Hydrology Lab's Research Modeling System (HLRMS)

(Modified on 14:15am 6/12/02)

1. Introduction

As a DMIP initiative, HL has developed the first version of a Research Modeling System (HLRMS) that combines lumped and distributed model features. While the system has a grid-based structure, it can be run in lumped or semi-distributed modes. The main goal of the HLRMS was to generate a flexible system that can easily incorporate different parameterizations of rainfall-runoff and routing processes. All parameterizations should be identifiable based on GIS and hydrological data.

2. System structure and hydrologic models

The first version of the HLRMS is consistent with an HRAP projection, such as computational elements are defined on a sequence of HRAP pixels. Each element consists of a rainfall-runoff component (in version 1 - SAC-SMA model) and a routing component (in version 1 - hillslope/channel kinematic wave model). Rainfall-runoff component generates fast (surface) and slow (subsurface/ground) runoffs. Within each element, a fast runoff is routed over conceptual hillslope to a channel, and then channel inflow combined with a slow runoff component and upstream pixel outflows is routed through a pixel conceptual channel. A conceptual hillslope consists of a number of uniform hillslopes (a number of hillslopes depends on a stream channel density of a pixel). A conceptual channel usually represents the highest order stream of selected pixel. Cell-to-cell connectivity sequence must be determined to move water from upstream to downstream elements and to a basin outlet. An implicit numerical scheme is used for kinematic equation solution.

Data format for input variables and model parameters are similar to 'xmrg' binary type grids with a minor difference in the header. All parameter grids are binary files with two header records. The first record is the same as the 'xmrg' record that consists of X- and Y-orifins of HRAP, and numbers of columns and rows in the file. The second record differs from 'xmrg' records. It includes two variables: scale factor to convert integer*2 values into real values, and number of bytes per value (2 for integer*2 format, and 4 for real*4 format). This allows to keep grids in integer*2 or real*4 formats. In the first run, the model will read archived 'xmrg' files directly. In the same time, a separate new binary file will be created for each selected non-nested basin. These new binary files contain precipitation data only for those grids contained in the selected basins. It will greatly reduce the run-time for further model test runs. State variables will be saved as the same format as newly created precipitation data except that they are corresponding to a specific time. During the run time, all grids are converted into one-dimensional arrays in a pixel connectivity order. A number of separate or nested basins can be processed simultaneously. Hydrographs at selected outlets are stored in an OH card format. Selected grids (input variables, states, water balance components) can be stored in an Arc/Info format for a selected period with desirable time interval.

Input variables. Precipitation 'xmrg' hourly grids. Climatological monthly evapotranspiration demand and PE adjustment factor grids. **NOTE:** Hourly/daily potential evaporation grids can be also used if available.

Initial state variables. Six SAC-SMA upper-lower zone states (UZTWC, UZFWC, LZTWC, LZFSC, LZFPC, ADIMPC), hillslope routing state (water depth of each pixel), and channel routing state (channel cross-section at each pixel) grids. Program will save all the states for the last time step in each run. Be noticed that in the current version, only the states for selected basins are saved, all values outside the selected basin region are set to -1.0.

Parameters. 16 SAC-SMA parameter grids. Hillslope slope, S_h, roughness coefficient, n_h, and stream channel density, D, grids. Channel slope, S_c, roughness coefficient, n_c, shape parameter, b, and top width parameter, a, grids. Shape and top width parameters are defined based on a relationship between channel top width and depth, ***B=a*H**b***.

The program converts hillslope parameter grids into one basic hillslope parameter grid, a specific hillslope discharge ($q_{0,h}$) per a unit depth, based on Chezy-Manning's equation:

$$q_{0,h} = 2k_q D \frac{\sqrt{S_h}}{n_h}$$

where $k_q = 10^5$ is a unit transformation coefficient. A unit of hillslope specific discharge is $mm^{-2/3}.s^{-1}$ if D is in km^{-1}.

Four channel grids are converted into two basic channel parameter grids, a specific channel discharge, $q_{0,c}$, per a unit channel cross-section, and a power value, m_c, in relationship between discharge and cross-section:

$$q_{0,c} = a^{-\frac{2}{3(b+1)}} (b+1)^{-\frac{2b}{3(b+1)}} \frac{\sqrt{S_c}}{n_c}$$

$$m_c = \frac{b + \frac{5}{3}}{b+1}$$

A unit of channel specific discharge is $m^{3-mc}.s^{-1}$ if a is in meters.

Grid replacement and adjustment. All variable and parameter grids except precipitation can be replaced by some uniform values for each selected basin/subbasin. They can be also adjusted by some ratio for each selected basin/subbasin. An information on replacement or adjustment is provided in an input deck of the HLRMS (see input deck description below).

3. Source code description

Currently about 42 subroutines (20 C subroutines and 22 FORTRAN subroutines) have been developed for the HLRMS. These subroutines do not include other preprocessors that are needed to prepare input data for the hydrologic simulations. Table 1 lists calling structure between subroutines. The function and parameter list for each subroutine are described in the Appendix I.

There are five major components consisted in the RMS. These components deal with data reading, data processing, runoff simulations, channel routing, and data output. Detailed components and related subroutines are list as follows:

1). Reading and processing an input deck which defines all necessary information in order to run the model. See **System Execution** for input deck details.
rd_deck1.f, rd_deck2.f, rd_card.f, slt_var.f

2). Read and process a channel connectivity sequence, and rearrange input deck arrays.
init_rut1.f, init_rut2.f, rd_hed_seq.f, vec_seq1.f, reorder.f

3). Read and replace/adjust rainfall-runoff and routing parameters.
read_runf_par.c, read_rut_par.c, read_one_par.c, read_new_xmrg.c

4). Read and replace/adjust model states.
read_rut_st.c, read_one_par.c

5). Generate simulation time loop, read xmrg files, and calculate ET demand.
xmrg_path.f, rd_hed_hrp.f, read_xmrg.c, get_ped.c, fill_miss.c

6). Run rainfall-runoff model (SAC-SMA).
do_sac.c, fland1.f

7). Run hillslope and channel routing models.
do_route.f, hslope.f, hstrem.f

8). Generate output time series and store selected grids. Arc/Info grids can be created for all/selected parameters as well as states (for a specified time) and precipitation (for a specified time period).
output_aigrid.c, write_avgrid.c, delete_avgrid.c, save_states.c, write_ave_pcp.c

9). Save states for the last hour. States are named as their original names plus the time when the states were saved. In order to read these states for next run, the start time should be the immediate next hour to the hour that states were saved. Routing states are now split into two group of states: channel routing states (cross section for all sub-reaches, km^2), and hillslope states (depth in mm).
save_states_f.c

4. System execution

How to run:

To run the program, issue '***hlrms.exe <input.deck name>***' where ***hlrms.exe*** is the executable program, and <***input.deck name***> is a file that defines simulation time period, parameter values/scale factors for each basin, input grid file paths, output path, states path, and how the program will be run, etc.

Input Deck Entries:

The input deck file contains many entries defined as symbol **@** plus a letter followed entry values appropriate for this entry. It uses a free format input for all cards (Description of these free format input rules is in the Appendix II). There is no particular order for most entries. Those parameter entries corresponding to a specific basin (defined in the entry ***@I***) have to be placed after each basin's entry, ***@I.*** Again, there is no required order in giving parameter entries for each basin. Each entry is described as following:

@A, defines start and end simulation time (in mmddyyy hour), time step in hours, precipitation mode (1, distributed-default or 0, lumped precipitation), and index for whether runoff simulation is conducted (=1) or only process parameters (=0), what precipitation averaging scale will be used (based on HRAP grid scale, e.g. if input is 3, then the precipitation will be average by 3x3 HRAP grids), whether or not create Arc/Info grids for SAC parameters, routing parameters, monthly PE, monthly PE adjustment, initial SAC states, and initial routing states (1=true, 0=false).
e.g. *@A 06011993 0 07312000 23 1 1 1 2 0 0 0 0 0 0*

@C, One line of comment (without space between letters) about this run.
e.g. *@C This_run_was_based_on_lumped_calib_param.*

@O, path to output files.
e.g. *@O /bulk/1/SAC_dst_data*

@P, path to model parameter grid files.
e.g. *@P /fs/home/vkoren/devl/parameters/hrap*

@Q, path to sequence file.
e.g. *@Q /fs/home/vkoren/SEQ_FILES/abrfc.seq*

@S, path to 1d-xmrg files.
e.g. *@S /fs/home/vkoren/devl/1d_xmrg*

@X, path to original xmrg files and number of characters denoting year.
e.g. *@X /fs/hydro/Hydro_Data/ABRFC/PRECIPITATION/RADAR/STAGE3/RAD_97-99 4*

@V, time period, interval and indicators whether or not to create Arc/Info grids for precipitation, ET, surface flow, subsurface flow, and routing flow. (Indicators: 1 is true, 0 is false and is default. There are five indicators for precipitation, ET, surface flow, subsurface flow, and routed flow respectively, one for six SAC states, and one for two routing states). The time format is 'mmddyyy hh' where hh (hour) should be in the range of 0-23.
e.g. *@V 06081993 15 06081993 23 1 1 0 0 0 0 0 0*
NOTE: When output of Arc/Info grids is desired, the output directory must all be lower case!

@F, runoff model ID, number of parameter grids, number of model parameters, and number of states.
e.g. *@F sac 16 16 6*

@G, routing model ID, number of parameter grids, number of model parameters, and number of states.
e.g. *@G rutpix 7 3 2* Or *@G rutpix 9 3 2*

@W, longitudinal and latitude of sub-window. LON_{left}, LON_{right}, LAT_{up}, LATdown.
e.g. *@W 97.0 94.0 37.0 33.3*
(***This entry is no longer necessary. Program will determine the window automatically***.)

@I, five-letter basin ID and an indicator whether it is a part of a nested basins: 0 if nested basin; otherwise means non-nested basin with an indicator equals a number of nested basins (including itself) in this basin,

e.g. *@I BLU18 0*	*Means nested basin*
@I BLUO2 2	*Non-nested basin that consists of two sub-basins*

It is followed by following entries,
+ISRT, routing states entry: Channel cross-sections, *AREAC*; Hillslope water depth, *DEPTH*
e.g. *+ISRT 5.0 0.*

+ISRF, SAC-SMA states entry as ratios of model parameters, uztwc uzfwc lztwc lzfsc lzfpc adimpc
e.g. *+ISRF .75 0.14 0.56 0.11 0.46 1.0*
Or *+ISRF -1 -1 -1 -1 -1 -1* (When state grids are exist)

+IPRF, SAC parameter entry
UZTWM UZFWM UZK PCTIM ADIMP RIVA ZPERC REXP LZTWM LZFSM LZFPM LZSK LZPK PFREE SIDE RSERV
e.g. *+IPRF -0.442 -0.99 -1.096 0.004 0. 0.028 -0.812 -1.056 -1.04 -0.87 -1.044 -1.302 -1. -1.11 0. 0.3* (should be in one line)

+IPRT, routing parameter entry: channel slope, *SLOPC*; channel roughness coefficient,

ROUGC; channel shape parameter (exponent), *BETAC*; channel shape parameter (top width at depth = 1), *ALPHC*; hillslope slope, *SLOPH*; channel density, *DS*; hillslope roughness coefficient, *ROUGH*
e.g., *+IPRT 0.0005 0.05 1.2 2.0 0.002 1.5 0.35*

+IPE, monthly PE entry (from Jan to Dec)
e.g. *+IPE 1.19 1.33 1.84 2.63 3.70 4.91 5.21 4.33 3.49 2.52 1.67 1.25*

+IPEA, monthly PE adjustment entry (from Jan to Dec)
e.g. *+IPEA 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0*

<u>Note</u>: If any one of the parameter value entries is missing, then their entry values are defaulted as -1. The values for each entry can be given as either positive or negative ones separated by a space(s).

If a value is **positive**, then program will use this value for that parameter instead of reading grid file.

If it is **negative**, then program will read girdded data file with the negative value being used as multiplies to the grid values. In this case, program will exit if no grid for the parameter is found. Value **-1** can be seen as just use grid value only since it is multiplied by 1.

Gridded files are in binary format. They have two lines of header with first line defining window info in HRAP and the second line defining a scale factor to be divided by the data to get real values. They are similar to XMRG files.

+IUH, unit hydrograph entry: number of UHG ordinates followed by UHG values
e.g. +IUH 7 1.5 4.5 9.5 15.5 12.0 6.5 2.0
Note: If there is no entry for a basin, then distributed channel routing be used as default

Entry ***@I*** and its parameter entries are repeated for each basin. It allows to replace/modify each basin/subbasin differently.

@STOP, End of input deck entries

5. Time Series & Grid Outputs

Several types of outputs can be created in the defined output path depending on user's choices.

1). Parameters:
sac_par.out, SAC parameter values for each grid of all selected basins. It also contains each basin averaged values and whole basin averaged values. The number of

grids/pixels and area for each basin are also included. All values are arranged for each basin.

sac_st.out, SAC states for each basin. Data are arranged in the same way as above.

rut_par.out, Routing parameters. Data are arranged in the same way as above.

rut_ch_st.out, Channel routing states (cross section m^2) for sub-reaches. Data are arranged in the same way as above.

rut_hill_st.out, hillslope routing state (depth mm) for each segment.

pe.out, Monthly PE values arranged in the same way as above.

pe_adj.out, Monthly PE adjustment values arranged in the same way as above.

The above six files are created automatically each run.

.out.1d, *.out.2d, *.whole_win, are output for individual parameter in 1-D, 2-D grids, and whole window. The * represents letter ID and parameter name. The outputs here are optional and are mainly for data checking.

2). Time series:

.hyd, ***.pcp***, Hydrograph output and local MAPX (in OH format) for each basin defined in the input card . * represents five letter basin ID.

.mapx, OH format MAPX time series for basins having nested sub-basins selected. * represents five letter basin ID.

3). Grids:

*+***x***, Created Arc/Info grids for xmrg. * represents time in hours. x is appended.

*+***st_name***+***x***, Created Arc/Info grids for states. * represents time in hours. ***st_name*** is state variable name.

*+***x***, Created Arc/Info grids for parameters. * represents parameter name.

st_name+_***, Created state variables in binary grid format for the last hour *.

4). Other:

debug.info, A file for debugging purpose. It contains initial parameters given in the input deck, file checking info, etc. It is automatically created in the directory where the program is run.

.xmrg, a binary file containing rainfall data only for those grids within the selected basin. * represents a basin's ID. The new xmrg data will created only for basins that the number after the @I entry is not zero. New data will automatically appended if the end time of simulation is later than the end time defined in the 1d xmrg file. If multiple basins are selected, the program only read 1d xmrg data for the common period when 1d xmrg data for each basin covers this period. The original xmrg data will be read for the rest of simulation. If any basin's 1d xmrg does not exist, the program will read original xmrg data for whole simulation period. In other words, the program will not read mixed xmrg data; it will either read original xmrg files or read 1d-xmrg files. Since the program will append new data to the 1d xmrg files if necessary, a user needs to make sure the file permission is set as writable for group/other. (Same applied to directories)

6. Known Limitations and Bugs

1). Length of each line in input deck should not exceed 128 characters
2). Basin name in input deck should not exceed 6 characters
3). Maximum number of selected pixels for all basins should not exceed 50000
4). Maximum number of columns in selected window should not exceed 7000
5). Maximum number of basins/subbasins should not exceed 500
6). Default maximum length of channel routing subreach is 2000m, and maximum number of subreaches of one pixel should not exceed 4.
7). Cumulative flow comparison after the simulation incorrect for parent basins of nested basins.
8) new data can be appended to 1D xmrg files, they cannot be inserted.

Table 1. Major Program Calling Structure and Short Explanations

A. Calling Structure

Main Program & Subroutines

main_rut.c

```
    |
    |--- rd_deck1.f -|--- rd_card.f
    |
    |--- setdefaults.c
    |
    |--- rd_deck2.f --|--- rd_card.f
    |
    |--- init_rut1.f
    |
    |--- init_rut2.f
    |
    |---  read_runf_par.c -|
    |                      |
    |---  read_rut_par.c  --|--- read_one_par.c
    |                      |
    |---  read_rut_st.c   ---|
    |
    |— output_par.c
    |
    |---  xmrg_path.f
    |
    |---  read_xmrg.c –|– fill_miss.c
    |
    |— read_new_xmrg.c
    |
    |— output_aigrid.c
    |
    |---  basin_average.c
    |
    |---  get_ped.c
    |
    |---  do_sac.c   --|-- fland1.f
    |                --|-- save_states.c
    |— get_tci.c
    |
    |— uhg_to_hg.c
    |
    |---  do_route.f -|--- hslope.f
    |                 -|--- hstream.f
    |— save_states_f.c
    |
    |— write_ave_pcp.c
```

B. Short Explanations:

- **main_rut.c:** main program to call other subroutines. It outputs hydrographs for selected basin outlets.
- **rd_deck1.f**: Reads an input deck to get basic input information.
- **rd_card.f**: Reads an input card from an input deck.
- **setdefaults.c**: Sets initial value as -1 for several parameters.
- **rd_deck2.f**: Reads an input deck second time (after '*rd_deck1.f*' to get input information for each defined outlet.
- **init_rut1.f**: Reads channel connectivity file and selects all connected pixels for defined outlets.
- **init_rut2.f**: Redefines HRAP coordinates and sequence numbers into relative coordinates of the selected sub-window. It is also calculate the distance interval for the routing model.
- **read_runf_par.c**: Reads parameter values for runoff process based on grids or values defined the input deck.
- **read_rut_par.c**: Reads routing parameter values for routing process based on grids or values defined in the input deck
- **read_rut_st.c**: Reads routing state values for selected basins. Whether to read grids or use provided values depends on values given in the input deck**.**
- **read_one_par.c**: Reads one gridded parameter file in binary format and creates a 1-D array of values.
- **output_par.c:** Output parameter values for each grid within each basin.
- **output_aigrid.c:** Creates a Arc/View grid for given parameters in a defined region.
- **xmrg_path.f**: Returns 'xmrg' file name before actual reading.
- **read_xmrg.c:** Reads radar xmrg data in each time step.
- **read_new_xmrg.c**: Reads newly created precipitation data for each time step for faster processing.
- **basin_average.c:** To get a basin averaged value based on gridded info if necessary.
- **get_ped.c:** To get/compute hourly PE for each grid based.
- **do_sac.c**: Conducts runoff simulation using SAC-SMA model.
- **save_states_f.c**: Saves states at the last time step.
- **get_tci.c:** To get total channel inflow for basins having UHG defined
- **uhg_to_hg.c**: Convert unit hydrograph to flow at a basin's outlet
- **do_route.f**: Main subroutine to prepare input information and to call kinematic routing models.
- **hslope.f**: Performs hillslope routing based on kinematic wave model.
- **hstream.f**: Perform channel routing for a conceptual pixel channel based on kinematic wave model.

7. References

Gorbunov, Yu. V., 1971. Calculation of a stream network water storage based on morphological laws, *Meteorologiya and Hydrologiya*, No. 2, 57-67 (In Russian).

Tokar, A. S., and P. Johnson, 1995. Optimization of river characteristics in the Blue Nile watershed, *NFS 3.0 Reference manual, No 121.*

Manual on Hydrological predictions, 1989. *Hydrometeoizdat, Leningrad, vol. 2. (In Russian).*

Willemin, J. H., 2000. Hack's law: Sinuosity, convexity, elongation, *WRR, vol. 36, No. 11, 3365-3374.*

Appendix I: Subroutines and Parameter Lists

main_rut.c

This is the main program. It defines all variables that need to be passed to and from subroutines. All major subroutines including those that are reading/processing parameters outside the time loop and those that reading xmrg data and conducting rainfall-runoff simulations within each time loop are called directly by ***main_rut.c***. Besides calling different subroutines, some important information is generated within this program. It outputs parameter info for grids within selected basins. It also write out hydrographs and MAPX files for each basin in OH format that can be read directly by mcp3/ICP.

setdefaults.c

Sets initial value as -1 for several parameters.
Parameter list:
  IN:
    np, number of parameters for a process, e.g. 16 for SAC-SMA
    nbasin, number of basins

  OUT:
    pcvalue, an 1-D array of parameter values for all selected basins

read_runf_par.c

Reads parameter values for runoff process based on grids or values given at the input deck.
Parameter list:
  IN:
    lftx, rgtx, dny, upy –HRAP coordinates (lower left corner and upper right corner) for the sub-window that covers all selected basins.
    npix, number of pixels that the selected basins have,
    col, row, column and row number (hrap) of all grids contained in selected basins relative to the sub-window. *col* and *row* are 1-D vector of dimension of *npix.*
    npar, number of parameters needed.
    num_fpt, 1-D array (*npix*) of index for defining/distinguishing whether a grid belongs to which basin.
    nfpt, number of basins selected to run.
    npar_read, number of parameters/grid files to be read. e.g. 16 for SAC, 12 for PE
    pcrs, 1-D array of scale factors or values (given in the input deck) for input variables; if its value is negative, then use its absolute value as a factor to grid value, if its value is positive, then use this value instead of reading a grid data for this variable.
    runfid, a letter ID to identify which runoff model will be used, e.g. ‘sac’ SAC model
    dirpar, directory string defining where the grid parameter files are located
    par_names, a array of parameter names in a specific order

dirout, directory where output files will be written

OUT:
par_1d, 1-D array of parameter values appending one after another

read_rut_par.c

Reads routing parameter values for routing process based on grids or values given at the input deck

Parameter list:
IN:
lftx, rgtx, dny, upy –HRAP coordinates (lower left corner and upper right corner) for the sub-window that covers all selected basins.
npix, number of pixels that the selected basins have,
col, row, column and row number (hrap) of all grids contained in selected basins relative to the sub-window. *col* and *row* are 1-D vector of dimension *npix*.
npar, number of parameters needed.
num_fpt, 1-D array (*npix*) of index for defining/distinguishing whether a grid belongs to which basin.
nfpt, number of basins selected to run.
listfpt, array of *nfpt*, it contains outlet grid ID number for each basin.
npar_read, number of parameters/grid files to be read.
pcrt, 1-D array of scale factors or values (given in the input deck) for routing parameters, if its value is negative, then use its absolute value as a factor to grid value, if its value is positive, then use this value instead of reading a grid data for this parameter.
rutid, a letter ID to identify which routing model will be used, e.g. 'rutpix'
dirpar, directory string defining where the grid parameter files are located
par_names, a array of parameter names in a specific order
fspix, array of npix containing areas above each basin's outlet
dirout, directory where output files will be written

OUT:
par_1d, 1-D array of parameter values appending one after another

read_rut_st.c

Reads routing state values for selected basins. Whether to read grids or use provided values depends on values given in the input deck

Parameter list:
IN:
lftx, rgtx, dny, upy –HRAP coordinates (lower left corner and upper right corner) for the sub-window that covers all selected basins.

npix, number of pixels that the selected basins have,
col, row, column and row number (hrap) of all grids contained in selected basins relative to the sub-window. *col* and *row* are 1-D vector of dimension *npix*.
npar, number of parameters needed.
num_fpt, 1-D array (*npix*) of index for defining/distinguishing whether a grid belongs to which basin.
nfpt, number of basins selected to run.
npar_read, number of parameters/grid files to be read.
pcrs, 1-D array of scale factors or values (given in the input deck) for routing states, if its value is negative, then use its absolute value as a factor to grid value, if its value is positive, then use this value instead of reading a grid data for this parameter.
rutid, a letter ID to identify which routing model will be used, e.g. 'rutpix'
dirpar, directory string defining where the grid parameter files are located
par_names, an array of parameter names in a specific order
dirout, directory where output files will be written

OUT:
par_1d, 1-D array of state values appending one after another.

read_one_par.c

Reads one gridded parameter file in binary forma and creates a 1-D array of values. Called by other subroutines like **read_runf_par.c**, **read_rut_par.c** and **read_rut_st.c**.

Parameter list:
IN:
lftx, rgtx, dny, upy –HRAP coordinates (lower left corner and upper right corner) for the sub-window that covers all selected basins.
npix, number of pixels that the selected basins have,
col, row, column and row number (hrap) of all grids contained in selected basins relative to the sub-window. *col* and *row* are 1-D vector of dimension *npix*.
par_fname_in, name of a parameter to be read
dirout, directory where output files will be written

OUT:
par_each, 1-D array of read values in dimension of *npix*.

output_par.c

Output parameter values for grids that contained in each basin. Files are created for each parameter.

Parameter list:
IN:
nfpt –number of selected basins.

npix, number of pixels that the selected basins have,
np, number of parameters.
**fp_out*, file pointer to be created.
id_names, five-letter ID for each basin.
fptarea, basin area.
npix_sub, number of grid for each basin.
pnames, name of a parameter to be output.
par, 1d array of parameter values
num_fpt, 1-D array (*npix*) of index for defining/distinguishing whether a grid belongs to which basin.

OUT: a file *.out will be created.

read_xmrg.c

Reads radar xmrg data in each time step. Be noticed that xmrg data at 00z is for last hour of the previous day. So for any day, 24 hour of radar data are corresponding to 01z, 02z, ...23z, 00z (next day)
Parameter list:
IN:
lftx, rgtx, dny, upy –HRAP coordinates (lower left corner and upper right corner) for the sub-window that covers all selected basins.
npix, number of pixels that the selected basins have,
col, row, column and row number (hrap) of all grids contained in selected basins relative to the sub-window. *col* and *row* are 1-D vector in dimension of *npix*.
ihed, number of lines in xmrg file header.
par_fname_in, name of a parameter to be read
avt_idx, index for whether ArcView grid need to be created for this hour
avp_idx, index for whether ArcView grids need to be created or not
thour, current time in hours.
dirout, directory where output files will be written

OUT:
par_each, 1-D array of read rainfall values in dimension of *npix*.

read_new_xmrg.c
Reads newly created precipitation data for each time step. The data contains start time, end time, number of grids, and data info.

Parameter list:
IN:
npix, number of pixels that the selected basins have,
nfpt_sep, number of non-nested basins
fp_xmrgs, file pointers for new xmrg data.
npix_sep, number of pixels for each non-nested basin.

tstart, start time of simulation.
tend, end time of simulation.
tloop, current time in time loop

OUT:
par_each, 1-D array of read rainfall values in dimension of *npix*.

output_aigrid.c

Parameter list:
IN:
npix, number of pixels that the selected basins have,
lftx, dny, –HRAP coordinates (lower left corner and upper right corner) for the sub-window that covers all selected basins
col, row, column and row number (hrap) of all grids contained in selected basins relative to the sub-window. *col* and *row* are 1-D vector of dimension *npix*.
npar, number of parameters.
par_1d_all, 1d array of all concatenated parameter values.
par_names, list of parameter names.
outdir, output directory

OUT:
**_aig*, Arc/Info grids named as *_aig will be created where * represents parameter name.

get_ped.c

To get/compute hourly PE for each grid based on daily PE value which is in turn derived by interpolating values at 16th day of each month.
Parameter list:
IN:
y, m, d, –time in year, month, and day.
npix, number of pixels that the selected basins have.
dtm, time step in minutes.
pe, array of monthly PE values for all grids in *npix*..
pe_adj, monthly PE adjustment factor in array of *npix*.

OUT:
ped, PE demand for each time step for grids of *npix*

get_daily_pe.c

To get/compute daily PE values for all grids. They are derived from interpolating values at 16th day of each month. It is called by **get_ped.c.**
Parameter list:
IN:

yr, mon, day, –time in year, month, and day.
npix, number of pixels that the selected basins have.
dtm, time step in minutes.
pe, array of monthly PE values for all grids in *npix**12..
pe_adj, monthly PE adjustment factor in array of *npix*12*.

OUT:
pe_day, daily PE for all grids of *npix* as an array of *npix**12

do_sac.c

Conducts runoff simulation using SAC-SMA model.
Parameter list:
IN:
npix, number of pixels that the selected basins have.
listfpt, array of *nfpt*, it contains outlet grid ID number for each basin.
num_fpt, 1-D array (*npix*) of index for defining/distinguishing whether a grid belongs to a specific basin.
runf_par, 1-D array of runoff parameters in *npix***nprunf*.
runf_st, 1-D array of runoff state values in *npix***nsrunf*
nfpt, number of basins selected to run.
pcp, *ped*, 1-D array of precipitation and PE demand in *npix*.
dtm, time step in minutes.
tloop, a time in the time loop.
tstart, start simulation time in hours.

OUT:
surf, 1-D array of surface runoff for all grids of *npix*.
subf, 1-D array of baseflow for all grids of *npix*.
tet, total ET for all grids of *npix*.

get_tci.c

Calulates total channel inflow for basins that unit hydrograph was defined/selected
Parameter list:
IN:
npix, number of pixels that the selected basins have.
nfpt, number of basins selected to run.
num_fpt, 1-D array (*npix*) of index for defining/distinguishing whether a grid belongs to a specific basin.
nordk, 1-D array of number of unit hydrograph ordinates for each basin.
surfl, grided surface runoff.
subfl, gridded subsurface runoff.

OUT:

tci, total channel inflow for each basin.
subf, 1-D array of baseflow for all grids of *npix*.

uhg_to_hg.c

Converts unit hydrograph to flow at the outlet of each basin.

Parameter list:
IN:
tci, total channel inflow for each basin.
uhg, unit hydrograph for all basins in a 1-D vector format (in cms/mm).
nordi, 1-D array of number of unit hydrograph ordinates for each basin.

OUT:
qflow, 2-D array (for nordi[k] of each basin k) of converted flow from UHG.

save_states.c

Saves states for a specified time, usually the tend, last time step of simulation.
Parameter list:
IN:
runf_st, 1-D array of state values.
npix, number of pixels that the selected basins have
st_names, array of strings for state names
stdir, directory where states are located

OUT: binary files named as time plus state names. The state variable files contain the time for which states are saved, number of pixels, and their values.

write_avgrid.c

Creates a ArcView grid for a defined region based on a HRAP grid.
Parameter list:
IN:
rectin, 2-D array of values.
name1, ArcView grid name to be created.
csize, cell size. (4762.5 if convert from hrap to polar stereographic coord)
bndbox, a four element array defining a rectangular boundaries.
pe_adj, monthly PE adjustment factor in array of *npix*.

OUT:
An ArcView grid name as name1 will be created.

delete_avgrid.c

Deletes a ArcView grid created by write_avgrid.c.
IN:

name1, ArcView grid name to be deleted.

basin_average.c

To get a basin averaged value based on gridded info.
Parameter list:

IN:

npix, number of pixels that the selected basins have.
nfpt, number of basins selected to run.
num_fpt, 1-D array (*npix*) of index for defining/distinguishing whether a grid belongs to a specific basin.

OUT:

mean, basin averaged values in *nfpt*.

write_ave_pcp.c

Output MAPX time series for basins having nested sub-basins selected.
Parameter list:

IN:

bsn_id, five-letter basin ID.
fp_pcp, file pointer of output file with the name as bsn_id.mapx.
fp_xmrg, file pointer to 1-D xmrg files.

OUT:

files named as *bsn_id.mapx* created at the defined output directory.

rd_deck1.f
Reads an input deck to get basic input information.
Parameter list:

IN:

iu, unit number which an input deck file is connected to
flinp, file name of an input deck

OUT:

n1, start run time, in hours from 1900 year
n2, end run time, in hours from 1900 year
dt, simulation time step, in hours
dirpar, directory of parameter grids
dirxmrg, directory of 'xmrg' grids
lxmrg, number of characters in 'xmrg' directory name
dirst, directory of model states
dirout, directory of output files
seqfl, directory including file name of connectivity file
runfid, rainfall-runoff model ID, < 9 characters (‘sac’ for SAC-SMA model)
rutid, routing model ID, < 9 characters (‘rutpix’ for kinematic routing)

nprunfg, number of parameter grid files of rainfall-runoff model to be read
nprunf, number of rainfall-runoff model parameters (for SAC-SMA *nprunf*=*nprunfg*=16)
nsrunf, number of rainfall-runoff model states
npruth, number of parameter grid files of routing model to be read
nprut, number of routing model parameters (for kinematic routing *nprut*=3, while *nprutg*=7)
nsrut, number of routing model states
lftx, *rgtx*, *dny*, *upy*, HRAP coordinates of a selected subwindow (left corner, right corner, bottom corner, and upper corner respectively)
nfpt, number of selected outlets to generate hydrographs
ipcp, option to use precipitation: 1 if distributed rainfall, 0 if it is lumped
comm, comments string
arcv, directory name to store grids for ArcView plots
exe, index for execution control and whether or not create Arc/Info for certain parameters.

rd_deck2.f

Reads an input deck second time (after '*rd_deck1.f*' to get input information for each defined outlet.

Parameter list:

IN:

nprunfg, number of parameter grid files of rainfall-runoff model to be read
nsrunf, number of rainfall-runoff model states
nprutg, number of parameter grid files of routing model to be read
nsrut, number of routing model states
iu, unit number which an input deck file is connected to
flinp, file name of an input deck

OUT:

nfpt, number of selected outlets to generate hydrographs
fptid(nfpt), outlet IDs
*runfpc(nfpt*nprunfg)*, 1-D array of scale factors or values for rainfall-runoff parameters; if its value is negative, then use its absolute value as a factor to grid value, if its value is positive, then use this value instead of reading a grid data for this parameter
*runfsc(nfpt*nsrunf)*, 1-D array of scale factors or values for rainfall-runoff states; if its value is negative, then use its absolute value as a factor to grid value, if its value is positive, then use this value instead of reading a grid data for this state
*rutpc(nfpt*nprutg)*, 1-D array of scale factors or values for routing parameters; if its value is negative, then use its absolute value as a factor to grid value, if its value is positive, then use this value instead of reading a grid data for this parameter
*rutsc(nfpt*nsrut)*, 1-D array of scale factors or values for routing states; if its value is negative, then use its absolute value as a factor to grid value, if its value

is positive, then use this value instead of reading a grid data for this state

*pec(nfpt*12)*, 1-D array of scale factors or values for potential evaporation; if its value is negative, then use its absolute value as a factor to grid value, if its value is positive, then use this value instead of reading a grid data for this variable

*peac(nfpt*12)*, 1-D array of scale factors or values for PE adjustment factor; if its value is negative, then use its absolute value as a factor to grid value, if its value is positive, then use this value instead of reading a grid data for this variable

nsubc(*nfpt*), 1-D array that defines how many nested basins are in each outlet; default value is 1 meaning that it is just one not nested basin; value 0 means nested basin; all other values mean how many nested basins are in this outlet

*gen_input(nfpt*12)*, 1-D array of basic variables to generate routing parameter grids. This array is used only in preprocessor mode, but not in simulation mode.

rd_card.f

Reads an input card from an input deck.

Parameter list:

IN:

iu, unit number which an input deck file is connected to

nrd(7), array defines number of rainfall-runoff & routing parameter grids, number of rainfall-runoff & routing parameters, number of grids/values of potential evaporation & PE adjustment factors (in this version 12 values each), and number of basic parameters to generate routing parameter grids (in this version equals 12) respectively

IN/OUT:

nline, counter of input deck lines

OUT:

flname, any name from input card field if there is one

lfl, string length for '*flname*' name

nn, number of variables (e.g., parameters, states, etc.) from an input card

vars(nn), array of input variables from an input card

nvar, maximum number of input variables in one input card. This variable is set to 20 as a constant parameter in subroutines *rd_deck1* and *rd_deck2*

crdid, card label

fend, index of the end of an input deck; -1 means normal end, 77 means error on reading of deck

in, number of outlet Ids that have been read

nnrd, order number of input card variables consistent with the order in array *nrd*.

slt_var.f

Selects all fields from an input card.

Parameter list:

IN:

ist, position of selected field of an input card; returns adjusted value for the next field position

*text*128*, character string of an input card

OUT:

*textx*lnt*, character string of selected field

lnt, string length of selected field

iend, index of the end of an input card; -1 means a normal end.

init_rut1.f

Reads channel connectivity file and selects all connected pixels for defined outlets.

Parameter list:

IN:

lftx, rgtx, lowy,iupy, sub-window NEXRAD coordinates

seqfl, connectivity file name

OUT:

npix, number of selected pixels

col(npix), row(npix), HRAP coordinates of selected pixels in connectivity order

seq(npix), sequence numbers of selected pixels

down(npix), sequence numbers of neighbor downstream pixels

num_pix(npix), outlet numbers for each selected pixel

fpix(npix), pixel area, km2

nfptc, number of defined outlets

listfpt(nfptc), sequence numbers of each selected outlet

fptarea(nfptc), local areas for each selected outlet

fptid, character string consists of outlet IDs

llx, HRAP coordinate of a lower left corner defined in a channel connectivity file

ury, HRAP coordinate of a upper right corner defined in a channel connectivity file

seqx, temporary array that consists of sequence numbers of pixels selected for the entire sub-window defined by *lftx, rgtx, lowy,iupy*

indexx(nfptc), index table that defines outlets in connectivity order.

init_rut2.f

Redefines HRAP coordinates and sequence numbers into relative coordinates of the selected sub-window. It is also calculate the distance interval for the routing model.

Parameter list:

IN:

lftx,lowy, HRAP X and Y ordinates of parameter grid origin

llx0, iupy0, HRAP X and Y ordinates of a defined sub-window

npix, number of selected pixels

col(npix), row(npix), HRAP coordinates of selected pixels in connectivity order

seq(npix), sequence numbers of selected pixels

down(npix), sequence numbers of neighbor downstream pixels

fpix(npix), pixel area

numfptpix(npix), outlet numbers for each selected pixel
nfptc, number of defined outlets
fptarea(nfptc), local areas for each selected outlet

OUT:

icol(npix), irow(npix), relative coordinates of selected pixels for defined sub-window
listfpt(nfptc), sequence numbers of each selected outlet
idown(npix), relative sequence numbers of neighbor downstream pixels
num_fpt(npix), outlet numbers for each selected pixel
pixarea(npix), pixel area
fabove(nfpt), area above each outlet
fspix(npix), area above each selected pixel
length(npix), conceptual channel length for each pixel
ndx, number of sub-reaches in kinematic routing for each pixel

vec_seq1.f

Reads information from a connectivity file and creates vector outputs in the routing simulation order for selected outlets.

Parameter list:

IN:

nlist, number of defined outlets
list(nlist), sequence numbers of defined outlets
seqx(), temporary array filled by -1 values

IN/OUT:

These arrays are input/output arrays. The subroutine will select a subset of data based on defined outlets:
col(npix), row(npix), HRAP coordinates of selected pixels in connectivity order
seq(npix), sequence numbers of selected pixels
down(npix), sequence numbers of neighbor downstream pixels
fpix(npix), pixel area

OUT:

npix, number of selected pixels
fptarea(npix), pixel area, km2
nfptpix(npix), outlet numbers for each selected pixel

rd_hed_seq.f

Reads header records of a channel connectivity file.

Parameter list:

IN:

iu, unit number to read connectivity file
nvar, number of variables to read from a header line

OUT:

fptname, outlet ID's from a header line
var(nvar-1), variable values from a header line

reorder.f
Reorder input variable arrays of each outlet in corrected outlet order.
Parameter list:

IN:

nprunfg, number of parameter grid files of rainfall-runoff model to be read
nsrunf, number of rainfall-runoff model states
nprutg, number of parameter grid files of routing model to be read
nsrut, number of routing model states
nfpt, number of outlets
index(nfpt), index table that defines outlets in connectivity order

IN/OUT:

These arrays are input/output arrays. The subroutine just will change the order of array elements:
*runfpc(nfpt*nrunfg)*, see *rd_deck2.f* description
*runfsc(nfpt*nsrunf)*, see *rd_deck2.f* description
*rutpc(nfpt*nprutg)*, see *rd_deck2.f* description
*rutsc(nfpt*nsrut)*, see *rd_deck2.f* description
*pec(nfpt*12)*, see *rd_deck2.f* description
*peac(nfpt*12)*, see *rd_deck2.f* description
*gen_input(nfpt*12)*, see *rd_deck2.f* description
nsubc(nfpt), see *rd_deck2.f* description

xmrg_path.f
Returns 'xmrg' file name (including entire directory path).
Parameter list:

IN:

itime, julian date in hours from 1900
nyy, number of year digits in 'xmrg' file name
grd_dir, path to a directory of 'xmrg' files
ln_dir, character number in the path name *grd_dir*
preff, prefix for 'xmrg' file name (default is *xmrg*)
lnp, character number in the prefix name

OUT:

grd_read, 'xmrg' file name including directory path
ihed, number of header records in 'xmrg' file; it uses default setting although it may be changed when 'xmrg' file is processed

rd_hed_hrp.f
Opens 'xmrg' file, and checks how many header records are there.
Parameter list:

IN:

iunit, device number to open 'xmrg' file
fname, 'xmrg' file name
iyx, year of 'xmrg' file to process

IN/OUT:

ihed, number of a header records (may be adjusted by this subroutine)

OUT:

xor, X-origin of 'xmrg' file (HRAP coordinates)
yor, Y-origin of 'xmrg' file (HRAP coordinates)
mx, number of columns per record
my, number of rows per file
ifile, status of file reading; 0 if no problem

fland1.f
Performs calculations for the Sacramento rainfall-runoff model.
Parameter list:

IN:

pxv, rainfall per one time interval
edmnd, potential evapotranspiration
dt, simulation time interval
uztwm, uzfwm, uzk, pctim, adimp, riva, zperc, rexp, lztwm, lzfsm, lzfpm, lzsk, lzpk, pfree, side, saved, SAC-SMA parameters

IN/OUT:

uztwc, uzfwc, lztwc, lzfsc, lzfpc, adimc, SAC-SMA states

OUT:

surf, simulated fast runoff
grnd, simulated slow runoff
tet, simulated actual evapotranspiration

do_route.f
Main subroutine to prepare input information and to call kinematic routing models.
Parameter list:

IN:

surf(npix), fast runoff at each pixel, *mm*(dt)sec*
ground(npix), slow runoff at each pixel, *mm*(dt)sec*
dt, simulation time interval, *sec*
*rut_par(3*npix)*, hillslope & channel routing parameters (one for hillslope, and 2 for channel)
npix, number of selected pixels
fpix(npix), pixel area, *km2*
ndx, number of sub-reaches in kinematic routing for each pixel
length(npix), conceptual channel length for each pixel
idown(npix), sequence numbers of neighbor downstream pixels

IN/OUT:

*rut_st(2*npix)*, channel & hillslope routing states (channel cross-section, and hillslope water depth) at each pixel
*dx_st(ndx*npix)*, channel states at each sub-reach for each pixel. These states are available only during running time. At the end of run time only *rut_st* array will be stored.

OUT:

q(npix), routed runoff at outlet of each pixel, *cms*

hstream.f
Perform channel routing for a conceptual pixel channel based on kinematic wave model.
Parameter list:

IN:

qinp, inflow from the upstream pixels, *cms*
r, rt, lateral inflow (hillslope routing output) to the pixel channel at *t* and *t+dt* time intervals, in *mm/s*
qspmn, m are channel routing parameters in relationship *qmn=qspmn*h**m*
length, channel length, *m*
ndx, number of sub-reaches to break down a pixel channel length
fpix, pixel area, *km2*
dt, simulation time interval, *sec*
error, accuracy criteria exiting iterations, default is 0.000001
niter, maximum number of iterations to exit, default is 10
teta, weight parameter of a numerical scheme to control stability (scheme is unconditionally stable if *teta* > or = 0.5), default is 1.0

IN/OUT:

h, channel cross-section (routing state), *m2*

OUT:

qmn, routed discharge, *cms*

hslope.f
Performs hillslope routing based on kinematic wave model.
Parameter list:

IN:

qspmn, hillslope routing parameter in relationship qsmn=qspmn*hs
rs, inflow in a hillslope (SCA-SMA fast runoff), mm*(dt)sec
dt, simulation time interval, sec
error, accuracy criteria exiting iterations, default is 0.0000001
niter, maximum number of iterations to exit, default is 10
teta, weight parameter of a numerical scheme to control stability (scheme is unconditionally stable if *teta* > or = 0.5), default is 1.0

IN/OUT:

hs, hillslope water depth, mm

OUT:

qsmn, routed fast runoff per unit area, mm/s

Additional utilities. There are also a number of subroutines to perform some standard procedures. Some of them were obtained from the OFS software package:

days_of_month.c
Determines number of days in a given month for a given year.

date_to_hr.c
Converts a date into number of hours starting at 1900.

hr_to_date.c
Converts a hour value starting from Jan. 1, 1900 into a date as mm/dd/yyyy hh:00

get_fname.c
Extract a file name from a long string which contains path and file name.

fill_miss.c
Fills the missed data within xmrg grid (only for those grids within selected basins)

maxv.c
Get the maximum value from an array.

minv.c
Get the minimum value from an array.

array1d_to_2d.c
Converts a 1D array into 2D array

flip_2d_array.c
Flips a 2D array upside down.

get_bndbox.c
Get region info and dimension info for Arc/Info grids.

is_leap_year.c
Check whether a given year is a leap year or not.

ddgch2.f
Gets hour-sum since January 1900 from month, year, day, and hour.
See OFS package for description.

ddgcj.f
Gets Julian day from year(four digits), month, and day.
See OFS package for description.

ddgdj2.f
Gets Julian day and year (four digits) from day-sum from January 1900.
See OFS package for description.

ddghc2.f
Gets calender date (four digit year, month, day, hour) from hour-sum since January 1900.
See OFS package for description.

ddgjc.f
Gets four digit year, month, and day from Julian day.
See OFS package for description.

ddgjd2.f
Gets day-sum since January 1900 from Julian day and four digit year.
See OFS package for description.

ddycdl.f
Updates year for calendar date.
See OFS package for description.

ddrmcl.c
Reads calendar date from machine.
See OFS package for description.

kktrim.f
Gets begin and end character location in a string.
See OFS package for description.

dmg.f
This Function gets day, month, year, and leep year index from calender date, MMDDYYYY.
Parameter list:

IN:

dat, calender date, MMYYDD

OUT:

nd, day
nm, month
year, four digit year

RETURN:

Leap year index: 0 if leap year; non-zero otherwise.

gf.f
This Function returns Gamma-function value.
Parameter list:

y, argument to calculate Gamma-function

iindexx.f
Generates index array to arrange variables in an increasing order. From 1986-92 Numerical Recipes Software.
Parameter list:

IN:

n, number of variables to reorder

arr(n), array of variables

OUT:

indx(n), array of indexes

int2char.f

Converts integer value into a character string.

Parameter list:

IN:

in, integer value to convert

IN/OUT:

ny, desired length of an output character string. If *ny* is bigger than number of digits in *in* value, zeros will be added in the front of string; if *ny* is negative, a string length will be equal a number of digits in *in* value, and *ny* will be adjusted to this length

OUT:

ich(ny), character string.

intconv.f

This Function returns converts character string into integer value.

Parameter list:

IN:

m, string length

x(m), character string

RETURN:

Integer value.

nchar.f

This Function returns a number of characters in a string.

Parameter list:

n, string length

name(m), character string.

ncharc.f

This Function returns a position number of defined character in a string. If there was no defined character in the string, it will return 0.

Parameter list:

n, string length

name(n), character string

c, defined character.

order.f

Reorders an 1-D array structured as a *(i-1)*m+j* sequence based on the defined index array.

Parameter list:

IN:

nsd, number of values per one subset (*m*) of data
nfpt, number of subsets
index(nfpt), defined index array

IN/OUT:

*array(nsd*nfpt)*, array to be reordered.

pagrid.f
Generates pathname of 'xmrg' file from directory name, file type, and calender date. Slightly changed the OFS code.
See OFS package for description.

rlconv.f
Transforms character string into real value. The Function returns a real value.
Parameter list:

n, string length
cinval(n), character string.

sbllgd.f
Converts the Longitude and Latitude coordinate location on Earth to the HRAP grid system location (and vice-versa). The subroutine was changed slightly by John Schaake from the OFS code.
See OFS package for description.

Appendix II: Input deck format.

The program uses free format input for all cards. Free format input rules are:

1. The @ in the first position of the line denotes a card label. Card labels can be on the same line or the line above the fields of the card. For example:

   ```
   @A 05021999 05311999 2
   ```

   can be specified as

   ```
   @A
           05021999 05311999 2
   ```

2. If a card is not needed for particular run, its label and all fields must be omitted.

3. Only columns 1-128 can be used for input. All columns beyond 128 are ignored.

4. One comma or at least one blank can be used as a delimiter between fields.

5. Blanks or commas are not allowed in character fields.

6. All fields are required. A null field (double commas) must be used to denote single fields for which default values are to be used. If N consecutive fields use default values, N+1 commas must be used. The following input implies that defaults are to be used for the first two fields on card C:

   ```
   @C , , ,
   ```

7. Not all fields have valid defaults. If the documentation does not specify a default, the input must be specified.

8. Consecutive commas at the end of a card can be omitted. For example,

   ```
   @E 1, , ,
   ```

   Can be specified as

   ```
   @E 1
   ```

9. Any number of comment lines can be used in input card order. Each comment line should have the # character in the first position of the comment line.

Appendix III.

A. Some feature to generate routing parameter grids for selected basins

HLRMS assumes that there are default grids of SAC-SMA and Routing model parameters. Otherwise, constant parameter values for each defined basin should be provided in an input deck. If there are information on measured discharges including channel cross-sections and top widths, an extension of HLRMS software can generate grids of channel roughness coefficient and top width parameters using empirical geomorphological relationships.

Channel roughness coefficient, n, at an outlet can be estimated from the Chazy-Manning's equation

$$n = \frac{A^{5/3}\sqrt{S}}{Q B^{2/3}} \qquad \textbf{(1)}$$

where A is a channel cross-section, S is a channel slope, Q is a discharge, and B is a top width of the channel. A number of n values can be estimated using measured discharges. Most representative value (discharges measured at high stages are preferable) should be used in further calculations.

Empirical Equation [Tokar & Johnson, 1995] is used to generate a roughness coefficient at each pixel above an outlet depending on channel slope (S_i), and area above selected pixel (F_i):

$$n_i = n_0 S_i^{k1} F_i^{k2} \qquad \textbf{(2)}$$

where $k1$=0.272, and $k2$=-0.00011. These values were estimated from data for about 20 streams in the USA. Parameter n_0 can be estimated from Eq. (2) for known values of selected roughness coefficients, slopes, and basin areas at an outlet.

Channel top width parameter, *a*, of relationship between channel top width and depth, H,

$$B = aH^m \qquad \textbf{(3)}$$

at an outlet can be estimated by correlation of measured data of channel cross-sections and top widths. Then parameters of a top width-cross-section relationship can be easily transformed into Eq. (3) parameters

$$m = \frac{1}{m^* - 1}, \qquad a = \left(\frac{m^* - 1}{a^* m^*}\right)^{1/(m^* - 1)} \qquad \textbf{(4)}$$

where a^*, and m^* are parameters of a top width-cross-section relationship.

If we assume that the channel shape parameter, m, does not depend on the channel order, a variable parameter a of Eq. (2) can be generated at each pixel using a few step procedure:

a) A channel order, k_i, for each pixel and an outlet pixel, k_o, can be estimated from Table 1. The Table 1 was generated based on the Rzhanitsyn's stream order definition [Manual, 1989] as a function of the stream length. Area above was calculated using stream length data [Willemin, 2000]:

$$L = 1.6\,F^{0.58} \tag{5}$$

In Eq. (5), channel length, L, is in *km*, and area above, F, is in *km²*.

Table 1. Channel order definition

Channel order	Channel length, km	Area above, km^2
1	0.8	0.3
2	1.5	0.9
3	2.8	2.7
4	5.1	7.6
5	9.3	21.6
6	16.9	61.3
7	31.0	176.3
8	57.0	509.4
9	104.0	1452.0
10	190.0	4149.0
11	338.0	11317.0
12	620.0	32562.0

b) Calculate cross-section ratios of each pixel and outlet from the morphological relationship suggested by Gorbunov [1971]:

$$r_i = \frac{A_i}{A_o} = \frac{0.013^{0.83^{k_i} - 0.83^{k_o}}}{R_L^{o-i}} \tag{6}$$

It is assumed that the Horton parameter of channel length, R_L, equals its average value of 2.1.

c) Estimate channel velocity at each pixel, v_i, from measured discharge-cross-section data at an outlet assuming that runoff above outlet distributed uniformly, and can be calculated as a ratio of contributed area

$$v_i = \frac{Q_o}{r_i A_o} \frac{F_i}{F_o} \tag{7}$$

d) From the Shezy-Manning's formula, estimate an average water depth at each pixel

$$H_{avg,i} = (v_i \frac{n_i}{\sqrt{S_i}})^{3/2} \tag{8}$$

e) Calculate a top width parameter at each pixel

$$a_i = r_i A_o (m+1)^{-m} H_{avg,i}^{-(m+1)} \tag{9}$$

Note: This approach assumes that slopes, roughness coefficients, and channel shape parameter do not depend on a water stage/discharge. As a result, different grids of parameter a_i can be derived for different selected pares of discharge-cross-section values at outlet. Preference should be done to higher discharges to represent better flood conditions. Future versions may use a few parameter sets for different ranges of the water depth.

B. Extension of the HLRMS code to generate routing parameter grids.

a) A few additional subroutines were written to generate parameter grids for a required basin where measured discharge data are available

do_route.f, dstwidth.f, wrt_genpar.f

Because the main program of the HLRMS was also changed, a separate executable program was created

gengrid.exe

b) An additional entry line of the input deck is required to run this program:

+IPRG *<n_values> <parameter #> <variable_1, variable_2,..., variable_(n_value-2)>*

<n_values> is a number of values in the line,

<parameter #> is the order number of the selected parameter to generate grid. The order number for routing parameters is defined in the HLRMS as: *SLOPC, ROUGC, BETAC, ALPHC, SLOPH, DS, ROUGH*,

<variable_i> is required input variables to generate grid of a selected parameter *<parameter #>.*

As described above, only two variable grids can be generated by recent version, *ROUGC* and *ALPHC*. For other parameters, all pixels of a selected basin can be filled by a constant value from an **+IPRG** line. Input variables to generate *ROUGC* parameter are a 'representative' roughness coefficient at outlet, and two parameters, *k1* (default value is 0.272) & *k2* (default value is -0.00011):

+IPRG 5 2 n_o *k1* *k2*

Input variables to generate ALPHC grid are 'representative' discharge, cross-section, and *a*

parameter at outlet:
+IPRG 5 4 Q_o A_o a_o
To generate other constant parameters:
+IPRG 3 <*n*> <*value*>
where <*n*> is an order number of selected parameter, and <*value*> is a constant value to fill.

+IPRG entry should be provided for each basin ID. All other input deck entries can be same as described in an Appendix II.