GRID_IO LIBRARY & UTILITIES

(SFWMM Technical Training Series)

August 13, 2002

Larry Brion, Dave Welter
and Ken Tarboton
(HSM, SFWMD)

GRID_IO Overview

- A general software library that can be used to manipulate (read, write and and search) 2-dimensional (2-D) data in binary (grid_io) format

- Originally developed at SFWMD by Bill Perkins (1991) and updated (dynamic memory allocation) by Randy VanZee (1993)

- First used in NSM and later on to SHEET-2D and SFWMM v2.1

- Grid_io: one of two binary formats used in some of the SFWMM input and output files

- The library is callable within C/C++ and Fortran programming languages

GRID_IO Basics

- Grid_io: an efficient way of storing 2-D data; stores information (in vector-form, i.e. 1-D array) for active nodes/cells in a 2-D grid

- Binary files created using the grid_io library consists of a header and a series of titled data set or snapshots

- Header - contains information about the data and grid configuration (title, number of rows, node vertical and horizontal size, total number of nodes, location of nodes in each row and cumulative count of nodes for each row)

- Snapshot - consists of the data title (tag) and an array of floating point values

GRID_IO Basics (cont.)

- Assumptions about data stored in grid_io format
  - grid cells are rectangular
  - node spacing is consistent throughout the area represented
  - all rows are contiguous (no gaps in active nodes within each row)
  - all grid cells specified by the grid definition are active and have corresponding data value/s

- Grid_io utilities are stand-alone applications used to facilitate manipulation of grid_io binary files

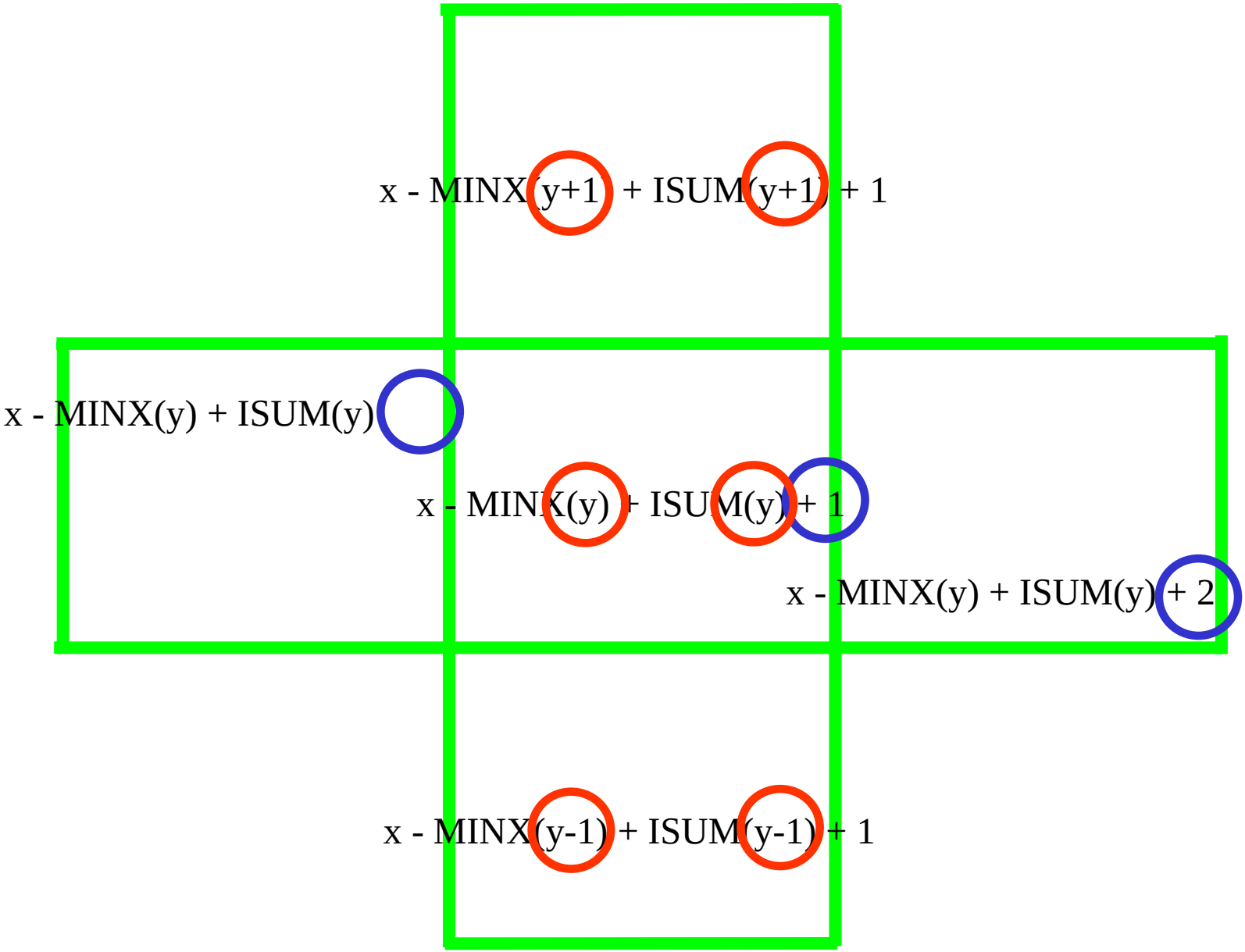
- Grid_io library routines are callable functions and subroutines used to create grid_io utilities

Grid_io Basics (Variable Definition)

- Pertinent variable names:
  - **ISUM(y) = total number of active cells below row y**
  - MAXX(y) **= maximum column number in row y**
  - **MINX(y) = minimum column number in row y**
  - MAXY **= maximum row number**
  - MINY **= minimum row number**

- Given rowno & colno in 2-D data array, find nodeno in grid_io data array
  - nodeno = ISUM(rowno) + colno - MINX(rowno) + 1
- Given nodeno in grid_io data array, find rowno and colno in 2-D data array
  - do y = MAXY, MINY, -1
    - if [ISUM(y) < nodeno] then
      - rowno = y
      - colno = nodeno - ISUM(y) + MINX(y) - 1
      - done
    - endif
  - enddo

Grid_io Basics (Node Numbering)



Using Grid_io Utilities

- cell_cat - produces the entire ASCII time-series of data pertinent to a cell or group of cells as defined in a binary input file in grid-io format

- cell_plot - functions similar to cell_cat but with plotting (using tsplot) capabilities

- cell_sum - produces ASCII tabular monthly and yearly sums of input binary data that is in grid_io format, for user specified cell/s

- gr_cut - produces a binary file in grid-io format containing an areal subset of an existing binary file in grid_io format; needs ASCII control file that defines extent of areal subset

Using Grid_io Utilities

- grid_freq* - produces a binary file in grid_io format that contains summary statistics from spatial data stored in grid_io format

- grid_peek* - produces the following text description of a binary file in grid-io format**:** header information, grid extents, and data title (usually date tags) of all grid snapshots

- grid_shot* - extracts in ascii format data (col-row-value) contained in grid_io format for a specified snapshot

- gr_summary* - produces several statistical summary files like annave, monave, annsum, monsum, etc. in grid_io format (except one .dat file in ASCII format) for daily OR monthly data stored in a grid_io format

Using Grid_io Utilities

- hydroperiod* - computes hydroperiods from daily ponding data stored in a grid_io data format. Output is also in grid_io format

- line_sum - like cell_sum, produces ASCII tabular monthly and yearly sums of input binary data that is in grid_io format, for user specified line of cell(s). In addition to row & column location of cell/s, user has to specify the direction of flow

- xgridview* - spatial display (graphical) of data stored in grid_io format file. one-directional spatial data
can be viewed as different colors and two-directional data can be viewed as vectors

- . . . et cetera

GRID_IO Utilities (misc.)

- addlake * - adds Lake Okeechobee (LOK) to a grid_io formatted binary file that does not include LOK areal extent. The output is also a grid_io binary file. (LB)

- cellcat2dss - reads cell_cat output and writes back to standard output in stoDSS format (DW)

- gr_bud - produces water budget for NSM (RV)

- gr_thsn - spatially interpolates time series data from a collection of points to cells defined in the grid_io file (RV)

- grid2gms - converts data from grid_io format to GMS format (RV)

- grid_angle - Computes the resultant angle in a 2-directional grid-io file (KT)

grid_angle

grid_angle - Version 1.1

```
**************************************************
* grid_angle computes the angle of the resultant   *
* vector for grid files A and B,  & the difference *
* in grids (generated in Perkins grid_io format),  *
* C is a resultant grid in the same binary format. *
**************************************************
```

Enter Binary Grid A filename(in quotes):

Enter Binary Grid A filename(in quotes):

"/vol/hsm1/data/sfwmm/RESTUDY/ALT6/OUT_AD13R_NOV98/MAPS/OVFLO/ovflow.annave”

Enter number of initial grid A snapshots to skip:

0

Enter Binary Grid B filename(in quotes):

"/vol/hsm/data/nsm/output/nsm45/MAPS/OVFLO/ovflow.annave"

Enter number of initial grid B snapshots to skip:

0

Enter Binary Grid C filename(in quotes):

"grid_angle.bin"

Enter Binary Grid C title(in quotes):

"ANGLE DIFFS (D13R-NSM45)"

Enter number of snapshots to process:

1

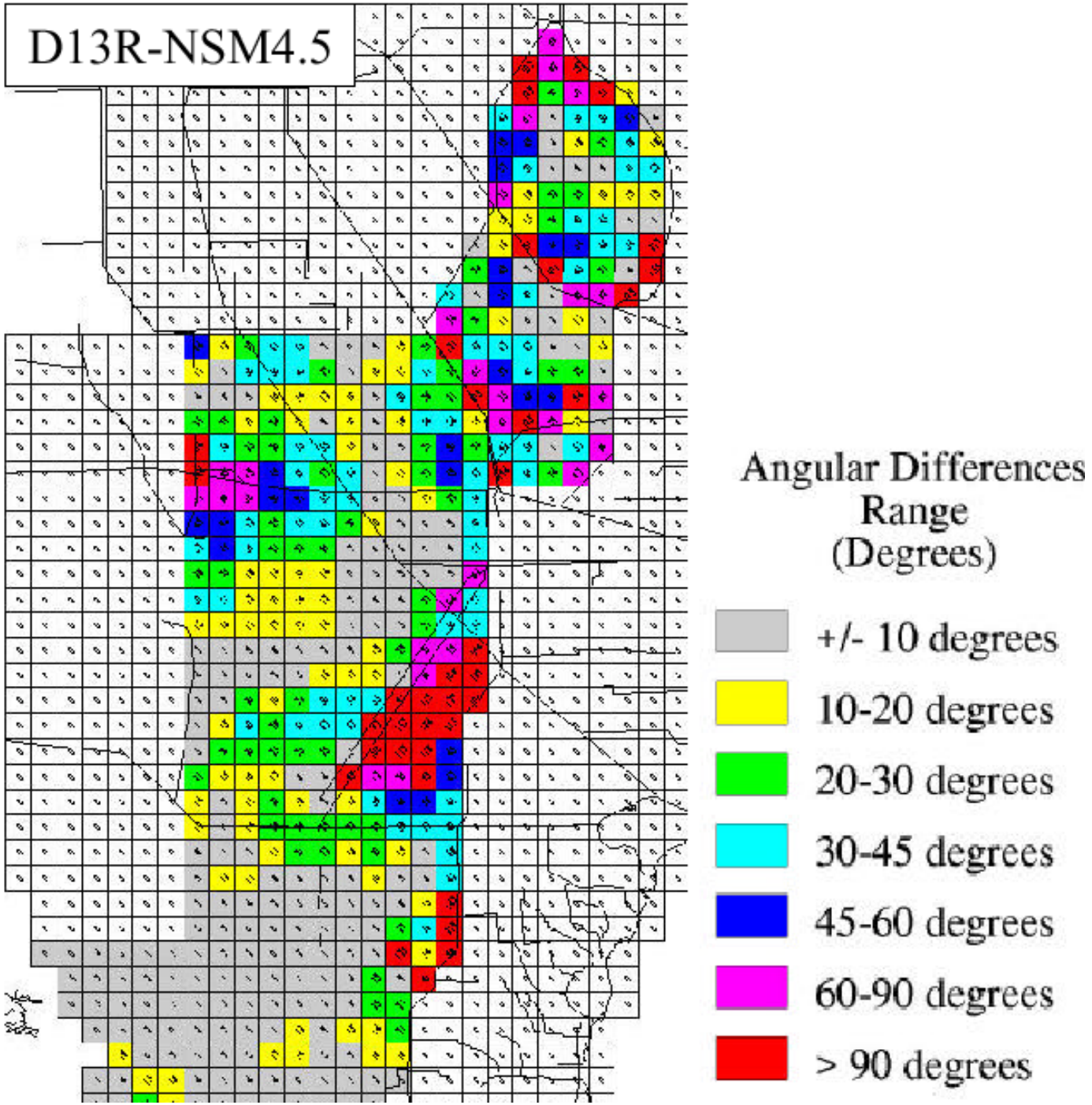
grid_angle.cf

```
"/vol/hsm1/data/sfwmm/RESTUDY/OUT_AD13R_XL672/MAPS/OVFLOW/ovflow.annave”
0
"/vol/hsm/data/nsm/output/nsm45/MAPS/OVFLO/ovflow.annave”
0
“grid_angle.bin”
"ANGLE DIFFS (XL672-NSM45)"
1
yes
```

gr_angle_epa.scr

```
# grid_angle < grid_angle.cf
gr_cut -d /newhomes/ktarbot/geog/cookie_cutters/epa.area -i
    grid_angle.bin -o grid_angle_epa.bin
grid_shot -n3 grid_angle_epa.bin > grid_angle_diff_epa.roco
awk 'BEGIN {print "-----------------------------------"} NR > 1 {if($3 < 0)
    {sum -= $3} else {sum += $3}} END {print "Average angular
    difference =", sum/(NR-1) }' grid_angle_diff_epa.roco


gr_angle_epa.scr
-----------------------------------
Average angular difference = 24.3624
```


D13R-NSM4.5
Angular Differences
Range
(Degrees)
+/- 10 degrees
10-20 degrees
20-30 degrees
30-45 degrees
45-60 degrees
60-90 degrees
> 90 degrees

GRID_IO Utilities (misc.)

- grid_freq* - computes summary statistics for data stored in a grid_io file (RV)

- grid_hpimp - compares hydroperiod improvement w.r.t. NSM for two simulations (LC)

- grid_lmscale - summarizes a monthly grid_io file (1-d or 2-d) into a long-term 12-month-snapshot grid_io file for selected water years (LB)

- grid_math - performs simple mathematical operations (+,-,*,/) on two grid_io files (of same spatial extent) and produces a third grid_io file in the same format. Needs ASCII control file that defines grid_io files on which to perform operation, etc. (CN)

GRID_IO Utilities (misc.)

- grid_ts_concat - concatenates (combines snapshots) two grid_io files that are spatially identical (same header information) and produces another grid_io file (LB)

- grid_ts_cut - takes a subset of snapshots from an existing grid_io file and writes data into a new grid_io file (LB)

- gridvel - computes velocities in the x- and y- directions (ft/month) given the monthly oveland flow and mean monthly ponding depth grid_io files; produces another grid_io file (CN)

- ts2wmmgrid - store values of SFWMM grid cells from ASCII to grid_io format (LB)

Using Grid_io Library Routines

Fortran Interface

- getgrid/setgrid - get or set grid header information for grid_io file

- gridrhd - reads header information from a grid_io file
- gread - read an areal data set or snapshot from a grid_io data file

- gridwhd - writes header information to a grid_io file
- gwrite - write an areal data set or snapshot to a grid_io data file

- gridskip - skip an arbitrary number of grid data sets or snapshots in a grid_io file
- grid_bottom - move to bottom (position of last snapshot)of a grid_io file
- grid_top - move to top (position of first snapshot)of a grid_io file

Grid_io File Structure

HEADER

Title - AltD13R (RESTUDY)
number_of rows - 65
number_of_nodes - 1746
size_x - 10560
size_y - 10560
xstart (array) - ...
xend (array) - ...
cum_node_count (array) - ...

Record

tag - January 1, 1965
data (array) - ...

Record

tag - January 2, 1965
data (array) - ...

Grid_io C Interface

- `int grid_read_header(FILE *file, GRID *grid)`
- `int grid_write_header(FILE *file, GRID *grid)`
- `int grid_write(FILE *file, GRID *grid, char *tag, float *values)`
- `int grid_read(FILE *file, GRID *grid, char *tag, float *values)`
- `int grid_skip(FILE *file, GRID *grid, int count)`
- `grid_top(FILE *file, GRID *grid)`
- `grid_bottom(FILE *file, GRID *grid)`
- `int grid_count_snapshots(FILE *file, GRID *grid)`
- `int grid_tag_search(FILE *file, GRID *grid, char *string)`
- `int grid_node(GRID *grid, int row, int column)`
- `int grid_free(GRID *grid)`

Grid_io FORTRAN Interface

- `call gridrhd(fd, errs)`
- `call gridwhd(fd, errs)`
- `call gwrite(fd, tag, values, errs)`
- `call gread(fd, tag, values, errs)`
- `call gridskp_(fd, cnt, errs)`
- `call setgrid_(title, nrows, nnodes, xsize, ysize, xstart, xend, cum_count)`
- `call getgrid_(title, nrows, nnodes, xsize, ysize, xstart, xend, cum_count)`
- `call openfilef77(unit, filename, access)`
- `call closefilef77(unit)`

Variable Types

```
INTEGER fd, errs, nrows, nnodes, unit, xstart(NROWS), xend(NROWS),
   cum_count(NROWS)
REAL xsize, ysize, values(NNODES)
CHARACTER*80 tag, title, filename, access
```

FORTRAN Example

```
PROGRAM FTEST
INTEGER ierr
character*80 tag
real values(55000)
character*80 title_et_in
real x_size, y_size
INTEGER nrows, num_lec_nodes
integer isum_lec(75), max_x_lec(75), min_x_lec(75)
call openfilef77(1, "daily_stg_minus_lsel.bin", "rb")
call gridrhd(1, ierr)
call getgrid(title_et_in, nrows, num_lec_nodes, x_size, y_size,
  +          min_x_lec, max_x_lec, isum_lec)
print*, title_et_in
print*, nrows
print*, num_lec_nodes
call gread(1, tag, values, ierr)
print*, tag
call gread(1, tag, values, ierr)
print*, tag
call gread(1, tag, values, ierr)
print*, tag
call closefilef77(1)
stop
end
```

Program Output

```
ALT D13R (RESTUDY)
71
1922
January  1, 1965
January  2, 1965
January  3, 1965
```

C Example

```
#include <stdio.h>
#include "grid_io.h”

main(){
 FILE *file = stdin;
 char tag[GRID_TAG_LENGTH];
 int i, test, done = 0;
 char c;
 GRID grid;
 float values[2500];
 file = fopen(“daily_stg_minus_lsel.bin”, "rb");
/* read the file header */
 test = grid_read_header(file, &grid);  print_header(grid);
printf("Run title:  \"%s\"\n", grid->header.title);
printf("Number of rows: %d\n", grid->header.number_of_rows);
printf("Number of nodes: %d\n", grid>header.number_of_nodes);
/* read snap shot */
 test = grid_read(file,
                  &grid, tag, values);
 printf("tag: \"%s\"\n", tag);

 fclose (file);
 grid_free(grid);
}
```

Program Output

Run title: "ALT D13R(RESTUDY)”
Number of rows: 71
Number of nodes:1922
Size: 10560 by 10560
tag: "January 1, 1965"

THANKS !

(SFWMM Technical Training Series)

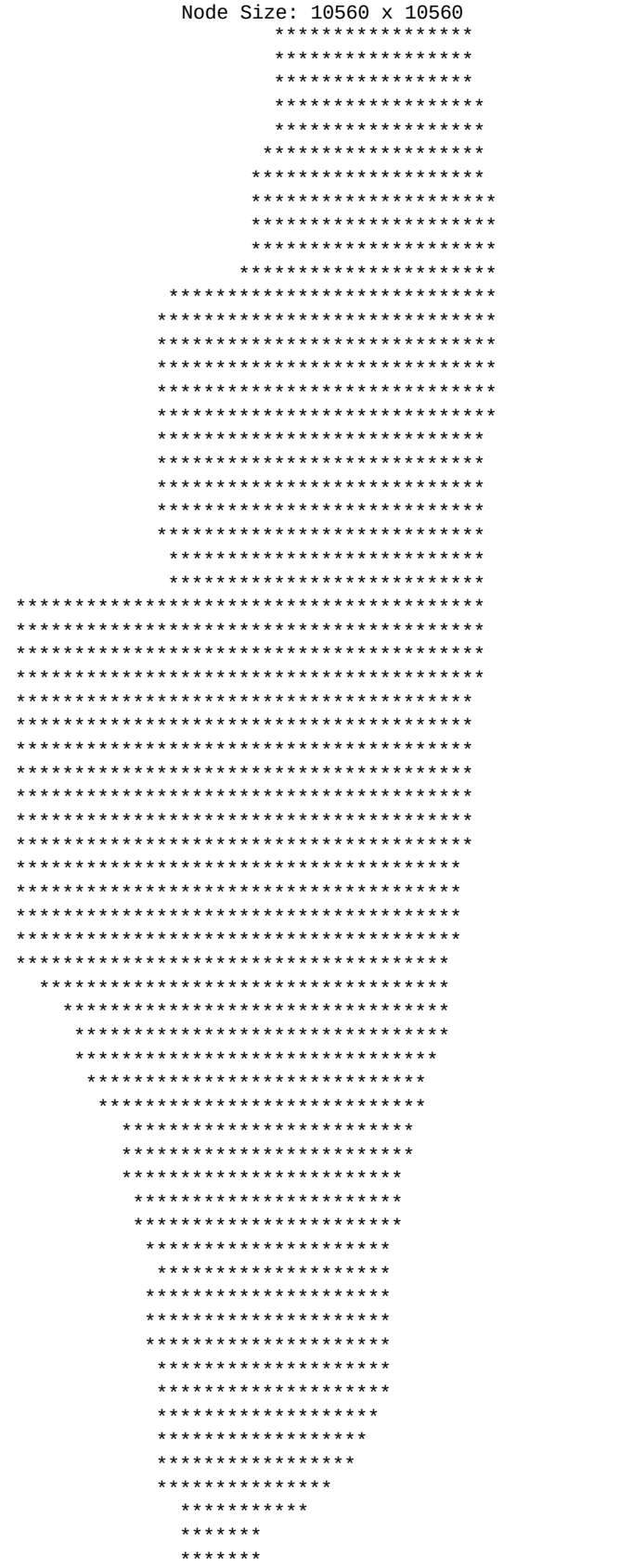
August 13, 2002

Larry Brion, Dave Welter
and Ken Tarboton
(HSM, SFWMD)

Usage grid_peek [-l] [file]

memo:> **grid_peek** -l stage.bin

```
Data File: "stage.bin"
               Run Title: "ALT D13R NOV98 (RESTUDY)"
               Number of Rows: 65
               Number of Nodes: 1746
               Node Size: 10560 x 10560
```



```
  0: January  1, 1965
  1: January 31, 1965
  2: February 28, 1965
           :
           :
           :
371: November 30, 1995
372: December 31, 1995
```

```
Usage line_sum [-o output] [-i cell_list_input] [-d daily output] [-L length]
               [-D #decimals] [-z dssfile] [input]
       -i = list of cells (row, column, direction)
       -z = name of the ascii file to transfer output to dss
       -d = write out DAILY output also (assumes input is daily)
memo:> cat 40mibend.in
22 17 s r
22 18 s r
22 19 s r
22 20 s r
22 21 s r
memo:> line_sum -i 40mibend.in surface_flow.bin
```

Year	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	TOTAL
1965	34.7	17.3	9.9	1.3	0.2	0.4	1.3	6.3	25.8	53.1	60.6	36.8	247.6
1966	23.0	11.8	7.2	4.9	3.9	18.9	97.8	154.5	147.0	137.2	110.3	60.8	777.5
1967	30.1	15.5	8.0	3.7	1.0	10.2	22.4	54.8	52.0	97.1	78.8	41.9	415.5
1968	22.9	10.3	5.6	1.8	6.3	43.0	121.5	174.2	106.1	112.1	84.0	44.7	732.6
1969	32.1	20.2	14.7	12.0	11.9	68.0	84.7	91.7	86.5	104.2	182.4	151.6	860.0
1970	118.5	80.7	101.0	138.9	92.1	114.4	121.3	114.2	80.2	68.7	37.6	19.1	1086.7
1971	9.5	4.6	1.6	0.3	0.0	1.6	2.2	3.6	10.2	21.2	26.3	20.0	101.1
1972	14.1	10.3	4.1	4.0	6.4	14.3	22.1	31.1	35.3	27.6	14.1	7.4	190.7
1973	5.1	2.2	1.2	0.5	0.2	0.2	4.1	8.9	21.1	34.9	17.5	9.0	105.0
1974	5.0	1.3	0.1	0.0	0.0	0.8	4.5	45.2	59.9	56.0	34.7	33.4	240.9
1975	17.4	8.5	3.2	0.2	0.1	1.0	9.3	21.7	39.3	82.0	49.5	24.4	256.5
1976	10.6	4.8	3.5	0.9	0.8	12.4	31.7	58.5	61.6	59.6	36.2	18.1	298.8
1977	10.1	4.0	1.7	0.4	2.7	2.4	1.2	8.7	24.6	32.8	23.4	17.3	129.3
1978	10.6	8.2	12.1	7.3	3.6	5.7	24.6	32.9	70.6	88.4	61.4	35.7	361.0
1979	35.0	22.8	14.4	6.9	16.2	9.6	15.9	11.0	17.4	50.3	49.2	52.2	300.8
1980	55.8	47.8	44.7	45.6	28.6	18.9	19.6	24.5	28.7	26.5	21.3	13.2	375.1
1981	6.9	4.3	2.3	0.8	0.2	0.1	1.1	14.6	55.2	72.0	47.4	19.0	223.9
1982	7.2	3.5	2.3	1.8	2.1	40.6	97.1	93.6	53.5	65.9	59.7	30.6	458.0
1983	20.4	34.3	79.4	80.8	45.4	69.6	51.2	51.4	80.4	55.5	40.6	39.4	648.3
1984	37.7	24.1	22.2	12.1	9.9	24.7	35.7	26.3	26.2	31.8	19.1	12.4	282.1
1985	7.8	3.5	1.4	0.3	0.1	-0.0	5.7	40.7	59.5	71.2	56.7	30.9	277.7
1986	22.0	10.7	10.2	7.1	3.2	7.0	22.7	58.5	53.9	33.1	21.7	15.9	266.0
1987	16.7	10.1	20.1	11.1	5.3	2.6	2.0	9.2	15.0	20.3	24.9	18.0	155.2
1988	13.2	7.6	3.6	1.2	0.4	13.9	18.6	32.7	41.6	32.6	11.5	6.2	183.0
1989	2.7	0.7	0.1	0.0	0.0	0.0	0.5	1.6	4.8	12.6	9.1	5.8	37.8
1990	2.6	1.0	0.4	0.0	0.1	1.5	5.7	14.2	14.1	15.9	11.7	5.4	72.7
1991	2.9	1.2	0.8	0.3	5.5	14.2	58.7	77.8	80.1	97.8	64.2	49.8	453.4
1992	40.4	30.4	24.0	18.9	9.1	38.4	92.4	109.9	122.2	77.8	51.3	39.3	654.0
1993	43.3	46.4	52.2	39.0	23.0	38.8	29.8	32.0	56.9	135.3	88.4	40.4	625.5
1994	25.9	15.5	13.1	6.0	5.0	11.3	21.3	53.2	91.2	172.1	214.1	256.8	885.6
1995	261.2	182.9	140.7	73.9	47.0	70.9	75.9	117.9	200.1	273.0	219.3	144.6	1807.4
											AVERAGE		435.8