Supplementary Material

Table S1. Univariable nonlinear regression coefficients estimated without the weight function and the corresponding information criterion.

	Alpha	gamma	beta1	beta2	beta3	beta4	AIC	AICc	BIC	CAIC	RMSE	Ordinary	Adjusted
Logistic	113445	30.276132	4.28E-05				3023.249	3023.427	3032.053	3035.053	12512.4	0.744838	0.74298
	88877.33	**126.02203**		0.002567			3080.693	3080.871	3089.497	3092.497	15384.4	0.614259	0.61144
	89287.81	**106.89921**			0.008454		3097.374	3097.552	3106.177	3109.177	16335.76	0.565076	0.55868
	133667.5	47.308796				0.004533	2955.681	2955.858	2964.484	2967.484	9776.73	0.844216	0.84308
Gompertz	126728.5	3.9927719	2.01E-05				3016.859	3017.036	3025.662	3028.662	12228.06	0.756303	0.75452
	93528	9.2704796		0.001343			3086.569	3086.747	3095.372	3098.372	15713.02	0.597604	0.59467
	90704.24	8.1420232			0.004501		3102.76	3102.938	3111.563	3114.563	16594.44	0.551193	0.54792
	147546.8	4.8720879				0.002137	2956.971	2957.148	2965.774	2968.774	9858.244	0.841608	0.84045
Chapman-Richards	183491.1	1.3625123	6.59E-06				3013.466	3013.644	3022.27	3025.27	12079.74	0.762179	0.76044
	92754.94	8.451908		0.001335			3087.763	3087.941	3096.566	3099.566	15780.64	0.594133	0.59117
	90640.45	6.2761634			0.004151		3102.711	3102.889	3111.514	3114.514	16652.4	0.548052	0.54475
	161035.8	2.5840282				0.001458	2964.968	2965.146	2973.772	2976.772	10145.97	0.832227	0.831
Weibull	170492.2	**2.207E-07**	1.272214				3013.632	3013.809	3022.435	3025.435	12086.94	0.761896	0.76016
	89193.93	7.941E-12		3.34349			3082.459	3082.637	3091.263	3094.263	15482.43	0.609327	0.60933
	85945.33	1.559E-09			3.171833		3098.169	3098.346	3106.972	3109.972	16382.52	0.562583	0.56258
	136964.4	**5.273E-07**				2.086343	2959.68	2959.858	2968.484	2971.484	9954.804	0.83849	0.83731
Modified logistic	230047.7	**6.986E-08**	1.356326				3013.238	3013.416	3022.041	3025.041	12069.83	0.762569	0.76084
	110959.9	3.462E-11		3.134637			3088.627	3088.804	3097.43	3100.43	15829.75	0.591603	0.5916
	96509.45	7.138E-10			3.322003		3102.251	3102.429	3111.055	3114.055	16624.89	0.549544	0.54954

	176622.1	**3.163E-07**				2.144988	2964.087	2964.265	2972.89	2975.89	10113.86	0.833287	0.83207
Lundqvist	1966089	75.817467	0.270111				3013.942	3014.12	3022.745	3025.745	12100.43	0.761364	0.75962
	109033.3	1718048		1.921635			3091.985	3092.163	3100.788	3103.788	16022.13	0.581616	0.57856
	113268.3	**17414.702**			1.563732		3105.634	3105.811	3114.437	3117.437	16828.39	0.538449	0.53166
	460940.3	123.67662				0.614541	2972.016	2972.194	2980.82	2983.82	10406.49	0.823501	0.82221

Note: The bold font denotes that the *p* value of the two-sided t test for the indicated estimated coefficient was larger than the selected significance level (0.05).

Table S2. Multivariable nonlinear regression coefficients estimated with different weight functions and the corresponding information criterion.

	Weight Function	alpha	gamma	beta1	beta2	beta3	beta4	AIC	AICc	BIC	CAIC	RMSE	Ordinary	Adjusted
Logistic	None	135149.0823	37.49313349	1.26473E-05	**0.000229046**	**-0.001175575**	0.003134057	2920.657048	2921.293412	2938.263892	2944.263892	8466.365593	0.883176904	0.879689647
	Andrews	154936.6291	162.4585113	-1.54232E-05	0.00087081	0.001396589	0.003603587	4529.521831	4530.158195	4547.128675	4553.128675	15254.30583	0.995009764	0.99489887
	Bisquare	154971.8214	161.9912456	-1.54119E-05	0.000885088	0.001346106	0.003598053	4529.721412	4530.357776	4547.328256	4553.328256	15254.56931	0.99491816	0.994805231
	Cauchy	152698.0914	55.86846435	2.18135E-05	**2.6666E-05**	**-0.000739561**	0.00277323	3200.691086	3201.327449	3218.297929	3224.297929	9516.295191	0.980893842	0.980323509
	Fair	143821.4625	48.96545902	2.11873E-05	**7.35522E-05**	**-0.000900031**	0.002723388	3068.913848	3069.550212	3086.520692	3092.520692	9018.784603	0.93881806	0.936991733
	Huber	150267.3788	50.15566415	2.60797E-05	**-0.000137128**	**-0.00056993**	0.002599345	3200.932967	3201.569331	3218.539811	3224.539811	9501.337992	0.968339913	0.967394836
	Logistic	149039.6825	51.27849186	2.64607E-05	**-0.000157799**	**-0.000515828**	0.002632182	3171.909258	3172.545622	3189.516102	3195.516102	9481.968055	0.964296116	0.963230328
	Talwar	155278.1435	155.0494226	-1.50301E-05	0.000776505	0.001634023	0.003577631	4502.367969	4503.004333	4519.974813	4525.974813	15119.67789	0.994333436	0.994207513
	Welsch	156260.4671	177.8471322	-1.97264E-05	0.001820947	-0.000975069	0.003394389	5060.627253	5061.263617	5078.234097	5084.234097	17075.19946	0.99511662	0.9950081
Gompertz	None	159914.4094	4.206218532	7.83438E-06	**0.000162228**	-0.000946206	0.001279879	2900.168018	2900.804381	2917.774861	2923.774861	7864.820991	0.89918797	0.896178655
	Andrews	134614.4776	5.216220413	3.29443E-06	**2.75148E-05**	0.00048653	0.001641139	3376.754358	3377.390721	3394.361201	3400.361201	8988.283831	0.992238073	0.992006374
	Bisquare	134496.2043	5.212324719	3.31918E-06	**2.65195E-05**	0.000489489	0.001638736	3377.130191	3377.766554	3394.737034	3400.737034	8988.158212	0.992035958	0.991798225
	Cauchy	126450.6488	4.906275079	6.47532E-06	**-2.23524E-05**	0.000538446	0.001298123	3368.581116	3369.217479	3386.187959	3392.187959	9100.238315	0.989395832	0.98907929
	Fair	181053.7753	4.603343702	7.36103E-06	0.000107843	-0.000441295	0.000990585	3094.174565	3094.810928	3111.781408	3117.781408	8179.879457	0.956490394	0.9551916
	Huber	164048.8355	4.585447365	7.70745E-06	**8.76776E-05**	**-0.000300864**	0.001016771	3156.799681	3157.436045	3174.406525	3180.406525	8245.787522	0.952198727	0.950771823
	Logistic	163822.5194	4.611703951	7.66437E-06	**8.65101E-05**	**-0.000290103**	0.001026399	3149.057534	3149.693897	3166.664377	3172.664377	8246.599428	0.954017788	0.952645184
	Talwar	36584.91005	4.361486515	1.07077E-05	**7.17029E-05**	**0.000654642**	0.002774503	5919.178473	5919.814837	5936.785317	5942.785317	17922.94176	0.985259332	0.984819312
	Welsch	124858.8388	5.047543104	5.67861E-06	**-7.7946E-05**	0.000857776	0.001361292	3414.165621	3414.801985	3431.772465	3437.772465	9275.786993	0.994881284	0.994728487
Modified logistic	None	1553121.081	**1.09681E-06**	35.20713389	1.525607723	15.53046189	1.626074654	2953.640558	2954.276922	2971.247402	2977.247402	9532.881622	0.851890445	0.849712363
	Andrews	9947.589852	**3.86884E-05**	17.96107713	1.416005174	13878552.2	33.90500454	6918.796517	6919.432881	6936.403361	6942.403361	24845.92541	0.790847663	0.789321004
	Bisquare	10035.3485	**3.84538E-05**	17.99240689	1.415413409	14369711.74	33.9775578	6914.672647	6915.30901	6932.27949	6938.27949	24834.59196	0.789280033	0.787741931

	Cauchy	758073.1522	**1.6969E-07**	18.40164637	1.559925486	845.3468053	28.55041094	5238.106724	5238.743087	5255.713567	5261.713567	19786.57731	0.965181195	0.964927043
	Fair	104887.9891	1.05629E-08	16.68499824	2.290129789	16682865.98	496228.6222	4292.342075	4292.978438	4309.948918	4315.948918	17359.09442	0.84663815	0.84663815
	Huber	-1389387.05	-3.19401E-07	20.67856054	1.416166753	6.783946023	33.91585436	4551.644453	4552.280817	4569.251297	4575.251297	18918.32211	0.762588819	0.760855891
	Logistic	104150.5576	2.19892E-08	18.79019783	2.218912772	2.942157726	23.99673822	4314.005685	4314.642049	4331.612529	4337.612529	17330.0593	0.867930121	0.867930121
	Talwar	8531.740475	**2.66348E-05**	19.61500285	1.508567216	27959.71098	49.76627761	6984.631488	6985.267851	7002.238331	7008.238331	24970.06833	0.823637717	0.822350401
	Welsch	1484686.165	**2.64022E-07**	18.05412276	1.367469656	17090975.55	32.67678729	5805.32811	5805.964474	5822.934954	5828.934954	21655.18953	0.857689713	0.856650952

Note: The bold font denotes that the *p* value of the two-sided t test for the indicated estimated coefficient was larger than the selected significance level (0.05).

Figure S1. Main MATLAB code.

```
clc
close all
clear all
%% Exact basin
tic
% load('LiDAR_block.shp');

j = imread('DEMz1.tif');
load('FA.mat');
FA = FlowAccumulation;

FlowAccumulation = FlowAccumulation.^(1/2);
H0 = imread('DEMz1.tif');
H0 = double(H0);
[r3, c3] = size(H0);
H0 = reshape(H0,[],1);
H0(H0<0) = 0;
H0 = reshape(H0,r3,c3);
ExcelName = 'Final Results sum Lvbo.xlsx'; % Save data as the Excel file
K = ones(1)./1;
Limit = 2;

levelNo = 15; % the number of basin area choice for research (max = 15)
% which accords to the number you selected in GIS
%

submax = 5; % the number of sub-basin area choice for research in the each
% above basin area (Recommendation 5)
%

iterationmax = 10; % the number of surface runoff area choice
% in each basin area (tried)
%



LineNo = 10; % the number of surface runoff line choice
% in each surface runoff area (tried)
%

% total largest number of Figures would be levelNo * iterationmax * LineNo
%

Posi = [208,117,1220,737];

Zsmooth1 = conv2(FlowAccumulation,K,'same'); % Convolution
% Nothing changes if K = 1;
FlowAccumulation = Zsmooth1;
% FlowAccumulation = Zsmooth1.^(1/2);
FlowAccFig = FlowAccumulation ;
H3 = imread('Penyu.tif');
H3 = double(H3);

[r3, c3] = size(H3);
H3 = reshape(H3,[],1);
H3(H3<0) = 0;
% H1(H1>0) = 1;
```

```
H3 = reshape(H3,r3,c3);


Htpi = imread('myTPI.tif');
Htpi = double(Htpi);

[r3, c3] = size(Htpi);
r = r3;
c = c3;
Htpi = reshape(Htpi,[],1);
Htpi(Htpi<0) = 0;
Htpi = reshape(Htpi,r3,c3);

AreaMax = 2000;

%% Colormap set
cm = colormap('hot');
cm = flipud(cm);
colorhot = colormap('hot');
colorhot = rot90(colorhot,2);
%% Initial
maxi = max(H3,[],'all');

level = [1 57 113 174 229 273 327 394 453 506 556 608 661 729 795 842];
% According to the number given by GIS

for i = 1:length(level)-1
    H3(H3>level(i) & H3<=level(i+1)) = i;
end
H30 = H3;
t = 0;
FlowAccsub = zeros(r,c,70);
DEMsub = zeros(r,c,70);
TPIsub = zeros(r,c,70);
%% Preparation of Exact basin
for i = 1:levelNo

    if i < 15
        modification = 30;
    else
        modification = 1;
    end
    b = max(H3,[],'all');
    if b == 0
        break
    end
    [rb0, cb0] = find(H3 == i);
    %     [rb, cb] = find(H3 == i);
    FlowAccBasinArea = zeros(r3,c3);
    FlowAccsubtraction = zeros(r3,c3);
    for j = 1:length(rb0)
        FlowAccBasinArea(rb0(j),cb0(j)) = FlowAccumulation(rb0(j),cb0(j));
        FlowAccsubtraction(rb0(j),cb0(j)) = H3(rb0(j),cb0(j));
    end

    %% Basin Sub Area
```

```
sub = 0;
while size(rb0,1) > 10 && sub < 5
   Lengthsave0 = zeros(1,iterationmax*LineNo);
   sub = sub + 1;
   row = rb0(1);
   column = cb0(1);
   No = 2000;
   SubAreaId = zeros(No,2);
   %    TPIID = zeros(No,2);
   for n = 1:No

      if n == 1
         x = [row, column];
         y = FindContour(x,r,c);
         SubArea = zeros(length(y),2);
      else
         x = SubAreaId;
         y = FindContour(x,r,c);
         SubArea = zeros(length(y),2);
         old = SubAreaId;
      end
      k = 0;
      for m = 1:length(y)
         judge = H30(y(m,1),y(m,2));

         if judge == i
            k = k + 1;
            SubArea(k,:) =  y(m,:);
         end
      end
      SubArea(SubArea==0) = [];
      SubArea = reshape(SubArea,[],2);

      if size(SubArea,1) > 0 || n == 1

         if n == 1
            SubAreaId = [x;SubArea];
         else
            SubAreaId = [SubAreaId;SubArea];
            SubAreaId = unique(SubAreaId,'rows');
         end

      else
         break
      end
   end



   %% Runoff Area
   IDmin = min(SubAreaId,[],1);
   IDmax = max(SubAreaId,[],1);
   ID(sub,:) = [IDmin IDmax];
   ExtractSub = zeros(r,c);
   Hsub = zeros(r,c);
   DEMsubeach = zeros(r,c); % to save each value of DEM
```

```
TPIsubeach = zeros(r,c); % to save each value of TPI
for l = 1:size(SubArea,1)
    ExtractSub(SubArea(l,1),SubArea(l,2)) = ...
        FlowAccBasinArea(SubArea(l,1),SubArea(l,2));
    Hsub(SubArea(l,1),SubArea(l,2)) = ...
        H3(SubArea(l,1),SubArea(l,2));
    DEMsubeach(SubArea(l,1),SubArea(l,2)) = ...
        H0(SubArea(l,1),SubArea(l,2));
    TPIsubeach(SubArea(l,1),SubArea(l,2)) = ...
        Htpi(SubArea(l,1),SubArea(l,2));

end
[rb, cb] = find(Hsub == i);
if size(rb,1) < 100  % Discard the area less than 100 girds
    rb0(1) = [];
    cb0(1) = [];
    sub = sub -1;
    continue
end
%% Save Flow Accumulation, TPI and DEM
FlowAccsave0 = sum(ExtractSub.^2,'all');
t = t + 1;
FlowAccsub(:,:,t) = ExtractSub;
DEMsub(:,:,t) = DEMsubeach;
TPIsub(:,:,t) = TPIsubeach;

Hplot = Hsub;
Hplot(Hplot>0) = modification;
Hplot = reshape(Hplot,r3,c3);
figure1 = figure(2);
colormap(spring);
axes1 = axes('Parent',figure1);
hold(axes1,'on');
hold on
ss1 = surf(axes1,H30,'Parent',axes1,'EdgeColor','none','FaceAlpha',0);
ss2 = surf(axes1,Hplot,'Parent',axes1,'EdgeColor','none');
hold off

xlim(axes1,[min(cb)-2 max(cb)+2]);
ylim(axes1,[min(rb)-2 max(rb)+2]);

grid(axes1,'on');
hold(axes1,'off');
% Set the remaining coordinate area properties
set(axes1,'Color','none','Colormap',cm,'DataAspectRatio',[1,1,0.5]);
view(axes1,[0 -90]);
cb1 = colorbar(axes1,'Position',...
   [0.077 0.146862483311081 0.0229612109744559 0.671537516688922]);
cb1.Label.String = 'Level';
cb1.Label.FontSize = 12;
% Create axes
axes2 = axes('Parent',figure1);
hold(axes2,'on');
% Create surf
hold on
ss1 = surf(axes2,FlowAccumulation,'Parent',axes2,'EdgeColor','none','FaceAlpha',0);
```

```
ss2 = surf(axes2,ExtractSub,'Parent',axes2,'EdgeColor','none','FaceAlpha',0.8);
cb2 = colorbar(axes2,'Position',...
    [0.93355737704918,0.146862483311081,0.022961210974456,0.671537516688921]);
cb2.Label.String = 'SQRT of Flow Accumulation';
cb2.Label.FontSize = 12;
hold off
xlim(axes2,[min(cb)-2 max(cb)+2]);
ylim(axes2,[min(rb)-2 max(rb)+2]);
%       zlim(axes2,[0 40]);
grid(axes2,'on');
hold(axes2,'off');
set(axes2,'Color','none','Colormap',colorhot,...
    'DataAspectRatio',[1,1,0.5]); %,'XTick',[],'YTick',[]
view(axes2,[0 -90]);
set(figure1,'Position',Posi);
print(['An example',num2str(i),'(',num2str(sub),')'],'-djpeg','-r600')
close all
FlowAccBasinArea = FlowAccBasinArea - ExtractSub;
H30 = H30 - Hsub;


Boundary = boundary(SubArea(:,1),SubArea(:,2),1);
polyin = polyshape([SubArea(Boundary,1)';SubArea(Boundary,2)']');
%       plot(polyin)
%       view([90 90]);
Boundarycircumference = perimeter(polyin);
Boundarycircumference = Boundarycircumference.*2;
FindMaxLength = pdist2(SubArea(Boundary,:),SubArea(Boundary,:),'euclidean');
MaxLength = max(FindMaxLength,[],'all');

[xL, ~] = find(FindMaxLength == MaxLength);
MaxLength = MaxLength.*2;
if size(xL,1) == 0
    continue
end


% Create figure
close all
figure1 = figure;

% Create axes
axes1 = axes('Parent',figure1);


% Create plot
plot(SubArea(:,1),SubArea(:,2),'Marker','.','LineStyle','none');
hold on
plot(SubArea(Boundary,1),SubArea(Boundary,2),'-m','LineWidth',1.8);
plot([SubArea(Boundary(xL(1)),1) SubArea(Boundary(xL(2)),1)],...
    [SubArea(Boundary(xL(1)),2) SubArea(Boundary(xL(2)),2)],'r-s',...
    'LineWidth',4.5,'MarkerFaceColor','r','MarkerSize',8)
view(axes1,[90 90]);
box(axes1,'on');

% Set the remaining axes properties
```

```
286         set(axes1,'DataAspectRatio',[1 1 1],'PlotBoxAspectRatio',...
287             [76.0736196319018 60 1]);
288         set(figure1,'Position',[680,58,947,920]);
289
290         title({[num2str(i),'(',num2str(sub),') Boundary Circumference = ',num2str(Boundarycircumference),' m'];...
291             ['Max Length of Basin = ',num2str(MaxLength),' m']})
292         print(['An example',num2str(t),' Boundary'],'-djpeg','-r600')
293         close all
294
295         %% Save value
296         if i == 1 && sub == 1
297             S = size(SubArea,1)*2^2;
298             Name = num2str(t);
299             Name = convertCharsToStrings(Name);
300             Boundarycircumferencesave = Boundarycircumference;
301             MaxLengthsave = MaxLength;
302             FlowAccsave = FlowAccsave0;
303         else
304             S = [S;size(SubArea,1)*2^2];
305             Name = [Name;convertCharsToStrings(num2str(t))];
306             Boundarycircumferencesave = [Boundarycircumferencesave;Boundarycircumference];
307             MaxLengthsave = [MaxLengthsave;MaxLength];
308             FlowAccsave = [FlowAccsave;FlowAccsave0];
309         end
310
311         %% Extract Runoff Area
312         for iteration = 1:iterationmax
313             if sum(ExtractSub,'all') < 200 % Discard the area with Flow
314                 % Accumulation less than 200
315                 break
316             end
317             max1 = max(ExtractSub,[],'all');
318             [row, column] = find(ExtractSub == max1);
319             No = 500;
320             RunoffAreaId = zeros(No,2);
321             for n = 1:No
322                 if n == 1
323                     x = [row, column];
324                     y = FindContour(x,r,c);
325                     RunoffArea = zeros(length(y),2);
326                 else
327                     x = RunoffAreaId;
328                     y = FindContour(x,r,c);
329                     RunoffArea = zeros(length(y),2);
330
331                 end
332                 k = 0;
333                 for m = 1:length(y)
334                     judge = ExtractSub(y(m,1),y(m,2));
335
336                     if judge > Limit
337                         k = k + 1;
338                         RunoffArea(k,:) =  y(m,:);
339                     end
340                 end
341                 RunoffArea(RunoffArea==0) = [];
```

```
            RunoffArea = reshape(RunoffArea,[],2);
            if n == 1
                RunoffAreaId = [x;RunoffArea];
            else
                RunoffAreaId = [RunoffAreaId;RunoffArea];
                RunoffAreaId = unique(RunoffAreaId,'rows');
            end
        end
        %% Runoff Area
        IDmin = min(RunoffAreaId,[],1);
        IDmax = max(RunoffAreaId,[],1);
        ID(iteration,:) = [IDmin IDmax];
        Extract = zeros(r,c);

        for l = 1:length(RunoffArea)
            Extract(RunoffArea(l,1),RunoffArea(l,2)) = ...
                ExtractSub(RunoffArea(l,1),RunoffArea(l,2));
        end

        if sum(Extract,'all') < 10
            break
        end

        figure1 = figure(2);
        colormap(spring);
        axes1 = axes('Parent',figure1);
        hold(axes1,'on');
        hold on
        ss1 = surf(axes1,H30,'Parent',axes1,'EdgeColor','none','FaceAlpha',0);
        ss2 = surf(axes1,Hplot,'Parent',axes1,'EdgeColor','none');
        hold off
        xlim(axes1,[min(cb)-2 max(cb)+2]);
        ylim(axes1,[min(rb)-2 max(rb)+2]);
        %       zlim(axes1,[0.5 5000]);
        grid(axes1,'on');
        hold(axes1,'off');
        % Set the remaining coordinate area properties
        set(axes1,'Color','none','Colormap',cm,'DataAspectRatio',[1,1,0.5]);
        view(axes1,[0 -90]);
        cb1 = colorbar(axes1,'Position',...
            [0.077 0.146862483311081 0.0229612109744559 0.671537516688922]);
        cb1.Label.String = 'Level';
        cb1.Label.FontSize = 12;
        % Create axes
        axes2 = axes('Parent',figure1);
        hold(axes2,'on');
        % Create surf
        hold on
        ss1 = surf(axes2,FlowAccumulation,'Parent',axes2,'EdgeColor','none','FaceAlpha',0);
        ss2 = surf(axes2,Extract,'Parent',axes2,'EdgeColor','none','FaceAlpha',0.8);
        cb2 = colorbar(axes2,'Position',...
            [0.93355737704918,0.146862483311081,0.022961210974456,0.671537516688921]);
        cb2.Label.String = 'SQRT of Flow Accumulation';
        cb2.Label.FontSize = 12;
        hold off
```

```
xlim(axes2,[min(cb)-2 max(cb)+2]);
ylim(axes2,[min(rb)-2 max(rb)+2]);
zlim(axes2,[0 40]);
grid(axes2,'on');
hold(axes2,'off');
set(axes2,'Color','none','Colormap',colorhot,...
    'DataAspectRatio',[1,1,0.5],'XTick',[],'YTick',[]);
view(axes2,[0 -90]);
set(figure1,'Position',Posi);
print(['An example',num2str(t),'-',num2str(iteration)],'-djpeg','-r600')
close all
ExtractSub = ExtractSub - Extract;

%% Surface Line
for j = 1:LineNo
    if sum(Extract,'all') < 10
        break
    end
    max1 = max(Extract,[],'all');
    [row, column] = find(Extract == max1);
    Extract(row,column) = 0;
    No = 1200;
    RunoffId = zeros(No,2); % indice of surface runoff
    RunoffId(1,:) = [row, column];
    for m = 2:No
        if row-1 == 0 || row+1 == r || column-1 == 0 || column+1 == 0
            break
        else
            ninegrids = Extract(row-1:row+1,column-1:column+1);
        end
        ninegrids(2,2) = 0;
        if m > 2
            rowlast = RunoffId(m-2,1);
            columnlast = RunoffId(m-2,2);
            rowrecent = RunoffId(m-1,1);
            columnrecent = RunoffId(m-1,2);

            deltax = rowrecent - rowlast;
            deltay = columnrecent - columnlast;

            if deltax == -1 && deltay == 0 % ↑↑ North
                ninegrids(3,:) = 0;
            elseif deltax == -1 && deltay == 1
                ninegrids(2,1) = 0;
                ninegrids(3,1) = 0;
                ninegrids(3,2) = 0;
            elseif deltax == 0 && deltay == 1 % ¶« East
                ninegrids(:,1) = 0;
            elseif deltax == 1 && deltay == 1
                ninegrids(1,1) = 0;
                ninegrids(1,2) = 0;
                ninegrids(2,1) = 0;
            elseif deltax == 1 && deltay == 0 % ÄÍ South
                ninegrids(1,:) = 0;
            elseif deltax == 1 && deltay == -1
```

```
            ninegrids(1,2) = 0;
            ninegrids(1,3) = 0;
            ninegrids(2,3) = 0;
        elseif deltax == 0 && deltay == -1 % Î÷ West
            ninegrids(:,3) = 0;
        elseif deltax == -1 && deltay == -1
            ninegrids(3,2) = 0;
            ninegrids(3,3) = 0;
            ninegrids(2,3) = 0;
        end
    end
    M = max(ninegrids,[],'all');
    if M < 1
        break
    end
    [row, column] = find(Extract == M);
    Extract(row,column) = 0;
    RunoffId(m,:) = [row column];
end
RunoffId(RunoffId == 0) = [];
RunoffId = reshape(RunoffId,[],2);

%% The length of runoff line
LineLength = 0;
for m = 2:size(RunoffId,1)
    p1 = RunoffId(m-1,:);
    p2 = RunoffId(m,:);
    LineLength = LineLength + distanceHHS(p1,p2);
end
Lengthsave0(1,LineNo*(iteration - 1)+j) = LineLength;


%% Runoff Area
ExtractLine = zeros(r,c);
for k = 1:size(RunoffId,1)
    ExtractLine(RunoffId(k,1),RunoffId(k,2)) = ...
        FlowAccFig(RunoffId(k,1),RunoffId(k,2));
end

figure1 = figure(2);
colormap(spring);
axes1 = axes('Parent',figure1);
hold(axes1,'on');
hold on
ss1 = surf(axes1,H30,'Parent',axes1,'EdgeColor','none','FaceAlpha',0);
ss2 = surf(axes1,Hplot,'Parent',axes1,'EdgeColor','none');
hold off
xlim(axes1,[min(cb)-2 max(cb)+2]);
ylim(axes1,[min(rb)-2 max(rb)+2]);
zlim(axes1,[0.5 5000]);
grid(axes1,'on');
hold(axes1,'off');
% Set the remaining coordinate area properties
set(axes1,'Color','none','Colormap',cm,'DataAspectRatio',[1,1,0.5]);
view(axes1,[0 -90]);
```

```
                % Create colorbar
                cb1 = colorbar(axes1,'Position',...
                    [0.077 0.146862483311081 0.0229612109744559 0.671537516688922]);
                cb1.Label.String = 'Level';
                cb1.Label.FontSize = 12;
                % Create axes
                axes2 = axes('Parent',figure1);
                hold(axes2,'on');
                % Create surf
                hold on
                ss1 = surf(axes2,FlowAccumulation,'Parent',axes2,'EdgeColor','none','FaceAlpha',0);
                ss2 = surf(axes2,ExtractLine,'Parent',axes2,'EdgeColor','none','FaceAlpha',0.8);
                cb2 = colorbar(axes2,'Position',...
                    [0.933557377049180,0.146862483311081,0.022961210974456,0.671537516688921]);
                cb2.Label.String = 'SQRT of Flow Accumulation';
                cb2.Label.FontSize = 12;
                hold off
                title(['Length = ',num2str(LineLength),' m'])
                xlim(axes2,[min(cb)-2 max(cb)+2]);
                ylim(axes2,[min(rb)-2 max(rb)+2]);
                zlim(axes2,[0 40]);
                grid(axes2,'on');
                hold(axes2,'off');
                set(axes2,'Color','none','Colormap',colorhot,...
                    'DataAspectRatio',[1,1,0.5],'XTick',[],'YTick',[]);
                view(axes2,[0 -90]);
                set(figure1,'Position',Posi);
                print(['An example',num2str(t),'-',num2str(iteration),'-',num2str(j)],'-djpeg','-r600')
                close all
                Extract = Extract - ExtractLine;


            end

        end
        %% Save value of Length
        Lengthsave0 = sort(Lengthsave0,'descend');
        if i == 1 && sub == 1
            Lengthsave = Lengthsave0;
        else
            Lengthsave = [Lengthsave;Lengthsave0];
        end

    end
end
VariableName = ["ID","Flow Accumulation","S","Boundary circumferencesave","MaxLengthsave","Each Length of surface runoff"];
MatrixSave = [FlowAccsave';S';Boundarycircumferencesave';MaxLengthsave';Lengthsave']';
writematrix(VariableName,ExcelName,'Sheet',1,'Range','B1');
writematrix(Name,ExcelName,'Sheet',1,'Range','B2');
writematrix(MatrixSave,ExcelName,'Sheet',1,'Range','C2');
save('FlowAccsub.mat','FlowAccsub');
save('DEMsub.mat','DEMsub');
save('TPIsub.mat','TPIsub');
```

```
toc


function y = FindContour(x,r,c)
% Note:
% The function is used to find the coordinate index of Contour point (?)
% surrounding the given inner points (*).
%    ? ? ?
%    ? * ?
%  ? ? * ? ?
%  ? * * * ?
%  ? ? ? ? ?
% USAGES
% y = FindContour(x)
%
% INPUT:
% x: The coordinate index of inner points
% r: The maximum range of row value
% c: The maximum range of column value
% OUTPUT:
% y: The coordinate index of Contour point surrounding the inner points
%% read original data and find size
[rold, ~] = size(x);

%% creat a full matrix to save result data
Full = zeros(rold*8,2);

%% creat a direction matrix
direction = [...
    -1  0;...due north
    -1 +1;...northeast
    0 +1;...due east
    +1 +1;...southeast
    +1  0;...due south
    +1 -1;...southwest
    0 -1;...due west
    -1 -1];% northwest
for i = 1:rold
    Full(1+(i-1)*8:i*8,:) = x(i,:) + direction;
end

y = unique(Full,'rows');

[rtest,~] = find(y(:,1) == 0 | y(:,1) == r+1 | y(:,2) == 0 | y(:,2) == c+1);
y(rtest,:) = [];

end

function d = distanceHHS(p1,p2)

d = ((p1(1,1) - p2(1,1)).^2 + (p1(1,2) - p2(1,2)).^2).^(1/2);
end

%
% function L = maxLength(p)
% % Note: Find the maximum length of a series of given points
% %
```

```
% % USAGES
% % L = maxLength(p)
% %
% % INPUT:
% % p: Inputting a series of given points
% % OUTPUT:
% % y: Find the maximum length of a series of given points
% for i = size(p,1)
%
%
%
%
% end


% function F = PointSort(p)
% % Note: Sorting the vertex of a polygon clockwise
%
% % USAGES
% % F = PointSort(p)
% %
% % INPUT:
% % p: Inputting the vertex of a polygon
% % OUTPUT:
% % y: Sorting the vertex of a polygon as clockwise
% centerx = mean(p(:,1));
% centery = mean(p(:,2));
% p(:,1) = p(:,1) - centerx;
% p(:,2) = p(:,2) - centery;
% p = [p;p(1,:)]; %
% for i = 1:size(p,1)-1
%     for j = 1:size(p,1)-i
%         crossproduct = det([p(j,1) p(j,2);p(j+1,1) p(j+1,2)]);
%         if crossproduct < 0
%             p([j,j+1],:) = p([j+1,j],:);
%         end
%     end
% end
% p = unique(p,'rows','stable');
% F = p;
%
%
% end
%
```