                i_line = i_line + 1
                read(i_infile+10,160, err=161) line(i_line)
160          format(a254)

```

```

traf_sed(i_line) = line(i_line)(12:16)
end do
161   i_line = i_line - 1
      close(i_infile+10)

      call heapsort3(i_line, traf_sed, line)

      i_traf_sed=0
      i_bartell=0
      i_dups=0
      i_wrote=0
400   i_bartell = i_bartell + 1 + i_dups
      i_dups=0
      if(i_bartell .gt. numevents(i_infile)) goto 490
403   if(towconfig(i_bartell+i_dups+1) .eq.
      & towconfig(i_bartell)) then
      i_dups = i_dups + 1
      goto 403
      end if
      i_gotone=0
401   i_traf_sed = i_traf_sed + 1
      if(i_traf_sed .gt. i_line) goto 400
420   if(trim(towconfig(i_bartell)) .eq.
      & trim(traf_sed(i_traf_sed))) then
      i_gotone = -1
      do i=1, i_dups+1
      i_wrote = i_wrote + 1
      write(i_tmsortfile,402) line(i_traf_sed)
      end do
402   format(a254)
      i_traf_sed = i_traf_sed + 1
      goto 420
      else
      if(i_gotone .eq. 0) then
      goto 401
      else
      i_traf_sed = 0
      goto 400
      end if
      end if
      goto 400

490   write(*,*) 'Number of events for ', sl(i_infile),': ',
      & numevents(i_infile)
      write(*,*) 'Wrote ',i_wrote,' items for ',sl(i_infile)

221   end do
      close(i_tmsortfile)

      open(i_tmsortfile, file=trim(PATH)//'tmsort.tmp',
      & status='old')
      i_line=0
      do
      i_line = i_line + 1
      read(i_tmsortfile,.....160, end=450) line(i_line)
      cellid(i_line)=line(i_line)(2:10)
      end do
450   i_line = i_line - 1
      close(i_tmsortfile, status='delete')

      write(*,*) 'Sorting...'
      call heapsort2(i_line, cellid, line)
      write(*,*) 'Sorted ',i_line,' items by cellid'

      open(72,file='sortline.txt', status='unknown')
      do 383 iq=1,i_line
      write(72,*) trim(line(iq))
383   continue
      close(72)

*   open(i_probout, file=
*     & trim(PATH)//'bwprob/p'//pool//'_ '//trim(monthnum)//'.bwp',
*     & form='formatted',status='unknown')
open(i_probout, file=
  & trim(PATH)//'p'//pool//'_ '//trim(monthnum)//'.bwp',

```



```

12000      write(*,*) 'ERROR! Sample File '//trim(samplefile)//
&          ' was not found.'

90000      stop
          end

*****

      subroutine hist(path, infil, i_month, cellid,
&          i_tows1992, i_tows100pc, ambientc_inp, i_probout)
*
      character*254 path
      character*64 infil
      character*10 cellid
      integer i_month

      dimension val(10)

      character TRAFFIC*5, STG_SL*2, ID1*10
      DIMENSION PFP(5,5001), CLASS(4001), SUMCLASS(4001)
      DIMENSION PCGCLASS(4001), PROB(11), CUMPCT(4001)

      real massriv, massboat, masstotal

      integer CLASSNUM

      sumx = 0

      ih_debug=0

      i_loop=1

      write(*,*) 'In HIST subroutine using cellid ',cellid
*
      open(115,file=trim(path)//trim(infil), form='formatted',
&          status='old', action='read')
* NAVFPROB STATEMENT
      read(115,*) id1, traffic, STG_SL, massriv, massboat, masstotal

      do 1400 i_loop=1,3

      sumx=0
          avgx=0

2          CONTINUE

          rewind(115)

          PFPMIN = 100000000.
          PFPMAX = 0.

      do i_c=1,5000
          PFP(i_loop, i_c)=0.0
      end do

*
*          READ AND STORE DATA IN ARRAY, FIND MAX AND MIN
*
          I = 1

* NAVFPROB STATEMENT
1          read(115,*, end=50) id1,traffic,STG_SL,massriv,massboat
          masstotal=massriv+massboat
          val(1)=masstotal
          val(2)=massriv
          val(3)=massboat

* NAVF          if (val(i_loop) .ge. 9999.0) goto 1    ! No data

          PFP(i_loop,I)= val(i_loop)

          sumx=sumx+val(i_loop)

```

```

        IF(PFP(i_loop,I).GT.PFPMAX) PFPMAX = PFP(i_loop,I)
        IF(PFP(i_loop,I).LT.PFPMIN) PFPMIN = PFP(i_loop,I)
        I = I + 1
if (I .gt. 5001) then
    write(*,*)'More than 5000 tows. Data truncated'
    goto 50
end if
GOTO 1
*
50  NUMDAT = I-1
    if (NUMDAT .lt. 5000) then
        write(*,*)'Less than 5000 tow data elements. This cell skipped.'
*      Exit and Go to the next cell
        goto 1500
    end if
    RNUMDAT = NUMDAT

    avgx=sumx/RNUMDAT
    if (avgx .eq. 0.0) then
        ratio = 1.0
    else
        ratio=10*PFPMAX/avgx
    end if
*  COMPUTE CLASS WIDTH
    if(ratio .gt. 1000.0) ratio = 1000.0
    if(ratio .lt. 25.0) ratio = 25.0
    CLASSWID = avgx/ratio

    NN = 4 * ratio
    CLASSNUM = NN
*
    CLASS(1) = PFPMIN
    CLASS(NN+1) = PFPMAX
    SUMCLASS(1) = 0.0
    DO 100 II = 2 , NN
        CLASS(II) = CLASS(II-1) + CLASSWID
        SUMCLASS(II) = 0.0
100  CONTINUE
*
*  DETERMINE NUMBER OF VALUES IN EACH CLASS
*
    DO 300 II = 1, NUMDAT
    DO 400 I = 1, NN
        IF(PFP(i_loop,II).LT.CLASS(I)) GOTO 400
        IF(PFP(i_loop,II).GT.CLASS(I+1)) GOTO 400
        SUMCLASS(I) = SUMCLASS(I) + 1.
400  CONTINUE
300  CONTINUE
450  CONTINUE
*
*  FLAG = 0 (ALL INTERVALS HAVE AT LEAST ONE VALUE)
*  FLAG = 1 (ONE OR MORE INTERVALS HAVE NO VALUES WHICH MEANS
*  CLASSNUM WILL HAVE TO BE REDUCED FOR PROGRAM TO RUN)
*
    flag = 0.0
    SUMPCT = 0.0
    CUMPCT(1) = 0.0
*  WRITE(*,*) 'CLASS  LOWLIM  HILIM  CLASSREP  #HIT  CLPROB  CUMPROB'
    DO 500 J = 1 , NN
        if(sumclass(j).eq.0.0) flag = 1.0
*
*  write(*,*) 'RNUMDAT = ',RNUMDAT,'I=',I,'J=',J,'NN=',NN,
*  & 'i_loop=',i_loop

    if(RNUMDAT .le. 0.00001)then
        write(*,*) 'RNUMDAT = ',RNUMDAT,' in HIST. I=',I
        stop
    else
        PCGCLASS(J) = SUMCLASS(J)/RNUMDAT
    end if
    SUMPCT = SUMPCT + PCGCLASS(J)
    CUMPCT(J+1) = SUMPCT

```

```

500     CONTINUE
*
*     test for flag = 1, if so go to end and REDUCE CLASSNUM BY 1
*
***     IF(flag.eq.1.0) goto 1000
*
*     FIND PFP AT PROB 0-1.0 IN 0.1 INCREMENTS USING LINEAR INTERP
*     THIS IS BEING STORED IN THE SAME ARRAY AS THE ORIGINAL VALUES
*     WHICH ARE NO LONGER NEEDED. PFPMIN IS USED FOR PROB = 0,
*     PFPMAX IS USED FOR PROB = 1.0
*
        PROB(1) = 0
        PFP(i_loop,1) = PFPMIN
        DO 600 J=2,11
        PROB(J) = PROB(J-1) + 0.1
        DO 700 I = 1,NN
        IF(PROB(J).LT.CUMPCT(I)) GOTO 700
        DELT = PROB(J)-CUMPCT(I)

        if(PCGCLASS(I) .le. 0.000001)then
*           write(*,*) 'PCGCLASS(I) = ',PCGCLASS(I),' in HIST, I = ',I
           goto 700
        else
*           write(*,*) 'PCGCLASS(I) = ',PCGCLASS(I), ' I = ', I
           RATIO = DELT/PCGCLASS(I)
           end if
           DIFF = RATIO*CLASSWID
           PFP(i_loop,J) = CLASS(I) + DIFF
700     CONTINUE
600     CONTINUE
        PROB(11) = 1.0
        PFP(i_loop,11) = PFPMAX

        goto 1401
1000    CONTINUE

1401    WRITE(*,14) NUMDAT,i_loop
14      format(' NUMBER OF DATA POINTS = ',i6,' LOOP ', i2)

1400    continue

        WRITE(i_probout,126)  cellid,
&      (PFP(1,ii1),ii1=1,11),
&      (PFP(2,ii2),ii2=1,11),
&      (PFP(3,ii3),ii3=1,11)

125      format(6x,f3.1,6x,f12.3,1x,f12.3,1x,f12.3,1x,f12.3,1x,f12.3)
126      format(a10,1x,32(f15.3,' '),f15.3)

1500    CLOSE (115)

2000    return
        end

*****
c
c -----
c
c     subroutine heapsort(n, ra, ca)
c     ra is the "control" !
c     integer ra(n)
c     character*5 ca(n),cca
c     !
c     if(n .le. 1) then
c         return
c     end if

c         l=n/2+1
c         ir=n

10     continue
c         if(l .gt. 1) then
c             l=l-1
c             rra=ra(l)
c             cca=ca(l) !

```

```

else
  rra=ra(ir)
  ra(ir)=ra(1)
  cca=ca(ir) !
  ca(ir)=ca(1) !
  ir=ir-1
  if(ir .eq. 1) then
    ra(1)=rra
    ca(1)=cca !
    return
  endif
endif
endif
i=1
j=l+1
20 if(j .le. ir) then
  if(j .lt. ir) then
    if(ra(j) .lt. ra(j+1)) j=j+1
  endif
  if(rra .lt. ra(j)) then
    ra(i)=ra(j)
    ca(i)=ca(j) !
    i=j
    j=j+j
  else
    j=ir+1
  endif
  goto 20
endif
ra(i)=rra
ca(i)=cca !
goto 10
end

c
c -----
c
c      subroutine heapsort2(n, ra, ca)
c      ra is the "control" !
c      character*10 ra(n),rra
c      character*254 ca(n),cca
c      !
c      if(n .le. 1) then
c        return
c      end if

      l=n/2+1
      ir=n

10  continue
      if(l .gt. 1) then
        l=l-1
        rra=ra(l)
        cca=ca(l) !
      else
        rra=ra(ir)
        ra(ir)=ra(1)
        cca=ca(ir) !
        ca(ir)=ca(1) !
        ir=ir-1
        if(ir .eq. 1) then
          ra(1)=rra
          ca(1)=cca !
          return
        endif
      endif
      endif
      i=1
      j=l+1
20  if(j .le. ir) then
        if(j .lt. ir) then
          if(ra(j) .lt. ra(j+1)) j=j+1
        endif
        if(rra .lt. ra(j)) then
          ra(i)=ra(j)
          ca(i)=ca(j) !
          i=j
          j=j+j
        else
          j=ir+1
        endif
        goto 20
      endif
      ra(i)=rra
      ca(i)=cca !
      goto 10
    end
  
```

```

        else
            j=ir+1
        endif
        goto 20
    endif
    ra(i)=rra
    ca(i)=cca!
    goto 10
end

c
c -----
c
c      subroutine heapsort3(n, ra, ca)
c          ra is the "control" !
c          character*5 ra(n),rra
c          character*254 ca(n),cca
c          !
c          if(n .le. 1) then
c              return
c          end if

c          l=n/2+1
c          ir=n

10      continue
        if(l .gt. 1) then
            l=l-1
            rra=ra(l)
            cca=ca(l) !
        else
            rra=ra(ir)
            ra(ir)=ra(l)
            cca=ca(ir) !
            ca(ir)=ca(l) !
            ir=ir-1
            if(ir .eq. 1) then
                ra(1)=rra
                ca(1)=cca !
                return
            endif
        endif
        i=l
        j=l+1
20      if(j .le. ir) then
            if(j .lt. ir) then
                if(ra(j) .lt. ra(j+1)) j=j+1
            endif
            if(rra .lt. ra(j)) then
                ra(i)=ra(j)
                ca(i)=ca(j) !
                i=j
                j=j+j
            else
                j=ir+1
            endif
            goto 20
        endif
        ra(i)=rra
        ca(i)=cca!
        goto 10
    end

c
c -----
c
c      subroutine gettimes(UNITIN,starttm,endtm,npts,
c          *      nlns,nrm,timehist)
c          All parameters except UNITIN are returned.
c          *      common/com/timehist

c          real.....timehist(50000)
c          integer..... npts
c          integer..... nlns, nrm
c          integer..... i,j,k

```

```

integer.....UNITIN

read(UNITIN,*)
read(UNITIN,*) starttm, endtm, npts
read(UNITIN,*)
nlns = npts/ 10
nrm = mod(npts,10)
k=1
do i = 1, nlns
  read(UNITIN,15) (timehist(j), j=k, k+9)
  k=k+10
end do
if (nrm .gt. 0) then
  nrmm1=nrm-1
  read(UNITIN,15) (timehist(j), j=k,k+nrmm1)
end if

15      format(10f10.2)

return
end

c
c -----
c
c      subroutine resetinfile(UNITIN)
c      Skips past the times in the concentration time history file.

integer UNITIN
character*254 line

rewind(unit=UNITIN)

do i = 1, 4
  read(UNITIN,*) line
end do

return
end

c
c -----
c
c      subroutine getconc(UNITIN, nlns, nrm, conc, ambientc)
c      conc() is returned.

integer.....nlns, nrm
real.....conc(50000)
integer UNITIN
integer flag
real ambientc
integer i, j, k

read(UNITIN, *) flag, ambientc

if (flag .eq. 1) then

k=1
  do i = 1, nlns
    read(UNITIN, 15) (conc(j),j=k,k+9)
    k=k+10
  end do
  if (nrm .gt. 0) then
    nrmm1=nrm-1
    read(UNITIN, 15) (conc(j),j=k,k+nrmm1)
  end if
else
  conc(1) = -999.0
  do i = 2, (nlns*10) + nrm
    conc(i) = 0.0
  end do
end if

15      format(10f10.2)

return

```

```

                                end
c
c -----
c
c      subroutine getdeltaT(timehist, npts, deltaT)
c      deltaT() is returned.
c
c      real.....timehist(50000)
c      integer.....npts
c      integer.....deltaT(50000)
c
c      integer i
c
c      do i = 1, npts - 1
c        deltaT(i) = timehist(i+1)-timehist(i)
c      end do
c
c      return
c      end
c
c -----
c
c      subroutine get_rm(id1, c_rm, rm)
c      character*10 id1
c      real rm
c
c      character*6 c_rm, cc_rm
c
c      i_pos=scan(id1,'LR')
c      c_rm=trim(id1(i_pos+1:i_pos+6))
c      c_rm=adjustr(c_rm)
c      cc_rm=c_rm(6:6)
c      c_rm=trim(c_rm(2:5))//'. '//cc_rm
c      c_rm=adjustl(c_rm)
c      Convert rivermile string to a real
c      open(401,file='temp.tmp',status='unknown')
c      write(401,102) c_rm
102      format(a)
c      rewind(401)
c      read(401,103) rm
103      format(f5.1)
c      close(401, status='delete')
c
c      return
c      end

```

Appendix E

SED2BW Program

Files

Input

Pxx_TOWNUMBER.DAT (annual number of tows – in the poolxx directory – this file is optional based on runtime preferences – the value can optionally be input at the keyboard)

Pxx_TRAFMNTH.DAT (in the poolxx directory – percent values of traffic per month)

Pxx_BWSECAREA.DAT (area in acres of each backwater)

Pxx_mm.BWP (sediment loading probabilities)

Pxx_BWLENGTH.DAT (contains the sediment type information)

PxxBWSEC.TXT (from GIS)

where

xx = Pool Number

mm = month

Output

Pxx_ttttTOWS.S2B (traffic impact potentials)

where

xx = Pool Number

tttt = number of yearly tows used

► **Note:** See “4Output“ on page 13 for a complete description of the output format.

Source Code

```
program sed2bw
```

```

c          poolxx is the working directory...
c          This program requires input from:
c          TRAFMNTH.DAT
c          Pxx_BWSECAREA.DAT
c          Pxx_mm.BWP
c          Pxx_BWLENGTH.DAT
c          PxxBWSEC.TXT

c          Variable prefixes used...
c          i_ = integer
c          r_ = real
c          c_ = character
c          rec_ = record (structure)

USE MSFLIB ! for the $MAXPATH, PATH routines, etc.

* -----

          ! Final storage for all resulting bw/sec channel information
STRUCTURE /bwsec_info/
  CHARACTER*5 c_number
  integer i_area
  real r_totalacre_ft_peryear
  real r_cm_peryear
  integer i_color_by_rate ! via cm_peryear
END STRUCTURE
  record /bwsec_info/ rec_bwsecinfo(100)

STRUCTURE /bw_length/
  CHARACTER*10 c_cellid
  integer i_sedtype ! Cohesive, Noncohesive
  real r_sedspecweight
END STRUCTURE
  record /bw_length/ rec_bwlength(100)
c          Cohesive sediment is represented by 2, specific weight, 78.0 pounds/ft^3
c          Non-cohesive sediment is represented by 3, specific weight, 96.3 pounds/ft^3
  real specweight_sedtype(10)
  character*15 c_sedtype(10)

STRUCTURE /bw_sec/
  CHARACTER*8 c_feature
  character*10 c_cellid
  real r_width
END STRUCTURE
  record /bw_sec/ rec_bwsec(100)

STRUCTURE /bw_prob/
  character*10 c_cellid
  real r_prob(11)
END STRUCTURE
  record /bw_prob/ rec_bwprob(100)

          ! Final storage for all resulting cell information
STRUCTURE /bw_cellinfo/
  CHARACTER*8 c_feature
  character*10 c_cellid
  real r_probvalue
  real r_mass(12)
  real r_tonsperyear
  real r_acre_ft_peryear
  integer i_color_by_volume ! via acre_ft_peryear
  real r_acre_ft_peryear_permeter
  integer i_color_by_unit_volume ! via acre_ft_peryear_permeter
  integer i_sedtype ! Cohesive, Noncohesive
END STRUCTURE
  record /bw_cellinfo/ rec_cellinfo(100)
  !

  character*6 c_colors(3)
  integer i_worst_color

  character*256 PATH, bwprobdir
  character*10 c_totaltows

```

```

character*2 pool, month
integer i, i_month, i_percentile
integer i_dum

character*10 c_lastcellid

character*1 c_readtows_how

real r_trafmonthpercent(12)
real i_towspermonth(12)
integer i_trafmonth
character*255 c_trafpercentfile

integer i_notows(12)

character*10 c_cellid
character*6 c_inout

integer i_sedtype

character*8 c_feature
character*5 c_bwsec
character*255 c_bwsecareacomment1, c_bwsecareacomment2

real r_width
integer bwsec_i_totalcells
integer bwlength_i_totalcells
integer bwprob_i_totalcells

real r_us_tonspermonth(12)

character*255 c_infile
character*255 c_lin
character*255 c_trafpercentcomment1, c_trafpercentcomment2
character*255 c_townumbercomment1, c_townumbercomment2

CHARACTER($MAXPATH) dir
INTEGER(4) length

character*1 ccc

* -----
c          Assign Values...

specweight_sedtype(1) = 0.0   ! ??
specweight_sedtype(2) = 78.0
specweight_sedtype(3) = 96.3
specweight_sedtype(9) = 9999 ! No data

c_sedtype(1) = 'Do Not Know'
c_sedtype(2) = 'Cohesive'
c_sedtype(3) = 'Noncohesive'
c_sedtype(4) = 'Cohesive-Soft'
c_sedtype(5) = 'Cohesive-Medium'
c_sedtype(6) = 'Cohesive-Hard'
c_sedtype(7) = 'Cohesive- ???'
c_sedtype(9) = 'No Sedtype Data'

c_colors(1) = 'BLUE'
c_colors(2) = 'YELLOW'
c_colors(3) = 'RED'

do 99 i=1,12
i_notows(i) = 0
99        continue

* -----

write(*,*) 'What is the pool (2 characters - Ex: 08)?'
read(*,*) pool

*          write(*,*) 'What is the month (2 characters - Ex: 04)?'

```

```

*      read(*,*) monthnum
*      monthnum=trim(adjustr(monthnum))
*      if(monthnum(1:1) .eq. '') then
*          monthnum(1:1) = '0'
*      endif

c      This one is for when running in the debugger
*      PATH = 'c:\windows\desktop\uppermiss\pool'\trim(pool)'\sc\'
*      PATH = ''

      ! Show the working path...
      if (PATH .eq. '') then
      ! Get current directory for display info...
      dir = FILE$CURDRIVE
      length = GETDRIVEDIRQQ(dir)
      ! Add backslash if necessary
      i_ccc=len(trim(dir))
      ccc=dir(i_ccc:i_ccc+1)
      if (ccc .ne. '\') then
      dir = trim(dir)'\\'
      end if
      !
      IF (length .GT. 0) THEN
      WRITE (*,*) 'Current directory is: '
      WRITE (*,*) trim(dir)
      ELSE
      WRITE (*,*) 'Failed to get current directory'
      END IF
      PATH = trim(dir)
      else
      write (*,*) 'Current path is ', trim(PATH)
      end if

      bwprobdir = ''
      write(*,*)
      write(*,*) 'What is the subdirectory containing ' //
      & 'BW or SC Probability data (.bwp files)?'
      write(*,*) 'Normally BWPROB.'
      write(*,*) 'If the same as the default, enter a period (dot)'
      read(*,*) bwprobdir
      if (trim(bwprobdir) .eq. '0' .or. trim(bwprobdir) .eq. '.') then
      bwprobdir = ''
      else
      i_ccc=len(trim(bwprobdir))
      ccc=bwprobdir(i_ccc:i_ccc+1)
      if (ccc .ne. '\') then
      bwprobdir = trim(bwprobdir)'\\'
      end if
      end if

      write(*,*) 'How many total tows pass for the year? '
      write(*,*) '"E" = Enter value, "F" = Read from file'
      read(*,*) c_readtows_how
      if (c_readtows_how .eq. 'E' .or. c_readtows_how .eq. 'e') then
      write(*,*) 'How many tows?'
      read(*,*) i_totaltows
      else
      open (10, file=trim(path)'\..\p'\trim(pool)'\_townumber.dat',
      & status='old', action='read')
      read(10,550) c_townumbercomment1
      read(10,550) c_townumbercomment2
      read(10,*) i_totaltows
      close(10)
      end if

207      write(*,*) 'What percentile value do you wish to use?'
      write(*,*) 'Choose 0, 10, 20, 30, 40, 50, 60, 70, 80, 90, or 100%'
      read(*,*) i_percentile
      if (i_percentile .ne. 0 .and. i_percentile .ne. 10
      & .and. i_percentile .ne. 20 .and. i_percentile .ne. 30
      & .and. i_percentile .ne. 40 .and. i_percentile .ne. 50
      & .and. i_percentile .ne. 60 .and. i_percentile .ne. 70
      & .and. i_percentile .ne. 80 .and. i_percentile .ne. 90
      & .and. i_percentile .ne. 100) then

```

```

write(*,*) 'Invalid percentage value, re-enter: '
goto 207
end if
write(*,*)

c      Read the Traffic Percentages...
c_trafpercentfile=trim(path)//'..\p'//trim(pool)//'_ '//
& 'trafmnth.dat'
open(10, file=c_trafpercentfile, status='old', action='read')
!Skip the first two header lines
read(10,550,end=906) c_trafpercentcomment1
read(10,550,end=906) c_trafpercentcomment2
do 102 i_trafmonth=1,12
read(10,*) r_trafmonthpercent(i_trafmonth)
if (r_trafmonthpercent(i_trafmonth) .gt. 0) then
i_towspermonth(i_trafmonth) = i_totaltows /
& r_trafmonthpercent(i_trafmonth)
else
i_towspermonth(i_trafmonth) = 0
end if
102  continue
goto 107
906  write(*,*) 'Unexpected end of file reading ',
& trim(c_trafpercentfile)
107  continue

! Read the backwater/secondary channel areas...
c_infile = trim(path)//'p'//trim(pool)//'_bwsecarea.dat'
open(25, file=c_infile, status='old', action='read')
i = 0
! Read first two comment lines
read(25,550) c_bwsecareacomment1
read(25,550) c_bwsecareacomment2
do
i = i + 1
read(25,*, end=916) rec_bwsecinfo(i).c_number,
& rec_bwsecinfo(i).i_area
end do
916  continue
close(25)
i_totalbwsecs = i - 1

550  format(a255)

i_found_tow_data = 0
! Open .bwp files for the 12 months and prepare to read the data...
do 105 i_month=1,12
! Convert integer i_month to two characters
write(month,505) i_month
505  format(i2)
month=trim(adjustr(month))
if(month(1:1) .eq. '') then ! Fill blank space with zero
month(1:1) = '0'
endif
!
c_infile = trim(path)//trim(bwprobdir)//'p'//
& trim(pool)//'_ '//month//'.bwp'
write(*,*) trim(c_infile)
open(10+i_month, err=301, file=c_infile, status='old',
& action='read')
! Skip past header info
do
506  read(10+i_month,506, end=903, err=903) c_lin
format(a255)
if (c_lin(1:2) .eq. '--') then
goto 201
end if
end do
goto 201
903  continue ! If a read error...
write(*,*)

```

```

write(*,*) 'Error reading data from ', trim(PATH),
& trim(c_infile), '. File is not in the correct format.'
write(*,*)
goto 301
201 continue

i_found_tow_data = 1 ! Found some data - flag

goto 305
301 continue
i_notows(i_month) = 1
write(*,507) i_month
507 format(' ... No tows for month ', i2)
305 continue
105 continue
!

! If no tow data found then exit...
if (i_found_tow_data .eq. 0) then
write(*,*) 'No .bwp files were read. No tow data. Exiting...'
stop
end if

c Aquire the .bwp data...
i_cell = 0! Cell Counter
do
i_cell = i_cell + 1 ! Cell Counter
rec_bwprob(i_cell).c_cellid = ''
rec_cellinfo(i_cell).r_tonsperyear = 0.0 ! Initialize value for sum
i_haveread_thiscell = 0
do 106 i_month=1,12
if (i_notows(i_month) .ne. 0) then
goto 106
end if
if (i_haveread_thiscell .ne. 0) then
c_lastcellid = rec_bwprob(i_cell).c_cellid
end if
! Read the cellid and first set of 11 values (Mass Totals)...
read(10+i_month,*,end=904)
& rec_bwprob(i_cell).c_cellid,
&(rec_bwprob(i_cell).r_prob(i1),i1=1,11)
if (i_haveread_thiscell .ne. 0) then ! Check for cellid sync
if (trim(rec_bwprob(i_cell).c_cellid) .ne.
&trim(c_lastcellid)) then
write(*,*)'Problem: .bwp input files cellids',
&' do not sync.'
stop
end if
end if
i_haveread_thiscell = 1
! Calculate tons per month for cell i
r_us_tonspermonth(i_month) = i_towspermonth(i_month) *
&rec_bwprob(i_cell).r_prob((i_percentile/10)+1) *
&2.203 / 2000
! Keep a running total of tons for cell i for the year
rec_cellinfo(i_cell).r_tonsperyear =
&rec_cellinfo(i_cell).r_tonsperyear +
&r_us_tonspermonth(i_month)
106 continue
! Store the cellid in the 'Results' array
rec_cellinfo(i_cell).c_cellid =
&rec_bwprob(i_cell).c_cellid
end do

904 do 108 i_month=1,12
close(10+i_month)
108 continue

bwprob_i_totalcells = i_cell - 1

c Must match cellids in other files with those in .bwp file

```

```

c      Open pxx_bwlength.dat, read cellid and sediment type into array
      open(10, file=trim(path)//'p'//
&trim(pool)//'_'//'bwlength.dat', status='old', action='read')
      i_cell = 0
      do
      i_sedtexture = 0
      read(10,*, end=901) c_cellid, i_dum, i_sedtype, i_sedtexture
      ! Find the matching cellid stored earlier in the 'Results' array
      do 115 i=1,bwprob_i_totalcells
      if(rec_cellinfo(i).c_cellid .eq. c_cellid) then
      i_cell = i_cell + 1
      rec_bwlength(i_cell).c_cellid = c_cellid
      rec_bwlength(i_cell).i_sedtype = i_sedtype
      if (rec_bwlength(i_cell).i_sedtype .gt. 3) then
      rec_bwlength(i_cell).i_sedtype = 9

*      write(*,*) 'No "Type of sediment" data for cell ',
*      &trim(rec_bwlength(i_cell).c_cellid)
      end if
      rec_bwlength(i_cell).r_sedspecweight =
&specweight_sedtype(rec_bwlength(i_cell).i_sedtype)
      rec_cellinfo(i).i_sedtype =
&rec_bwlength(i_cell).i_sedtype
      if (i_sedtype .eq. 2) then
      if ((i_sedtexture .ge. 1) .and.
&
      (i_sedtexture .le. 3)) then
      rec_cellinfo(i).i_sedtype = i_sedtype + 1 +
&i_sedtexture
      else
      rec_cellinfo(i).i_sedtype = 7
      end if
      end if
      goto 116
      end if
      continue
115      continue
116      continue
      end do
      close(10)
      bwlength_i_totalcells = i_cell
*      call heapsort1(bwlength_i_totalcells,rec_bwlength.c_cellid,
*      &rec_bwlength)

c      Open pxxbwsec.dat, read cellid and other stuff
      open(10, file=trim(path)//'p'//
&trim(pool)//'bwsec.txt', status='old', action='read')
      read(10,*) c_lin ! Skip first line (description line)
      i_cell = 0
      do
      read(10,*, end=902) c_feature, c_cellid, dum, r_width, dum,
& dum,dum,dum,dum,dum,dum,dum,dum, c_inout
      if (trim(c_inout) .eq. 'INLET') then
      ! Find matching cellid in rec_cellinfo array
      do 117 i=1,bwprob_i_totalcells
      if(rec_cellinfo(i).c_cellid .eq. c_cellid) then
      i_cell = i_cell + 1
      rec_bwsec(i_cell).c_feature = c_feature
      rec_cellinfo(i).c_feature = c_feature.....! Fill in the feature value
      rec_bwsec(i_cell).c_cellid = c_cellid
      rec_bwsec(i_cell).r_width = r_width
      goto 118
      end if
      continue
117      continue
      end if
      continue
118      continue
      end do
      close(10)
      bwsec_i_totalcells = i_cell
*      call heapsort2(bwsec_i_totalcells,rec_bwsec.c_cellid,rec_bwsec)

      ! Calculate Acre-ft/year, need density value based on sedtype
      do 120 i_cell = 1,bwprob_i_totalcells
      if ((rec_bwlength(i_cell).r_sedspecweight .eq. 0) .or.
&(rec_bwlength(i_cell).i_sedtype .gt. 3)) then

```

```

write(*,*) 'Cannot calculate Acre-ft/year for cell ',
&trim(rec_bwlength(i_cell).c_cellid), '. ',
&'No "Type of sediment" data for this cell.'
end if
! Find Matching cellid in rec_bwlength array for Specific Weight Value
do 121 i=1,bwlength_i_totalcells
if (rec_cellinfo(i).c_cellid .eq.
&rec_bwlength(i_cell).c_cellid) then
rec_cellinfo(i).r_acre_ft_peryear =
&rec_cellinfo(i).r_tonsperyear * 2000 /
! Density value
&rec_bwlength(i_cell).r_sedspecweight /43560 ! 43560 ft^2/Acre
! Assign Impact Potential color by VOLUME
if (rec_cellinfo(i).r_acre_ft_peryear .lt. 1.0) then
rec_cellinfo(i).i_color_by_volume = 1
else
rec_cellinfo(i).i_color_by_volume = 2
end if
goto 122
end if
121 continue
122 continue

120 continue
!

! Calculate cm/year into each backwater/secondary channel for all associated
inlets
do 126 i_bwsec=1,i_totalbwsecs
rec_bwsecinfo(i_bwsec).r_cm_peryear = 0.0
rec_bwsecinfo(i_bwsec).r_totalacre_ft_peryear = 0.0
! Find matching bw/sc numbers in rec_bwsecinfo array
do 127 i_cell=1,bwprob_i_totalcells
c_bwsec = rec_cellinfo(i_cell).c_feature(1:5)
if (rec_bwsecinfo(i_bwsec).c_number .eq. c_bwsec) then
! Sum the Acre-ft/year for this Backwater/Secondary channel
rec_bwsecinfo(i_bwsec).r_totalacre_ft_peryear =
&rec_bwsecinfo(i_bwsec).r_totalacre_ft_peryear +
&rec_cellinfo(i_cell).r_acre_ft_peryear
end if
127 continue
! Finish the cm/year calculation
rec_bwsecinfo(i_bwsec).r_cm_peryear =
&( rec_bwsecinfo(i_bwsec).r_totalacre_ft_peryear /
&rec_bwsecinfo(i_bwsec).i_area ) * 30.48 ! Convert ft to cm
! Assign Impact Potential color by RATE
if (rec_bwsecinfo(i_bwsec).r_cm_peryear .lt. 0.1) then
rec_bwsecinfo(i_bwsec).i_color_by_rate = 1
else
if ((rec_bwsecinfo(i_bwsec).r_cm_peryear .ge. 0.1)
&.and.
&(rec_bwsecinfo(i_bwsec).r_cm_peryear .lt. 1.0)) then
rec_bwsecinfo(i_bwsec).i_color_by_rate = 2
else
rec_bwsecinfo(i_bwsec).i_color_by_rate = 3
end if
end if

126 continue

! Calculate Acre-ft/yr/meter for each cellid in rec_cellinfo array
do 130 i_cell = 1,bwprob_i_totalcells
! Find matching cellid from the rec_bwsec array
do 131 j_cell = 1, bwsec_i_totalcells
if (trim(rec_bwsec(j_cell).c_cellid) .eq.
&trim(rec_cellinfo(i_cell).c_cellid)) then
if (rec_bwsec(j_cell).r_width .eq. 0) then
write(*,*) 'Width for cell ',
&trim(rec_cellinfo(i_cell).c_cellid),
&' is 0. Must skip this cell.'
goto 130
end if
rec_cellinfo(i_cell).r_acre_ft_peryear_permeter =
&rec_cellinfo(i_cell).r_acre_ft_peryear /

```

```

&rec_bwsec(j_cell).r_width
! Assign Impact Potential color by UNIT VOLUME
if (rec_cellinfo(i_cell).r_acre_ft_peryear_permeter
&.lt. 0.01) then
rec_cellinfo(i_cell).i_color_by_unit_volume = 1
else
rec_cellinfo(i_cell).i_color_by_unit_volume = 2
end if
goto 130
end if
131 continue
130 continue

c Write Results
! Cloddege to convert integer to character
open(101,file='sed2bw.tmp',status='unknown')
write(101,*) i_totaltows
rewind(101)
read(101,*) c_totaltows
close(101, status='delete')
!
open(10, file=trim(path)//'p'//
&trim(pool)//'_'//trim(c_totaltows)//'tows.s2b',
&status='unknown')
write(10,*) 'SEDIMENTS TO BACKWATER OR SECONDARY CHANNEL ',
&'(SEE FEATURE), POOL',
&trim(pool)
530 write(10,530) i_percentile
format (' BASED ON ', i3, '% ROLLUP')
write(10,*) 'Total tows for year: ', i_totaltows
write(10,*)
write(10,531)
531 format( 'FEATURE AREA CELLID SEDIMENT_TYPE',
&' IMPACT_BY_VOLUME ',
&' IMPACT_BY_RATE ',
&' IMPACT_BY_UNIT_VOLUME WORST_CASE')
write(10,532)
532 format( ' acres ',
&' acre-ft/year ',
&' cm/year ',
&' acre-ft/year/meter')
write(10,533)
533 format(135('-'))

! Sort rec_cellinfo array by c_feature
call heapsort1(bwprob_i_totalcells, rec_cellinfo.c_feature,
&rec_cellinfo)
do 135 i_cell=1,bwprob_i_totalcells
do 136 i_bwsec=1,i_totalbwsecs
if (rec_bwsecinfo(i_bwsec).c_number .eq.
&rec_cellinfo(i_cell).c_feature(1:5)) then
i_worst_color=max(rec_cellinfo(i_cell).i_color_by_volume,
&rec_bwsecinfo(i_bwsec).i_color_by_rate,
&rec_cellinfo(i_cell).i_color_by_unit_volume)
write(10,536) rec_cellinfo(i_cell).c_feature,
&rec_bwsecinfo(i_bwsec).i_area,
&rec_cellinfo(i_cell).c_cellid,
&c_sedtype(rec_cellinfo(i_cell).i_sedtype),
&rec_cellinfo(i_cell).r_acre_ft_peryear,
&c_colors(rec_cellinfo(i_cell).i_color_by_volume),
&rec_bwsecinfo(i_bwsec).r_cm_peryear,
&c_colors(rec_bwsecinfo(i_bwsec).i_color_by_rate),
&rec_cellinfo(i_cell).r_acre_ft_peryear_permeter,
&c_colors(rec_cellinfo(i_cell).
& i_color_by_unit_volume),
&c_colors(i_worst_color)
end if
136 continue
135 continue
536 format(a8,2x,i7,3x,a10,1x,a15,2x,
&4x,f10.5,2x,a6,
&4x,f10.5,2x,a6,
&4x,f10.5,2x,a6,
&11x,a6)

```

```

        close(10)

        stop
        end

*****
c
c -----
c
        subroutine heapsort1(n, ra, ca)
c          ra is the "control" !
          character*8 ra(n), rra
        STRUCTURE /bw_cellinfo/
          CHARACTER*8 c_feature
          character*10 c_cellid
          real r_probvalue
          real r_mass(12)
          real r_tonsperyear
          real r_acre_ft_peryear
          integer i_color_by_volume ! via acre_ft_peryear
          real r_acre_ft_peryear_permeter
          integer i_color_by_unit_volume ! via acre_ft_peryear_permeter
          integer i_sedtype
        END STRUCTURE
          record /bw_cellinfo/ ca(n), cca

c          !
          if(n .le. 1) then
            return
          end if

          l=n/2+1
          ir=n

10         continue
          if(l .gt. 1) then
            l=l-1
            rra=ra(l)
            cca=ca(l) !
          else
            rra=ra(ir)
            ra(ir)=ra(l)
            cca=ca(ir) !
            ca(ir)=ca(l) !
            ir=ir-1
            if(ir .eq. 1) then
              ra(l)=rra
              ca(l)=cca !
              return
            endif
          endif
          i=l
          j=l+1
20         if(j .le. ir) then
          if(j .lt. ir) then
            if(ra(j) .lt. ra(j+1)) j=j+1
          endif
          if(rra .lt. ra(j)) then
            ra(i)=ra(j)
            ca(i)=ca(j) !
            i=j
            j=j+j
          else
            j=ir+1
          endif
          goto 20
        endif
        ra(i)=rra
        ca(i)=cca!
        goto 10
      end
c

```

```

c -----
c
      subroutine heapsort2(n, ra, ca)
c      ra is the "control" !
      character*10 ra(n), rra
STRUCTURE /bw_sec/
  CHARACTER*8 c_feature
  character*10 c_cellid
  real r_width
END STRUCTURE
      record /bw_sec/ ca(n), cca

c      !
      if(n .le. 1) then
        return
      end if

      l=n/2+1
      ir=n

10    continue
      if(l .gt. 1) then
        l=l-1
        rra=ra(l)
        cca=ca(l) !
      else
        rra=ra(ir)
        ra(ir)=ra(l)
        cca=ca(ir) !
        ca(ir)=ca(l) !
        ir=ir-1
        if(ir .eq. 1) then
          ra(l)=rra
          ca(l)=cca !
          return
        endif
      endif
      i=l
      j=l+1
20    if(j .le. ir) then
      if(j .lt. ir) then
        if(ra(j) .lt. ra(j+1)) j=j+1
      endif
      if(rra .lt. ra(j)) then
        ra(i)=ra(j)
        ca(i)=ca(j) !
        i=j
        j=j+j
      else
        j=ir+1
      endif
      goto 20
    endif
    ra(i)=rra
    ca(i)=cca!
    goto 10
  end

c -----
c
      subroutine heapsort3(n, ra, ca)
c      ra is the "control" !
      character*10 ra(n), rra
STRUCTURE /bw_prob/
  CHARACTER*10 c_cellid
  real r_prob(11)
END STRUCTURE
      record /bw_prob/ ca(n), cca

c      !
      if(n .le. 1) then
        return
      end if

      l=n/2+1

```

```

        ir=n
10  continue
    if(l .gt. 1) then
        l=l-1
        rra=ra(l)
        cca=ca(l)    !
    else
        rra=ra(ir)
        ra(ir)=ra(l)
        cca=ca(ir) !
        ca(ir)=ca(l) !
        ir=ir-1
        if(ir .eq. 1) then
            ra(l)=rra
            ca(l)=cca !
            return
        endif
    endif
endif
i=1
j=l+1
20  if(j .le. ir) then
    if(j .lt. ir) then
        if(ra(j) .lt. ra(j+1)) j=j+1
    endif
    if(rra .lt. ra(j)) then
        ra(i)=ra(j)
        ca(i)=ca(j) !
        i=j
        j=j+j
    else
        j=ir+1
    endif
    goto 20
endif
ra(i)=rra
ca(i)=cca!
goto 10
end

```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

1. REPORT DATE (DD-MM-YYYY) August 2004		2. REPORT TYPE Interim report		3. DATES COVERED (From - To)	
4. TITLE AND SUBTITLE User's Manual for SEDLOAD Sediment Loading of Backwaters and Secondary Channels				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) Clay LaHatte and Stephen T. Maynard				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Coastal and Hydraulics Laboratory, U.S. Army Engineer Research and Development Center, 3909 Halls Ferry Road, Vicksburg, MS 39180-6199				8. PERFORMING ORGANIZATION REPORT NUMBER ENV 47	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Army Engineer District, Rock Island, Rock Island, IL 61204-2004 U.S. Army Engineer District, St. Louis, St. Louis, MO 63103-2833 U.S. Army Engineer District, St. Paul, St. Paul, MN 55101-1638				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT The SEDLOAD package is a group of programs designed to determine the sediment load delivered to the inlets of backwaters and secondary channels of a river due to river traffic, specifically tow traffic. This user's manual describes how to set up batch files, input and output file formats, example outputs, and source code listings in order to run the program.					
15. SUBJECT TERMS Backwaters Computer program			Navigation Sediment User's manual		
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT UNCLASSIFIED	b. ABSTRACT UNCLASSIFIED	c. THIS PAGE UNCLASSIFIED			74

