Computer simulations of cobble structure on a gravel stream bed

by

Selina Tribe

B.Sc., University of British Columbia, 1994

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE STUDIES

(Department of Geography)

We accept this thesis as conforming
to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

September 1996

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of GEOGRAPHY

The University of British Columbia
Vancouver, Canada

Date SEPT 27, 1996

DE-6 (2/88)

Abstract

In Harris Creek, a gravel bed river in southcentral British Columbia, the large stones protruding from the surface armour of the bed are arranged as curvilinear ridges oriented at varying angles to the flow, defining a net-like surface structure. A computer simulation, using various rules governing the entrainment and deposition of stones, was written to reproduce these structures. The working hypothesis for the simulation is that the patterns are the result of the mutual interaction of stones as they are mobilized into the flow and collide with downstream obstacles. An opposing hypothesis would be that the pattern being imposed on the stones by the flow. A statistical algorithm was developed to identify cobble ridges and their orientations and to allow a comparison between the simulation results and the real example. Simulations using different sized stones in a bed with 10% of its area covered by stones produce flow-parallel, linear clusters resembling cluster bedforms. Simulations using a more densely populated bed, up to 25% covered area, produce a variety of gravel bedforms including cluster bedforms, transverse ribs, some trending at oblique orientations, and arcuate ridges. These results indicate that the magnitude of the stone population influences the suite of bedforms that emerge. Cluster bedforms may be the result of obstacles to the downstream movement of stones in a sparsely populated bed. Furthermore, cluster bedforms may be nuclei from which oblique ridges develop out of excess sediment. Finally, simulations in which large stones are rarely entrained and travel shorter distances than small stones, produce structures very similar to those found in Harris Creek.

Contents

List of Tables

List of Figures

Acknowledgements

Mike Church initially suggested this topic for study and the working hypothesis. He gave insightful comments, suggestions and encouragement throughout the course of this work. Birger Bergersen suggested the linked list approach and Rik Blok provided several tutorials on linked lists. Both Birger Bergersen and Brian Klinkenberg provided access to computational facilities. Jerry Maedel in the Forestry Department provided digitizing facilities and both he and Kurtis Halingten assisted in the digitizing. Kurtis Halingten assisted in developing the statistical algorithm, writing the Mathematica code, and solving numerous problems relating to conflicting file formats and non-compatible software. This work was done under the auspices of a University Graduate Fellowship.

SELINA TRIBE

The University of British Columbia
September 1996

Chapter 1

Introduction

When viewed from above, the cobbles protruding from the bed of Harris Creek are arranged into a net-like structure of curvilinear ridges (figure 1.1). How does this pattern form? The purpose of this study is to reproduce the cobble patterns in Harris Creek in a computer simulation. The design of the simulation is guided by the working hypothesis that the pattern is due to the mutual interaction of cobbles as they are entrained, transported and deposited, in contrast to the hypothesis that the pattern is imposed on the cobbles by the water flow. If the simulated structures resemble those in Harris Creek then inferences may be made about how the real structure is formed and possibly about coarse sediment transport in general.

For those readers not versed in sediment transport, the following explanatory statements will help demystify the jargon used here. Although most of the terms *cobble*, *gravel*, *clast* and *stone* have strict size definitions in sedimentology, they are used interchangeably here. The size of a stone is denoted D_{50} if its diameter is that of the median size of the population. In a like manner, a stone denoted D_{95} has a diameter as large as the 95^{th} percentile of stones in the population. The shape and orientation of irregular stones can be characterized as the 3 orthogonal axes, a, b and c of an approximating ellipsoid. The a axis defines the largest diameter of the stone and the b and c axes define the intermediate and smallest diameters respectively. The most hydrodynamically stable position for stones is to be oriented with the plane containing the ab axes dipping upstream because if they were positioned such that this plane was dipping downstream, the flow would catch the upturned end and flip the stone over. Stones in the stable orientation are called *imbricated.* The *lee* is the area immediately downstream of a stone while the *stoss* is the area immediately upstream of the stone. If a stone is positioned on the bed without any touching or near neighbours, it is in an *open* or *isolated* position. This is in contrast to stones touching or close to other stones which are said to be in *closed* or *shielded* positions. A stone is *entrained* at the precise moment it is mobilized into the flow and then it is *transported* along the river bed until finally it stops, hence becoming *deposited.*

Figure 1.1: Vertical airphoto of the Harris Creek study reach showing cobbles protruding from the stream bed and defining a net-like structure. This photo shows the upstream third of section B as mapped in figure 2.1. The scale is visible as two perpendicular white 1.2 meter long sticks resting on some stones. A rope by which the suspended camera was towed along the reach is visible in the bottom right.

Chapter 2

The Cobble Patterns in Harris Creek

2.1 Description

The cobble patterns in Harris Creek comprise curvilinear ridges of stones protruding from the plane of the river bed defining a moderately well-developed, net-like structure (figure 2.1). The ridges are commonly 1 to 3 stones thick and 1 stone high and composed of the coarsest cobble and boulder fraction available in the creek. The ridges extend from 3 to 20 stone diameters and any one ridge can span approximately 25 to 75% of the uncovered width of the reach. The ridges are straight in some regions and in other regions are arcuate with the concave side facing either upstream or downstream. Despite the irregularity in their pattern, the ridges tend to form a net-like or rib-like structure whose segments are oriented oblique to the flow direction (figure 2.3) and spaced on order of 1 to 2 meters or 5 to 10 stone diameters apart.

The cobble ridges are made up of sub-rounded to sub-angular cobbles and boulders of diverse lithologies with an average diameter of 20 centimeters (as measured from figure 2.1 and described in [9], riffle surface material) identifying them as the coarsest fraction of sediment available in the creek, comparable with the D_{95} of the bulk bed material. The cobble ridges have been observed to form during moderate floods capable of moving coarse stones and typical of snowmelt gravel bed streams. This arrangement of stones is also observed in Furry Creek, located in the southwestern Coast Mountains, British Columbia [10].

The study reach in which the cobble patterns are found is a 60 m long riffle portion of Harris Creek, a snowmelt flood regime creek draining a portion of the Okanagan highlands in southcentral British Columbia [9]. The reach is approximately 5 to 7 meters wide with an upstream boundary located approximately 6 m upstream of the first appearance of the structure and a downstream boundary located where the river narrows to half its average width resulting in a zone of convergent, super-critical flow. In the study reach, three domains of approximately 15 m long and 5 m wide exhibit representative examples of the cobble

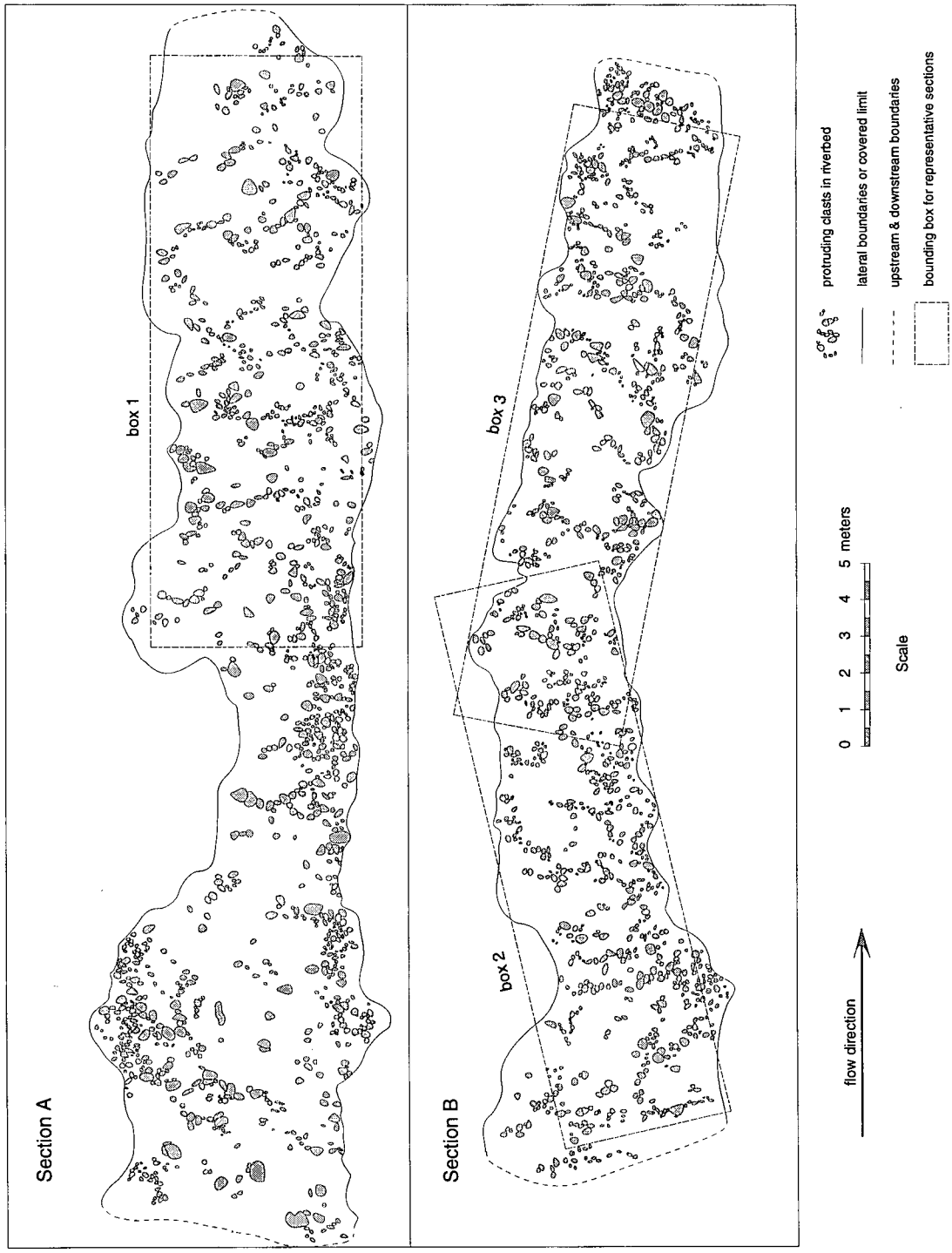


Figure 2.1: Plan view map of the cobble patterns in Harris Creek showing the outline of all protruding stones on the river bed. Section B continues approximately 5 m downstream of the end of section A. The three rectangular boxes outline representative examples of the cobbles patterns and are shown magnified in figure 2.2.

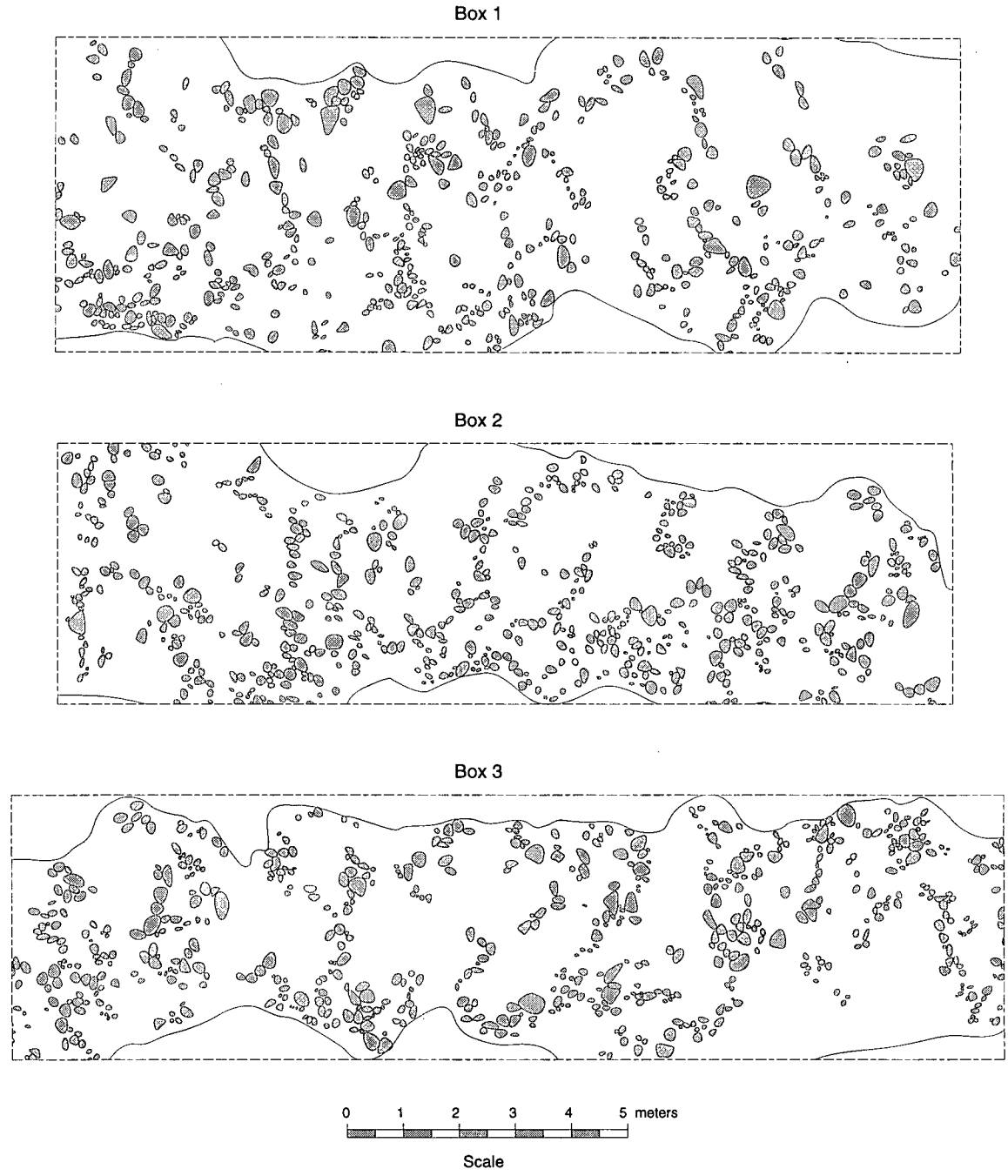


Figure 2.2: Large scale view of protruding stones in three select areas of Harris Creek where the net-like pattern of cobble ridges is well displayed. The cobble patterns can be seen to incorporate linear, arcuate and ramifying ridges of stones as well as isolated stones, clusters and rib elements. A well-developed cluster bedform is evident in the center of bed at the downstream margin of Box 1 and several others are visible in the other boxes. An example of a transverse rib can be found 3.5 m upstream of the cluster bedform in Box 1. Flow direction is to the right.

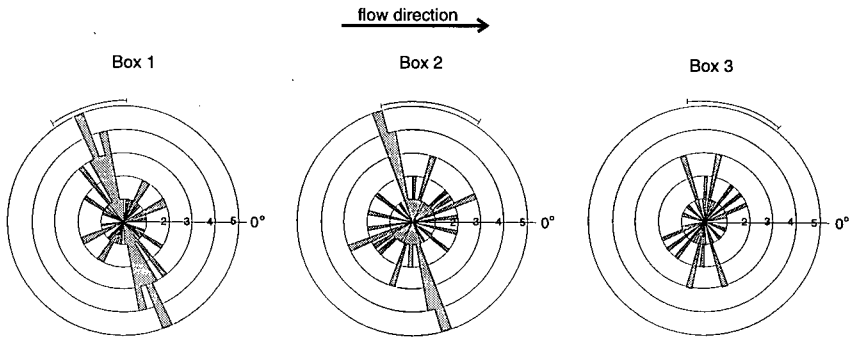


Figure 2.3: Rose diagrams for each of the representative segments of Harris Creek's cobble patterns. The arcs represent the angular deviation in the population. Flow direction is toward 0°. The statistics for these roses are tabulated in Table 2.1.

Table 2.1: Statistics of the rose diagrams (figure 2.3) for the line segment approximation of the cobble patterns (figure 2.4).

statistic	Box 1	Box 2	Box 3
N[a]	37	39	24
class interval	5°	5°	5°
maximum[b]	5	5	3
maximum percentage[c]	13.5	12.8	12.5
mode[d]	160°	165°	165°
mean[e]	342.7°	9.05°	15.25°
angular deviation	37.25°	48.85°	45.8°

[a]Total number of observations.
[b]Maximum number of observations in a single class.
[c]Maximum percentage of total observations in a single class.
[d]Mode: class interval with maximum number of observations.
[e]Mean: azimuth of resultant vector.

patterns and are outlined on figures 2.1, 2.4 and 2.7. The verisimilitude of the computer simulations is judged by comparing computed results with the areas within these three boxes.

Harris Creek's cobble ridges are aligned at sub-transverse angles to the mean flow direction as shown by the bi-directional rose diagrams[1] in figure 2.3. The associated statistics appear in Table 2.1. To generate these diagrams, cobble ridges of three or more stones in extent were approximated by line segments. For each of the boxes outlined in figures 2.1, 2.4 and 2.7, the angle that each line segment made with a line running perpendicular to the box's long dimension, taken to be transverse to the mean flow direction, was tabulated. Arcuate ridges require several measurements. This qualitative test for preferred orientation reveals that ridges tend to be aligned transverse rather than parallel to the flow.

[1]Rose diagrams made using *Vector Rose 2.08, Orientation analysis and statistics for directional data*, 1988-1992, software for the Macintosh written by P. Zippi, 1987.

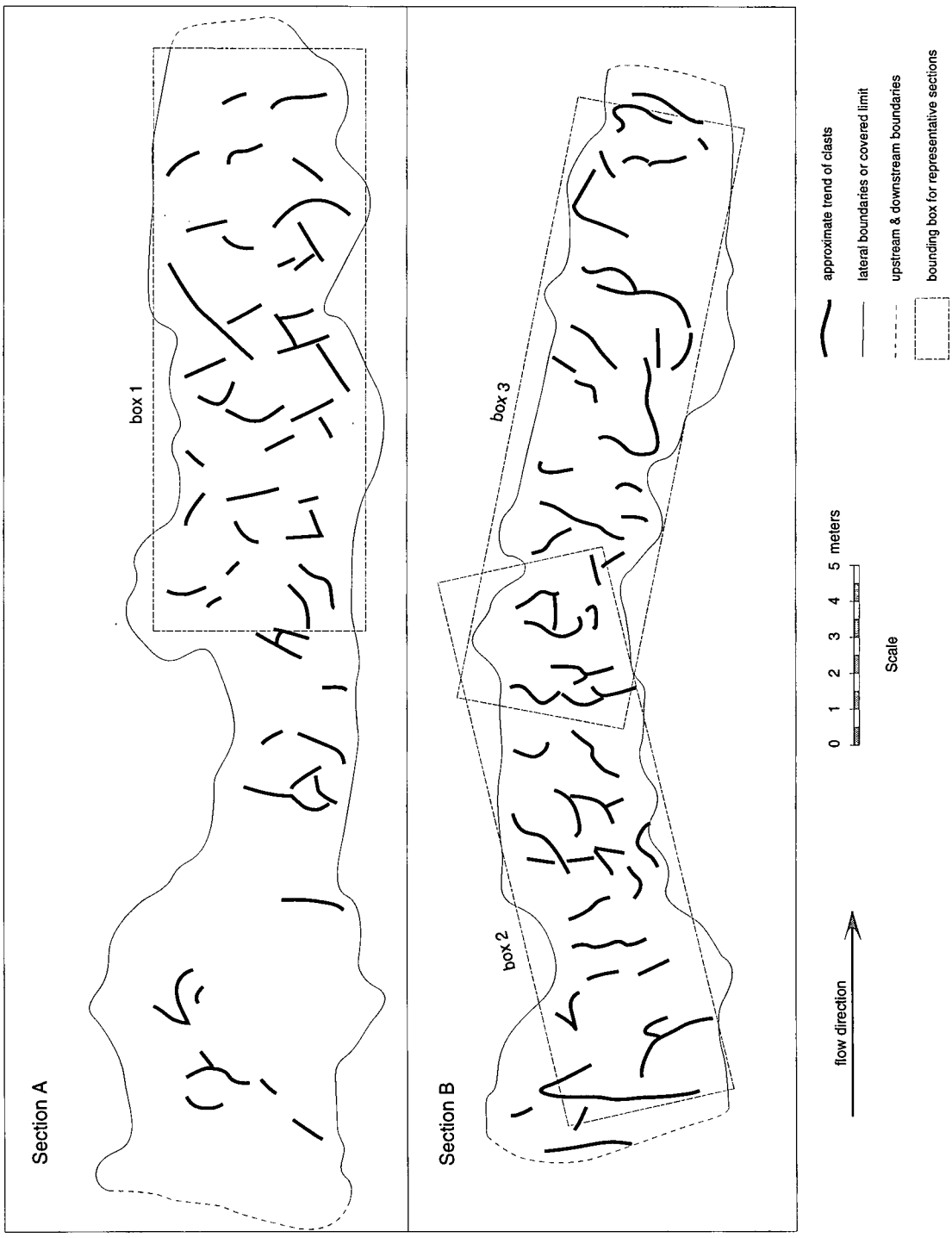


Figure 2.4: Line segment approximation of Harris Creek's cobble patterns drawn to the same scale as figure 2.1 and used to generate the data for the rose diagrams of figure 2.3. Each line represents the trend of a cobble ridge of 3 or more clasts in extent.

2.2 Generating the map of Harris Creek

In 1989, when the creek was at low flow, Harris Creek's cobble patterns were photographed with a gimbal-mounted camera suspended by helium-filled balloons. The camera and balloons were anchored and positioned by three people holding onto kevlar lines who slowly towed the apparatus above and along the reach. Every 5 to 10 meters the shutter was activated by remote control in order to obtain overlapping photographs for stereo viewing. The scale, in this case two 1.2 meter long sticks oriented perpendicular to each other, was laid on exposed rocks and repositioned every twenty meters or so. The resulting photographs provide an aerial, stereo view of the cobble structures on the river bed with a scale in almost every frame (figure 1.1).

The plan-view map of the cobble structures was made by viewing each portion of the reach in stereo and outlining on a mylar overlay every clast observed to protrude from the river bed. Care was taken to avoid excess translation or rotation of each set of stereo photos used in tracing successive portions of the reach. The lateral boundaries of the study reach are the limit of overhanging trees for approximately one quarter of the length and the river bank for the remaining length. After the cobble outlines and reach boundaries were traced onto the mylar, the overlay was scanned and used as a digital template in Adobe Illustrator to produce figure 2.1.

Cobbles protruding from the bed were identified by several criteria with varying degrees of ease and accuracy. Those that stuck out above the local water surface were the easiest to map, especially when leaves and other debris were trapped in their lee. Submerged cobbles often produced a disturbance of the water surface making them fairly easy to outline. Cobbles which were significantly submerged rarely disturbed the water surface; however, a careful observation of the colour, contrast and texture of the river bed through the water allowed clast outlines to be drawn. The submerged yet protruding stones were slightly more in focus with moderately more distinct outlines and a brighter colour than those stones locked into the basal armour of the creek bed. This outline can be thought of as representing roughly the plane defined by the *ac* or *ab* axes of the cobble. Some outlines may not be exact representations of the cobble sizes because of the variable orientation and uncertainties in the position and shape of the submerged clasts. In some case it is possible that submerged yet protruding clasts were not mapped because the water obscured them or that mapped clasts are not significantly proud of the bed and rightfully belong to the basal armour.

2.3 Comparison with Cluster Bedforms and Transverse Ribs

Our understanding of the form and process of gravel bedforms lags behind our understanding of these matters in sandy sediment. This is because of the logistical problems inherent in dealing with gravel. For example, since coarse clasts range in diameter from centimeters to meters and require high shear stresses to entrain them, flumes capable of handling these conditions are expensive to make while field observations are difficult

to accomplish through flood-force, turbid water. Nonetheless, a suite of gravel bedforms has been established by various researchers over the years, usually by observations of outwash plains [23, 15] or by examining the bed configuration of a coarse sediment stream after a flood has passed [22, 14].

Harris Creek's cobble patterns resemble cluster bedforms and transverse ribs. The cellular aspect to the pattern is similar to the step-pool geometry of steep mountain streams [14], though at a smaller scale and in a stream with a lower gradient.

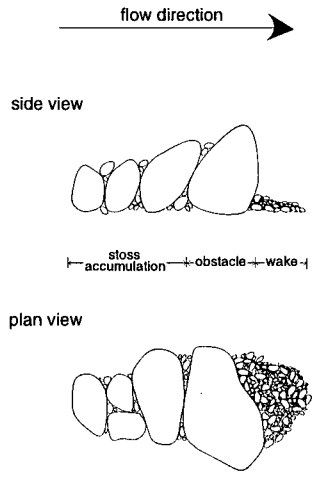


Figure 2.5: Diagram of a single cluster bedform showing its three major components. The height of the obstacle clast is on order of 20 to 30 cm.

Cluster bedforms (figure 2.5), sometimes referred to as pebble clusters [12] and imbricate clusters [14], are perhaps the most common and best understood gravel bedform [12, 7, 27, 5, 13]. They consist of upstream-trending accumulations of coarse particles immediately upstream of a coarser particle. The cluster is typically several clast diameters long and one or two clast diameters wide. A single cluster bedform can be divided into three parts: the obstacle, the stoss accumulation and the wake accumulation [5]. The obstacle is usually the central and largest clast of the cluster behind which pile up clasts of comparable but usually smaller diameter. These piled-up clasts are often imbricated and are termed the stoss accumulation. Sand and gravel clasts of significantly smaller diameter than the obstacle are trapped in the lee of the obstacle and comprise the wake accumulation.

Cluster bedforms are visible in the Harris Creek study reach. A good example can be found in the center of the bed at the downstream limit of Box 1 (figure 2.2) and less well developed examples elsewhere. A group of pebble clusters will define a pattern of points distributed in two dimensional space whereas Harris Creek's cobble patterns are more linear in two dimensional space. The linearity of the patterns and their preferred orientation sub-transverse to flow resembles transverse ribs.

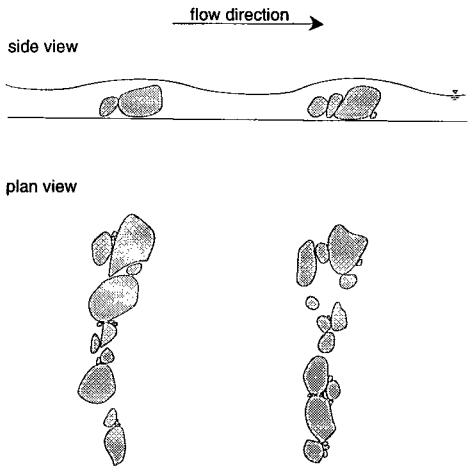


Figure 2.6: Sketch of transverse ribs. The side view includes a profile of an antidune wave illustrating that the increased water depth upstream of the antidune crests causes flow deceleration and the deposition of clasts which make up the ribs.

Transverse ribs (figure 2.6) are rows of gravel clasts typically 1 to 2 clasts thick, several clasts wide, oriented transverse to the mean flow direction and semi-regularly spaced along the stream at distances that vary proportionally with the diameter of the constituent clasts [15, 2]. Commonly ribs are roughly linear, though some are concave facing upstream [22], and the constituent clasts are imbricated. On gravel outwash plains, transverse ribs are observed to grade into a cellular arrangement of clasts usually 1 to 2 clasts thick with a diameter on order of 1 m called stone cells [15]. The stones constituting these cells are also imbricated.

In Harris Creek, an example of a transverse rib can be seen 4.5 m upstream of Box 1's downstream boundary (figure 2.2). The strong sub-transverse alignment of the cobble ridges resembles transverse ribs and the net-like, arcuate form traced by some the ridges resembles the description of stone cells [15]. Harris Creek's cobble patterns may be considered as the large scale structure defined by a continuum of gravel bedforms; an admixture of transverse ribs, stone cells, cluster bedforms and isolated stones. Moreover, cluster bedforms may be the articulation points of ridges.

2.3.1 A Hydraulic Calculation to Test for Transverse Ribs

Harris Creek's cobble patterns more strongly resemble transverse ribs than cluster bedforms, prompting a closer investigation of their similarity. It is a widely accepted hypothesis that transverse ribs form in almost critical to supercritical flow regimes. That they form in supercritical flows is supported by the fact that ribs form preferentially on gravel bars or riffle sections in gravel bed streams, braided streams and outwash plains [23, 15] where a shallower water depth causes high Froude number flows. Also, flume experiments in gravel

show that clasts tend to accumulate slightly upstream of a hydraulic jump [24]. That ribs form in almost critical flows is supported by observations that floods rarely became supercritical in regions where transverse ribs were known to form. Koster [19] suggests that the increase in flow depth at an antidune crest causes a decrease in flow velocity and subsequent deposition of coarse particles underneath the crest (figure 2.6). Rib formation during almost critical flows is also supported by observations of rooster tail and checkerboard antidunes, whose pattern of crests closely resembles the linear pattern of ribs and the more circular pattern of stone cells respectively, in regions where ribs and cells are known to form [15].

Assuming ribs require flows with a Froude number approaching or exceeding unity, if calculations show that Harris Creek has near critical flows then its cobble patterns could be transverse ribs. Alternatively, if the calculated Froude numbers are well below 1.0 this can be interpreted to mean that either the cobble patterns are distinct from transverse ribs or that transverse ribs have a formative mechanism that does not require critical flows. The latter interpretation would suggest that the widely held hypothesis for the origin of transverse ribs is in error.

The starting point for the calculation is the critical Shields' number, θ, which is related to the value of shear stress τ felt by a particle of diameter D at the moment of entrainment,

$$\theta = \frac{\tau}{(\rho_s - \rho)gD} \tag{3.1}$$

where $\rho_s = 2.7\,g/cm^3$ and $\rho = 1.0\,g/cm^3$ are the densities of sediment and water respectively, and $g = 9.8\,m/s^2$ is gravitational acceleration. This formula was developed for well-sorted natural gravels and for individual particles of varying sizes occupying open positions on a flume bed. Shields found that as the particle diameter increased, θ attained a constant value of 0.06. Later investigators found that θ can be as low as 0.03 for large, isolated stones and as high as 0.09 for shielded stones [10]. After obtaining a characteristic clast diameter for Harris Creek, Equation (3.1) can be used to determine the shear stress τ of the flow required to entrain that clast.

The value of shear stress resulting from assuming the Shields relation holds for a particular particle size can then be equated to the shear stress exerted by water of average depth d flowing uniformly down a bed of slope S,

$$\tau = \rho g d S \tag{3.2}$$

Given the slope S of the reach, equation (3.2) will give the depth d of the flow at the moment of entrainment. Using discharge-depth and discharge-velocity relations obtained from gauging station records near the study reach [30], the depth can be used to find the discharge of the flow which in turn is used to find the velocity v. The final step is to determine the Froude number,

$$Fr = \frac{v}{\sqrt{gd}} \tag{3.3}$$

where critical flows have $Fr = 1.0$.

The above calculation was performed for a bed slope S of 0.013 and D_{50} and D_{84} of bed material on the riffle surface of Harris Creek, approximately 75 mm and 110 mm respectively [9]. The resulting Froude numbers range from $Fr = 0.37$ for small stones in open positions, that is for D_{50} and $\theta = 0.03$, to $Fr = 0.69$ for large stones in shielded positions, for D_{84} and $\theta = 0.09$. This exercise indicates that even for large, locked clasts such as those comprising Harris Creek's cobble patterns, the average Froude number of the flow was well below 1.0 and therefore sub-critical, implying the patterns were formed during firmly sub-critical flows. This implication is supported by observations that the cobble patterns in Harris Creek were destroyed during peak flows and reformed after floods of lesser magnitude [10]. The calculation was also performed for the largest stones found in Harris Creek, D_{95} of 20 cm. The resulting Froude number ranges from 0.57 for stones in open positions to 0.94 for stones in shielded positions. This indicates the largest stones require sub-critical to critical flows to mobilize, especially if they are shielded by neighbors. As will be discussed later, the largest stones may not need to be entrained to form the cobble patterns since they appear to be scattered about the bed whereas the smaller stones are clustered together, defining the structure. It is important to keep in mind that because turbulent flows cause widely fluctuating shear stresses over the bed, this simple calculation may not embody the dynamics of Harris Creek's actual flows.

The ridges comprising Harris Creek's patterns have a sub-transverse arrangement resembling transverse ribs. That they are distinct from transverse ribs is indicated by the oblique orientation of the cobble ridges to the mean flow direction (figures 2.3 and 2.4) and by calculations showing that flows were sub-critical during the formation of the ridges.

2.4 Digitizing Harris Creek

The Harris Creek study reach was digitized to determine characteristic values for parameters used in the simulation and to provide data for the statistical algorithm, which would allow a more quantitative comparison between the real phenomenon and the simulation. Digitizing also allowed the stones in the reach to be displayed according to their size, revealing dissimilar distributions of large and small stones that are incorporated into the simulation.

On an enlargement of figure 2.1 each stone's largest diameter was measured by hand and the stone assigned to one of five size bins on the basis of its radius. The stone's largest diameter corresponds roughly to the a axis, a point that can be appreciated by considering that an ellipsoidal stone will most likely be deposited lying on its side, so to speak, in the stream bed and not positioned end up, balanced on its smallest cross-section.

The bins used are 0 - 6.25, 6.25 - 12.5, 12.5 - 18.75, 18.75 - 25.0 and 25.0 - 31.25 cm. Each stone is digitized as a circle of radius r equal to the maximum allowable radius of the bin the stone belongs to, i.e., a stone of radius 11 cm would fall into the second bin and be digitized as a circle of radius 12.5 cm. Each stone is written to a datafile as the x and y coordinates of the center point and radius r of the covering circle. The lateral boundaries of the reach were digitized as a sequence of points (x, y) defining line segments in space. The digitized reach is shown in figure 2.7 and separate maps showing the location of each stone in a size bin are given in figures 2.8 to 2.12. In practice, the digitizing software writes a graphic file which must be edited in order to get data files of (x, y, r) for each stone, one stone to a line, and of (x, y) for the points defining the lateral boundaries. Instead of using covering circles, the precise outline of each stone could have been digitized along with a point representing the center of each stone. However, the tedious task of outlining 2500 irregular shapes was passed over in favor of the simpler method.

The binned stones' radii (Table 2.2) are approximately lognormally distributed and correspond well with the size distribution measured on the riffle surface of a reach not far from the study section [9]. The stones in the two smallest size classes cluster together and exhibit the cobble patterns fairly well (compare figures 2.8 and 2.9 with figure 2.1 and 2.7). The stones in the three larger size classes do not delineate the cobble patterns, rather, they appear to be scattered throughout the reach (compare figures 2.10, 2.11 and 2.12 with figure 2.1 and 2.7). Separately displaying stones according to radius suggests that the larger stones are the keystones of the pattern while it is the accumulation of smaller stones that determines the pattern.

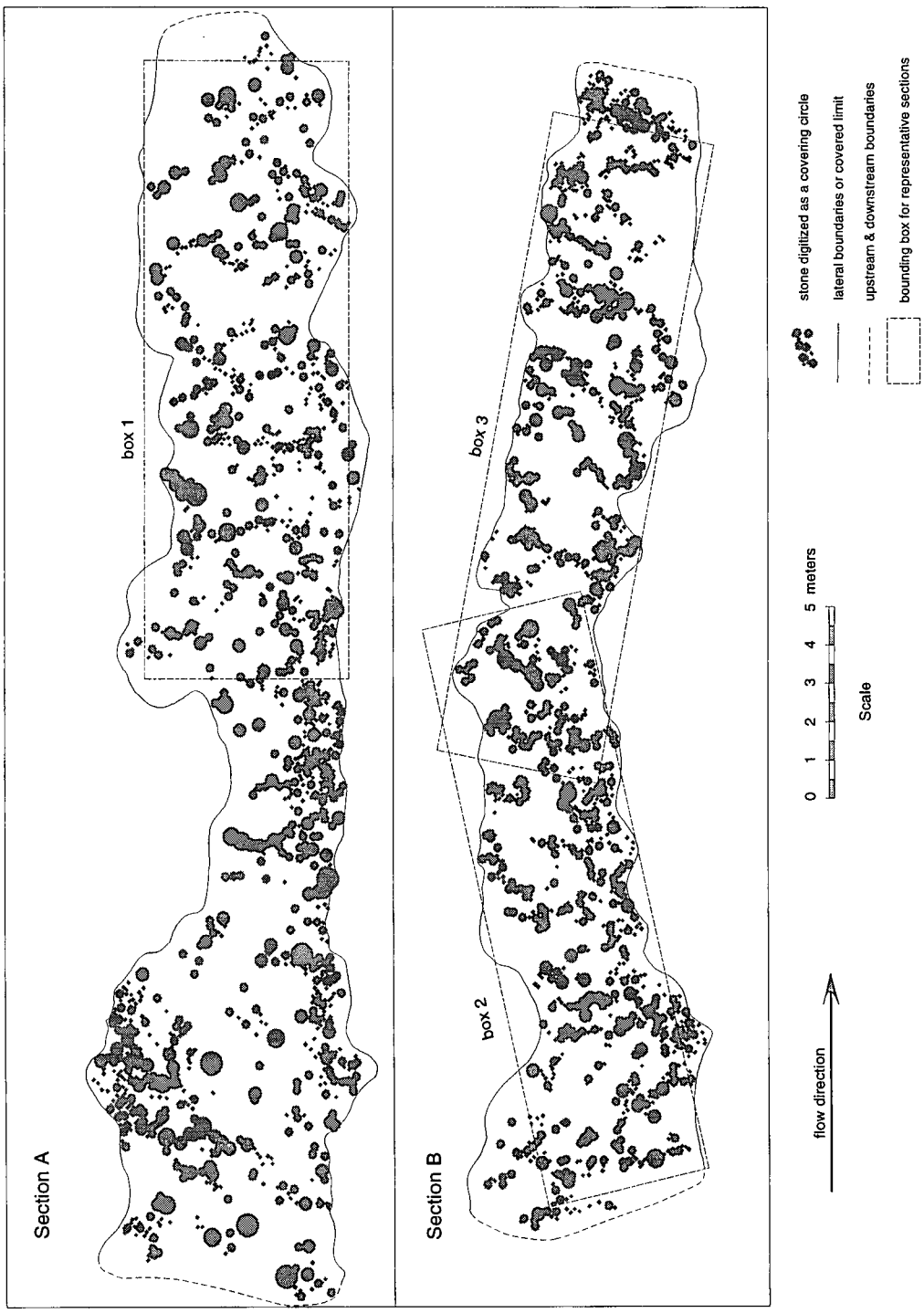


Figure 2.7: Stones digitized as covering circles of five different radii. This figure is drawn to the same scale as figure 2.1. Careful comparison of the two figures will reveal a few places where the covering circle of a large stone completely overlaps a smaller adjacent stone. In these instances, computed areal coverage will be reduced slightly but stone population counts will be accurate since the smaller stone, though invisible, still remains in the digital file.

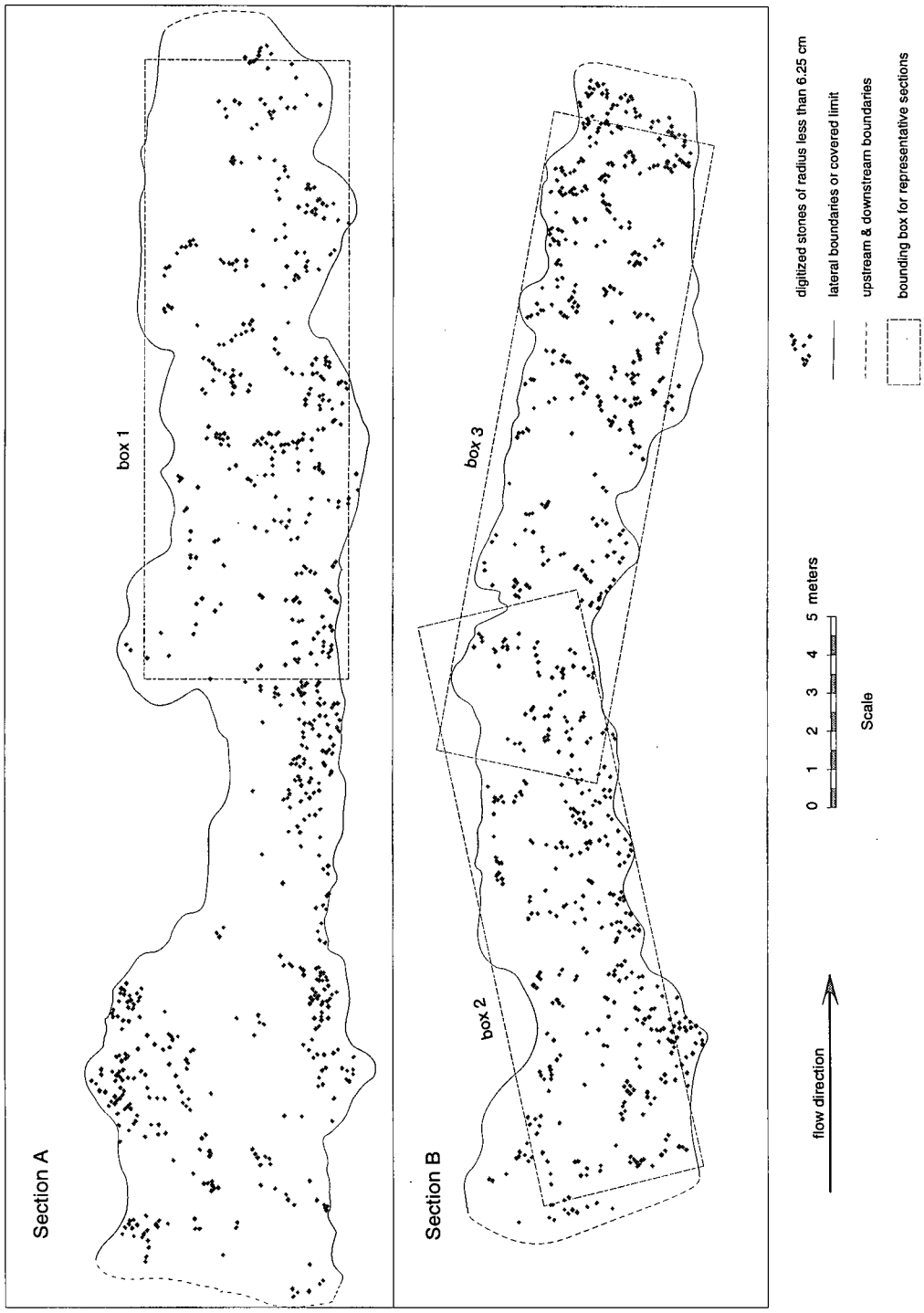


Figure 2.8: Stones of radius less than 6.25 cm. These stones tend to cluster together into ridges that trace the cobble pattern.

Figure 2.9: Stones of radius 6.25 cm to 12.5 cm. These stones tend to cluster together into ridges that in some places trace the cobble pattern.

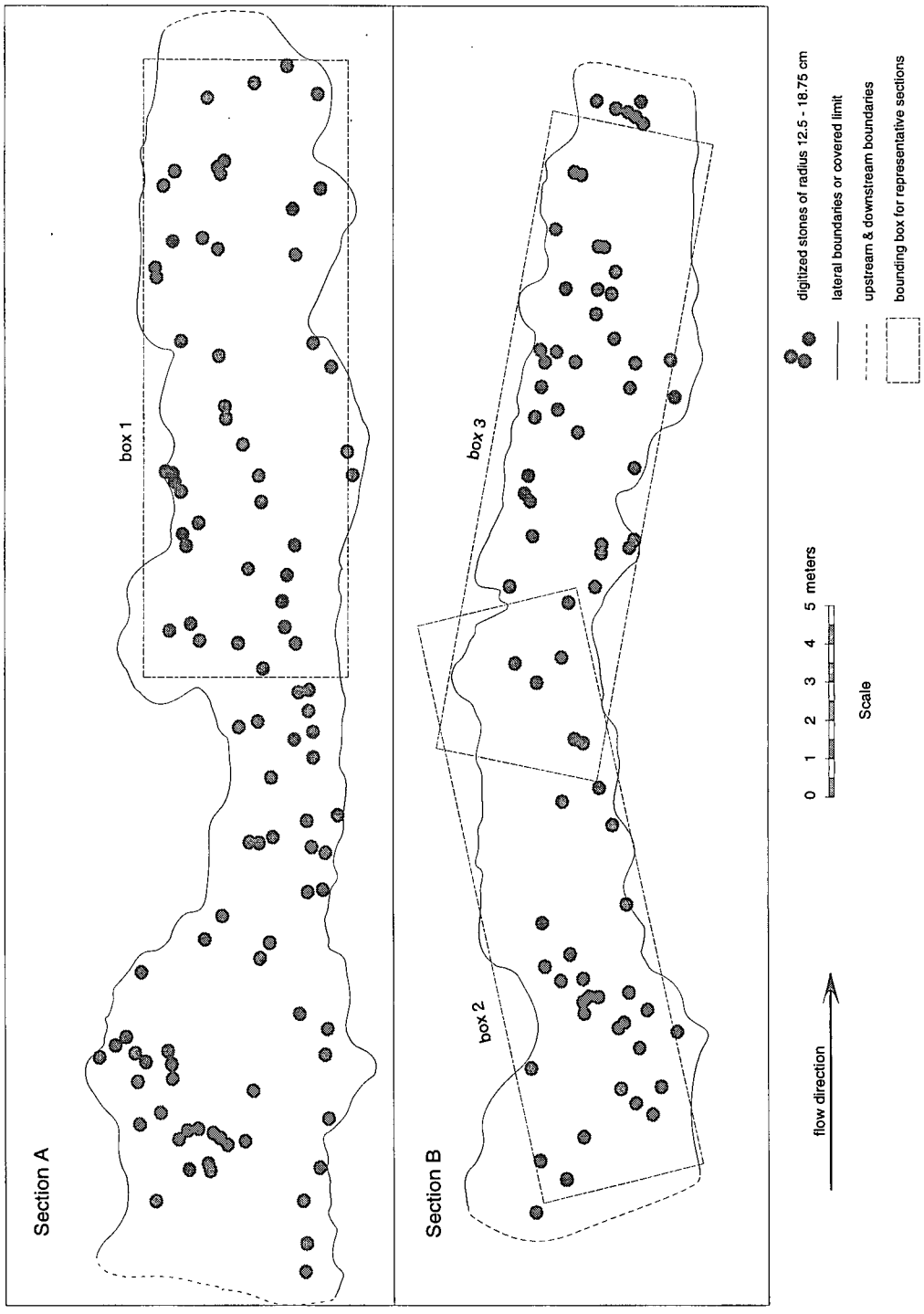


Figure 2.10: Stones of radius 12.5 cm to 18.75 cm. These stones are less clustered and more scattered throughout the reach than smaller stones.

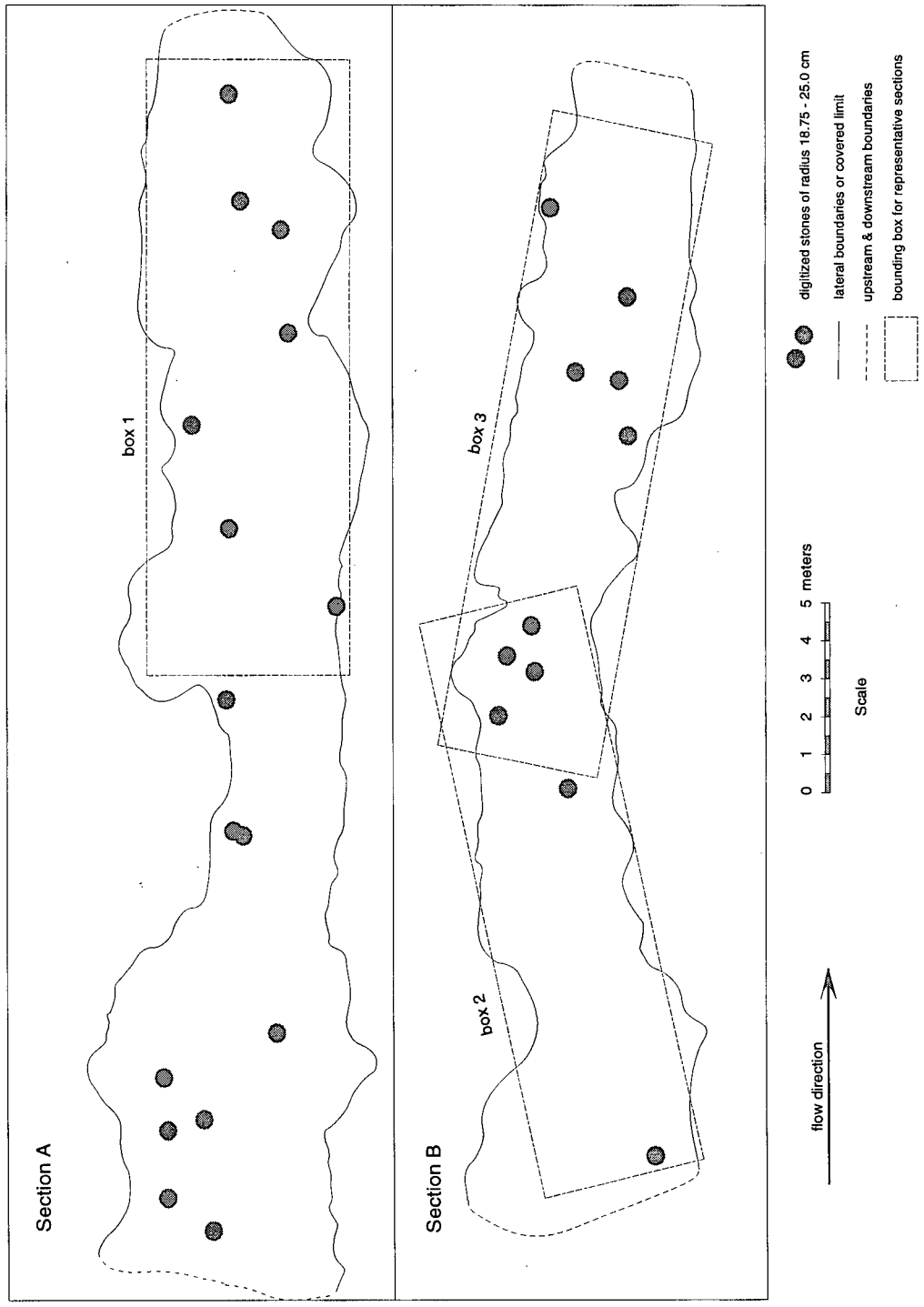


Figure 2.11: Stones of radius 18.75 to 25.0 cm. These stones appear randomly scattered throughout the reach and do not delineate the cobble patterns.

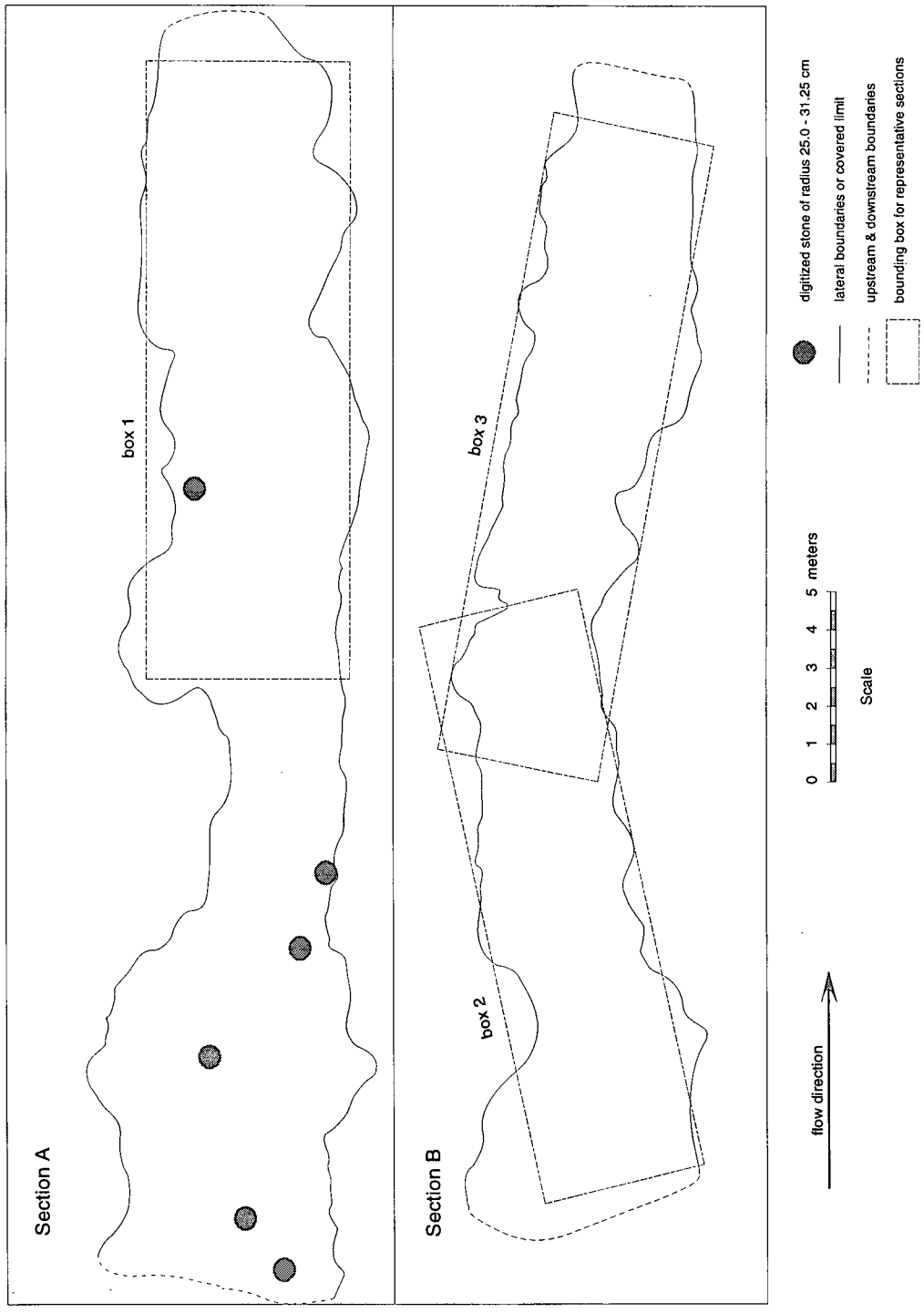


Figure 2.12: Stones of radius 25.0 to 31.25 cm.

bin size (cm)	Section A frequency	Section B frequency
0 - 6.25	663	713
6.25 - 12.5	438	461
12.5 - 18.75	100	74
18.75 - 25.0	16	11
25.0 - 31.25	6	0
section total	1223	1259
total clasts	2482	

Table 2.2: Size distributions of identified stones in the digitized Harris Creek reach.

The projected cross sectional area of stones in each size bin was computed from the digitizing information and is tabulated in Table 2.3. Note that these values are maxima since covering circles were used in the calculations. Taken together, the stones cover 23% to 30% of the bed. Experiments in sand showed that particles covering about 6% to 12% of the bed maximized the flow resistance [34]. A densely covered bed facilitates stability since the flow steps above the wake interference of the rough bed, no longer exerting force close to the bed to move stones [25]. The form of the bed, then, can be considered a stable configuration since the flow is displaced elsewhere. Harris Creek's dense coverage suggests the cobble patterns are an expression of the most stable configuration of the bed to shear flow, a situation supported by observations that the patterns are destroyed when the stresses in Harris Creek exceed about three times the threshold stress required to move the larger half of clasts [10]. Large clasts positioned as keystones of the patterns, the net-like or cellular arrangement of ridges, and the high population of stones contributing efficient energy dissipation [25], all attest to the stability of the cobble patterns.

Box 1 dimensions: 16.0 m by 5.6 m				
radius (cm)	frequency	absolute area (m^2)	relative area (%)	abs. covered area (%)
¡ 6.25	301	3.69	18.12	4.12
6.25 - 12.5	204	10.01	49.17	11.17
12.5 - 18.75	45	4.97	24.41	5.55
18.75 - 25.0	7	1.37	6.73	1.53
25.0 - 31.25	1	0.31	1.50	0.35
total	558	20.36		22.72
Box 2 dimensions: 15.6 m by 4.5 m				
radius (cm)	frequency	absolute area (m^2)	relative area (%)	abs. covered area (%)
0 - 6.25	329	4.04	18.85	5.75
6.25 - 12.5	256	12.57	58.66	17.89
12.5 - 18.75	33	3.64	16.99	5.18
18.75 - 25.0	6	1.18	5.51	1.68
25.0 - 31.25	0	0	0	0
total	624	21.43		30.50
Box 3 dimensions: 16.5 m by 4.5 m				
radius (cm)	frequency	absolute area (m^2)	relative area (%)	abs. covered area (%)
0 - 6.25	353	4.33	19.74	5.83
6.25 - 12.5	235	11.54	52.60	15.54
12.5 - 18.75	39	4.31	19.64	5.80
18.75 - 25.0	9	1.77	8.07	2.38
25.0 - 31.25	0	0	0	0
total	636	21.94		29.55

Table 2.3: Size distributions and area covered by stones in the digitized Harris Creek reach. Refer to figure 2.7 for location of boxes.

2.5 A Statistical Algorithm to Identify Ridge Structures

In order to compare the real phenomena as exemplified in Harris Creek with the computer simulation results, a statistical algorithm was developed that identifies ridge structures and their orientations. A close examination of figure 2.2 reveals a pattern of cobble ridges which occur in and are structurally associated with other structures such as pebble clusters, transverse ribs and isolated stones. What exactly is a cobble ridge and how can it be measured? A cobble ridge, the basic constituent of the cobble patterns, is defined here as a linear or arcuate alignment of 3 or more stones. The interclast spacing, the (non)linearity of the ridge, and the sizes of the component stones are several factors that complicate the definition of a cobble ridge pattern. Other complicating factors include the proximity of other patterns (clusters, river bed boundaries, isolated stones) and the angle of the ridge with respect to the flow.

A statistical algorithm was developed that would register linear accumulations of stones trending at any angle to the mean flow direction by measuring the percent of covered area in strips of different orientation (see Appendix A for a complete description of the statistical test). The rectangular riverbed is rotated from $-45°$ to $45°$ about its lower left corner and at each 1 degree interval the fraction of the bed covered by stones is calculated for vertical and horizontal strips and displayed as a histogram projected onto the reference x and y axes. During the rotation of the river bed, an obliquely trending ridge will show up on the histogram as a peak when it is rotated into parallelism with the strips along x or y. At all other angles it will contribute to a more uniform distribution. The areal percentage of the strip covered by stones at each angle is displayed as one pixel in a density plot of angle vs bin where a white pixel indicates zero covered area and a black pixel indicates a high percentage of covered area. The information in the density plot is condensed as a pseudo χ^2 (PC^2) graph, so called because it uses percent covered area in the calculation instead of the absolute values needed for a true χ^2 graph. Peaks in the PC^2 graph indicate the angle at which the distribution of stones is least uniform. In the Harris Creek maps and in simulation results, the downstream direction is defined as $0°$ and clockwise angles are positive, however, in this statistical algorithm clockwise angles are negative. This difference in convention necessitates that a peak occurring at some angle in the statistical graphs must be converted to find the corresponding angle of the ridge in the study reach.

In the density plot, if there are alternating light and dark regions along an angle, it is likely there are alternating low and high concentrations of stones at corresponding angles in the river bed. A candidate ridge structure is identified as a light area (low concentration of stones in the strip) changing to a dark area (high concentration of stones in the strip) and then back to a light area along a horizontal line. This records the rotation of a ridge at approximately that bin location into and out of parallelism with the bin strip.

The density plot along x and the PC^2 test for the cobble structures in Box 3 are shown in figure 2.13. The density plot is well banded with dark and light strips. The PC^2 graph has peaks at $15°$ and $-7°$ corresponding to ribs oriented at $-75°$ and $83°$ to the flow direction. The other two boxes were also

analyzed with this statistical method and the results are shown in figures 2.14 and 2.15.

Only the density plot and PC^2 graphs along x are shown because the widths of the strips spanning the x axis change much less than they do along y axis therefore contributing less error to the results. Also, the test was written so that ridges with a strong transverse component will register in the density plot along x.

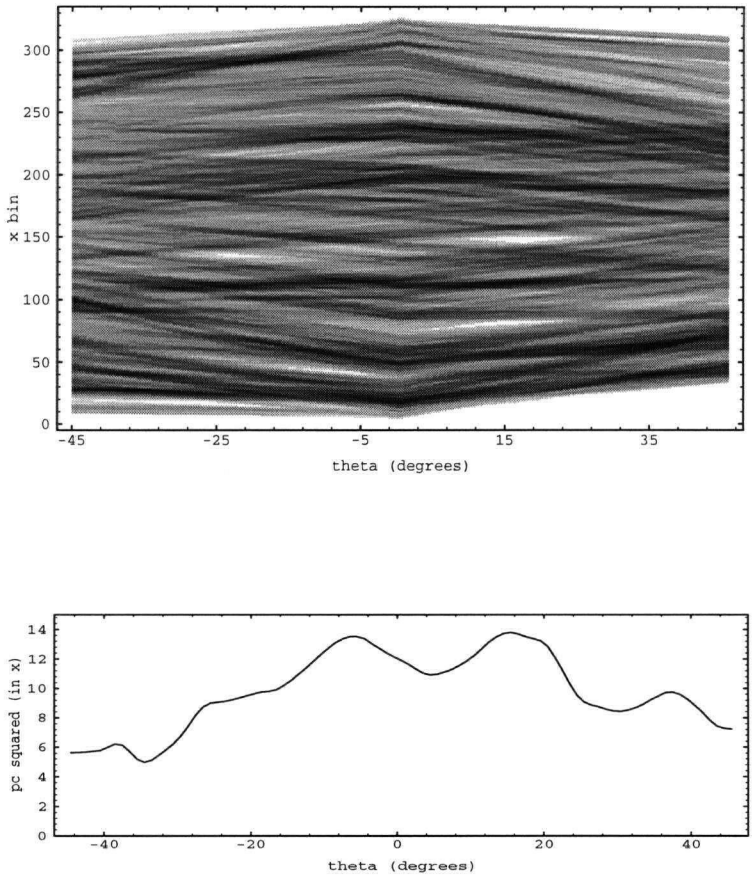
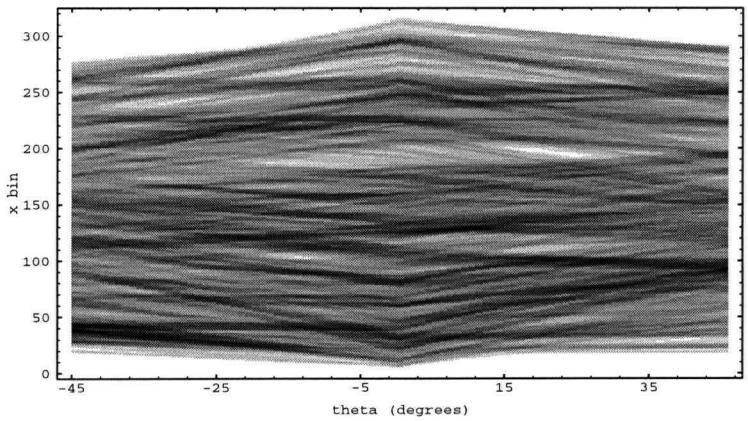


Figure 2.13: Density plot (top) and PC^2 graph (bottom) for Box 3 in Harris Creek along x. For the density plot, white pixels indicate regions with zero percentage of stones and black pixels indicate regions of high percentage of stones. The peaks in the PC^2 graph correspond to angles at which the distribution of stones is least uniform. In the PC^2 graph maxima appear at angles $+15°$ and $-7°$ and can be seen to be coincident with the angles at which there exist regions of alternating light and dark pixels in the density plot, that is, along vertical lines in the top graph. These angles correspond to ridges aligned at $-75°$ and $83°$ in the Harris Creek map.

The rose diagrams (figure 2.3) show that there is a predominant alignment approximately 60 to 77° either side of the flow direction. The PC^2 graphs for the three boxes show comparable maxima corresponding to approximately 72 to 86°. This indicates that the statistical test is detecting the predominant orientations of the ridges.



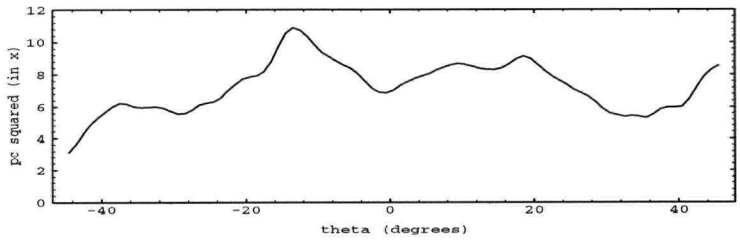


Figure 2.14: Density plot (top) and PC^2 graph (bottom) for Box 1 in Harris Creek along x. For the density plot, white pixels indicate regions with zero percentage of stones and black pixels indicate regions of high percentage of stones. The peaks in the PC^2 graph at $-13°$ and $19°$ correspond to ridge structures oriented at angles of $77°$ and $-71°$ in the Harris Creek reach.

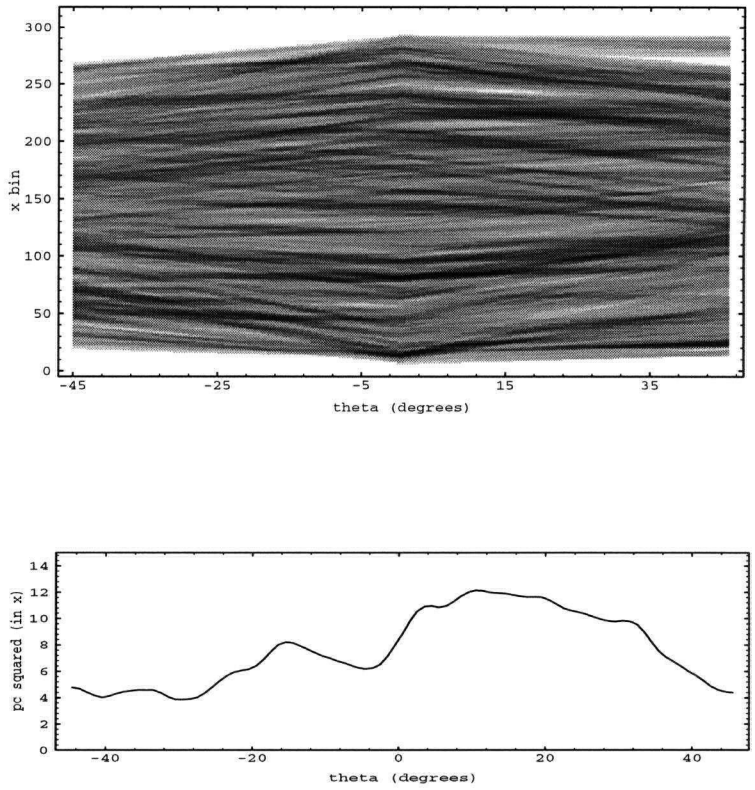


Figure 2.15: Density plot (top) and PC^2 graph (bottom) for Box 2 in Harris Creek along x. For the density plot, white pixels indicate regions with zero percentage of stones and black pixels indicate regions of high percentage of stones. The peaks in the PC^2 graph and the angles at which there are regions of alternating light and dark pixels in the density plot occur at $-15°$, $10°$ and $16°$ corresponding to ridge structures oriented at angles of $75°$, $-80°$ and $-74°$ in the Harris Creek reach.

Chapter 3

Computer Simulations of Cobble Patterns

The rise in popularity of computer simulation models in the physical sciences over the past ten years or so is due to the sophistication of computer modeling techniques, in particular the development of cellular automata, lattice models and new numerical techniques. Earth scientists have long noted the regular and distinctive patterns that can develop in earth materials yet only now do they have the computational tools with which to study such phenomena [16]. Several investigators have been successful in reproducing geomorphic patterns such as eolian ripples [3, 20, 31], beach cusps [32], non-periglacial sorted nets [1] and stone stripes [33] using computer simulation models. These simulations all share in common the desire to reproduce a naturally occurring pattern developed in noncohesive sediments of various sizes and to retain the discrete character of the sand or pebbles comprising the pattern. In contrast to using differential equations dealing with continuous variables [20], the discrete particles are programmed to move or behave according to the investigators' understanding of the local physics of the system and usually involve randomness in the algorithm. Aside from being currently in vogue, computer simulation experiments allow certain phenomena to be studied without the logistical problems accompanying physical experimentation. In this case, cobble transport is simulated without the problems of clast retrieval and equipment damage that so hamper empirical investigation.

It is essential to realize that a computer simulation, by virtue of its discrete nature, never completely reproduces real conditions. A successful computer simulation of a river, like the one presented here, only validates the hydrologic and sedimentologic scenario programmed into it. It is also crucial to remember that results of the `Riverbed` model are a function of the physical scenario programmed into the code, in the form of parameter values and rules, as well as the problem solving architecture, in the form of loops and logical

criteria. Both factors must be kept in mind as possibly contributing to simulation results.

Before describing the computer model in detail, reviewing the current knowledge about coarse sediment transport will sharpen our intuitive understanding of the physics of the system, outline the hydrologic and sedimentologic scenario used in the simulation, and uncover several principles that will be incorporated into the design of the model.

3.1 Facts about coarse sediment transport

In a gravel bed river the size of the stones is commensurate with the depth of water flow resulting in very rough and irregular flow boundary conditions. The stones are often observed to be imbricated against one another and packed such that some are shielded from the shear effects of the flow by virtue of being hidden by neighbors [21]. In contrast to a stone lying in an open position on the bed, shielded stones of equal size generally require higher shear stresses to be entrained, typically because they can't move until the obstacle, which is usually larger, moves. Traditionally, the shear stress acting parallel to the bed has been characterized as a function of the water velocity and grain size of the particle, an approach which works reasonably well for sand-sized sediment. However, on a packed gravel bed the shear stress acting on each stone must also be a function of the sheltering and stability afforded by surrounding stones (as alluded to in Section 2.3.1). All this is to say that the mobility of coarse clasts is a complex function of flow conditions, particle size and bed structure [21]. The difficulty in predicting the onset of movement lends itself to a stochastic modeling approach and says that the movement of stones in the model should be partially dependent on whether the stone is in an open or shielded position.

Given that a cobble has been successfully entrained into the flow, how does it travel along the bed? Because they are so large, cobbles in transport will roll and slide along the bottom of the bed. Consequently they are transported along a bumpy and irregular bed surface and encounter obstacles in their path resulting in an irregular transport trajectory. Coarse clasts are known to engage in sporadic and discontinuous motion, a situation which again supports the appropriateness of a stochastic modeling approach and does not contradict the irregular trajectories that the simulated stones travel.

A stone will tend to stop if it hits an obstacle from behind, causing an elastic rebound in the upstream direction which is quickly damped by the flow downstream. If a stone hits an obstacle on the side then the combined effects of a rebound trajectory not diametrically opposed to the flow and the force of the water pushing the stones downstream will tend to keep the stone in motion. Large stones travelling downstream, by virtue of their momentum, may tend to roll over small protruberances in the stream bed and only be stopped by impinging against a stone of comparable radius. Small stones travelling downstream may tend to stop behind all obstacles or be drawn into the obstacle's lee by virtue of being small enough to be affected

by the reduced lift and drag forces caused by the obstacle's perturbation of the flow [7]. In the simulation, the deposition of a stone can depend on the size of the obstacle and the angle at which it hits the obstacle.

3.2 `Riverbed` Computer Simulation Model

The prime motivation in writing the `Riverbed` model is to reproduce Harris Creek's collection of obliquely and variably trending cobble ridges. The adopted rules reflect this goal. `Riverbed` is written in C++ and version `7.0` can be found in Appendix C. The code is heavily documented, especially at the beginning of functions and major loops. In the following description of the algorithm, PARAMETERS which can be changed from one experiment to the next are indicated by small capitals and the various *rules* implemented are indicated by italics. Different versions of the code were written to incorporate different rules and parameters. For instance, `Riverbed 5.0` uses the rebound rule with clasts of equal radius while `Riverbed 6.2` uses rebound and obstacle rules with clasts of different radii. For ease of exposition, only one result from any particular set of rules and parameters is shown and is often referred to by the version of the code that produced it. In actuality, several experiments were performed for each of many combinations of rules and parameters. The result shown is representative of the results of these experiments.

In the `Riverbed` computer model, individual cobbles in the reach are successively picked at random, or with some nonuniform probability, as in the entrainment probability rule, and entrained along a randomly chosen trajectory. Upon impacting an obstacle, the cobble stops or continues depending on the cobble-obstacle geometry and the rules governing deposition. If a stone hits an obstacle right from behind then it will tend to stop but if it hits at an angle, the stone could bounce off and continue downstream in a different direction. After rebounding off the obstacle the cobble continues on its new path to further encounters or loses its momentum and becomes deposited, as in the travel distance rule, otherwise it remains at the position of impact.

The *rebound rule* is modeled after elastic collisions (figure 3.1). Given the stone's x, y coordinates, its trajectory and the coordinates of the obstacle, the rebound angle is calculated to be equal to the impact angle as measured about the normal to the tangent of the impact site (see figure 3.2 for reference diagram). If the rebound trajectory is within 90° either side of the mean flow direction, the cobble will continue down the reach on this new path, in other words it bounces off the obstacle. If the rebound trajectory is greater than $\pm 90°$ from the flow, the cobble stops in the position at impact. The rationale behind depositing those cobbles with a rebound trajectory greater than $\pm 90°$ is that otherwise the cobble would travel upstream. In this way the rebound rule models the general effects of the water flow, although hydrodynamics are not explicitly treated. The rebound rule may be slightly modified from one experiment to the next. The simplest way to change it is to change the angular range permitting rebounding by specifying the parameter FULCRUM

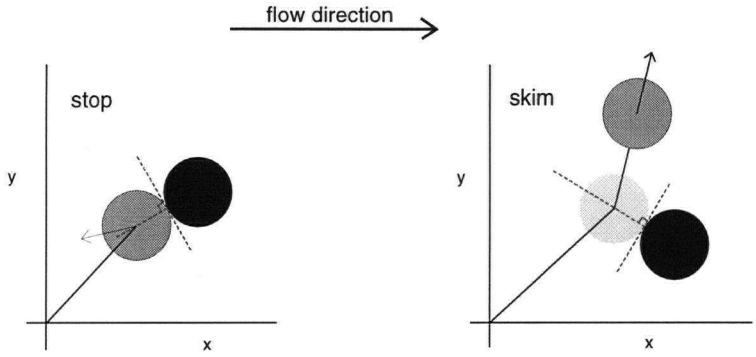


Figure 3.1: Schematic of the rebound rule showing that an incoming stone whose rebound angle is determined to be greater than 90 degrees from the flow direction (left diagram) will stop whereas a stone whose rebound angle is computed to be less than 90 degrees from the flow direction (right diagram) will be deflected by the obstacle and continue on a new trajectory.

to be other than 90. The rebound rule is used in all versions of `Riverbed`.

The *neighbor rule* is intended to model the shielding effects of neighboring stones: after a stone has been chosen to be randomly entrained, the neighbor rule counts the number of stones touching or within some small distance, typically 1 to 3 cm, of the chosen stone. If the number of neighbors, regardless of their spatial distribution, is greater than or equal to the critical value NN then this stone is not entrained and another candidate is chosen. If the neighbors number less than NN, the stone is entrained. The neighbor rule is used in `Riverbed 5.1`.

The *obstacle rule* models the effects of the relative size of impinging and obstacle clasts: the obstacle a stone is deposited against must have a radius greater than or equal to half the stone's radius, otherwise the stone rolls over it. The rationale for this rule is that a large stone in transport will tend to be stopped by obstacles of comparable or larger radius while small stones will be stopped behind anything they run into. The obstacle rule is used in `Riverbed 6.2`. When locating an appropriate obstacle, this rule takes precedence over the rebound rule.

An *entrainment probability rule* models the fact that larger stones are less likely to be entrained into the flow than are smaller stones at any given shear stress. For any entrained stone, the entrainment probability is calculated as the stone's radius multiplied by a uniform random number. If the result is less than some CRITERION the stone is entrained, otherwise it is not entrained. The entrainment probability is programmed in `Riverbed 7.0`.

The *travel distance rule* embodies the fact that a travelling stone will slow down and stop due to its inertia and friction with the bed and water. This rule is modelled after the results of experiments on the travel distance of stones in open positions in gravel bed rivers [8] where a stone's travel distance was found

to vary in inverse proportion to its size as

$$L = 1.77\,(1 - \log D)^{1.35} \tag{2.1}$$

where $L = L_i/L_{D_{50surface}}$, $D = D_i/D_{50subsurface}$ and L_i and D_i are respectively the travel distance and diameter of the stone in question. With this rule, each stone is assigned a maximum distance of travel derived from Equation 2.1. Once it is entrained, the stone travels along its trajectory until it stops behind an obstacle (according to the rebound rule) or until its cumulative travel distance has exceeded the maximum travel distance originally assigned to it. If the stone bounces off an obstacle, it continues on its new trajectory until the accumulated travel distance exceeds the maximum travel distance. This rule is implemented in `Riverbed 7.0`.

The entire simulation is essentially scale free except that a representative clast size must be declared in order to supply travel distances and stones sizes. To adopt this computer model to a different prototype with a different representative stone size, the radius of the stone must be known as well as the length and width of the simulated reach.

The computer simulation is written such that, all parameters and rules being equal, using the same integer to seed the random number generator will produce identical results. In this way the computer model is deterministic, determinism being used in the same sense as for deterministic chaotic systems, that is, although indivdual trajectories in a chaotic system may be impossible to predict, given exactly the same initial conditions, that trajectory will be exactly reproduced. The computer model is also stochastic in the sense that any outcome depends on the initial random number used. In fact, the model can be more stochastic if more rules and criterion incorporated random numbers. Therefore, a simulation result depends on the design of the computer program as well as on probability. Given that the simulation is fundamentally stochastic, it cannot be expected to reproduce the Harris Creek patterns exactly. At best, it can be expected to yield realistic and similar looking results for a high percentage of trials using the same rules and parameters with different random numbers.

3.2.1 Initial and Boundary Conditions

To initiate a simulation, the reach is seeded with randomly placed stones. The number of seed stones is determined by calculating how many non-overlapping circles are required to cover the fraction SEEDFRACTION of the reach area. The parameter SEEDFRACTION is typically 0.20 to 0.30 in accord with the fractional covered area computed for the digitized boxes in the Harris Creek reach. In `Riverbed 5.0` a radius of 10.0 cm is used to calculate the number of seeding stones. In `Riverbed 6.0` and later the average value of 10 cm is used even though the stones have different sizes. The x and y coordinates of the seed stones are determined by drawing uniform random numbers with zero mean and unit variance and scaling this number by the LENGTH

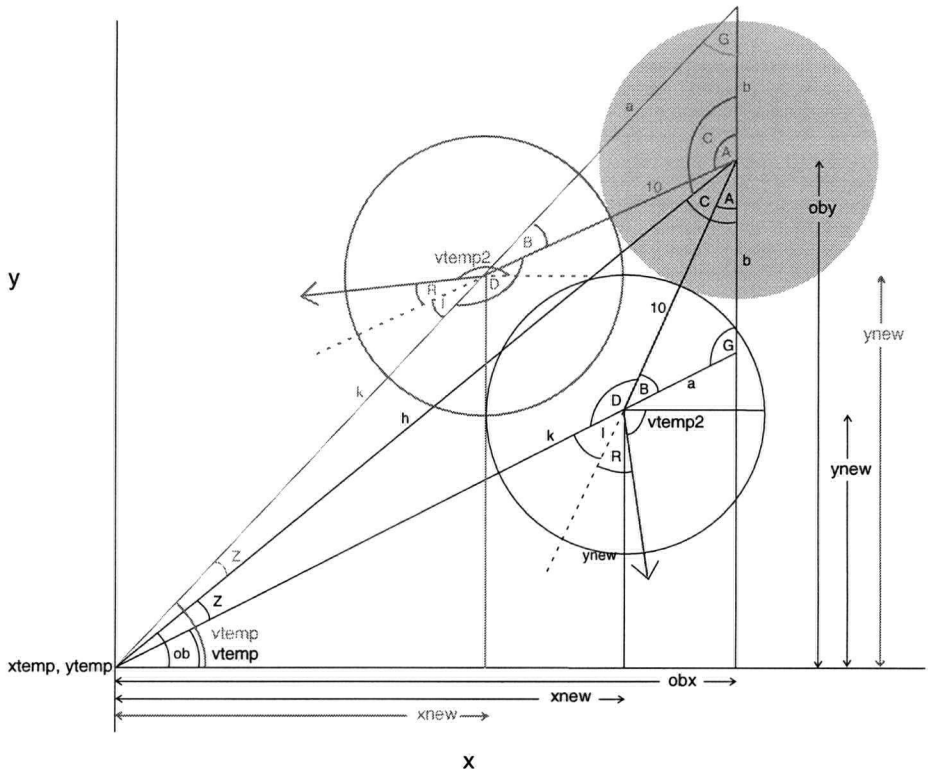


Figure 3.2: Diagram showing how **Riverbed** calculates the impact position $xnew, ynew$ and the rebound angle $vtemp2$ using knowledge of the stone's starting coordinates and trajectory, $xtemp, ytemp$ and $vtemp$ respectively, and the obstacles coordinates obx and oby. The equations used to calculate the impact position and rebound angle can be found in part 4 of the entrain() function in all versions of the **Riverbed** code. The grey filled stone is the obstacle, the black unfilled stone skims this obstacle and moves on as calculated by the black angles and vectors, and the grey unfilled stone stops behind the obstacle, as calculated with the grey angles and vectors. All angles and vectors have the same names as in the **Riverbed** code. Flow direction is along the positive x axis. The line segments labelled 10 are center-to-center lines between the obstacle and the impinging stone, both of radius 5.0. If different sized stones are used then this length must be adjusted accordingly.

and WIDTH of the reach respectively. Seed stones do not touch one another. Figure 3.3 shows seeded reaches with fractions of 0.10, 0.20 and 0.25 of the reach area covered by randomly placed stones. They are the initial conditions for an experiment.

The upstream and downstream boundaries of the simulated reach are open: once a stone manages to travel through the downstream boundary, a new stone is introduced on the upstream boundary. This setup maintains a constant population of stones in the reach at all times. The left and right boundaries of the reach are rigid: any stone that intersects them is rebounded off with a new trajectory that is equal to the incoming trajectory reflected about the normal to the boundary.

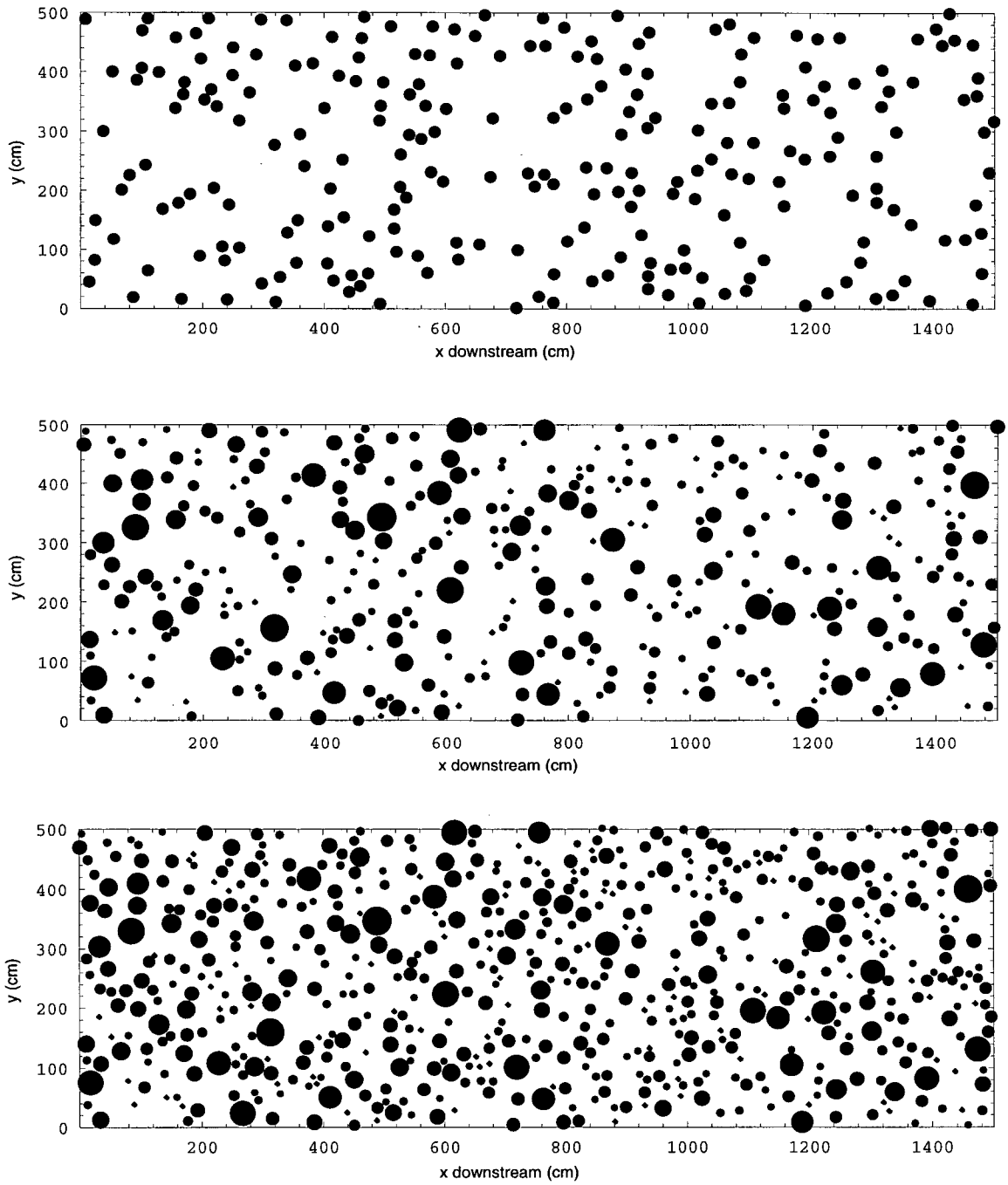


Figure 3.3: The initial seeded beds for experiments with 10% (top), 20% (center), and 25% (bottom) coverage. The top bed has stones of equal radius and is the initial condition for experiments using `Riverbeds 5.0` and `5.1`. The center bed is the initial condition for experiments using `Riverbeds 6.1` and `6.2`. The bottom bed is the initial condition for an experiment using `Riverbed 7.0`.

3.2.2 Experimental Setup

After the initial bed has been generated with seed stones and the x and y coordinates and radius of each stone have been written to a file, one of these stones is picked at random to be entrained into the flow. This stone is given a trajectory by drawing a number from a gaussian distribution, with zero mean and unit variance, and scaling by RANGE, allowing most stones to travel within RANGE degrees of the mean flow direction. The chosen stone then travels along this trajectory until it encounters an obstacle in its path at which point the rebound rule determines whether the stone glances off the obstacle and continues on a new path or whether it is deposited. If it is deposited then another stone in the reach is picked at random to be entrained into the flow.

Since `Riverbed` is a time-independent model, the length of the simulation is set by the number of random entrainements, RANDOMCLASTS, specified for each experiment. Typically 2000 to 3000 random entrainments are specified per simulation and once these have been done the positions of all the stones and their radii are written to a file. An experiment of length RANDOMCLASTS = 2000 using 200 stones corresponds to each stone in the river bed having been chosen to be entrained on average 10 times. Experiments using up to 6000 random entrainments do not reveal a pattern of stones significantly differing from the results of shorter duration. The initial and final files are read into a plotting program, in this case *Mathematica*, where a graph is made of the initial and final configurations of the simulated reach.

To maintain a steady state population, if a stone exits the downstream margin of the reach, a new stone is introduced on the upstream margin. The new stone's position is determined by drawing a uniform random number scaled by the WIDTH, and the trajectory is randomly chosen by drawing from a gaussian distribution and scaling by RANGE. All the simulations follow the same basic outline though different parameters or rules can be used.

All experiments were performed on a Macintosh Quadra 605 personal computer, using Symantec's C compiler. Simulation run times vary from approximately one and a half hours for 300 stones and 2000 random entrainments to three hours for 500 stones and 3500 random entrainments. Omitting the commands for generous screen output in the code will reduce run times significantly. The Mathematica script takes approximately two hours to complete on the Macintosh computer. When run on Sparc 20 computers, the script could take as little as five minutes providing a minimum of 35% of one processor was available for the computing task.

3.2.3 Model Calibration and Testing

The WIDTH of the simulated reach is set at 5 m, comparable to the average width of the Harris Creek study reach. The LENGTH is set at 15 m which is shorter than the length of the study reach but long enough to

provide a buffer zone for upstream and downstream boundary effects. To ameliorate the problem of the upstream boundary clogging up with stones, the program is modified such that if a newly introduced stone will impinge on a stone which itself lies very close to the upstream boundary, the new stone is discarded and another one chosen. Since stones near the downstream boundary have relatively fewer potential obstacles in their path, they tend to preferentially exit the reach. The result is that during the course of a simulation the downstream portion of the reach tends to empty out its stones.

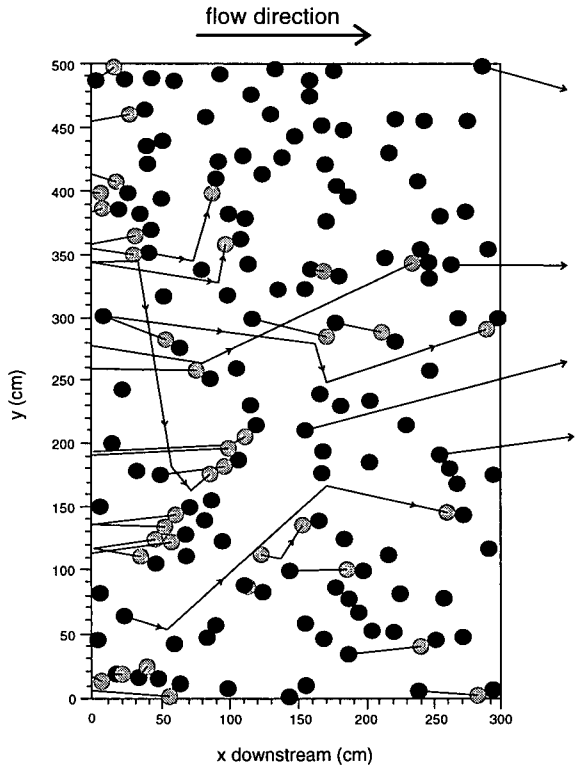


Figure 3.4: Diagram showing the river bed resulting from 20 randomly entrained stones and 20 newly injected stones in a short `Riverbed` test reach using the rebound rule on stones of equal size. Black circles represent the positions of stones at the initial conditions of the experiment. Entrained stones travel along vectors until they are deposited as a grey stone. Newly injected stones are transported along vectors originating on the upstream boundary (at $x = 0$) and are deposited in positions denoted by grey circles. Stones which travel out of the reach have vectors extending past the downstream boundary and are counted as throughput. Note that the upstream margin of the reach, at $y =$ 350 - 400, begins to clog up with stones. Even after only 40 entrainments the stones begin to cluster.

In experiments using stones of equal radius (as in `Riverbed 5.0`), the radius is chosen to be 10 cm corresponding to the D_{95} (20 cm) of the riffle surface in Harris Creek. In experiments using stones of different radii (as in `Riverbed 6.1` and later versions), the radii are drawn from a lognormal distribution, truncated at a minimum size of $D_{75} = 10$ cm, a maximum size of $D_{100} = 50$ cm and a median diameter of $D_{95} = 20$ cm, producing a population with a similar size distribution to Harris Creek. A census is taken of stone size

in the initial bed and the final bed and of the stones which exit the reach. Comparison of these three size distributions determines if stones of certain radii are preferentially transported through the reach.

Since a computer model rapidly grows into a complicated structure capable of unexpected behavior without rigorous attempts to test each subcomponent, `Riverbed` was tested at every step to ensure that the code performed as intended. For instance, all the random number generators were checked by graphing their distributions and calculating their mean and spread. The code writes to a file the values of key variables at intermediate stages of the simulation, providing all the information necessary to follow by hand the newly entrained stone, draw its transport vector, determine the appropriate obstacle, the impact and rebound angle of this configuration, and the subsequent movement or deposition of the stone. Such an exercise was performed several times during the course of programming producing diagrams similar to figure 3.4. The information in the file of key values was used to manually draw, on the initial bed, the vector for each entrained stone, following it along checking rebound angles and determining the appropriate obstacle and final position of the stone. The marked-up initial bed is then compared to the final bed to ensure a complete match.

3.2.4 Programming Details

`Riverbed` is based on linked lists where each list member represents one stone and contains memory space for the x and y coordinates of the stone in cm, the radius in cm, the trajectory when moving, where clockwise is positive, counterclockwise is negative and downstream is zero, the maximum travel distance for a stone of that size in cm, and other characteristics as needed. Linked lists are a dynamic data structure meaning that the list is only as long as the number of stones in the reach and shrinks or grows if stones are lost or added; it does not need its size declared beforehand as is necessary if using arrays. Linked lists have other advantages over modeling approaches using arrays, cellular automata or lattice models. For example, linked lists use real variables in $x - y$ space to position the stones which avoids the effects of rectangular or hexagonal grids and grid spacing.

One peculiarity of linked lists is that to determine the obstacle for any entrained cobble, the cobble-potential obstacle vector must be calculated for each member of the list. In other words, rather than following the transporting cobble along its path and seeing which stone it eventually collides with, linked lists require that the length and direction of the vector from every stone in the river bed to the cobble are calculated. The obstacle chosen is that list member which has the shortest vector within grazing angle of the cobble.

3.3 Simulations using equal sized stones

In order to start simply, the first simulation `Riverbed 5.0` uses only the rebound rule on stones of equal radius, in this case 10 cm. A typical result is shown in figure 3.5.

In an effort to produce structures with a stronger transverse component and to take into account the shielding effects of neighboring clasts, `Riverbed 5.1` introduces the neighbor rule, requiring stones to have few close neighbors to be entrained. Figure 3.6 shows the result of a simulation using the same parameters, random seed and rules as in the previous experiment with the addition of the neighbor rule.

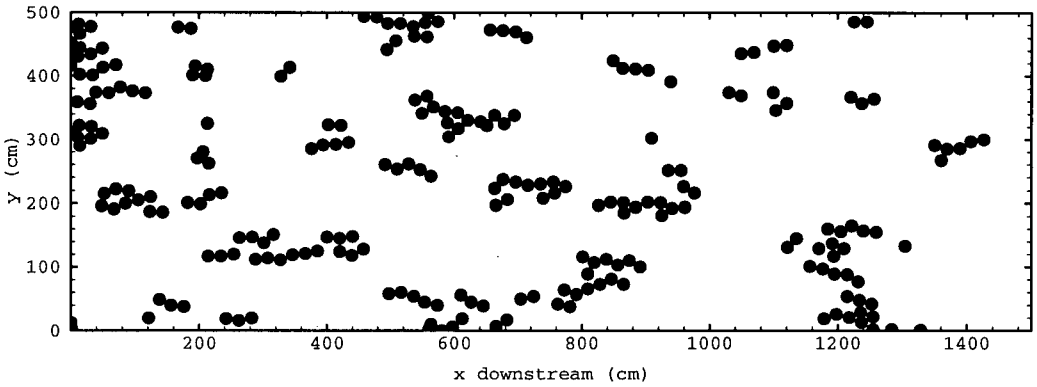


Figure 3.5: Final bed of `Riverbed 5.0` using the rebound rule after 2000 random entrainments of 238 stones of radius 10 cm with FULCRUM = 90 and SEEDFRACTION = 0.1. Flow is to the right.

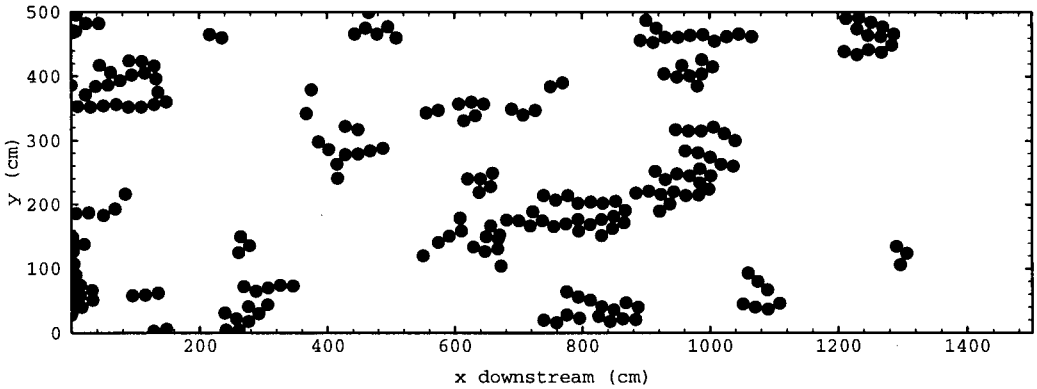


Figure 3.6: Final bed of `Riverbed 5.1` using the rebound and neighbor rules after 2000 random entrainments of the 238 stones of radius 10 cm in the bed. FULCRUM = 90 and SEEDFRACTION = 0.1. Flow is to the right.

3.3.1 Discussion

The bed resulting from `Riverbed 5.0` displays linear ridges consisting of approximately 3 to 15 members which trend, and tend to fan out, in the upstream direction. Approximately 5 % of the stones remain in open positions. Similarly, the results of `Riverbed 5.1` show the vast majority of stones comprising upstream undulating, elongate clusters of approximately 3 to 20 members. Hardly any stones remain in open positions. Comparing this result with the previous one, the ridges are less linear with a slightly wider spatial spread than when using only the rebound rule. Therefore, the rebound rule, with or without the neighbor rule, is insufficient to produce transverse bedforms with equal sized stones on a sparsely populated bed.

Both of these simulations produce longitudinal clusters which do not resemble the Harris Creek patterns. The neighbor rule was added in the hopes of introducing lateral clustering and can be seen to produce clusters with a less pronounced linear trend upstream but it does not significantly change the orientation of the clusters. Comparing the results of these two different sets of rules, it appears that the neighbor rule is redundant since the billiard ball rules impose it: stones cannot move downstream if other stones are in front of them.

The upstream trending clusters of stones produced with `Riverbed 5.0` resemble cluster bedforms (figure 2.5). Although flow perturbations are thought to be partially responsible for cluster bedforms, their presence in this simulation suggests they may be a result of obstacles to clast movement on a sparsely populated bed.

The difference in the number of stones remaining in open positions between these two experiments is a result of the design of the computer model which uses the number of random entrainments as the main counter governing the length of the simulation. In `Riverbed 5.1`, since a proportion of stones will be closely surrounded by other stones and therefore fail to meet the entrainment criterion imposed by the neighbor rule, the linked list must be searched more frequently to find a stone that can move before the random entrainment counter is incremented. Consequently, many more random numbers are culled, the list is more thoroughly searched, and any stones in open positions are likely to be entrained sooner than they would have in `Riverbed 5.0`. If the `Riverbed 5.0` simulation is run for a longer time, eventually all stones leave the open positions and cluster together. This effect demonstrates how simulation results may be affected by the design of the computer code and not be solely the expression of the physics programmed into it.

The statistical algorithm was performed on the `Riverbed 5.1` experiment (figure 3.7). Since the structures trend parallel to the flow direction, the density plot along x does not register them very well. They would show up better in the y density plot, however as stated previously, the y plots are unreliable due to excessive changes in bin widths. The density plot shows a low percentage of covered area for virtually all angles. The dark band at xbin = 10 to 25 is the signature of the accumulation of stones on the upstream margin of the reach. The large white area above xbin = 270 is the signature of the empty downstream

margin of the simulated reach. The two maxima in the PC^2 graph at 23° and −16° record the high covered area of strips trending at angles of −67° and −74° in the simulated bed.

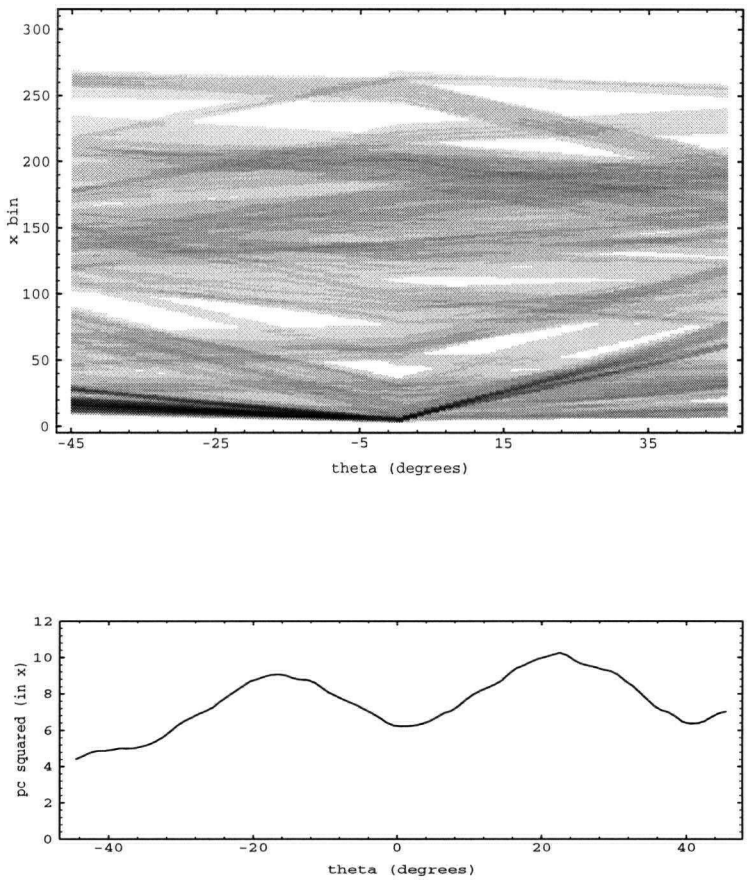


Figure 3.7: Density plot (top) and PC^2 graph (bottom) for the `Riverbed 5.1` result along x. For the density plot, white pixels indicate regions with zero percentage of stones and black pixels indicate regions of high percentage of stones. The peaks in the PC^2 graph correspond to angles at which the distribution of stones is least uniform. In the PC^2 graph maxima appear at angles +15° and −7° and can be seen to be coincident with the angles at which there exist regions of alternating light and dark pixels in the density plot, that is, along vertical lines in the top graph. These angles correspond to ridges aligned at −75° and 83° in the Harris Creek map.

3.4 Simulations using unequal sized stones

Since Harris Creek's cobbles comprise a range of grain sizes and cover 20 − 30% of the reach, it is possible that a denser population of unequal sized stones will introduce different structures into the simulated reach. `Riverbed 6.1` and all later versions use stones of unequal diameter drawn from a lognormal distribution truncated at D_{75} and D_{100} for Harris Creek, 10 cm and 50 cm respectively [9]. `Riverbed 6.1` uses only the

rebound rule on stones of unequal size. A typical result is shown in figure 3.8.

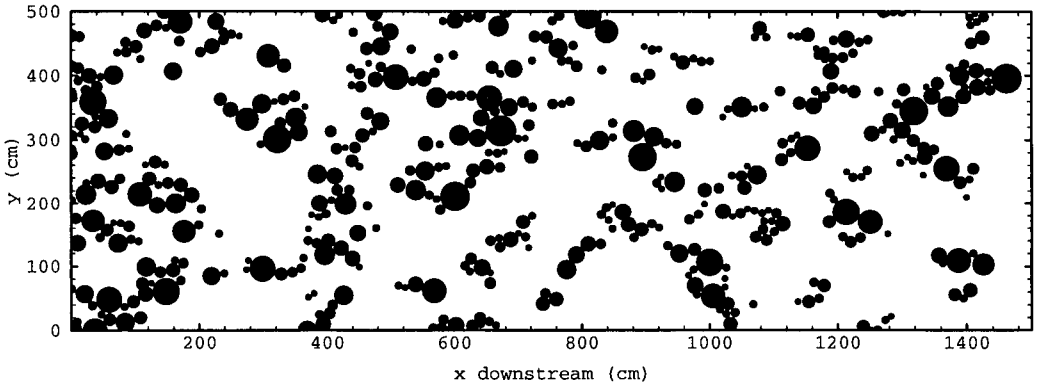


Figure 3.8: Final bed of `Riverbed 6.1` using 477 unequal sized stones and the rebound rule after 3000 random entrainments. SEEDFRACTION is 0.20 and FULCRUM is set to 60. Flow is to the right.

`Riverbed 6.2` introduces the obstacle rule requiring an obstacle to be at least half the size of the impacting stone before the stone will stop. The obstacle rule takes precedence over the rebound rule when choosing the obstacle for a stone. Consequently, if a candidate obstacle is small enough, the stone will pass over it until stopped by a large enough stone, causing some stones to overlap smaller stones when they finally come to rest. A typical result, using the same parameters as the `Riverbed 6.1` experiment, is shown in figure 3.9.

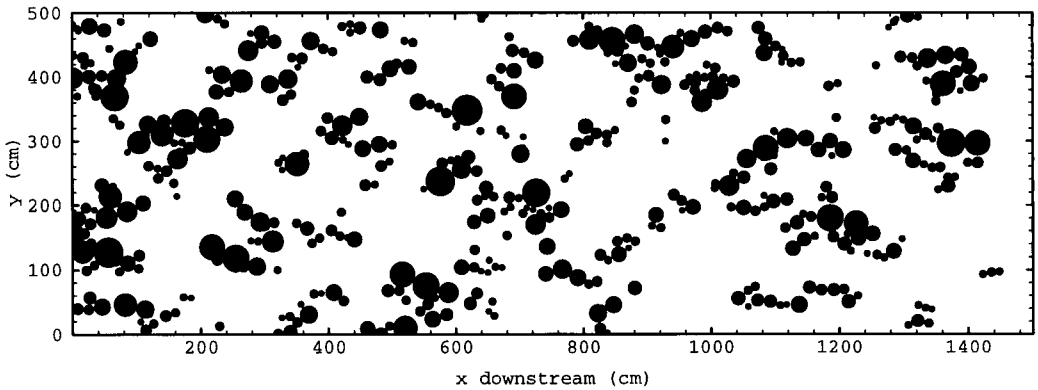


Figure 3.9: Final bed of `Riverbed 6.2` using the obstacle and rebound rules on 477 unequal sized stones after 3000 random entrainments. SEEDFRACTION is 0.20 and FULCRUM is set to 60. Flow is to the right. Because of the primacy of the obstacle rule over the rebound rule, some overlap of stones occurs.

3.4.1 Discussion

In both `Riverbed 6.1` and `6.2`, the stones tend to form a variety of structures including cluster bedforms and arcuate clusters trending at an angle to flow direction. Good examples of cluster bedforms occur in figure 3.8 at coordinates (1140, 300) and in figure 3.9 at (240, 400) and (350, 275). Arcuate ridges occur in figure 3.8 with apexes positioned at (875, 130) and (1280, 340) and in figure 3.9 centered at (650, 180). No good examples of transverse ribs occur. Both of these simulations produce the unrealistic structure consisting of a large stone upstream of a string of small stones. The addition of the obstacle rule sorts the stones and tends to deposit large stones near other large stones, producing an unnatural size distribution in the light of the Harris Creek size distributions shown in figures 2.8 to 2.12. Overall, the results of these two simulations are more realistic looking than the results using equal sized stones though unrealistic structures occur. These two simulations used twice as many stones as the `Riverbed 5` experiments. The increase in number of stones contributed to the growth of oblique clusters and ridges.

Using a FULCRUM of 60 allows stones to glance off an obstacle only when the rebound angle is less than 60° from the flow direction. This means that some stones stop in positions slightly more lateral to the obstacle than they would have if FULCRUM was 90. However, this effect is not extreme: comparison of simulations in `Riverbed 6.1` using different values for the FULCRUM parameter show that lateral clustering is not significantly enhanced by this change alone.

Comparing the histograms of size distributions (figure 3.10) for the initial and final beds with the distribution of clasts exiting the reach for the `Riverbed 6.2`, no size stone is proportionally more likely to exit than others.

The results of the statistical algorithm on the `Riverbed 6.2` experiment can be found near the end of Appendix B, the first of two cases shown. The density plot is less definitely banded compared to the density plots of the three boxes in Harris Creek (figures 2.13, 2.14 and 2.15), yet it is significantly more banded compared to the density plot of the `Riverbed 5.1`. The density plot displays moderately well-developed dark and light bands indicating that oblique linear clusters are created better than in `Riverbed 5.1`. Again, the dark band at xbin = 0 to 30 and the light band above xbin = 290 reflect the high density of stones at the upstream margin and the paucity of stones at the downstream margin, respectively. The PC^2 graph has peaks at $-23°$ and 12° corresponding to ridges oriented at 67° and $-78°$ in the simulated reach. The location of these ridges is hard to see in the in figure 3.9. This is because cobble ridges in the bed extend only part way across the bed while the strips used in the statistical algorithm extend the full width of the reach, registering any stones in that strip regardless of whether there are large gaps between adjacent stones or whether some of the stones more properly belong to other ridges trending at different angles but intersecting the strip.

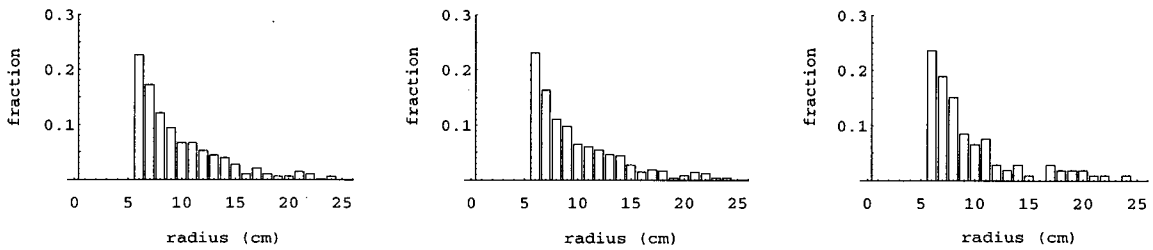


Figure 3.10: Size distribution of stones in the initial bed (left), final bed (center) and of the stones that travel through the reach (right) for Figure 3.9.

3.5 Simulations with reluctant large and mobile small stones

In gravel bed rivers, large stones are less likely to be entrained and travel shorter distances before stopping than small stones [8]. To incorporate this behavior into the simulation model `Riverbed 7.0` uses the entrainment probability rule, with CRITERION = 15, and the travel distance rule. The maximum travel distance for each stone is computed using equation (2.1) with $D_{50subsurface} = 10$ cm and $L_{D_{50surface}} = 236$ cm. $D_{50subsurface}$ for Harris Creek is 6 cm, however the higher value of 10 cm is used because this is the size of the smallest, and theoretically the most mobile, stones in the simulation. A typical result of a simulation using these rules along with the rebound rule is shown in figure 3.11. The bed is 25% covered to mirror the coverage of the Harris Creek boxes.

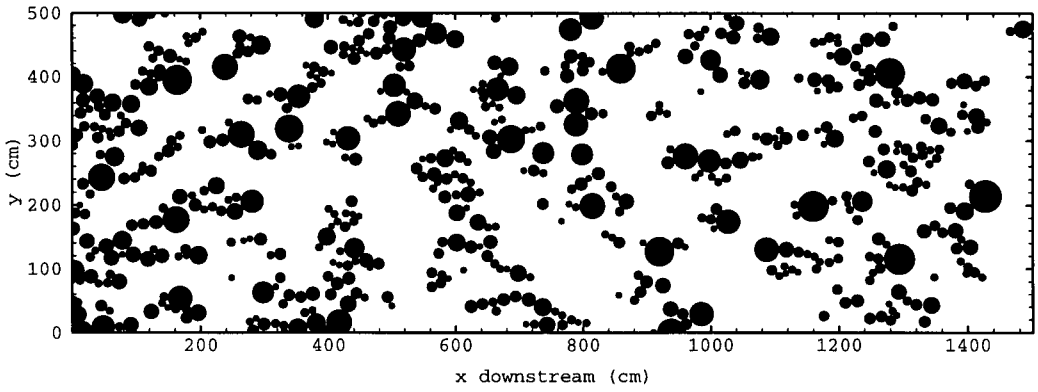


Figure 3.11: Final bed of `Riverbed 7.0` using 596 unequal sized stones, the entrainment probability, travel distance and rebound rules after 3500 random entrainments. SEEDFRACTION is 0.25 and FULCRUM is set to 60. Flow is to the right.

3.5.1 Discussion

This combination of rules produces a variety of structures including clusters bedforms, at (1290, 145) and (1030, 160), transverse ribs, extending from (580, 260) to (780, 0), (800, 210) to (800, 500) and (420, 0) to (440, 220), and circular stone cells, centered at (440, 360) and (570, 400). Close examination of figure 3.11 shows the large stones are more or less scattered throughout the bed while the small stones tend to cluster together in the vicinity of large stones, a distribution that mirrors Harris Creek's distribution (refer to figures 2.8 to 2.12). The stones are less tightly packed than in previous experiments because they can now stop in open positions on the bed and not only flush against an obstacle.

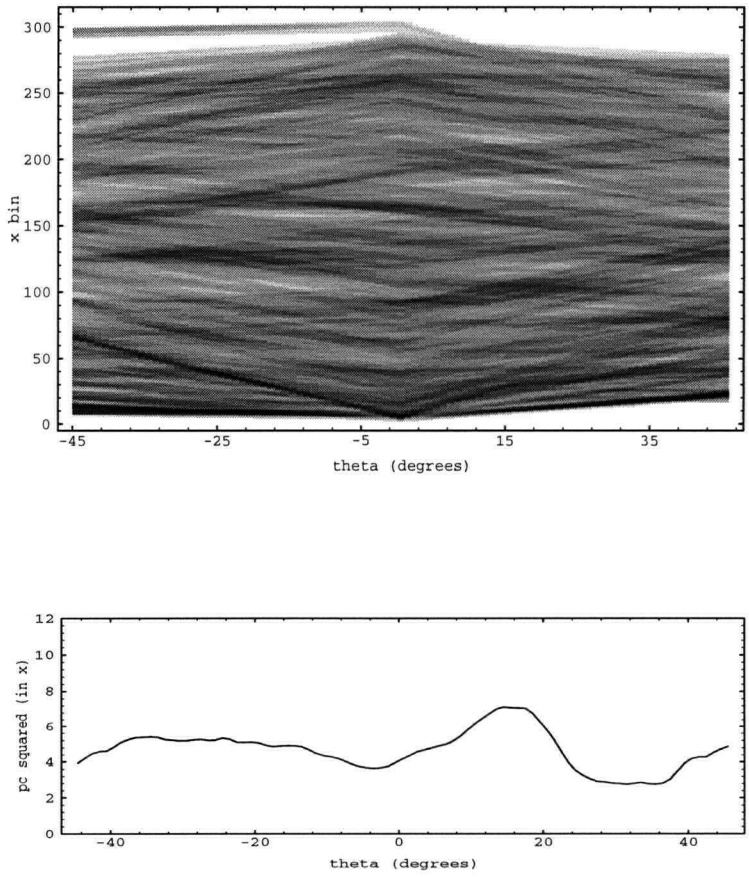


Figure 3.12: Density plot (top) and PC^2 graph (bottom) along x for the `Riverbed 7.0` simulation result in figure 3.11. The density plot exhibits alternate regions of light and dark pixels at an angle of $-19°$. The PC^2 graph maxima appear at angles $+15°$ and $-35°$. These angles correspond to ridges aligned at 71°, 55° and $-75°$ in the simulated creek bed.

The results of the statistical algorithm on the `Riverbed 7.0` bed are shown in figure 3.12. The density plot registers the upstream margin clogged with stones and the downstream margin empty of stones

as in all previous experiments. The density plot and PC^2 maxima correspond to ridge structures oriented at angles of 71°, 55° and −75° in the simulated reach. These orientations compare well with the trends of the ridges as determined in the rose diagrams (figure 2.3). The density plot is better banded than those of all previous simulations and best resembles the results for the Harris Creek boxes.

In addition to a variety of structures, `Riverbed 7.0` also produces a bed composed almost entirely of cluster bedforms when the stone population is small enough to cover only 10% of the reach area. A typical result of an experiment using SEEDFRACTION of 0.10, the same value used in `Riverbed 5.0` and `5.1`, is shown in figure 3.13. Good cluster bedforms occur at (840, 180), (1240, 80) and (320, 220).

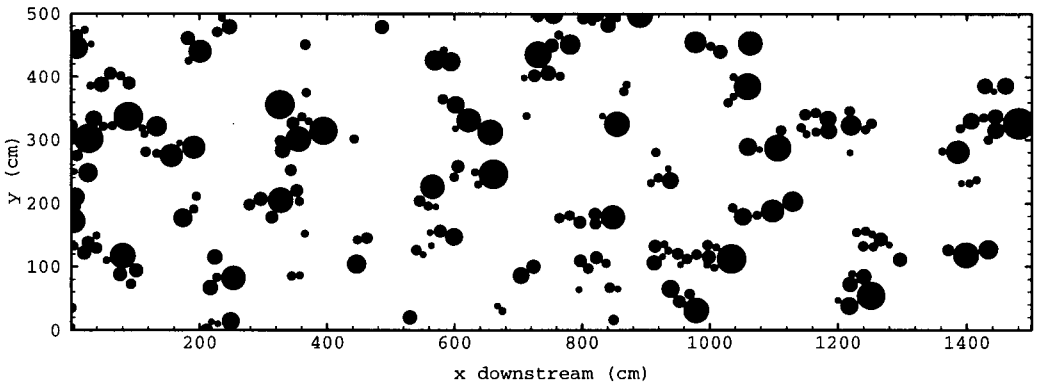


Figure 3.13: Final bed of `Riverbed 7.0` using 238 unequal sized stones, the entrainment probability, travel distance and rebound rules, after 2500 random entrainments. SEEDFRACTION is 0.10 and FULCRUM is set to 60. Flow is to the right.

Chapter 4

Conclusions

Harris Creek displays a net-like pattern of cobble ridges, one or two cobbles thick, with a variable yet strong transverse alignment. The patterns can be characterized as a framework of smaller stones articulated about randomly positioned large stones, as suggested by the separate maps of the stones in each size bin (figures 2.8 to 2.12). Alternatively, the patterns can be thought of as an admixture of several different structures: cluster bedforms, transverse ribs, arcuate ridges and isolated stones. Crude hydraulic calculations support observations that the cobble patterns form in sub-critical flows and not in the near critical to critical flows thought to be necessary to form transverse ribs.

Experiments on a bed with 10% of its area covered by stones almost exclusively produce cluster bedforms regardless of the rules used or whether the stones are all the same size or not. This finding suggests that cluster bedforms are a result of obstacles to clast movement in a sparsely populated bed. In a more densely populated bed, the large obstacle clasts in cluster bedforms may become the nuclei from which oblique ridges develop out of the excess sediment. Experiments with unequal sized stones introduce a greater variety of orientations to the resulting structures than those developed in equal sized stones. This suggests that the magnitude of the stone population and the range of sizes influences the suite of bedforms that emerge.

Simulations of `Riverbed 7.0` using entrainment probabilities and maximum travel distances, both of which vary inversely proportional to stone radius, produce patterns most closely resembling Harris Creek's patterns. These simulations produce a bed with pebble clusters, transverse and oblique ribs, arcuate ridges and isolated stones. The density plot is narrowly and finely banded and is similar to the density plot of the Harris Creek boxes. The density plot also registers ridges at a similar orientation as indicated by the rose diagrams.

The rebound rule appears to be an adequate representation of how stones will impact a downstream

obstacle and serves as an adequate base upon which to build different rules. It can be substantially enhanced by incorporating several changes. More randomness can be programmed into the impact to deliver the unpredictable rebound trajectories resulting from irregular shaped stones colliding with one another. The momentum exchange between obstacle and impinging stone can be calculated to determine how a smaller stone might be launched into transport as a result of gaining momentum from the impact of a larger stone.

As programmed, the neighbor rule is unnecessary since its inclusion does not produce structures differing from using only the rebound rule. A more realistic and effective neighbor rule would discriminate between stones in front of and behind a stone and implement the decrease and increase, respectively, in the lift forces associated with such positioning [7]. Lateral neighbors should be discriminated too although no experimental results exist on the influence of lateral neighbors on a particle's mobility.

The obstacle rule sorts the stones according to size resulting in large stones depositing close to other large stones. The positions of the largest stones in Harris Creek are not clustered together like the smaller stones, but scattered throughout the reach. Therefore, the obstacle rule leads to unrealistic results. This implies that large stones can stop behind any size obstacle or that they stop before they collide with anything.

That stones do not necessarily need an obstacle to their continued downstream movement in order to stop is supported by the results of `Riverbed 7.0` simulations. The entrainment probability rule is a realistic rule, supported as it is by empirical investigation [8] and consideration of the fact that at a given shear stress, smaller stones in open positions are more likely to move than larger stones in similar positions. This rule uses a simple linear expression for the probability of entrainment. It may benefit from using a nonlinear probability of entrainment.

The rule which seems to contribute most to generating realistic cobble structures is the travel distance rule. The advantage of this rule is that it has an empirical basis [8] and allows stones to stop in open positions. All other rules deposit stones flush against other stones, achieving a strict clustering that is not seen in real river beds. Another advantage of the travel distance rule is that it condenses several hydrologic effects into a single relation. For example, the friction between a stone and the surrounding water and between a stone and the river bed will act to slow a transporting stone, eventually causing it to stop. Also the inertia of a stone will affect its travel distance in the same manner, and the variability in shear stress in a turbulent flow may be strong enough to entrain a stone at one instance but drop significantly the next instant, causing the stone to be deposited.

The statistical algorithm does register linear ridges of stones and their corresponding orientations although the results are somewhat muddied by other parts of the bed that intersect the bin strips whose covered area is being calculated.

The `Riverbed` computer model is both stochastic and deterministic. Randomness enters the model via the choice of seed value used in the random number generator. If the same value is used and all parameters

and rules remain the same, the outcome will be exactly duplicated. For this reason, the simulation cannot be expected to exactly reproduce the Harris Creek patterns but it can be expected to produce similar results over a large number of trials using different seed values.

4.1 Future Work

A computer simulation can always be improved, though usually with considerable cost in the form of increasing the complexity of the program and longer computation times. Many of the enhancements to the rebound and neighbor rules and the entrainment probability as discussed in the preceeding section can be programmed to improve realism. Flow effects are only qualitatively modeled but could be incorporated by utilizing equations governing water velocity, shear stress, and other pertinent hydrodynamic factors. The size distributions of the stones that exit the downstream margin of the simulated reach can be compared to data on sediment transport collected for Harris Creek to further calibrate and validate the model.

The statistical algorithm can be improved to discriminate ridges. However the exact form of computationally possible improvements is complicated. At present the algorithm only indicates areas of possible ridges.

Reproducing Harris Creek's patterns was the motivation for the suite of experiments presented here. The success of `Riverbed 7.0` warrants a more systematic investigation into the effects of different parameter values. It may be possible to extend this version of the model to simulate coarse sediment transport in more general situations. Other changes that can be made include allowing stones to exit the reach without having to introduce new ones, no longer maintaining a constant stone population.

Appendix A

Statistical Algorithm to Identify Ridge Structures

A search for a suitable statistical test to identify transverse ridge patterns in stones uncovered comparable problems in forestry and marine biology. The spatial distribution of the clasts is similar to the spatial distribution of trees in a forest [29] and of acorn barnacles on the side of a ship [11]. The data of these problems record the spatial coordinates (x and y) and the radius r of the data points and the analyses cater to cluster formations and their associated statistics to determine measures of spatial randomness, regularity, and clustering. They do not address the spatial statistics of linear accumulations. For this reason, the following statistical algorithm was developed to identify cobble ridges. Suppose two cobble ridges span part of the width of the stream as illustrated in figure A.1. By rotating the rectangular bounding box about the lower left corner point and calculating the fraction of area covered by the stones in vertical and horizontal strips, as the ridges rotate into parallelism with the bins, a maximum of covered area will register. By calculating the covered area all dimensions are rendered unnecessary.

In order to cover all possible angles in a 360° range that the ridges might trend, the rectangular bounding box was rotated over a span of 90° from −45° to 45°, where 0° is along the positive x direction of the reference axes and also the downstream direction of the unrotated box. Ridges oriented transverse or sub-transverse to the flow would rotate into parallelism with the strips spanning the reference x axis and produce a peak in the histogram of percent area covered along the reference x axis. This is illustrated in figure A.1 by the transverse ridge on the right. For this ridge, the bins along x register a high percentage of covered area and produce a peak in the x histogram but no peak in the histogram spanning the y axis. The more longitudinally oriented rib on the left contributes to a uniform distribution in the x histogram and a peak in the y histogram.

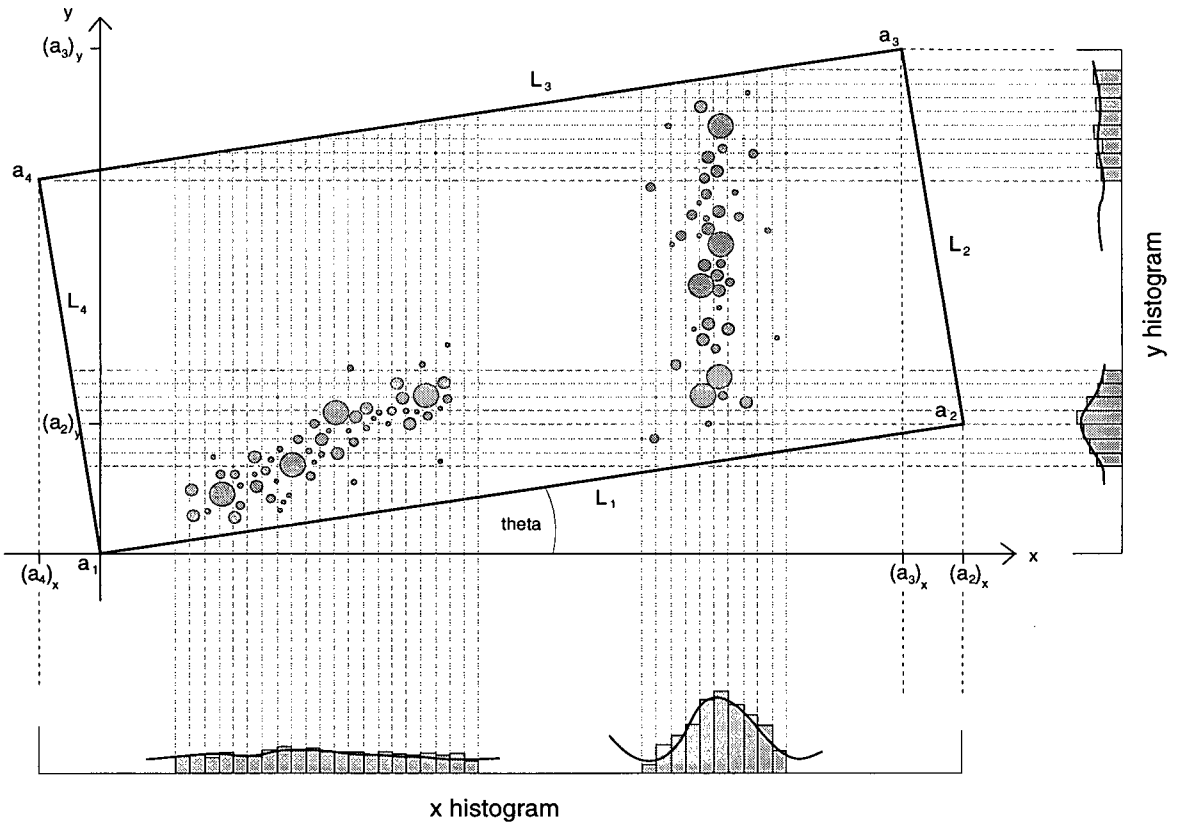


Figure A.1: Schematic diagram illustrating how ridges oriented at different angles register in the x and y histograms. The transverse ridge on the right causes a maximum in the x histogram since, for this angle of θ, it is parallel to the bin strips along x. The longitudinal ridge on the left causes a peak in the y histogram. Flow direction is parallel to the box and in the positive x direction when the box is rotated to 0°.

After the x and y histograms along the reference axes have been calculated at every degree for the box rotated from $-45°$ to 45°, the PC^2 test is calculated along x and y. The PC^2 test is defined as

$$PC^2 = \sum_{i=1}^{n} \frac{(o_i - e)^2}{e} \tag{0.1}$$

where e is the fraction of the total bed covered by the entire stone population, o_i is the covered fraction of a single strip and n is the number of bins.

A constant number of histogram bins along the reference x and y axes was used for all angles of rotation to simplify the *Mathematica* algorithm. Looking at figure A.1, it is evident that the far corners of the box will span a greater dstance along the reference x axis when it is rotated from 0°. Consequently, the width of the bins and the span of the histogram changes as the bed is rotated in order to maintain a constant number of bins. Using wide bins tends to smooth the histograms whereas using narrow bins tends to show finer detail. However, smaller bins require significantly more computation time. At $\theta = 0$, the bin

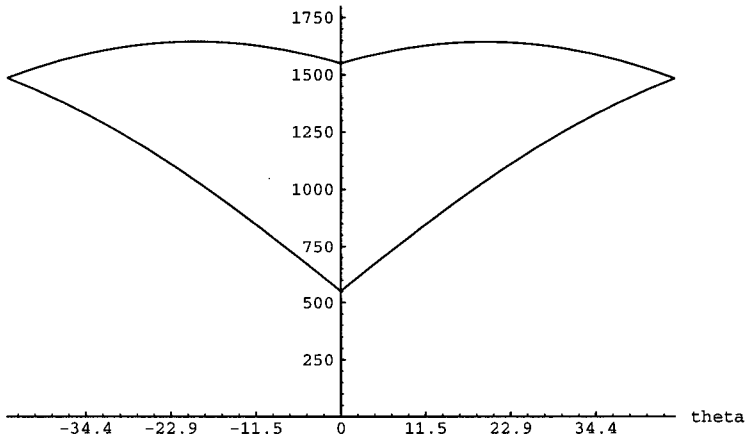


Figure A.2: A graph showing how the span of the x (top curve) and y (bottom curve) histograms change depending on the angle the bounding box is rotated to. The vertical axis is the length of the histogram in cm and the horizontal axis is θ in degrees. The span of the x histogram is more constant than that of the y histogram.

width is set to 5 cm, a width that ensures that the smallest stone in the simulation will register in at least two adjacent strips.

It is important to note that this heuristic does not unequivocally identify cobble ridges. The statistical algorithm is only effective at locating regions where the area covered by the stones differs from an assumed uniform distribution of stones as indicated by maxima in the PC^2 graph. A high percentage of covered area for a strip implies a high probability of finding a ridge only if adjacent strips have a low percent of covered area. This statistical algorithm would faithfully identify the trends of cobble ridges generously spaced along a line because then the pattern is effectively one dimensional. However, Harris Creek's patterns range over two spatial dimensions with a fair amount of variability. Arcuate ridges which span several adjacent strips, regardless of rotation angle, are difficult to isolate by this method.

The following is a description of the algorithm. The *Mathematica* code which performs these calculations is in Appendix B.

1. Read in each clast as a vector of the form:

$$\begin{bmatrix} x & y & r \end{bmatrix}$$

2. Given a value of θ, multiply by the rotation matrix A

$$A = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

while preserving the radius to get

$$\begin{bmatrix} x\cos\theta - y\sin\theta & x\sin\theta + y\cos\theta & r \end{bmatrix}$$

3. Since stones might intersect the edges of the original bounding box, and in order to significantly simplify the algorithm, the bounding box was increased in size in both dimensions by two times the radius of the largest contained stone. This ensures every stone is completely contained in the interior of the box. Thus for any stone

$$\begin{bmatrix} x\cos\theta - y\sin\theta + r_{max} & x\sin\theta + y\cos\theta + r_{max} & r \end{bmatrix}$$

4. The four corner points of the bounding rectangle, where x_m and y_m are the maximum dimensions in the x and y dimensions, respectively, are mapped as follows:

$$a_1 \;:\; (0,0) \longrightarrow (0,0) \tag{0.2}$$

$$a_2 \;:\; (x_m,0) \longrightarrow (x_m\cos\theta, x_m\sin\theta) \tag{0.3}$$

$$a_3 \;:\; (x_m,y_m) \longrightarrow (x_m\cos\theta - y_m\sin\theta, x_m\sin\theta + y_m\cos\theta) \tag{0.4}$$

$$a_4 \;:\; (0,y_m) \longrightarrow (-y_m\sin\theta, y_m\cos\theta) \tag{0.5}$$

5. The four lines forming the bounding box have the following equations (for all non-zero θ)

$$L_1 \;:\; y = (\tan\theta)\,x \tag{0.6}$$

$$L_2 \;:\; y = -(\cot\theta)\,x + \frac{x_m}{\sin\theta} \tag{0.7}$$

$$L_3 \;:\; y = (\tan\theta)\,x + \frac{y_m}{\cos\theta} \tag{0.8}$$

$$L_4 \;:\; y = -(\cot\theta)\,x \tag{0.9}$$

6. The projection of the 'outermost' points of the box onto the x and y reference axes defines the span along x and y of the histograms. For $\theta \in [0°, 45°]$ the x histogram has a span from the x-coordinate of the a_4 point, denoted $(a_4)_x$ in figure A.1, to the x-coordinate of the a_2 point, denoted $(a_2)_x$. Expressed in absolute units, this is $y_m\sin\theta + x_m\cos\theta$. The span of the y histogram is from $(a_1)_y$ to $(a_3)_y$ or $y_m\cos\theta + x_m\sin\theta$. By symmetry, for $\theta \in [-45°, 0°]$ the span changes in a similar manner as illustrated in figure A.2. It is important to note that the span of either histogram is a function of θ and can change dramatically.

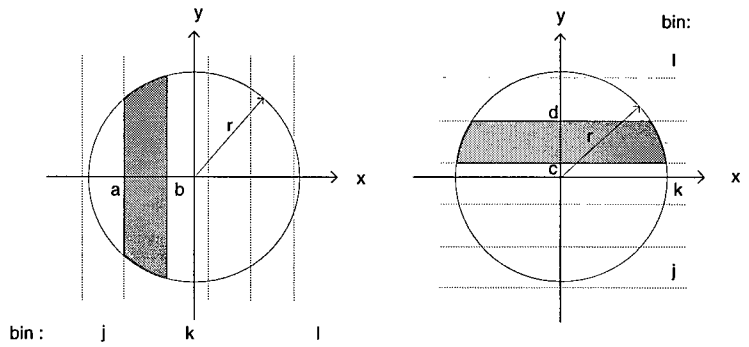


Figure A.3: Given a clast that spans several bins in either the x or y dimension, it is necessary to determine which bin the leftmost (bottommost) point, the centre, and the rightmost (topmost) point are contained. The area of the stone in each bin is computed by the integrals I_x or I_y.

7. To calculate the area of each clast in any given bin, I_x:

$$I_x = \int_a^b \int_{-\sqrt{r^2-x^2}}^{+\sqrt{r^2-x^2}} dy\,dx \tag{0.10}$$

$$= 2\int_a^b \sqrt{r^2-x^2}\,dx \tag{0.11}$$

$$= b\sqrt{r^2-b^2} + r^2 \arcsin\frac{b}{r} - a\sqrt{r^2-a^2} - r^2 \arcsin\frac{a}{r} \tag{0.12}$$

$$= b\sqrt{r^2-b^2} - a\sqrt{r^2-a^2} + r^2\left[\arcsin\frac{b}{r} - \arcsin\frac{a}{r}\right] \tag{0.13}$$

For horizontal strips, the area of each clast in any given bin, I_y, is the same:

$$I_y = d\sqrt{r^2-d^2} - c\sqrt{r^2-c^2} + r^2\left[\arcsin\frac{d}{r} - \arcsin\frac{c}{r}\right] \tag{0.14}$$

8. The area of any strip can be calculated from the equations of the four perpendicular lines making up the bounding box. Since the number of bins on either axis is specified and as the span of the histograms is known, the width and area of each bin can be calculated.

9. The PC^2 test, as defined in equation 0.1, was redefined to minimize the contribution of bins at either end of the bounding box which, by virtue of experiencing dramatic width changes during rotation, can return anomalous values. For example, if the bounding box was rotated counterclockwise by one degree, the leftmost and rightmost bins of the x histogram have significantly smaller areas than those in the interior of the box. It is very likely that a stone could cover a high percentage of the area of this small strip and return an usually high value for the fraction of covered area for that strip. The number of bins in both histograms was set to 310 (thus for $\theta = 0°$ the width of a bin in the x histogram would have an initial width of 5) thus 80 bins on either side were ignored to calculate the PC^2 test, redefined

as

$$PC^2 = \sum_{i=81}^{230} \frac{(o_i - e)^2}{e} \tag{0.15}$$

Appendix B

Mathematica Script for Statistical Algorithm

The following is an edited version of the *Mathematica* script that performs the statistical test for ridge structures. The script generates data for a range of θ from $-45°$ to $45°$. The datafile bed1.dat used in this example is the final result of the `Riverbed 6.1` experiment shown in figure 3.8. The density plot and PC^2 graph for this result is shown in the script using the data file of the final condition of the `Riverbed 6.1` simulation as shown in figure 3.8.

```
adat=ReadList["bed1.dat",{Number,Number,Number}];

rmax=25;
nx=310;
ny=310;
xmax=1500+2 rmax;
ymax=500+2 rmax;
ms=Dimensions[adat]

sa=0;
bdat=Table[0,{ms[[1]]},{ms[[2]]}];
For[i=1,i<=ms[[1]],i=i+1,bdat[[i,1]]=adat[[i,1]]+rmax;
                         bdat[[i,2]]=adat[[i,2]]+rmax;
                         bdat[[i,3]]=adat[[i,3]];
                         sa+=N[Pi adat[[i,3]]^2];];
N[sa,10]

156653.979

A[theta_]:=N[{{Cos[N[theta Pi/180]],-Sin[N[theta Pi/180]],0},
              {Sin[N[theta Pi/180]],Cos[N[theta Pi/180]],0},{0,0,1}}];
xmat=Table[0,{91},{nx}];
ymat=Table[0,{91},{ny}];
a1={0,0};
a2={N[xmax Cos[theta Pi/180]],N[xmax Sin[theta Pi/180]]};
a3={N[xmax Cos[theta Pi/180]-ymax Sin[theta Pi/180]],
```

```
    N[xmax Sin[theta Pi/180]+ymax Cos[theta Pi/180]]};
a4={N[-ymax Sin[theta Pi/180]],N[ymax Cos[theta Pi/180]]};
Ix=Compile[{a,b,r},b Sqrt[r^2-b^2]-a Sqrt[r^2-a^2]
       +r^2 (ArcSin[b/r]-ArcSin[a/r])];
Iy=Compile[{c,d,r},d Sqrt[r^2-d^2]-c Sqrt[r^2-c^2]
       +r^2 (ArcSin[d/r]-ArcSin[c/r])];

For[theta=0,theta<=45,theta=theta+1,
       cdat=N[Transpose[A[theta].Transpose[bdat]]];
       ddat=Table[0,{ms[[1]]},{ms[[2]]}];
       a1={0,0};
       a2={N[xmax Cos[theta Pi/180]],N[xmax Sin[theta Pi/180]]};
       a3={N[xmax Cos[theta Pi/180]-ymax Sin[theta Pi/180]],
           N[xmax Sin[theta Pi/180]+ymax Cos[theta Pi/180]]};
       a4={N[-ymax Sin[theta Pi/180]],N[ymax Cos[theta Pi/180]]};
       For[i=1,i<=ms[[1]],i=i+1,ddat[[i,1]]=cdat[[i,1]]+Abs[a4[[1]]];
                                ddat[[i,2]]=cdat[[i,2]];
                                ddat[[i,3]]=cdat[[i,3]];
       ];
       dx=N[(-a4[[1]]+a2[[1]])/nx];
       dy=N[a3[[2]]/ny];
       For[i=1,i<=ms[[1]],i=i+1,
              x=ddat[[i,1]];
              r=ddat[[i,3]];
              If[r!=0,r=r,r=0.00001];
              j=Floor[(x-r)/dx]+1;
              k=Floor[x/dx]+1;
              l=Floor[(x+r)/dx]+1;
              del=x-(k-1) dx;
              eps=k dx-x;
              m=k-j;
              n=l-k;
              If[m>0,For[s=m,s>m-0.5,s=s-1,xmat[[theta+46,k-s]]+=Ix[-r,(k-s) dx-x,r]];
                     For[s=m-1,s>=1,s=s-1,xmat[[theta+46,k-s]]+=Ix[(k-s-1) dx-x, (k-s) dx-x,r]];
                     For[s=0,s>=-0.5,s=s-1,xmat[[theta+46,k]]+=Ix[(k-1) dx-x,0,r]];,
                     xmat[[theta+46,k]]+=Ix[-r,0,r]];
              If[n>0,For[s=n,s>n-0.5,s=s-1,xmat[[theta+46,k+s]]+=Ix[(k+s-1) dx-x,r,r]];
                     For[s=n-1,s>=1,s=s-1,xmat[[theta+46,k+s]]+=Ix[(k+s-1) dx-x,(k+s) dx-x,r]];
                     For[s=0,s>=-0.5,s=s-1,xmat[[theta+46,k]]+=Ix[0,k dx-x,r]];,
                     xmat[[theta+46,k]]+=Ix[0,r,r]];
       ];
       For[i=1,i<=ms[[1]],i=i+1,
              y=ddat[[i,2]];
              r=ddat[[i,3]];
              If[r!=0,r=r,r=0.00001];
              j=Floor[(y-r)/dy]+1;
              k=Floor[y/dy]+1;
              l=Floor[(y+r)/dy]+1;
              del=y-(k-1) dy;
              eps=k dy-y;
              m=k-j;
              n=l-k;
              If[m>0,For[s=m,s>m-0.5,s=s-1,ymat[[theta+46,k-s]]+=Iy[-r,(k-s) dy-y,r]];
                     For[s=m-1,s>=1,s=s-1,ymat[[theta+46,k-s]]+=Iy[(k-s-1) dy-y, (k-s) dy-y,r]];
                     For[s=0,s>=-0.5,s=s-1,ymat[[theta+46,k]]+=Iy[(k-1) dy-y,0,r]];,
                     ymat[[theta+46,k]]+=Iy[-r,0,r]];
              If[n>0,For[s=n,s>n-0.5,s=s-1,ymat[[theta+46,k+s]]+=Iy[(k+s-1) dy-y,r,r]];
                     For[s=n-1,s>=1,s=s-1,ymat[[theta+46,k+s]]+=Iy[(k+s-1) dy-y,(k+s) dy-y,r]];
                     For[s=0,s>=-0.5,s=s-1,ymat[[theta+46,k]]+=Iy[0,k dy-y,r]];,
                     ymat[[theta+46,k]]+=Iy[0,r,r]];
       ];
];
```

```
For[theta=-1,theta>=-45,theta=theta-1,
       cdat=N[Transpose[A[theta].Transpose[bdat]]];
       ddat=Table[0,{ms[[1]]},{ms[[2]]}];
       a1={0,0};
       a2={N[xmax Cos[theta Pi/180]],N[xmax Sin[theta Pi/180]]};
       a3={N[xmax Cos[theta Pi/180]-ymax Sin[theta Pi/180]],
           N[xmax Sin[theta Pi/180]+ymax Cos[theta Pi/180]]};
       a4={N[-ymax Sin[theta Pi/180]],N[ymax Cos[theta Pi/180]]};
       For[i=1,i<=ms[[1]],i=i+1,ddat[[i,1]]=cdat[[i,1]];
                                ddat[[i,2]]=cdat[[i,2]]+Abs[a2[[2]]];
                                ddat[[i,3]]=cdat[[i,3]];
       ];
       dx=N[(a3[[1]])/nx];
       dy=N[(a4[[2]]-a2[[2]])/ny];
       For[i=1,i<=ms[[1]],i=i+1,
              x=ddat[[i,1]];
              r=ddat[[i,3]];
              If[r!=0,r=r,r=0.00001];
              j=Floor[(x-r)/dx]+1;
              k=Floor[x/dx]+1;
              l=Floor[(x+r)/dx]+1;
              del=x-(k-1) dx;
              eps=k dx-x;
              m=k-j;
              n=l-k;
              If[m>0,For[s=m,s>m-0.5,s=s-1,xmat[[theta+46,k-s]]+=Ix[-r,(k-s) dx-x,r]];
                     For[s=m-1,s>=1,s=s-1,xmat[[theta+46,k-s]]+=Ix[(k-s-1) dx-x, (k-s) dx-x,r]];
                     For[s=0,s>=-0.5,s=s-1,xmat[[theta+46,k]]+=Ix[(k-1) dx-x,0,r]];,
                     xmat[[theta+46,k]]+=Ix[-r,0,r]];
              If[n>0,For[s=n,s>n-0.5,s=s-1,xmat[[theta+46,k+s]]+=Ix[(k+s-1) dx-x,r,r]];
                     For[s=n-1,s>=1,s=s-1,xmat[[theta+46,k+s]]+=Ix[(k+s-1) dx-x,(k+s) dx-x,r]];
                     For[s=0,s>=-0.5,s=s-1,xmat[[theta+46,k]]+=Ix[0,k dx-x,r]];,
                     xmat[[theta+46,k]]+=Ix[0,r,r]];
       ];
       For[i=1,i<=ms[[1]],i=i+1,
              y=ddat[[i,2]];
              r=ddat[[i,3]];
              If[r!=0,r=r,r=0.00001];
              j=Floor[(y-r)/dy]+1;
              k=Floor[y/dy]+1;
              l=Floor[(y+r)/dy]+1;
              del=y-(k-1) dy;
              eps=k dy-y;
              m=k-j;
              n=l-k;
              If[m>0,For[s=m,s>m-0.5,s=s-1,ymat[[theta+46,k-s]]+=Iy[-r,(k-s) dy-y,r]];
                     For[s=m-1,s>=1,s=s-1,ymat[[theta+46,k-s]]+=Iy[(k-s-1) dy-y, (k-s) dy-y,r]];
                     For[s=0,s>=-0.5,s=s-1,ymat[[theta+46,k]]+=Iy[(k-1) dy-y,0,r]];,
                     ymat[[theta+46,k]]+=Iy[-r,0,r]];
              If[n>0,For[s=n,s>n-0.5,s=s-1,ymat[[theta+46,k+s]]+=Iy[(k+s-1) dy-y,r,r]];
                     For[s=n-1,s>=1,s=s-1,ymat[[theta+46,k+s]]+=Iy[(k+s-1) dy-y,(k+s) dy-y,r]];
                     For[s=0,s>=-0.5,s=s-1,ymat[[theta+46,k]]+=Iy[0,k dy-y,r]];,
                     ymat[[theta+46,k]]+=Iy[0,r,r]];
       ];
];

l3xl4x=Compile[{a,b,theta},1/2 (b^2-a^2) (Tan[theta Pi/180]+
         Cot[theta Pi/180])+(b-a) (ymax/Cos[theta Pi/180])];
l3xl1x=Compile[{a,b,theta},(b-a) ymax/Cos[theta Pi/180]];
l2xl1x=Compile[{a,b,theta},1/2 (a^2-b^2) (Tan[theta Pi/180]+
         Cot[theta Pi/180])+(b-a) (xmax/Sin[theta Pi/180])];
```

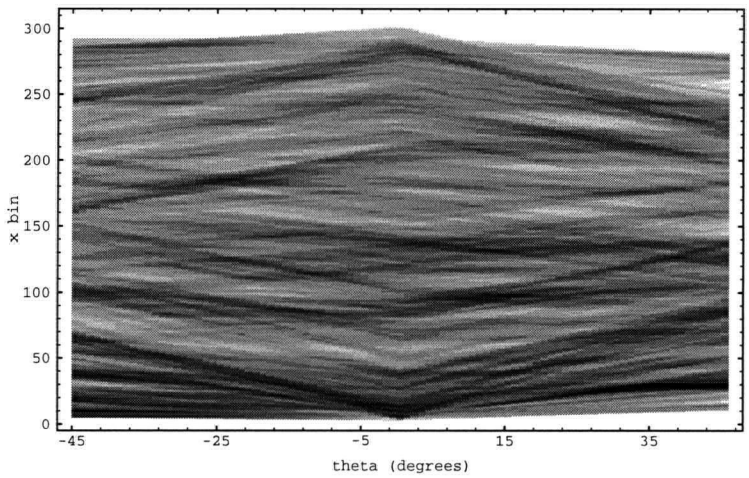
```
normxmat=Table[0,{91},{nx}];

For[theta=0,theta<=0.5,theta=theta+1,
       a1={0,0};
       a2={N[xmax Cos[theta Pi/180]],N[xmax Sin[theta Pi/180]]};
       a3={N[xmax Cos[theta Pi/180]-ymax Sin[theta Pi/180]],
           N[xmax Sin[theta Pi/180]+ymax Cos[theta Pi/180]]};
       a4={N[-ymax Sin[theta Pi/180]],N[ymax Cos[theta Pi/180]]};
       dx=N[(-a4[[1]]+a2[[1]])/nx];
       dy=N[a3[[2]]/ny];
       For[i=1,i<=nx,i=i+1,
              normxmat[[theta+46,i]]+=ymax dx;
       ];
];
For[theta=1,theta<=45,theta=theta+1,
       a1={0,0};
       a2={N[xmax Cos[theta Pi/180]],N[xmax Sin[theta Pi/180]]};
       a3={N[xmax Cos[theta Pi/180]-ymax Sin[theta Pi/180]],
           N[xmax Sin[theta Pi/180]+ymax Cos[theta Pi/180]]};
       a4={N[-ymax Sin[theta Pi/180]],N[ymax Cos[theta Pi/180]]};
       dx=N[(-a4[[1]]+a2[[1]])/nx];
       dy=N[a3[[2]]/ny];
       i1=Floor[(-a4[[1]]+a1[[1]])/dx]+1;
       i2=Floor[(-a4[[1]]+a3[[1]])/dx]+1;
       For[i=1,i<=i1-1,i=i+1,
              normxmat[[theta+46,i]]+=l3xl4x[a4[[1]]+(i-1) dx,a4[[1]]+i dx,theta];
              normxmat[[-theta+46,i]]+=l3xl4x[a4[[1]]+(i-1) dx,a4[[1]]+i dx,theta];
       ];
       For[i=i1,i<=i1+0.5,i=i+1,
              normxmat[[theta+46,i]]+=l3xl4x[a4[[1]]+(i-1) dx,0,theta];
              normxmat[[theta+46,i]]+=l3xl1x[0,a4[[1]]+i dx,theta];
              normxmat[[-theta+46,i]]+=l3xl4x[a4[[1]]+(i-1) dx,0,theta];
              normxmat[[-theta+46,i]]+=l3xl1x[0,a4[[1]]+i dx,theta];
       ];
       For[i=i1+1,i<=i2-1,i=i+1,
              normxmat[[theta+46,i]]+=l3xl1x[a4[[1]]+(i-1) dx,a4[[1]]+i dx,theta];
              normxmat[[-theta+46,i]]+=l3xl1x[a4[[1]]+(i-1) dx,a4[[1]]+i dx,theta];
       ];

       For[i=i2,i<=i2+0.5,i=i+1,
              normxmat[[theta+46,i]]+=l3xl1x[a4[[1]]+(i-1) dx,a3[[1]],theta];
              normxmat[[theta+46,i]]+=l2xl1x[a3[[1]],a4[[1]]+i dx,theta];
              normxmat[[-theta+46,i]]+=l3xl1x[a4[[1]]+(i-1) dx,a3[[1]],theta];
              normxmat[[-theta+46,i]]+=l2xl1x[a3[[1]],a4[[1]]+i dx,theta];

       ];
       For[i=i2+1,i<=nx,i=i+1,
              normxmat[[theta+46,i]]+=l2xl1x[a4[[1]]+(i-1) dx,a4[[1]]+i dx,theta];
              normxmat[[-theta+46,i]]+=l2xl1x[a4[[1]]+(i-1) dx,a4[[1]]+i dx,theta];
       ];
];

l4yl1y=Compile[{c,d,theta},1/2 (d^2-c^2) (Tan[theta Pi/180]+Cot[theta Pi/180])];
l4yl2y=Compile[{c,d,theta},xmax/Cos[theta Pi/180] (d-c)];
l3yl2y=Compile[{c,d,theta},(xmax/Cos[theta Pi/180]+ymax/Sin[theta Pi/180]) (d-c)
         -1/2 (d^2-c^2) (Tan[theta Pi/180]+Cot[theta Pi/180])];
l3yl1y=Compile[{c,d,theta},ymax/Sin[theta Pi/180] (d-c)];

normymat=Table[0,{91},{ny}];

For[theta=0,theta<=0.5,theta=theta+1,
```

```
        a1={0,0};
        a2={N[xmax Cos[theta Pi/180]],N[xmax Sin[theta Pi/180]]};
        a3={N[xmax Cos[theta Pi/180]-ymax Sin[theta Pi/180]],
            N[xmax Sin[theta Pi/180]+ymax Cos[theta Pi/180]]};
        a4={N[-ymax Sin[theta Pi/180]],N[ymax Cos[theta Pi/180]]};
        dy=N[a3[[2]]/ny];
        For[i=1,i<=ny,i=i+1,
              normymat[[theta+46,i]]+=xmax dy;
        ];
];

For[theta=1,theta<=19,theta=theta+1,
        a1={0,0};
        a2={N[xmax Cos[theta Pi/180]],N[xmax Sin[theta Pi/180]]};
        a3={N[xmax Cos[theta Pi/180]-ymax Sin[theta Pi/180]],
            N[xmax Sin[theta Pi/180]+ymax Cos[theta Pi/180]]};
        a4={N[-ymax Sin[theta Pi/180]],N[ymax Cos[theta Pi/180]]};
        dy=N[a3[[2]]/ny];
        i1=Floor[(a2[[2]])/dy]+1;
        i2=Floor[(a4[[2]])/dy]+1;
        For[i=1,i<=i1-1,i=i+1,
              normymat[[theta+46,i]]+=l4yl1y[(i-1) dy,i dy,theta];
              normymat[[-theta+46,i]]+=l4yl1y[(i-1) dy,i dy,theta];
        ];
        For[i=i1,i<=i1+0.5,i=i+1,
              normymat[[theta+46,i]]+=l4yl1y[(i-1) dy,a2[[2]],theta];
              normymat[[theta+46,i]]+=l4yl2y[a2[[2]],i dy,theta];
              normymat[[-theta+46,i]]+=l4yl1y[(i-1) dy,a2[[2]],theta];
              normymat[[-theta+46,i]]+=l4yl2y[a2[[2]],i dy,theta];
        ];
        For[i=i1+1,i<=i2-1,i=i+1,
              normymat[[theta+46,i]]+=l4yl2y[(i-1) dy,i dy,theta];
              normymat[[-theta+46,i]]+=l4yl2y[(i-1) dy,i dy,theta];
        ];

        For[i=i2,i<=i2+0.5,i=i+1,
              normymat[[theta+46,i]]+=l4yl2y[(i-1) dy,a4[[2]],theta];
              normymat[[theta+46,i]]+=l3yl2y[a4[[2]],i dy,theta];
              normymat[[-theta+46,i]]+=l4yl2y[(i-1) dy,a4[[2]],theta];
              normymat[[-theta+46,i]]+=l3yl2y[a4[[2]],i dy,theta];
        ];
        For[i=i2+1,i<=ny-1,i=i+1,
              normymat[[theta+46,i]]+=l3yl2y[(i-1) dy,i dy,theta];
              normymat[[-theta+46,i]]+=l3yl2y[(i-1) dy,i dy,theta];
        ];
        For[i=ny,i<=ny+0.5,i=i+1,
              normymat[[theta+46,i]]+=l3yl2y[(i-1) dy,a3[[2]],theta];
              normymat[[-theta+46,i]]+=l3yl2y[(i-1) dy,a3[[2]],theta];
        ];
];

For[theta=20,theta<=45,theta=theta+1,
        a1={0,0};
        a2={N[xmax Cos[theta Pi/180]],N[xmax Sin[theta Pi/180]]};
        a3={N[xmax Cos[theta Pi/180]-ymax Sin[theta Pi/180]],
            N[xmax Sin[theta Pi/180]+ymax Cos[theta Pi/180]]};
        a4={N[-ymax Sin[theta Pi/180]],N[ymax Cos[theta Pi/180]]};
        dy=N[a3[[2]]/ny];
        i1=Floor[(a4[[2]])/dy]+1;
        i2=Floor[(a2[[2]])/dy]+1;
        For[i=1,i<=i1-1,i=i+1,
              normymat[[theta+46,i]]+=l4yl1y[(i-1) dy,i dy,theta];
```

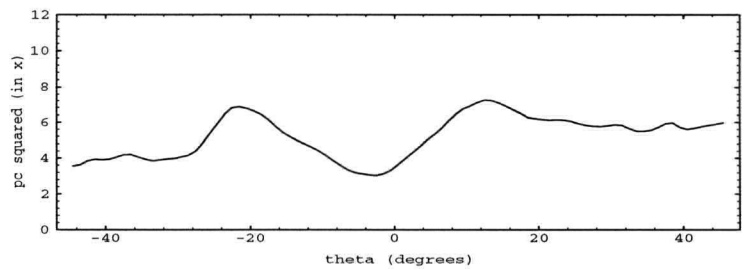
```
            normymat[[-theta+46,i]]+=l4yl1y[(i-1) dy,i dy,theta];
        ];
        For[i=i1,i<=i1+0.5,i=i+1,
            normymat[[theta+46,i]]+=l4yl1y[(i-1) dy,a4[[2]],theta];
            normymat[[theta+46,i]]+=l3yl1y[a4[[2]],i dy,theta];
            normymat[[-theta+46,i]]+=l4yl1y[(i-1) dy,a4[[2]],theta];
            normymat[[-theta+46,i]]+=l3yl1y[a4[[2]],i dy,theta];
        ];
        For[i=i1+1,i<=i2-1,i=i+1,
            normymat[[theta+46,i]]+=l3yl1y[(i-1) dy,i dy,theta];
            normymat[[-theta+46,i]]+=l3yl1y[(i-1) dy,i dy,theta];
        ];

        For[i=i2,i<=i2+0.5,i=i+1,
            normymat[[theta+46,i]]+=l3yl1y[(i-1) dy,a2[[2]],theta];
            normymat[[theta+46,i]]+=l3yl2y[a2[[2]],i dy,theta];
            normymat[[-theta+46,i]]+=l3yl1y[(i-1) dy,a2[[2]],theta];
            normymat[[-theta+46,i]]+=l3yl2y[a2[[2]],i dy,theta];
        ];
        For[i=i2+1,i<=ny-1,i=i+1,
            normymat[[theta+46,i]]+=l3yl2y[(i-1) dy,i dy,theta];
            normymat[[-theta+46,i]]+=l3yl2y[(i-1) dy,i dy,theta];
        ];
        For[i=ny,i<=ny+0.5,i=i+1,
            normymat[[theta+46,i]]+=l3yl2y[(i-1) dy,a3[[2]],theta];
            normymat[[-theta+46,i]]+=l3yl2y[(i-1) dy,a3[[2]],theta];
        ];
];

xresmat=Table[0,{91},{nx}];
For[i=1,i<=91,i=i+1,
        For[j=1,j<=nx,j=j+1,xresmat[[i,j]]=N[xmat[[i,j]]/normxmat[[i,j]]]];
];
xresmax=Max[xresmat]

0.634253

f[x_]:=Which[      0 == x , 1.0,
              0.0 < x <= 0.05, 0.95,
              0.05 < x <= 0.1, 0.9,
              0.1 < x <= 0.15, 0.85,
              0.15 < x <= 0.2, 0.8,
              0.2 < x <= 0.25, 0.75,
              0.25 < x <= 0.3, 0.7,
              0.3 < x <= 0.35, 0.65,
              0.35 < x <= 0.4, 0.6,
              0.4 < x <= 0.45, 0.55,
              0.45 < x <= 0.5, 0.5,
              0.5 < x <= 0.55, 0.45,
              0.55 < x <= 0.6, 0.4,
              0.6 < x <= 0.65, 0.35,
              0.65 < x <= 0.7, 0.3,
              0.7 < x <= 0.75, 0.25,
              0.75 < x <= 0.8, 0.2,
              0.8 < x <= 0.85, 0.15,
              0.85 < x <= 0.9, 0.1,
              0.9 < x <= 0.95, 0.05,
              0.95 < x <= 1.0, 0.0];

xtest=Table[0,{91},{nx}];
For[i=1,i<=91,i=i+1,
        For[j=1,j<=nx,j=j+1,xtest[[i,j]]=N[xresmat[[i,j]]/xresmax]];
```

```
]
testx=Map[f,xtest,{2}];
```

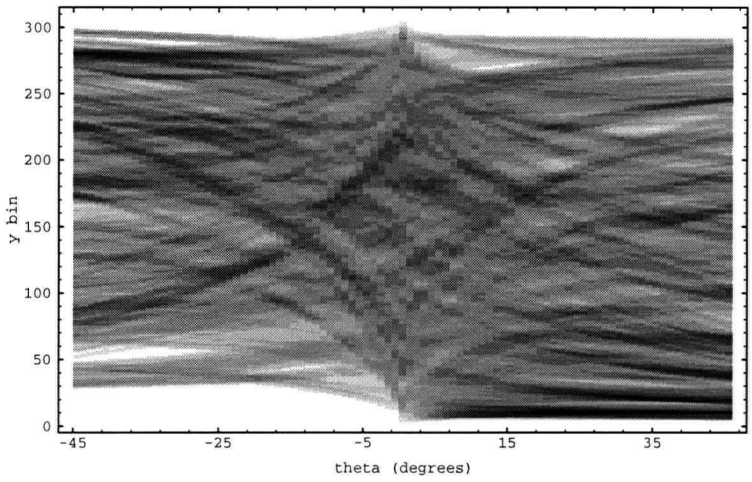
```
Show[DensityGraphics[Transpose[testx]], Mesh->False, AspectRatio->0.618,Frame->True,
      FrameLabel->{"theta (degrees)","x bin"}, PlotRange->{{-1,92},{-5,nx+5},{0.0,1.0}}]
```



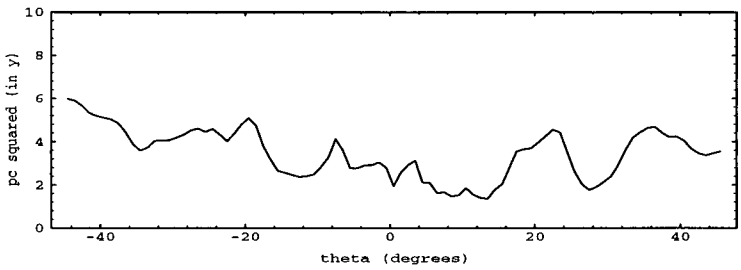

```
xmidpoints=Table[-0.5+j-45,{j,91}];
xchitest=Table[0,{91}];
xoffset=10;
For[i=1,i<=91,i=i+1,
      For[j=1+xoffset,j<=nx-xoffset,j=j+1,
            xchitest[[i]]+=(xresmat[[i,j]]-sa/(xmax ymax))^2/(sa/(xmax ymax))];
];
ListPlot[Transpose[{xmidpoints,xchitest}], PlotRange->{5,20}, Frame->True,
      FrameLabel->{"theta (degrees)","pc squared (in x)"}, AxesOrigin->{-50,0},
      PlotJoined -> True, PlotStyle->AbsoluteThickness[0.5],AspectRatio->0.4]
```



```
yresmat=Table[0,{91},{ny}];
For[i=1,i<=91,i=i+1,
       For[j=1,j<=ny,j=j+1,
              yresmat[[i,j]]=N[ymat[[i,j]]/normymat[[i,j]]]];
];
yresmax=Max[yresmat]
0.671906

ytest=Table[0,{91},{ny}];
For[i=1,i<=91,i=i+1,
       For[j=1,j<=ny,j=j+1,ytest[[i,j]]=N[yresmat[[i,j]]/yresmax]];
]
testy=Map[f,ytest,{2}];
Show[DensityGraphics[Transpose[testy]],Mesh->False, AspectRatio->0.618, Frame->True,
       FrameLabel->{"theta (degrees)","y bin"}, PlotRange->{{-1,92},{-5,ny+5},{0.0,1.0}}]
```




```
ymidpoints=Table[-0.5+j-45,{j,91}];
ychitest=Table[0,{91}];
yoffset=10;
For[i=1,i<=91,i=i+1,
       For[j=1+yoffset,j<=ny-yoffset,j=j+1,
              ychitest[[i]]+=(yresmat[[i,j]]-sa/(xmax ymax))^2/(sa/(xmax ymax))];
];
ListPlot[Transpose[{ymidpoints,ychitest}],PlotRange->{0,20}, Frame->True,
       FrameLabel->{"theta (degrees)","pc squared (in y)"}, AxesOrigin->{-50,0},
       PlotJoined -> True, PlotStyle->AbsoluteThickness[0.5],AspectRatio->0.4]
```


10
8
6
4
2
0
pc squared (in y)
-40
-20
0
20
40
theta (degrees)

Appendix C

Computer Code for Riverbed

```
#include <iostream.h>
#include <fstream.hpp>                  //for file class
#include <math.h>
#include <stdio.h>
#include <stdlib.h>                     //for exit
#define Width 500.0                     //width of reach in cm
#define Length 1500.0                   //length of reach in cm
#define SeedFraction 0.20               //fraction of riverbed covered by seed stones
                                        //belonging to D_95.
#define RandomClasts 3001               //number of random entrainments, main counter
                                        //for program
#define Range 10.0                      //for angular range of trajectories
#define Lambda 4.0                      //for lognormal distribution of radii
#define Epsilon 0.0                     //for lognormal distribution of radii
#define D_75 10.0                       //Harris Creek minimum riffle surface grain size, cm.
#define D_95 20.0                       //grain size of the 95th coarsest percentile
                                        //in Harris Creek in cm.
#define D_100 50.0                      //maximum grain size in cm allowed in Harris Creek.
#define D_50sub 10                      //mean diameter of subsurface clasts in cm.
#define L_D50surf 236                   //mean distance of movement in cm for D_50 surface
                                        //material.
#define FULCRUM 60.0                    //skim/stop criterion in entrain() in degrees
                                        //either side of downstream.
#define Criterion 15.0                  //criterion for clast entrainment.

// RIVERBED 7.0
// Riverbed 7.0 models coarse clast transport in a stream bed with the rebound rule along with
// an entrainment probability and travel distance for each stone that varies inversely
// proportionally to the radius. A stone is introduced into the simulated reach from a random
// position on the upstream margin or is randomly entrained from the seeded stone population
// produced in the initial conditions. It travels downstream with a randomly chosen trajectory
// and may encounter obstacles. Upon impact with an obstacle, the stone's rebound angle is
// calculated by the rebound rule to be equal to the incident angle measured about the normal
// to tangent of the impact site. It stops if rebound angle is greater than FULCRUM degrees;
// otherwise the stone moves along the new trajectory to further encounters. If a stone exits
// the reach or travels through the bed without encountering obstacles, a new stone is
// introduced upstream to take its place. Riverbed 7.0 also uses different radii drawn from
// a truncated lognormal distribution, and an anti-clogging rule (Vanish) for the upstream
// margin. The river bed width and length are in cm. Detailed notes on the algorithm appear at
// the beginning of crucial loops and for each function.
// Written August 5-13, 1996 by Selina Tribe.
```

```
     struct NodeType {                    //dynamic memory allocation to make a linked list
          float                    x;     //x position
          float                    y;     //y position
          float                    r;     //clast radius
          float                    v;     //clast trajectory
          float                    l;     //maximum travel length
          NodeType*       link;           //pointer to next struct
     };
     typedef NodeType* NodePtr;
     NodePtr          head;               //head of list
     NodePtr      endPtr;                 //points to end of list
     long seed=-34581;                    //initial seed for ran1
     float SeedNumber;                    //number of stones in the bed at initial conditions.
     long *idum;                          //for random number generator
     float tr,g;                          //results of ran1() and gasdev().
     int totalClasts;                     //counter: total number of stones in the reach,
                                          //= SeedNumber + input - throughput.
     int throughput;                      //counter: number of stones passing through the
                                          //downstream boundary of reach.
     int adjust;                          //counter: how many times new positions were adjusted
                                          //because they fell out of the reach.
     int input;                           //counter: number of new stones introduced into the
                                          //active bed.
     int rebound;                         //counter: total number of rebounds on reach margins
     int vanishedRock;                    //counter for number of repicks of new clast to avoid
                                          //clogging the upstream margin.
     int reversal;                        //counter for number of trajectory reversals
                                          //performed on bed margins.
     int deleteClast;                     //flag signifying a re-entrained clast.
     int stop;                            //flag signifying a clast has been intercepted and
                                          //stopped, to get out of entrain().
     int vanish;                          //flag signalling that obstacle picked is too close
                                          //to upstream margin so repick new stone position.
     int Number;                          //counter: number of nodes in list.
     int exitClast;                       //flag:inject new stone since one has passed through.
     float xtemprc, ytemprc, rtemprc;     //coordinates, radius of randomly entrained stone.
     float vtemprc, ltemprc;              //angle, maximum distance of randomly entrained stone.
     float D;                             //D=2*radius/D_50sub.
     float L;                             //L= l/L_D50surf.
     streampos savedPos3=0;               //for file "values.dat"
     float ran1(long *idum);
     float gasdev(long *idum);
     void seedbed();
     void entrain();
     void fileSeedBed();
     void filebed ();

void main ()
{
     int rc,i;                            //counters
     NodePtr          delPtr;             //pointer to randomly entrained node to be deleted
                                          //from list
     NodePtr          prevPtr;            //pointer to node before entrainPtr allowing for
                                          //easy deletion
     int Position;                        //position in list chosen to be deleted=stone chosen
                                          //to be re-entrained.
     int moveable;                        //flag indicating whether stone has fulfilled
                                          //entrainment criterion.

     // Initialize riverbed by seeding it with clast. Set counters and flags to zero.
     idum=&seed;
```

```
seedbed();
cout<<"bed seeded."<<"\n";
fileSeedBed();
cout<<"bed filed"<<"\n";
throughput = 0;
input = 0;
adjust = 0;
rebound = 0;
reversal = 0;
stop = 0;
deleteClast = 0;
vanishedRock = 0;
vanish = 0;
moveable = 0;

// random entrain loop
// 1. This procedure will randomly chose a stone located at some position in the riverbed
// to be entrained into the flow. Each stone picked randomly will have a probability of
// entrainment inversely proportional to size: a large stone is less likely to be picked.
// The stone is chosen from the existing list by picking a uniform random number ranging
// from 1 to Number, the length of the list. A trajectory is picked from a normal
// distribution with the same variance as used for the new stones. Any node in the list
// can be entrained: if the first node happens to be chosen then the head pointer and all
// other pointers, which are always repositioned to the list head, are shifted one down.
//    The pointer delPtr is positioned at the node in the list representing the chosen
// stone then x, y, r, travel distance, v are written to temporary variables (xtemprc,
// ytemprc, rtemprc, ltemprc, vtemprc). The flag deleteClast is set to 1. The pointer
// prevPtr travels down the list to the node immediately prior to delPtr. The pointers
// point to new positions and delPtr is deleted resulting in the list being one node
// shorter. Xtemprc, ytemprc, rtemprc, vtemprc are fed into the entrain() function to
// compute where the stone ends up.
for(rc=1;rc<RandomClasts;rc++) {              //begin rc loop
     fstream file("values.dat");
     if (!file) {
          cerr<<"failed to open file 'values-nc'\n";
          exit(EXIT_FAILURE);
     }
     while (moveable == 0) {
          file.seekp(savedPos3);
          tr = ran1(idum);
          Position = int(tr*Number) ;         //chosen clast is a random number scaled
                                              //by list size
          cout<<"rc= "<<rc<<" tr= "<<tr<<" Position= "<<Position<<"\n";
          file<<"rc= "<<rc<<" tr= "<<tr<<" Position= "<<Position<<"\n";
          delPtr  = head;
          if (Position > 1) {
               for (i=2; i<(Position+1); i++) {
                    delPtr = delPtr->link;
                    if (delPtr == NULL) {
                         cout<<"ERROR : delPtr POINTS TO NULL"<<"\n";
                         file<<"ERROR :  delPtr POINTS TO NULL"<<"\n";
                    }
               }
          }

          tr=ran1(idum);
          if ((delPtr->r*2*tr) < Criterion ) {        //Probability of entrainment: larger
                                                      //stones less likely to be entrained.
               moveable = 1;
               xtemprc = delPtr->x;                   //coordinates are copied from list
                                                      //into the bedload.
               ytemprc = delPtr->y;
```

```
                rtemprc = delPtr->r;
                ltemprc = delPtr->l;
                g = gasdev(idum);
                vtemprc = g*Range;                          //get trajectory from gaussian dist.
                if (ytemprc <= 0.0 && vtemprc > 0.0) {
                     vtemprc = - vtemprc;
                }
                if (ytemprc >= Width && vtemprc < 0.0) {
                     vtemprc = - vtemprc;
                }
                deleteClast = 1;
                cout<<"xtemprc= "<<xtemprc<<" ytemprc= "<<ytemprc<<" ltemprc= "<<ltemprc
                    <<" rtemprc= "<<rtemprc<<" vtemprc= "<<vtemprc<<"\n";
                file<<"xtemprc= "<<xtemprc<<" ytemprc= "<<ytemprc<<" ltemprc= "<<ltemprc
                    <<" rtemprc= "<<rtemprc<<" vtemprc= "<<vtemprc<<"\n";
                if (delPtr == head) {
                     head = head->link;               //reposition head and all other
                                                      //pointers one node down.
                } else {
                     prevPtr = head;
                     while (prevPtr->link != delPtr) {
                          prevPtr = prevPtr->link;
                     }
                     prevPtr->link = delPtr->link;
                }
                if (prevPtr->link == NULL) {
                     endPtr = prevPtr;
                }
                delete delPtr;
                Number --;
                cout<<"Randomly entrain # "<<Position<<"  "<<Number<<" nodes left in list"
                    <<" i= "<<i<<"\n";
                file<<"Randomly entrain # "<<Position<<"  "<<Number<<" nodes left in list"
           <<" i= "<<i<<"\n";
                savedPos3=file.tellg();
                file.close();
                entrain();
           }
      }
      moveable = 0;
      stop = 0;
      if(exitClast == 1) {
           exitClast = 0;
           entrain();
           stop = 0;
      }
}                                                  // end rc loop

totalClasts = SeedNumber + input - throughput;
filebed();
fstream file("values.dat");
if (!file) {
     cerr<<"failed to open file 'init'\n";
     exit(EXIT_FAILURE);
}
file.seekp(savedPos3);
cout<<" rc= "<<rc<<"total clasts= "<<totalClasts<<"\n";
file<<" rc= "<<rc<<"total clasts= "<<totalClasts<<"\n";
cout<<" input = "<<input<<" throughput= "<<throughput<<"\n";
file<<" input = "<<input<<" throughput= "<<throughput<<"\n";
cout<<" adjusts= "<<adjust<<" rebounds= "<<rebound<<" vanished= "<<vanishedRock
    <<" reversals= "<<reversal<<"\n";
```

```
     file<<" adjusts= "<<adjust<<" rebounds= "<<rebound<<"\n";
     savedPos3=file.tellg();
     file.close();
}                                                    // end main
//-------------------------------------------------------------------------------------
void entrain()
// This function projects a stone along a trajectory and determines which obstacle it will
// impact, how it will rebound and where it will eventually rest. Stop must be zero to enter
// this function and stop = 1 to exit the function. Detailed notes are at the beginning of each
// numbered section. The stone's initial position is xtemp and ytemp, its radius rtemp, and the
// trajectory is vtemp in degrees where negative vtemp sends a stone counterclockwise down the
// river and positive clockwise. A stone does not leave this function until it is stopped at
// which point its coordinates are determined and added to the end of the list. Values are
// written to file values.dat. An anti-clogging rule has been written to avoid clogging up of
// the upstream margin. When an obstacle is found, if it has an x position less than the radius
// of the incoming clast, then the clast vanishes and vanishedRock counter is incremented.
{
     NodePtr          tempPtr;                          //pointer to find obstacle clast
     NodePtr      obstaclePtr;                          //pointer to obstacle clast stone hits
     float xtemp, ytemp, rtemp, vtemp, ltemp, mileage;  //temporary coordinates, radius,
                                                        //angle, travel length, distance
                                                        //travelled for entrained clast.
     float oldDist, touchDistance, ratio;               //vector distance between new clast
                                                        //and potential obstacle
     float a, b, hy, k, obx, oby;                       //vectors in the encounter triangles
     float A, B, C, G, M, Ob, Q, Z;                     //angles in the encounter triangles
     float hnew, xnew, ynew, alpha, xlag, ylag;         //vectors for position of new clast
                                                        //at encounter point
     float xfree, yfree, hfree, xN, yN, lN, xS, yS, lS; //for boundary conditions
     float xtemp2, ytemp2, vtemp2, ltemp2;              //temporary coordinates and angle for
                                                        //skimming clast.
     int reboundedClastN, reboundedClastS, shortmargin; //flags.
     int reboundsN, reboundsS;                          //counters: number of rebounds along
                                                        //north and south margin.
     int obstacle, skim, Case;                          //flag: obstacle found; skimming stone;
                                                  //orientation of encounter triangle.
     int checkedNode ;                                  //counts number of list nodes checked.

     reboundsN = 0;
     reboundsS = 0;
     reboundedClastN = 0;
     reboundedClastS = 0;
     shortmargin = 0;
     skim = 0;
     while(stop == 0) {                              //stop condition
          fstream file("values.dat");
          if (!file) {
               cerr<<"failed to open file 'values-entrain() .'\n";
               exit(EXIT_FAILURE);
          }
          file.seekp(savedPos3);
          // 1. The entrained stone can be a randomly entrained clast (when deleteClast = 1)
          // a clast rebounding off the northern and southern rigid boundaries, a clast which
          // has just skimmed an obstacle and continues to move downstream, a clast repicked
          // since the previous one would have clogged the upstream border, or a new clast
          // introduced on upstream margin in which case xtemp = 0, ytemp is drawn from a
          // uniform distribution and vtemp is drawn from a gaussian distribution.
          if (reboundedClastN == 1) {               //begin margin&skim loop
               ytemp = yN;
               xtemp = xN;
               vtemp = alpha;
               ltemp = lN;
```

```
     reboundedClastN = 0;
} else if (reboundedClastS == 1) {
     ytemp = yS;
     xtemp = xS;
     vtemp = alpha;
     ltemp = lS;
     reboundedClastS = 0;
} else if (skim ==1) {
     ytemp = ytemp2;
     xtemp = xtemp2;
     vtemp = vtemp2;
     ltemp = ltemp2;
     skim = 0;
     if (vtemp > 0 && ytemp == 0) {       //to avoid errors
          vtemp = -vtemp;
          reversal ++;
     } else if (vtemp < 0 && ytemp == Width) {
          vtemp = -vtemp;
          reversal ++;
     }
} else if (deleteClast ==1) {
     ytemp = ytemprc;
     xtemp = xtemprc;
     rtemp = rtemprc;
     vtemp = vtemprc;
     ltemp = ltemprc;
     deleteClast = 0;
} else if (vanish == 1){                  //repick to avoid clogging upstream border.
     tr=ran1(idum);
     ytemp = tr*Width;
     xtemp = 0.0;
     g = gasdev(idum);
     rtemp = Lambda*exp(g) + Epsilon;     //radii drawn from lognormal distribution.
     while (rtemp < (D_75*0.5) || rtemp > (D_100*0.5)) {
          g = gasdev(idum);
          rtemp = Lambda*exp(g) + Epsilon;
     }
     g = gasdev(idum);
     vtemp = g*Range;
     D = 2*rtemp/10;
     L =1.77*(pow(1-log10(D),1.35));
     ltemp = L*L_D50surf;
     file<<"\t"<<tr<<"\t"<<g<<"\n";
     cout<<"\t"<<tr<<"\t"<<g<<"\n";
     vanish = 0;
     vanishedRock ++;
} else {                                  //inject a new clast at upstream margin.
     tr=ran1(idum);
     ytemp = tr*Width;
     xtemp = 0.0;
     g = gasdev(idum);
     rtemp = Lambda*exp(g) + Epsilon;
     while (rtemp < (D_75*0.5) || rtemp > (D_100*0.5)) {
          g = gasdev(idum);
          rtemp = Lambda*exp(g) + Epsilon;
     }
     D = 2*rtemp/10;
     L =1.77*(pow(1-log10(D),1.35));
     ltemp = L*L_D50surf;
     g = gasdev(idum);
     vtemp = g*Range;
     file<<"\t"<<tr<<"\t"<<g<<"\n";
```

```
     cout<<"\t"<<tr<<"\t"<<g<<"\n";
     input ++;
}                                                  //end margin&skim loop
tempPtr = head;
obstaclePtr = head;
obstacle = 0;
vanish = 0;
checkedNode = 0;
oldDist = 2000;

// 2. Starting at the head of the list, compare the stone's initial position and
// trajectory with each member of the list positioned downstream from the obstacle
// using tempPtr. If the stone encounters an obstacle and if the distance to this
// obstacle is less than what is was previously then the pointer obstaclePtr is
// positioned at that point in the list and the obstacle flag is set to one.
while(tempPtr != NULL && vanish == 0) {      //begin while list
     if ((tempPtr->x) > xtemp) {             //begin forward loop,
                                             //only looks at stones in front of it
          oby = sqrt((tempPtr->y-ytemp)*(tempPtr->y-ytemp));
          hy = sqrt((tempPtr->x-xtemp)*(tempPtr->x-xtemp) +
               (tempPtr->y-ytemp)*(tempPtr->y-ytemp));
          Ob = asin(oby/hy)*57.3;
          if (tempPtr->y > ytemp) {
               Ob = - Ob;
          }
          touchDistance = rtemp + tempPtr->r;
          Q = (atan(touchDistance/hy))*57.3;
          if ((fabs(vtemp - Ob) < Q) &&  hy < oldDist) {//if new clast will touch the
                                                       //obstacle and if distance
                                                       //gets closer
               obstaclePtr = tempPtr;
               oldDist = hy;
               obstacle = 1;
               tempPtr = tempPtr->link;
               if (obstaclePtr->x < rtemp) {
                    obstacle = 0;
                    vanish = 1;     //set flag to redraw to avoid upstream clog
                    cout<<"Obstacle too close to upstream margin so vanish."<<"\n";
                    file<<"Obstacle too close to upstream margin so vanish."<<"\n";
               }
          } else {
               tempPtr = tempPtr->link;
          }
     } else {
          tempPtr = tempPtr->link;
     }                       // end forward loop
     checkedNode ++;
}                            // end while list
tempPtr = head;         // reposition to head of list to avoid deletion difficulties.

// 3. The entire list has been checked against the trajectory of the entrained stone
// and no obstacle has been found (obstacle=0). Therefore we must check to see if the
// stone will travel its maximum distance before hitting the north or south boundaries
// or pass through the downstream margin of the bed. If the stone passes through, it is
// counted as throughput, stop is set to 1, and the entrain function is exited. If it
// hits the north or south boundaries the stone is given marginal coordinates xN, yN or
// xS, yS and rebound flags are set to 1. These coordinates are sent back to part 1 where
// the list is checked for potential obstacles to the rebounder.
if (obstacle == 0 && vanish == 0) {     //begin obstacle loop
     cout<<"No obstacle, boundary conditions"<<"\n";
     file<<"No obstacle, boundary conditions"<<"\n";
     cout<<"xtemp= "<<xtemp<<" ytemp= "<<ytemp<<" ltemp="<<ltemp<<" rtemp= "
```

```
        <<rtemp<<" vtemp= "<<vtemp<<"\n";
    file<<"xtemp= "<<xtemp<<" ytemp= "<<ytemp<<" ltemp="<<ltemp<<" rtemp= "
        <<rtemp<<" vtemp= "<<vtemp<<"\n";
    if (vtemp <= 0) {                    //begin rebound loop
         yfree = Width - ytemp;
         xfree = xtemp + yfree/(tan(-vtemp*0.01745));
         hfree = yfree/(sin(-vtemp*0.01745));
         if (ltemp <= hfree) {
              shortmargin = 1;
              stop = 1;
              xlag = ltemp*cos(-vtemp*0.01745);
              ylag =  ltemp*sin(-vtemp*0.01745);
         } else if ((xfree > Length) && (ltemp > hfree)) {
              throughput ++;
              stop = 1;                  //to get out of stop condition
              exitClast = 1;             //flag to get a new stone injected into bed.
              cout<<"throughput"<<"\n";
              file<<"throughput"<<"\n"<<"\n";
         } else {
              yN = Width;
              xN = xtemp + xfree;
              lN = ltemp - hfree;
              reboundedClastN = 1;
              reboundsN ++;
              rebound ++;
              alpha = 0 - vtemp;
              cout<<"N rebounder"<<"\n";
              file<<"N rebounder"<<"\n";
         }
    } else if (vtemp > 0) {
         yfree = ytemp;
         xfree = xtemp + yfree/(tan(vtemp*0.01745));
         hfree = yfree/(sin(vtemp*0.01745));
         if (ltemp < hfree) {
               shortmargin = 1;
               stop = 1;
              xlag =  ltemp*cos(vtemp*0.01745);
              ylag =  ltemp*sin(vtemp*0.01745);
         } else if ((xfree > Length) && (ltemp > hfree)) {
              throughput ++;
              stop = 1;
              exitClast = 1;
              cout<<"throughput"<<"\n";
              file<<"throughput"<<"\n";
         } else {
              yS = 0.0;
              xS = xtemp + xfree;
              lS = ltemp - hfree;
              reboundedClastS = 1;
              reboundsS ++;
              rebound ++;
              alpha = 0 - vtemp;
              cout<<"S rebounder"<<"\n";
              file<<"S rebounder"<<"\n";
         }
    } else {
         cout<<"ERROR IN REBOUND LOOP"<<"\n";
         file<<"ERROR IN REBOUND LOOP"<<"\n";
    }                                          //end rebound loop
} else if (obstacle ==1 && vanish == 0) {      //continue obstacle loop

    // 4. The entire list has been checked and an obstacle has been found.
```

```
// Trigonometry on the triangles formed by the stone and the obstacle give
// parameters which will be used in part 5 to get new coordinates at point of
// contact. Capital letters are angles always converted to degrees and small
// letters are lengths of triangle sides. Case keeps track of which set of
// triangles is being calculated. Reference diagram given in figure 15 of
 // thesis text.
obx = sqrt((obstaclePtr->x-xtemp)*(obstaclePtr->x-xtemp));
oby = sqrt((obstaclePtr->y-ytemp)*(obstaclePtr->y-ytemp));
cout<<"xtemp= "<<xtemp<<" ytemp= "<<ytemp<<" ltemp= "<<ltemp<<" rtemp= "
    <<rtemp<<" vtemp= "<<vtemp<<"\n";
cout<<"obstx= "<<obstaclePtr->x<<" obsty= "<<obstaclePtr->y<<" obrad= "
    <<obstaclePtr->r<<" obx= "<<obx<<" oby= " <<oby<<"\n";
file<<"xtemp= "<<xtemp<<" ytemp= "<<ytemp<<" ltemp= "<<ltemp<<" rtemp= "
    <<rtemp<<" vtemp= "<<vtemp<<"\n";
file<<"obstx= "<<obstaclePtr->x<<" obsty= "<<obstaclePtr->y<<" obrad= "
    <<obstaclePtr->r<<" obx= "<<obx<<" oby= " <<oby<<"\n";
hy = sqrt((obstaclePtr->x-xtemp)*(obstaclePtr->x-xtemp) +
     (obstaclePtr->y-ytemp)*(obstaclePtr->y-ytemp));
Ob = asin(oby/hy)*57.3;
touchDistance = rtemp + obstaclePtr->r;
if (obstaclePtr->y > ytemp) {
     Ob = - Ob;
}
Q = (atan(touchDistance/hy))*57.3;
cout<<"hy= "<<hy<<" Ob= "<<Ob<<" Q= "<<Q<<" touchDist= "<<touchDistance<<"\n";
file<<"hy= "<<hy<<" Ob= "<<Ob<<" Q= "<<Q<<" touchDist= "<<touchDistance<<"\n";
Case = 0;
if ((vtemp<0.0 && Ob<0.0) && fabs(vtemp) < fabs(Ob)) {//start case loop, case 1
     C = 90.0 + Ob;
     Z = - Ob + vtemp;
     b = oby - obx*tan(-vtemp*0.017452);
     Case = 1;
} else if ((vtemp<0.0 && Ob<0.0) && fabs(vtemp) > fabs(Ob)) { //case 2
     C = 90.0 - Ob;
     Z = - vtemp + Ob;
     b = obx*tan(-vtemp*0.017452) - oby;
     Case = 2;
} else if (vtemp>0.0 && Ob>0.0 && vtemp<Ob) {                  //case 3
     C = 90.0 - Ob;
     Z = Ob - vtemp;
     b = oby - obx*tan(vtemp*0.017452);
     Case = 3;
} else if (vtemp>0.0 && Ob>0.0 && vtemp>Ob) {                  //case 4
     C = 90.0 + Ob;
     Z = vtemp - Ob;
     b = obx*tan(vtemp*0.017452) - oby;
     Case = 4;
} else if (vtemp>0.0 && Ob<= 0.0)  {                           //cases 5
     C = 90.0 + Ob;
     Z = - Ob + vtemp;
     b = oby + obx*tan(vtemp*0.017452);
     Case = 5;
} else if (vtemp<0.0 && Ob>= 0.0) {                            //case 6
     C = 90.0 - Ob;
     Z =  Ob - vtemp;
     b = oby + obx*tan(-vtemp*0.017452);
     Case = 6;
} else {
     cout<<"ERROR IN CASE LOOP"<<"\n";
     file<<"ERROR IN CASE LOOP"<<"\n";
}                                                              //end case loop
cout<<"C= "<<C<<" Z= "<<Z<<" b= "<<b<<" case= "<<Case<<"\n";
```

```
file<<"C= "<<C<<" Z= "<<Z<<" b= "<<b<<" case= "<<Case<<"\n";
G = 180.0 - Z - C;
B = (asin((sin(G*0.017452))*b/touchDistance))*57.3;
A = 180.0 - B - G;
a = touchDistance*sin(A*0.017452)/sin(G*0.017452);
k = hy*sin(C*0.017452)/sin(G*0.017452);
cout<<"G= "<<G<<" B= "<<B<<" A= "<<A<<" a= "<<a<<" k= "<<k<<"\n";
file<<"G= "<<G<<" B= "<<B<<" A= "<<A<<" a= "<<a<<" k= "<<k<<"\n";
hnew = k - a;
obstaclePtr = head;     //reposition to head of list to avoid deletion problems

// 5. Using the triangles solved above, the coordinates of the center of the
// stone are determined at its point of contact with the obstacle. If the stone
// happens to end up slightly out of the digital reach then it is nudged back to
// the zero border, a message signals this adjustment and adjust is incremented.
// At the point of contact, the stone will rebound at an angle equal to the
// incident angle as measured from the normal to the encounter surface. If the
// rebound angle is greater than FULCRUM degrees the stone stop's and is
// deposited at the impact position. If the angle is less than FULCRUM then the
// stone skims the obstacle, stop flag is zero so entrain() cannot be exited.
// The new coordinates, trajectory, and remaining travel distance, xtemp2, ytemp2,
// vtemp2 and ltemp are sent back to part 1.
if ((Case == 1) || (Case == 2)) {      //begin skim/stop loop
     ynew = hnew*sin(-vtemp*0.017452);
     xnew = hnew*cos(-vtemp*0.017452);
     xtemp2 = xtemp + xnew;
     ytemp2 = ytemp + ynew;
     if (Case ==1) {
          vtemp2 = 180 - 2*B + vtemp;
     } else {
          vtemp2 = vtemp -180 + 2*B;
     }
} else if ((Case == 3) || (Case == 4)) {
     ynew = hnew*sin(vtemp*0.017452);
     xnew = hnew*cos(vtemp*0.017452);
     xtemp2 = xtemp + xnew;
     ytemp2 = ytemp - ynew;
     if (Case == 3) {
          vtemp2 = -180 + 2*B + vtemp;
     } else {
          vtemp2 = 180 -2*B + vtemp;
     }
} else if ((Case == 5) || (Case == 6)) {
     if (Case == 5) {
          ynew = hnew*sin(vtemp*0.017452);
          xnew = hnew*cos(vtemp*0.017452);
          xtemp2 = xtemp + xnew;
          ytemp2 = ytemp - ynew;
          vtemp2 = 180 - 2*B + vtemp;
     } else {
          ynew = hnew*sin(-vtemp*0.017452);
          xnew = hnew*cos(-vtemp*0.017452);
          xtemp2 = xtemp + xnew;
          ytemp2 = ytemp + ynew;
          vtemp2 = vtemp -180 + 2*B;
     }
} else {
     cout<<"ERROR IN OUTERSKIM/STOP LOOP"<<"\n";
     file<<"ERROR IN OUTER SKIM/STOP LOOP"<<"\n";
}                                                  //end skim/stop loop
if (vtemp2 < FULCRUM && vtemp2 > -FULCRUM) {  //skim/stop assignment
     skim = 1;
```

```
} else if (vtemp2 > FULCRUM || vtemp2 < -FULCRUM) {
     stop = 1;
} else {
     cout<<"ERROR IN SKIM/STOP ASSIGNMENT"<<"\n";
     file<<"ERROR IN SKIM/STOP ASSIGNMENT"<<"\n";
}
if (xtemp2 < 0.0) {          //adjustments if stone is slightly out of bounds
     xtemp2 = 0.0;
     adjust ++;
     cout<<"x adjustment made to 0."<<"\n";
     file<<"x adjustment made to 0."<<"\n";
}
if (ytemp2 < 0.0) {
     ytemp2 = 0.0;
     adjust ++;
     cout<<"y adjustment made to 0."<<"\n";
     file<<"y adjustment made to 0."<<"\n";
} else if (ytemp2 > Width) {
     ytemp2 = Width;
     adjust ++;
     cout<<"y adjustment made to width."<<"\n";
     file<<"y adjustment made to width."<<"\n";
}

ltemp2 = ltemp - (hy - touchDistance); //re-evaluate allowable travel distance

cout<<"hnew= "<<hnew<<" ynew= "<<ynew<<" xnew= "<<xnew<<" xtemp2= "<<xtemp2
    <<" ytemp2= "<<ytemp2<<" ltemp2= "<<ltemp2<<" vtemp2= "<<vtemp2<<"\n";
     if (skim ==1) { cout<<"  skim"<<"\n"; } else {cout<<"  stop"<<"\n"; }
file<<"hnew= "<<hnew<<" ynew= "<<ynew<<" xnew= "<<xnew<<" xtemp2= "<<xtemp2
    <<" ytemp2= "<<ytemp2<<" ltemp2= "<<ltemp2<<" vtemp2= "<<vtemp2<<"\n";
     if (skim ==1) { file<<"  skim"<<"\n"; } else {file<<"  stop"<<"\n"; }

// 6. The stone ends up stopping therefore a node is added to the end of the
// list into which the stone's coordinates are written. Number, representing
// the length of the list, is incremented. The entrain function is completed
// and exited.
if (stop == 1 || ltemp2 < 0.0) {      //begin stop portion of obstacle loop
     endPtr->link = new NodeType;
     endPtr->link->link = NULL;
     endPtr = endPtr->link;
     if (ltemp2 < 0.0 && vtemp < 0.0) {
          endPtr->x = xtemp + ltemp*cos(-vtemp*0.01745);
          endPtr->y = ytemp +ltemp*sin(-vtemp*0.01745);
          stop = 0;
     } else if (ltemp2 < 0.0 && vtemp >= 0.0) {
          endPtr->x = xtemp + ltemp*cos(vtemp*0.01745);
          endPtr->y = ytemp - ltemp*sin(vtemp*0.01745);
          stop = 0;
     } else if (stop ==1) {
          endPtr->x = xtemp2;
          endPtr->y = ytemp2;
     }
     stop = 1;
     endPtr->r = rtemp;
     D = 2*rtemp/10;
     L =1.77*(pow(1-log10(D),1.35));
     endPtr->l  = L*L_D50surf;      //reset ltemp
     endPtr->v = 1.0;
     Number ++;
     cout<<"x= "<<endPtr->x<<" y="<<endPtr->y<<" ltemp="<<endPtr->l<<" r="
         <<endPtr->r<<" v="<<endPtr->v<<"\n";
```

```
                    file<<"x= "<<endPtr->x<<" y="<<endPtr->y<<" ltemp="<<endPtr->l<<" r="
                        <<endPtr->r<<" v="<<endPtr->v<<"\n";
                    cout<<"\n";
                    file<<"\n";
                }                                       //end stop portion of obstacle loop
            }                                           //end obstacle loop
            if (shortmargin == 1) {
        if ((xtemp + xlag)<= Length){       //stopped before hitting boundary
             endPtr->link = new NodeType;
                    endPtr->link->link = NULL;
                    endPtr = endPtr->link;
                    endPtr->x = xtemp + xlag;
    if (vtemp<=0){
      ylag = -ylag;
    }
    endPtr->y = ylag;
                    endPtr->r = rtemp;
                    D = 2*rtemp/10;
                    L =1.77*(pow(1-log10(D),1.35));
                    endPtr->l  = L*L_D50surf;           //reset ltemp
                    endPtr->v = 1.0;
                    stop = 1;                           //to exit entrain()
                    cout<<"x= "<<endPtr->x<<" y="<<endPtr->y<<" ltemp="<<endPtr->l<<" r="
                        <<endPtr->r<<" v="<<endPtr->v<<"\n";
                    file<<"x= "<<endPtr->x<<" y="<<endPtr->y<<" ltemp="<<endPtr->l<<" r="
                        <<endPtr->r<<" v="<<endPtr->v<<"\n";
                    cout<<"\n";
                    file<<"\n";
        } else if ((xtemp + xlag)>Length){      //passed through downstream boundary
    throughput++;
    exitClast=1;
    cout<<"throughput"<<"\n";
    file<<"throughput"<<"\n";
        } else {
    cout<<"error at downstream margin"<<"\n";
    file<<"error at downstream margin"<<"\n";
        }
  }
  savedPos3=file.tellg();
          file.close();
     }                                          //end stop condition
}

//-------------------------------------------------------------------------------------------
void seedbed()
// This function generates a river bed covered with SeedFraction non-touching stones whose
// positions are drawn from a uniform distribution in x and y, whose radii are drawn from a
// lognormal distribution and whose maximum travel distance, l, is calculated.
{
     NodePtr checkPtr;
     int overlap;
     float xtemp, ytemp, rtemp, xdiff, ydiff;
     float root, Distance, touchDistance;

     SeedNumber = int((Width*Length*SeedFraction)/(D_95*0.5*D_95*0.5*3.14159));
     Number = 0;
     head = new NodeType;
     head->link = NULL;
     endPtr = head;                             //positions pointer to top of list
     tr=ran1(idum);                             //fill in first item in list=first clast
     endPtr->x = tr*Length;
     tr=ran1(idum);
```

```
        endPtr->y = tr*Width;
        g = gasdev(idum);
        endPtr->r = Lambda*exp(g) + Epsilon;
        while ( endPtr->r < (D_75*0.5) ||endPtr->r > (D_100*0.5)) {
             g = gasdev(idum);
             endPtr->r = Lambda*exp(g) + Epsilon;
        }
        endPtr->v = 0.0;                              //seeded clast signifier
        D = 2*endPtr->r/10;
        L =1.77*(pow(1-log10(D),1.35));
        endPtr->l  = L*L_D50surf;
        endPtr->link = new NodeType;
        endPtr->link->link=NULL;
        endPtr = endPtr->link;
        Number ++;
        overlap = 0;
        while (Number < SeedNumber) {
             tr=ran1(idum);                          //fill in first item in list=first clast
             xtemp = tr*Length;
             tr=ran1(idum);
             ytemp = tr*Width;
             g = gasdev(idum);
             rtemp = Lambda*exp(g) + Epsilon;
             while (rtemp < (D_75*0.5) || rtemp > (D_100*0.5)) {
                  g = gasdev(idum);
                  rtemp = Lambda*exp(g) + Epsilon;
             }
             checkPtr = head;
             overlap = 0;
             while (checkPtr->link !=NULL) {
                  xdiff = checkPtr->x - xtemp;
                  ydiff = checkPtr->y - ytemp;
                  root = (xdiff*xdiff) + (ydiff*ydiff);
                  Distance = sqrt (root);
                  touchDistance = checkPtr->r + rtemp;
                  if (Distance < touchDistance) {
                       checkPtr = endPtr;
                      overlap = 1;
                  } else if (Distance >= touchDistance) {
                       checkPtr=checkPtr->link;
                  } else {
                       cout<<"ERROR in DISTANCE LOOP"<<"\n";
                  }
             }
             if (overlap == 0) {
                  endPtr->x = xtemp;
                  endPtr->y = ytemp;
                  endPtr->r = rtemp;
                  endPtr->v = 0.0;
                  D = 2*endPtr->r/10;
                  L =1.77*(pow(1-log10(D),1.35));
                  endPtr->l  = L*L_D50surf;
                  Number ++;
             }
             if (overlap == 0 && Number < SeedNumber) {
                  endPtr->link = new NodeType;
                  endPtr->link->link=NULL;
                  endPtr = endPtr->link;
             }
        }
        cout<<"SeedNumber= "<<SeedNumber<<" Number= "<<Number<<"\n";
}
```

```
//-------------------------------------------------------------------------------------------
void fileSeedBed ()
// Prints the initial conditions to a file resulting in a table of x,y coordinates, radii and
// travel length. It opens file (init.dat) and uses the pointer visitPtr to write the contents
// of the list to file. Includes error handling and returns visitPtr to head after use to
// avoid problems when deleting nodes.
{
     NodePtr      visitPtr;
     streampos savedPos1=0;                          //for file "init.dat"
     fstream file("init.dat");
     if (!file) {
          cerr<<"failed to open file 'init'\n";
          exit(EXIT_FAILURE);
     }
     file.seekp(savedPos1);
     visitPtr = head;
     while (visitPtr != NULL) {
          file<<visitPtr->x<<"     "<<visitPtr->y<<" "<<visitPtr->r<<"\n";
          visitPtr = visitPtr->link;
     }
     savedPos1= file.tellg();
     visitPtr = head;
     file.close();
     cout<<SeedNumber<<"\n";
}
//-------------------------------------------------------------------------------------------
void filebed ()
// Prints the final conditions to file resulting in a table of x,y coordinates, radii and travel
// length. It opens file bed.dat and uses visitPtr to visit each node in the list and write
// contents to file. Includes error handling and returns visitPtr to head after use.
{
     NodePtr      visitPtr;
     streampos savedPos2=0;                          //for file "bed.dat"
     fstream file("bed.dat");
     if (!file) {
          cerr<<"failed to open file 'bed'"<<"\n";
          exit(EXIT_FAILURE);
     }
     file.seekp(savedPos2);
     visitPtr = head;
     while (visitPtr != NULL) {
          file<<visitPtr->x<<"     "<<visitPtr->y<<"     "<<visitPtr->r<<"\n";
          visitPtr = visitPtr->link;
     }
     savedPos2=file.tellg();
     visitPtr = head;
     file.close();
     cout<<totalClasts;
}
//-------------------------------------------------------------------------------------------
#define IA 16807
#define IM 2147483647
#define AM (1.0/IM)
#define IQ 127773
#define IR 2836
#define NTAB 32
#define NDIV (1+(IM-1)/NTAB)
#define EPS 1.2e-7
#define RNMX (1.0-EPS)
float ran1(long *idum)
// Random number generator RAN1 from Numerical Recipes in C. Returns a uniform random deviate
// between (0.0,1.0). Call with idum a negative integer to initialize and do not change
```

```
// idum between successive deviates in a sequence. RNMX should approximate the largest
// floating value that is less than 1.
{
     int j;
     long k;
     static long iy=0;
     static long iv[NTAB];
     float temp;
     if (*idum<=0||!iy) {
          if (-(*idum)<1) *idum=1;
          else *idum=-(*idum);
          for (j=NTAB+7;j>=0;j--) {
    k=(*idum)/IQ;
               *idum=IA*(*idum-k*IQ)-IR*k;
               if (*idum,0) *idum+=IM;
               if (j<NTAB) iv[j]=*idum;
          }
          iy=iv[0];
     }
     k=(*idum)/IQ;
     *idum=IA*(*idum-k*IQ)-IR*k;
     if (*idum<0) *idum+=IM;
     j=iy/NDIV;
     iy=iv[j];
     iv[j]=*idum;
     if ((temp=AM*iy)>RNMX) return RNMX;
     else return temp;
     }
#undef IA
#undef IM
#undef AM
#undef IQ
#undef IR
#undef NTAB
#undef NDIV
#undef EPS
#undef RNMX
//-------------------------------------------------------------------------------------------
float gasdev(long *idum)
// Returns normally distributed deviate with zero mean and unit variance using ran1(idum)
// as the source of uniform deviates. From Numerical Recipes in C.
{
     float ran1(long *idum);
     static int iset = 0;
     static float gset ;
     float fac, rsq, v1, v2;
     if (iset == 0) {
          do {          v1=2.0*ran1(idum)-1.0;
                         v2=2.0*ran1(idum)-1.0;
                         rsq=v1*v1+v2*v2;
          }
          while (rsq >= 1.0|| rsq == 0.0);
          fac=sqrt(-2.0*log(rsq)/rsq);
          gset=v1*fac;
          iset=1;
          return v2*fac;
     } else {
          iset=0;
          return gset;
     }
}
```

Bibliography

[1] Ahnert, F., 1994. Modelling the development of non-periglacial sorted nets. *Catena* **23**, 43-63.

[2] Allen, J.R.L., 1983. A simplified cascade model for transverse stone-ribs in gravelly streams. *Proceedings, Royal Society of London A*,**385**, 253-266.

[3] Anderson, R.S., 1990. Eolian ripples as examples of self-organization in geomorphological systems. *Earth-Science Reviews* **29**, 77-96.

[4] Bluck, B.J., 1987. Bedforms and clast size changes in gravel-bed rivers, *in* Richards, K.J. (Ed.), River channels; environment and process. *Special Publication, Institute of British Geographers* **18**, 159-178.

[5] Brayshaw, A.C., 1984. Characteristic and origin of cluster bedforms in coarse-grained alluvial channels, *in* Koster, E.H., and Steel, R.J. (Eds.), Sedimentology of Gravels and Conglomerates *Canadian Society of Petroleum Geologists Memoir* **10**, 77-85.

[6] Brayshaw, A.C., 1985. Bed microtopography and entrainment thresholds in gravel-bed rivers. *Geological Society of America Bulletin* **96**, 218-223.

[7] Brayshaw, A.C., Frostick, L.E., and I. Reid, 1983. The hydrodynamics of particle clusters and sediment entrainment in coarse alluvial channels. *Sedimentology* **30**, 137-143.

[8] Church, M. and M.A. Hassan, 1992. Size and distance of travel of unconstrained clasts on a streambed. *Water Resources Research* **28** (1), 299-303.

[9] Church, M., Wolcott, J.F., and W.K. Fletcher, 1991. A test of equal mobility in fluvial sediment transport: behavior of the sand fraction. *Water Resources Research* **27**, (11), 2941-2951.

[10] Church, M. 1996. Personal communication.

[11] Cressie, N.A.C., 1993. *Statistics for Spatial Data*, Revised edition, John Wiley & Sons, Inc. New York.

[12] Dal Cin, R., 1968. "Pebble Clusters": their origin and utilization in the study of paleocurrents. *Sedimentary Geology* **2**, 233-241.

[13] De Jong, C., 1991. A reappraisal of the significance of obstacle clasts in cluster bedform dispersal. *Earth Surface Processes and Landforms* **16**, 737-744.

[14] De Jong, C. and P. Ergenzinger, 1995. The interrelations between mountain valley form and river-bed arrangement, *in* Hickin, E.J. (Ed.), *River Geomorphology*, John Wiley & Sons.

[15] Gustavson, T.C., 1974. Sedimentation on gravel outwash fans, Malaspina Glacier Foreland, Alaska. *Journal of Sedimentary Petrology* **44**, 374-389.

[16] Hallet, B., 1990. Spatial self-organization in geomorphology: from periodic bedforms and patterned ground to scale-invariant topography. *Earth-Science Reviews* **29**, 57-75.

[17] Hassan, M.A., 1993. Bed material and bedload movement in two ephemeral streams. *Special Publications of the International Association of Sedimentologists* **17**, 37-49.

[18] Hassan, M.A. and M. Church, 1992. The movement of individual grains on the streambed *in* Billi, P., Hey. R.D., Thorne, C.R., and P. Tacconi (Eds.), *Dynamics of Gravel-bed Rivers*, John Wiley & Sons Ltd.

[19] Koster, E.H., 1978. Transverse Ribs: Their characteristic origin and paleohydraulic significance. *Canadian Society of Petroleum Geologists Memoirs* **5**, 161-186.

[20] Landry, W. and B.T. Werner, 1994. Computer simulations of self-organized wind ripple patterns. *Physica* **D**, 238-260.

[21] Laronne, J.B. and M.A. Carson, 1978. Interrelationships between bed morphology and bed-material transport for a small, gravel-bed channel. *Sedimentology* **23**, 67-85.

[22] Martini, I.P., 1977. Gravelly flood deposits of Irvine Creek, Ontario, Canada. *Sedimentology* **24**, 603-622.

[23] McDonald, B.C. and I. Banerjee, 1971. Sediments and bedforms on a braided outwash plain. *Canadian Journal of Earth Sciences* **8**, 1282-1301.

[24] McDonald, B.C. and T.J. Day, 1978. An experimental flume study on the formation of transverse ribs. *Geological Survey of Canada Paper* **78-1A**, 441-451.

[25] Nowell, A.R. and M. Church, 1979. Turbulent flow in a depth-limited boundary layer. *Journal of Geophysical Research* **84**, 4816-4824.

[26] Press, W.H., Teukolsky, S.A., Vetterling, W.T. and B.P. Flannery, 1994. *Numerical Recipes in C.* Second Edition. Cambridge University Press.

[27] Reid, I and L.E. Frostick, 1984. Particle interaction and its effect on the thresholds of initial and final bedload motion in coarse alluvial channels, *in* Koster, E.H., and Steel, R.J. (Eds.), Sedimentology of Gravels and Conglomerates *Canadian Society of Petroleum Geologists Memoir* **10**, 61-68.

[28] Reid, I. and M.A. Hassan, 1992. The influence of microform bed roughness elements on flow and sediment transport in gravel bed rivers. *Earth Surface Processes and Landforms* **15**, 739-750.

[29] Rossi, R.E., Mulla, D.J., Journel, A.G., and E.H. Franz, 1992. Geostatistical Tools for Modeling and Interpreting Ecological Spatial Dependence, *Ecological Monographs*, **62**,(2), 217-314.

[30] Weatherby, H., 1996. Unpublished M.Sc. thesis, Department of Earth and Ocean Science, University of British Columbia.

[31] Werner, B.T., 1995. Eolian dunes: computer simulation and attractor interpretation. *Geology* **23** (12), 1107-1110.

[32] Werner, B.T. and T.M. Fink, 1993. Beach cusps as self-organized patterns. *Science* **260**, 968-971.

[33] Werner, B.T. and B. Hallet, 1993. Numerical simulation of self-organized stone stripes. *Nature* **361**, 142-145.

[34] White, C.M., 1940. The equilibrium of grains on the bed of a stream. *Proceedings of the Royal Society Series A*, **174**, 322-338.