# DESIGN SPECIFICATION FOR

# OPEN-BOOK WATERSHED MODEL

**BY**

**NITIN KATIYAR**

**May, 2007**

## Preface

***Before reading this manual reader is advised to read the thesis "Development of an open-book watershed model for rapid prototyping of satellite-based flood forecasting in international river basins" submitted to Dept. of Civil Engineering, Tennessee Tech University, Cookeville, TN, and Paper "Katiyar, N., and Hossain, F. 2007. An Open-book Watershed Model for Prototyping Space-borne Flood Monitoring Systems in International River Basins. Environmental Modeling and Software. (In press; doi:10.1016/j.envsoft.2006.12.005)"to understand the concept illustrated and applied in manual and implemented code .***

## Disclaimer

This manual is in no way comprehensive. Users with no background in or understanding of distributed hydrology are strongly advised against using this code in any mode, particularly in operational mode. Besides knowledge of basic hydrology, experience with typical numerical techniques used in physically-based hydrodynamic models is recommended as it will help the user grasp capabilities and limitations of this model. Users are encouraged to experiment with the model and venture in hydrology textbooks and journal papers to learn more about the topics touched upon in this manual. No claims are made regarding the suitability of the model for any particular purpose. The model was written for research and educational purposes with a particular focus on rapid prototyping of satellite based flood forecasting systems in International River Basins."

# Input Files

| ***File Name*** | ***Format*** |
|---|---|
| *dem.txt* | *No of pixels in x direction "TAB" No of pixels in y direction "TAB" Pixel Length of the square pixel (this program assumes square grid) "NEW LINE" Elevation values in grid format.*  Dem.txt |
| *channel_width.txt* | *This File is in grid format with the x and y number of grid according to dem.txt file. It contains 0 if the land belonging to this pixel is NOT a channel flow otherwise it will have channel width. Note wherever we have a channel pixel the width of the pixel will be taken from this file not from dem file so dem file will have extra pixel.*  channel_width.txt |
| *rainfall.txt* | *Rainfall Period (in min) "NEW LINE" Rainfall Values in grid format unit mm/hr*  Rainfall.txt |
| *parameter.txt* | *Viscosity (in MKS) "TAB" Manning's Roughness "TAB" Threshold Reynold's Number (to decide about the laminar/turbulent) "TAB" Gravitational Acceleration "TAB" Time Step for the calculation in min "TAB" Saturated Hydraulic conductivity in cm/hr "TAB" Field Capacity* |

| | |
|---|---|
| |  parameter.txt |
| *soil_storage.txt* | *Percentage Storage in grid format. Soil storage will be calculated using depth of bed rock and porosity.*  soil_storage.txt |
| *porosity.txt* | *Porosity in grid format.*  Porosity.txt |
| *depth.txt* | *Depth of bed rock in grid format in m.*  Depth.txt |
| *output.txt* | *Flow at outlet point of watershed in $m^3/s$.*  Output.txt |

## Naming convention

| | |
|---|---|
| *Function (Subroutine)* | *fn_* |
| *Input File Name* | *fl_in_* |
| *Output File Name* | *fl_out_* |
| *File Pointer* | *fp_* |
| *Array* | *a_* |
| *Variable* | *v_* |
| *Local Variable* | *v_ and l_v_* |
| *Counters* | *i's; ii's; i1's* |

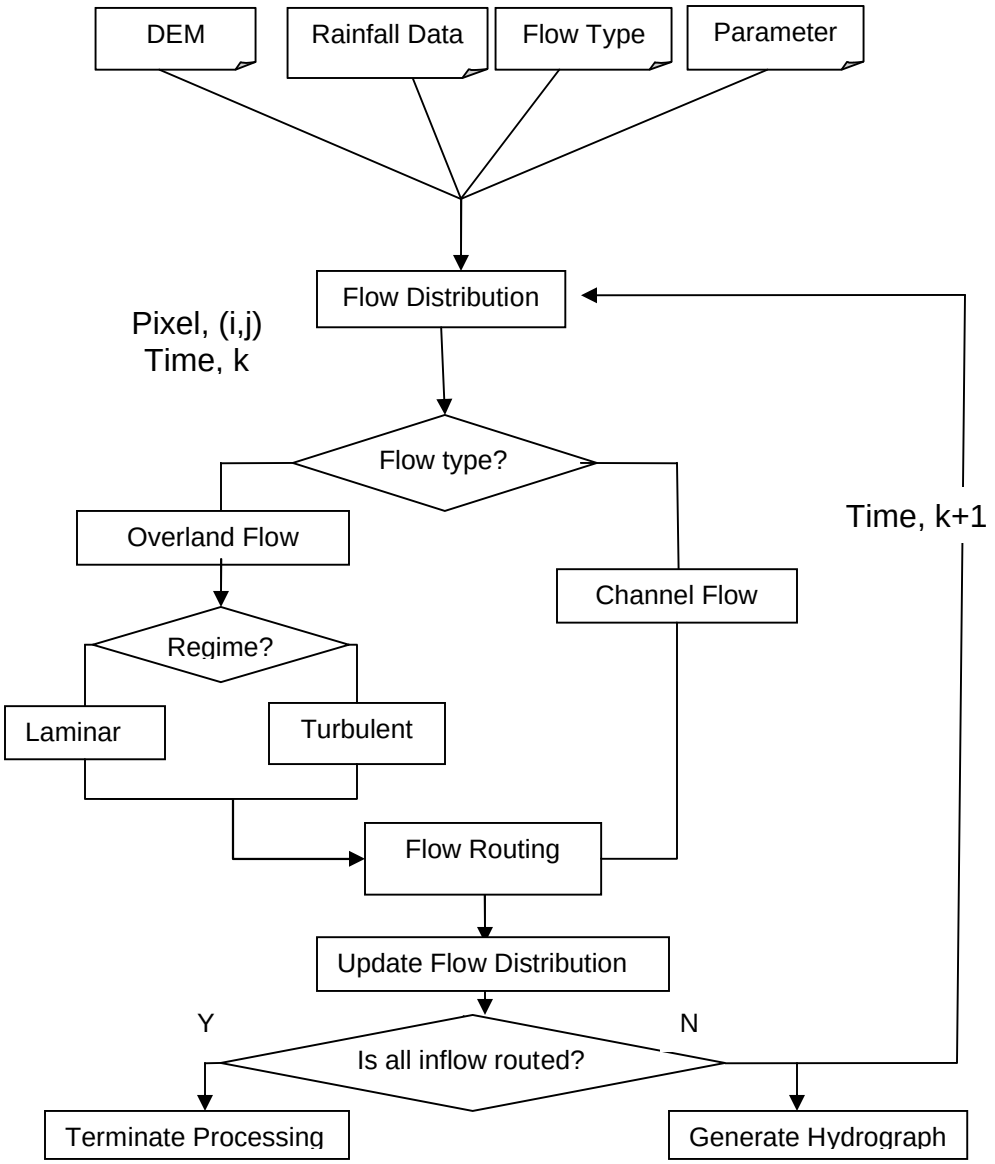
## Defining some global variables

| | |
|---|---|
| *a_slope_n_river_width[250][150][3]* | *This 3-D array contains –Elevation – River width – Slope* |
| *a_resultant_direction[250][150][2]* | *This 3-D array contains resultant flow direction of the pixel.* |
| *a_cosine[250][150]* | *Cosine of the slope angle* |
| *a_rainfall[250][150]* | *Rainfall for a particular rainfall period* |
| *a_inflow_2_pixel[250][150]* | *Inflow to pixels for all the pixels from other pixels* |
| *a_destination_pixel[2]* | *Location of the destination pixel* |
| *l_a_temp_inflow_2_pixel[250][150]* | *Temporary storage of inflow to pixels* |
| *a_soil_storage* | *Soil storage* |
| *a_grid_Response_time[250][150]* | *Grid response time* |
| *a_saturation_excess_flow[250][150]* | *Saturation excess flow* |
| *a_porosity* | *Porosity* |
| *a_depth* | *Depth of bed rock* |

# Technical Specifications

## Algorithm Formulation of the Model and Computer Implementation

Computer processing of the open-book model involves several routines. Details about how the code works and input/output files are documented in manual prepared for the model. These routines have been written in computer language C to implement equations mentioned in previous sections. Figure illustrates the flow diagram of the model. The model basically requires seven different types of ASCII input files with strict formatting namely DEM file, Rainfall Data, Channel Width data, Depth (or soil column), Porosity, Initial Soil Moisture Storage, and Parameter files. Model input is converted into SI units. Preprocessing of the data is required by the model. The slope values for each pixel are generated, which is the steepest gradient among the surrounding eight pixels in its neighborhood. Time varying rainfall is read from the input file and then converted to overland flow. For each pixel, excess rainfall is calculated. For this purpose a linear storage-discharge reservoir scheme is followed. Subsurface flow is nonzero if soil moisture storage in the soil column for that pixel is greater than field capacity. Overland saturation excess flow is calculated, if soil moisture storage is greater than soil storage capacity, otherwise it is zero. Using Overland flow and subsurface flow, soil moisture is updated for every time step using the mass balance equation 1. The overland flow is assumed to be sheet flow over the surface. Using a Reynolds number threshold value provided in the parameter input file, identification of the overland flow regime as laminar or turbulent is conducted. If the overland flow in laminar, flow is routed. For every time




Flow diagram of computer data processing of the model

step, it is checked if the flow is still. If flow is turbulent based on Reynolds number criteria, Manning's Equation is used to rout the flow until the flow meets the channel. Overland sheet flow meets the channel flow and becomes part of channel. For channel flow Manning's Equation is used and for faster convergence of this equation Newton method is used. This routed flow is reported for the outlet pixel of the open-book watershed as the flow for a given time of the flood hydrograph. For the benefit of future users of the developed computer code, a user's manual has also been prepared that details how the code is to be applied on a computer system.

## Main Routine

*Open files dem.txt, channel_width.txt, rainfall.txt, parameter.txt, soil_storage.txt, porosity.txt, depth.txt in read mode and output.txt in write mode. If error in opening file output file will contain the error.*

*Read x and y direction pixels, pixel length and dem values in v_x_pixels, v_y_pixels, v_pixel_length and a_slope_n_river_width[i][j][0]. a_slope_n_river_width is three dimensional array. Read river width values in a_slope_n_river_width[i][j][1]. Call subroutine fn_populate_slope_file. Call subroutine fn_populate_cos_file. Read viscosity, manning's roughness, threshold reynolds number, gravitational acceleration, time step, saturated hydraulic conductivity and field capacity from parameter file into v_viscosity, v_mannings_coeffcient, v_thre_reynlds_no, v_gravitation_accln, v_time_step, v_sat_hydraulic_conduc and v_field_capacity. Read porosity values into global array a_porosity and depth values into global array a_depth. Read percentage soil storage into global array a_soil_storage after multiplying this value by porosity and depth of the corresponding pixel.*

*Convert unit of saturated hydraulic conductivity from cm/hr to m/s. Call subroutine fn_populate_grid_response_time_array.*

*Rainfall file may contain several sets of dataset (time period and data in grid format). Store it in v_time_period and Call fn_get_rainfall_data. Every period of rainfall (storm period) may contain several time steps. So for each time step - Call fn_linear_storage_discharge and fn_overland_flow_processing subroutines. Print final output at the end of this time step in the output file and reinitialize array a_inflow_2_pixel. Currently subroutine which check if a_inflow_2_pixel to all pixels is zero and rainfall is also zero or not for termination condition, has been removed. It can be again added at the same place. Currently, the program runs for 200 more timesteps after rainfall ends, to stabilize itself. In every run it calls Call fn_linear_storage_discharge and fn_overland_flow_processing subroutines. Close all the open files at the end. Any error that occurred while closing is written in output file.*

## Fn_get_rainfall_data Routine

*Read rainfall data from the file and convert the unit from mm/hr to m/s by multiplying it with $2.777777X10^{-7}$. Declaration of the variable as double is used for storing this.*

## Fn_populate_slope_file Routine

*For every pixel in the grid get the 8 neighboring pixels (for boundary pixels neighboring pixels will be less than 8. If the neighboring pixel's x coordinate or y coordinate is same as that of the pixel under consideration then the slope to that pixel will be elevation difference upon pixel length, because both pixels are side by side. On the other hand if the pixel is neighboring pixel without satisfying the above criteria then the slope will be elevation difference divided by pixel length multiplied with square root of two, because both pixels are diagonally situated. For every pixel under consideration find out the steepest slope and store it in a_slope_n_river_width[i][j][2] and resultant pixel direction in a_resultant_direction[i][j][0] and a_resultant_direction[i][j][1].*

## Fn_populate_cos_file Routine

*Store the cosine of the slope angle into a_cosine array. This array is calculated using cos ( arc tan (slope)) formula.*

## Fn_linear_storage_discharge

*Excess rainfall is calculated from input rainfall using linear discharge conceptualization. Time step and v_time_period are in min so for further calculation it needs to be multiplied with 60 for it to be in sec.*

*For each pixel on the watershed following processing is performed. If the soil moisture is greater than field capacity (a_soil_storage[i][j] > v_field_capacity) subsurface flow is calculated using* $q_{ss} = ( S(t) - S_f )/t_c$ *and stored in v_subsurface_flow, otherwise it is zero. If soil moisture storage is greater than storage capacity ( a_soil_storage[i][j] > (a_porosity[i][j]*a_depth[i][j]) ), overland flow is calculated using* $q_{se} = ( S(t) - S_b )/ \Delta(t)$ *and stored in a_saturation_excess_flow[i][j], otherwise it is zero.* $q_{se}$ *is also known as excess rainfall. Using* $q_{ss}$ *and* $q_{ss}$ *values update the soil moisture storage (* $S(t+\Delta(t)) = S(t) + ( Rainfall - q_{ss} - q_{se})$**time step ).*

## Fn_overland_flow_processing

*Time step is taken as input for this routine. Basic purpose of this routine is to rout the flow as far as it can reach in this given time step. In this beginning l_v_time_remaining = v_time_period. For each pixel following processing takes place.*

*Check if the pixel is a river pixel. If river width value for this pixel is nonzero i.e. it has a nonzero entry in channel width file for this pixel. It can be confirmed by checking that a_slope_n_river_width[i][j][1] if very small. For this code small value considered is 0.001 some one may choose a very small value as the double precision is considered. If the flow is not river flow then, calculate flow by calling routine fn_total_inflow_2_pixel and store it in v_flow. Initialize inflow to pixel array for this pixel (a_inflow_2_pixel[i][j]*

*= 0.0). Call fn_sheet_flow_processing for overland sheet flow processing. l_a_temp_inflow_2_pixel is a temporary array, which stores the inflow to pixel values for all the destination pixels. v_out_flow quantity which is nothing but the v_flow value. This value is added to the mentioned temporary array. Destination pixel is calculated and discussed in fn_sheet_flow_processing routine. Assign zero to v_out_flow as it has already been added to l_a_temp_inflow_2_pixel array.*

*If the flow is channel flow following processing takes place. An important thing that should be noticed is, that unit here is $m^3/s$ not $m^2/s$ as in sheet flow. Channel flow is calculated by calling fn_total_inflow_2_pixel_channel routine and stored in v_flow_channel. Initialize inflow to pixel array for this pixel (a_inflow_2_pixel[i][j] = 0.0). Call fn_channel_flow_processing where channel flow processing takes place. l_a_temp_inflow_2_pixel is a temporary array, which stores the inflow to pixel values for all the destination pixels. v_out_flow quantity which is nothing but the v_flow_channel value. This value is added to the mentioned temporary array. Assign zero to v_out_flow as it has already been added to l_a_temp_inflow_2_pixel array.*

*Add these temporary inflow to pixel values to the a_inflow_2_pixel array.*

## Fn_sheet_flow_processing

*This routine is called for each pixel in routine fn_overland_flow_processing. If the pixel under consideration is a river pixel follow the following instructions.*

*Calculate the Reynolds number of the flow and store in l_v_reynolds_no (Reynolds Number = 4q/ע). Now the Reynolds number criteria given in the parameter input file and stored in l_v_thres_reynlds_no decides about the type of regime (laminar/turbulent). If the flow is laminar velocity of the flow is calculated using fn_laminar_overland_flow_processing function otherwise it is calculated using fn_turbulent_overland_flow_processing. Length of movement will be pixel under consideration to the destination pixel from the vector matrix. Knowing the length and velocity, time taken for movement of water packet from pixel under consideration to destination pixel is calculated. This calculated time is stored in l_v_travel_time variable.*

*If the time remaining in the time step (calculated in overland flow processing) is less than half of the time required, no movement of water takes place, otherwise water moves to next pixel. If it moves to next pixel this same routine is called recursively for movement of water in the remaining time until the time remaining is less than half of the time required for the next movement. When it comes out of this routine the location of water packet will be the destination pixel value. And with these values destination pixel matrix will be updated later.*

*On the other hand, if flow is channel flow, fn_channel_flow_processing is called using l_v_channel_flow flow value. Note the unit of l_v_channel_flow is $m^3/s$ not $m^2/s$.*

## Fn_total_inflow_2_pixel

*This routine calculates total inflow to pixel. Flow from the pixel will be, inflow to pixel under consideration and rainfall taking place of the pixel during the given time step q = inflow to pixel + rainfall. Rainfall is calculated using this formula - rainfall = i * L * cos t.*

## Fn_total_inflow_2_pixel_channel

*This routine calculates total inflow to pixel for a channel pixel. Processing of this routine is same as routine fun_total_inflow_2_pixel except the unit for flow in channel pixel is not unit width.*

## Fn_laminar_overland_flow_processing

*This routine calculates the velocity of the sheet flow, if the flow regime is laminar. Resistant coefficient is calculated using formula Resistance Coefficient($C_L$) = 96 + 108* rainfall(i)$^{0.4}$ and stored in local variable l_v_resis_coeff. l_v_friction_factor is calculated by dividing resistant coefficient by Reynolds number. Water depth is calculated using*

*Darcy-Weisbach Equation: Depth(y) = ( Friction Factor(f) * Flow(qo)^2 / ( 8 * Gravitation Acceleration(g)* Slope(So) ) ) ^ (1/3). Hence velocity of flow is Velocity of flow(V) = Flow(qo) / Depth(y).*

## Fn_turbulent_overland_flow_processing

*This routine calculates the velocity of the sheet flow, if the flow regime is turbulent. Depth of water is calculated using Manning's Equation: Water Depth(y) = ( Manning's Equation(n)*Flow(qo) / ( sqrt(Slope(So)) )^(3/5) and stored in l_v_water_depth. Using this value, l_v_velocity is calculated by dividing l_v_flow by l_v_water_depth.*

## Fn_channel_flow_processing

*Since the outlet point of the watershed is at the end of channel flow. As soon as water flows to the outlet point, no more movement takes place for this water packet. If during processing, pixel under consideration is outlet pixel (provided in the parameter file). Assign the whole flow to this pixel for the current time step and terminate the processing for this time step for the pixel under consideration.*

*Velocity of flow is calculated by calling fn_river_flow_processing and velocity is stored in l_v_velocity. Since we know the velocity and the length of the movement. Again if the remaining time is less than the half of the l_v_travel_time, no more movement takes place. Water packet has reached its destination for the current time step. Otherwise again rountine fn_channel_flow_processing is called recursively.*

## Fn_river_flow_processing

*This routine is used for velocity calculation in channel flow. To start with new depth (l_v_new_depth) is taken as 1.00, and the old depth (l_v_old_depth) is assumed to*

*be very small number -9999.0. Until we get the negligible difference between new calculated depth and depth in the previous step, keep doing the following processing. Newton's Method is used for faster convergence*

$Q_n = 1/n*(So^{1/2})*B*Y_n*((BY_n/(B+2Y_n))^{(2/3)})$

*,and*

$Y_{n+1} = Y_n - (1 - Q/Q_n) /( (5B + 6Y_n)/(3Y_n(B+2Y_n)))$

*After these iterations, the final depth will be the stable depth for Manning's equation. Velocity is calculated by flow/depth formula.*